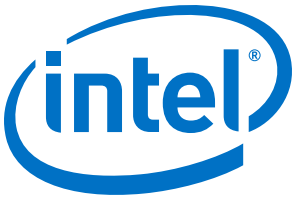


HDMI Intel® FPGA IP User Guide

Updated for Intel® Quartus® Prime Design Suite: **19.3**

IP Version: **19.1.0**

Subscribe
Send Feedback

UG-HDMI | 2019.10.10
Latest document on the web: **PDF** | **HTML**

Contents

1. HDMI Intel® FPGA IP Quick Reference

The Intel® FPGA High-Definition Multimedia Interface (HDMI) IP provides support for next-generation video display interface technology. The HDMI Intel FPGA IP is part of the Intel FPGA IP Library, which is distributed with the Intel Quartus® Prime software.

Information		Description
IP Core Information	Core Features	• Conforms to the *High-Definition Multimedia Interface (HDMI) Specification versions 1.4 and 2.0b* • Supports transmitter and receiver on a single device transceiver quad • Supports pixel frequency up to 600 MHz • Supports RGB and YCbCr 444, 422, and 420 color modes • Accepts standard H-SYNC, V-SYNC, data enable, RGB video format, and YCbCr video format • Supports up to 32 audio channels in 2-channel and 8-channel layouts. • Supports 1, 2, or 4 symbols per clock • Supports 8, 10, 12, or 16 bits per component (bpc) • Supports single link Digital Visual Interface (DVI) • Supports High Dynamic Range (HDR) InfoFrame insertion and filter through the provided design examples • Supports up to 1,536 kHz audio sample frequency • Supports the High-bandwidth Digital Content Protection (HDCP) feature for Intel Arria® 10 devices
	Typical Application	• Interfaces within a PC and monitor • External display connections, including interfaces between a PC and monitor or projector, between a PC and TV, or between a device such as a DVD player and TV display
	Device Family	Supports Intel Stratix® 10 (H-tile and L-tile), Intel Arria 10, Intel Cyclone® 10 GX, Arria V, and Stratix V FPGA devices
	Design Tools	• Intel Quartus Prime software for IP design instantiation and compilation • Timing Analyzer in the Intel Quartus Prime software for timing analysis • ModelSim* - Intel FPGA Edition or ModelSim - Intel FPGA Starter Edition, NCSim, Riviera-PRO*, VCS*, VCS MX, and Xcelium* Parallel software for design simulation

Note: The HDMI Intel FPGA IP provides offline support for the High-bandwidth Digital Content Protection (HDCP) feature. For further information, contact Intel at https://www.intel.com/content/www/us/en/broadcast/products/programmable/applications/connectivity-solutions.html.

Related Information

- HDMI Intel Arria 10 FPGA IP Design Example User Guide
  For more information about the Intel Arria 10 design examples.

- HDMI Intel Cyclone 10 GX FPGA IP Design Example User Guide
  For more information about the Intel Cyclone 10 GX design examples.
- HDMI Intel Stratix 10 FPGA IP Design Example User Guide
  For more information about the Intel Stratix 10 design examples.
- HDMI Intel FPGA IP User Guide Archives on page 116
  Provides a list of user guides for previous versions of the HDMI Intel FPGA IP.

2. HDMI Overview

The HDMI Intel FPGA IP core provides support for next generation video display interface technology.

The HDMI standard specifies a digital communications interface for use in both internal and external connections:

- Internal connections—interface within a PC and monitor
- External display connections—interface between a PC and monitor or projector, between a PC and TV, or between a device such a DVD player and TV display.

The HDMI system architecture consists of sinks and sources. A device may have one or more HDMI inputs and outputs.

The HDMI cable and connectors carry four differential pairs that make up the Transition Minimized Differential Signaling (TMDS) data and clock channels. You can use these channels to carry video, audio, and auxiliary data.

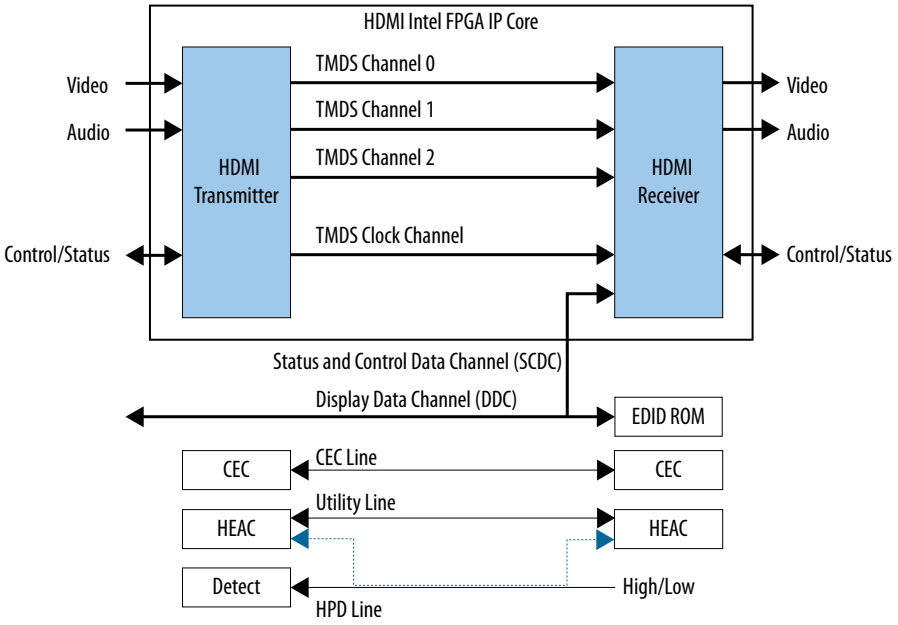
The HDMI also carries a Video Electronics Standards Association (VESA) Display Data Channel (DDC) and Status and Control Data Channel (SCDC). The DDC configures and exchanges status between a single source and a single sink. The source uses the DDC to read the sink's Enhanced Extended Display Identification Data (E-EDID) to discover the sink's configuration and capabilities.

The optional Consumer Electronics Control (CEC) protocol provides high-level control functions between various audio visual products in your environment.

The optional HDMI Ethernet and Audio Return Channel (HEAC) provides Ethernet compatible data networking between connected devices and an audio return channel in the opposite direction of TMDS. The HEAC also uses Hot-Plug Detect (HPD) line for link detection.

Figure 1. HDMI Intel FPGA Core Block Diagram

The figure below illustrates the blocks in the HDMI Intel FPGA IP core.



Based on TMDS encoding, the HDMI protocol allows the transmission of both audio and video data between source and sink devices.

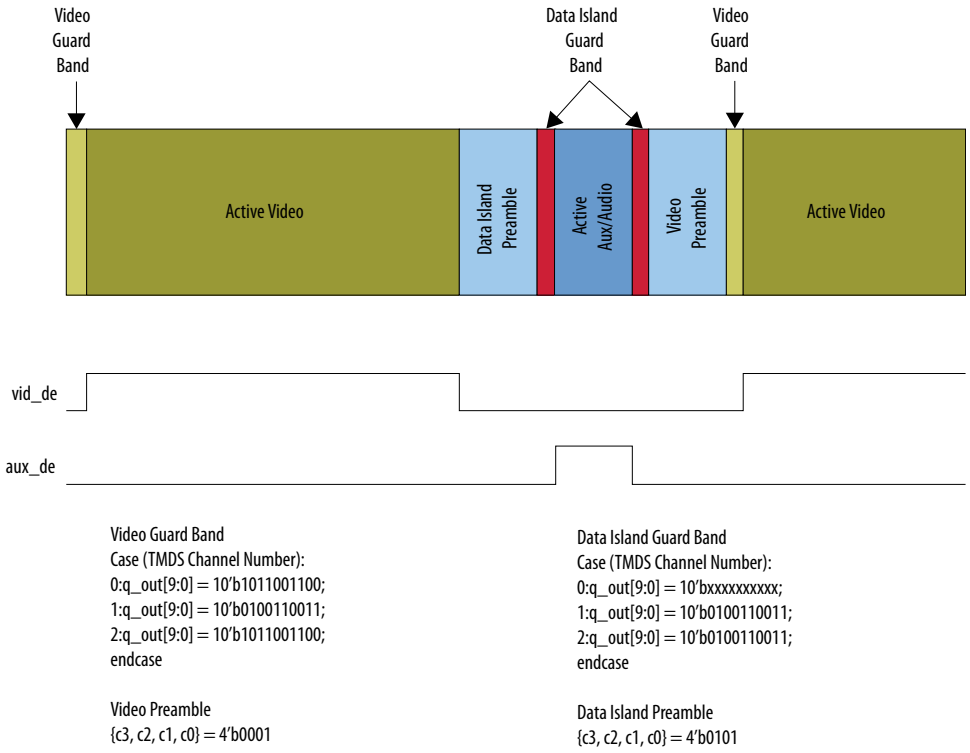
An HDMI interface consists of three color channels accompanied by a single clock channel. You can use each color line to transfer both individual RGB colors and auxiliary data.

Note: Refer to *AN 837: Design Guidelines for Intel FPGA HDMI* to know more about the channel mapping to the RGB colors.

The receiver uses the TMDS clock as a frequency reference for data recovery on the three TMDS data channels. This clock typically runs at the video pixel rate.

TMDS encoding is based on an 8-bit to 10-bit algorithm. This protocol attempts to minimize data channel transition, and yet maintain sufficient transition so that a sink device can lock reliably to the data stream.

Figure 2. HDMI Intel FPGA Video Stream Data



The figure above illustrates two data streams:

- Data stream in green—transports color data
- Data stream in dark blue—transports auxiliary data

Table 1. Video Data and Auxiliary Data

The table below describes the function of the video data and auxiliary data.

Data	Description
Video data	• Packed representation of the video pixels clocked at the source pixel clock. • Encoded using the TMDS 8-bit to 10-bit algorithm.
Auxiliary data	• Transfers audio data together with a range of auxiliary data packets. • Sink devices use auxiliary data packets to correctly reconstruct video and audio data. • Encoded using the TMDS Error Reduction Coding–4 bits (TERC4) encoding algorithm.

Each data stream section is preceded with guard bands and pre-ambles. The guard bands and pre-ambles allow for accurate synchronization with received data streams.

The following figures show the arrangement of the video data, video data enable, video H-SYNC, and video V-SYNC in 1, 2, and 4 symbols per clock.

Figure 3. Video Data, Video Data Valid, H-SYNC, and V-SYNC—1 Symbol per Clock

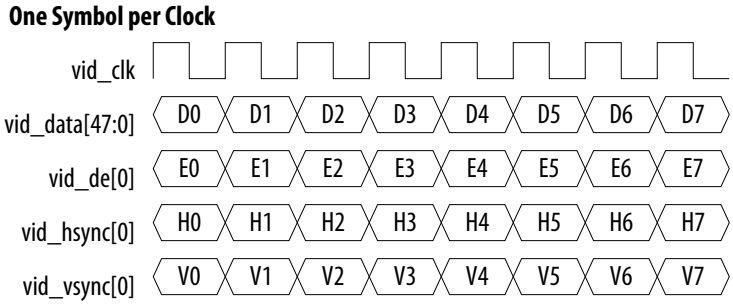


Figure 4. Video Data, Video Data Valid, H-SYNC, and V-SYNC—2 Symbols per Clock

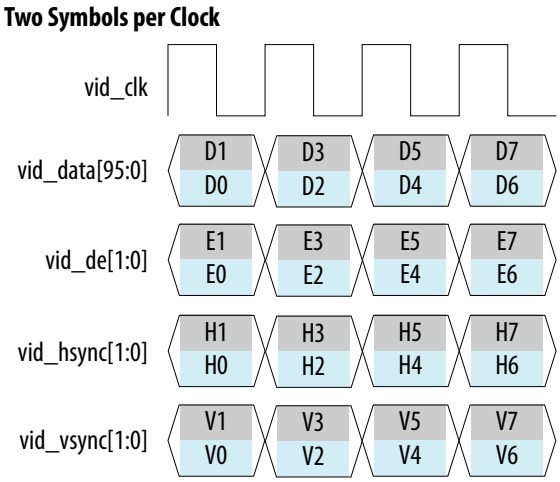
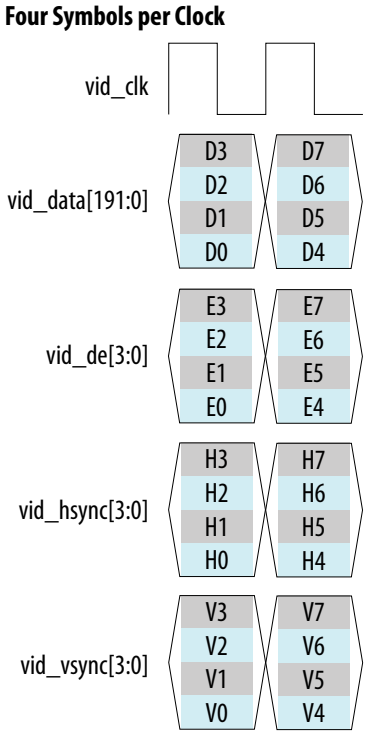


Figure 5. Video Data, Video Data Valid, H-SYNC, and V-SYNC—4 Symbols per Clock



Related Information

AN 837: Design Guidelines for Intel FPGA HDMI

2.1. Release Information

IP versions are the same as the Intel Quartus Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

The IP versioning scheme (X.Y.Z) number changes from one software version to another. A change in:

- X indicates a major revision of the IP. If you update your Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Table 2. HDMI Intel FPGA IP Release Information

Item	Description
IP Version	19.1.0
Intel Quartus Prime Version	19.3
Release Date	September 2019
Ordering Code	IP-HDMI

2.2. Device Family Support

Table 3. Intel Device Family Support

Device Family	Support Level
Intel Stratix 10 (H-tile and L-tile)	Final
Intel Arria 10	Final
Intel Cyclone 10 GX	Final
Arria V	Final
Stratix V	Final

The following terms define device support levels for Intel FPGA IP cores:

- Advance support—the IP core is available for simulation and compilation for this device family. Timing models include initial engineering estimates of delays based on early post-layout information. The timing models are subject to change as silicon testing improves the correlation between the actual silicon and the timing models. You can use this IP core for system architecture and resource utilization studies, simulation, pinout, system latency assessments, basic timing assessments (pipeline budgeting), and I/O transfer strategy (data-path width, burst depth, I/O standards tradeoffs).
- Preliminary support—the IP core is verified with preliminary timing models for this device family. The IP core meets all functional requirements, but might still be undergoing timing analysis for the device family. It can be used in production designs with caution.
- Final support—the IP core is verified with final timing models for this device family. The IP core meets all functional and timing requirements for the device family and can be used in production designs.

2.3. Resource Utilization

The resource utilization data indicates typical expected performance for the HDMI Intel FPGA IP.

Table 4. HDMI Data Rate

The table lists the maximum data rates for HDMI Intel FPGA IP configurations of 1, 2, and 4 symbols per clock.

Devices	Maximum Data Rate (Mbps)		
	1 Symbol per Clock	2 Symbols per Clock	4 Symbols per Clock
Intel Stratix 10	Not Supported	5,940	Not Supported

continued...

Devices	Maximum Data Rate (Mbps)		
	1 Symbol per Clock	2 Symbols per Clock	4 Symbols per Clock
		(Example: 4Kp60 8 bpc)	
Intel Arria 10	Not Supported	5,940 (Example: 4Kp60 8 bpc)	Not Supported
Intel Cyclone 10 GX	Not Supported	5,940 (Example: 4Kp60 8 bpc)	Not Supported
Arria V GX	1,875 (Example: 1080p60 10 bpc)	3,276.8 (Example: 4Kp30 8 bpc)	5,940 (Example: 4Kp60 8 bpc)
Stratix V	2,970 (Example: 4Kp30 8 bpc)	5,940 (Example: 4Kp60 8 bpc)	Not Supported

Table 5. Color Depth Supported for Each Pixel Encoding

Pixel Encoding	Color Depth			
	8	10	12	16
RGB	Yes	Yes	Yes	Yes
YCbCr 4:4:4	Yes	Yes	Yes	Yes
YCbCr 4:2:2(1)	Yes	Yes	Yes	Not applicable
YCbCr 4:2:0	Yes	Yes	Yes	Yes

Table 6. HDMI Intel FPGA Resource Utilization

The table lists the performance data for the different Intel FPGA devices.

Device	Symbols per Clock	Direction	ALMs	Logic Registers		Memory	
				Primary	Secondary	Bits	M10K or M20K
Intel Stratix 10 H-tile	2	RX	5.041	6,633	902	38,400	14
	2	TX	4,975	7,559	1,368	37,568	13
Intel Stratix 10 L-tile	2	RX	5,025	6,584	967	38,400	14
	2	TX	4,966	7,539	1,425	37,568	13
Intel Arria 10	2	RX	3,952	5,716	1,049	38,400	14
	2	TX	4,422	7,016	1,701	36,968	13
Intel Cyclone 10 GX	2	RX	4,000	5,768	965	38,400	14
	2	TX	4,484	7,167	1,629	36,968	13
Arria V GX	1	RX	2,630	4,039	402	35,712	13
	1	TX	2,700	4,462	417	11,108	11
	2	RX	3,446	4,656	531	38,400	14
	2	TX	3,759	6,091	450	12,680	13

continued...

(1) According to *HDMI 1.4b Specification Section 6.5.1*, 8 and 10 bpc use the same pixel encoding as 12 bpc, but the valid bits are left-justified with zeros padding the bits below the least significant bit.

Device	Symbols per Clock	Direction	ALMs	Logic Registers		Memory	
				Primary	Secondary	Bits	M10K or M20K
	4	RX	4,895	5,937	614	43,776	20
	4	TX	6,135	9,156	445	15,824	18
Stratix V	1	RX	2,592	3,946	398	35,712	13
	1	TX	2,634	4,415	461	11,108	11
	2	RX	3,337	4,619	440	38,400	14
	2	TX	3,644	5,919	680	12,680	13

Table 7. Recommended Speed Grades for Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX Devices

Device	Lane Rate (Mbps)	Interface Width (bits)	Speed Grades
Intel Stratix 10	6,000	20	-1, -2
Intel Arria 10	6,000	20	-1, -2
Intel Cyclone 10 GX	6,000	20	-5

Table 8. HDCP Resource Utilization

The table lists the HDCP resource data for HDMI Intel FPGA IP configurations of 2 symbols per clock for Intel Arria 10 devices.

HDCP IP	ALMs	Combinatorial ALUTs	Registers	M20K	DSP
HDCP 2.3 TX	6,336	10,568	12,004	10	3
HDCP 2.3 RX	6,950	11,768	12,648	11	3
HDCP 1.4 TX	1,815	3,019	3,886	2	0
HDCP 1.4 RX	1,368	2,315	3,108	3	0

3. HDMI Intel FPGA IP Getting Started

This chapter provides a general overview of the Intel IP core design flow to help you quickly get started with the HDMI Intel FPGA IP core. The Intel FPGA IP Library is installed as part of the Intel Quartus Prime installation process. You can select and parameterize any Intel FPGA IP core from the library. Intel provides an integrated parameter editor that allows you to customize the HDMI Intel FPGA IP core to support a wide variety of applications. The parameter editor guides you through the setting of parameter values and selection of optional ports.

Related Information

- Introduction to Intel FPGA IP Cores
  Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.
- Creating Version-Independent IP and Platform Designer Simulation Scripts
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- Project Management Best Practices
  Guidelines for efficient management and portability of your project and IP files.

3.1. Installing and Licensing Intel FPGA IP Cores

The Intel Quartus Prime software installation includes the Intel FPGA IP library. This library provides many useful IP cores for your production use without the need for an additional license. Some Intel FPGA IP cores require purchase of a separate license for production use. The Intel FPGA IP Evaluation Mode allows you to evaluate these licensed Intel FPGA IP cores in simulation and hardware, before deciding to purchase a full production IP core license. You only need to purchase a full production license for licensed Intel IP cores after you complete hardware testing and are ready to use the IP in production.

The Intel Quartus Prime software installs IP cores in the following locations by default:

Figure 6. IP Core Installation Path

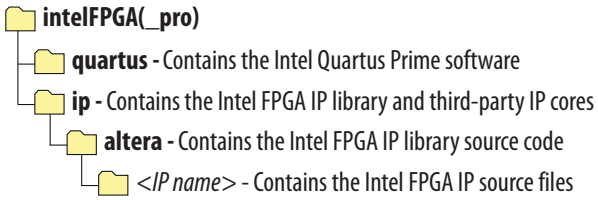


Table 9. IP Core Installation Locations

Location	Software	Platform
<*drive*>:\intelFPGA_pro\quartus\ip\altera	Intel Quartus Prime Pro Edition	Windows*
<*drive*>:\intelFPGA\quartus\ip\altera	Intel Quartus Prime Standard Edition	Windows
<*home directory*>:/intelFPGA_pro/quartus/ip/altera	Intel Quartus Prime Pro Edition	Linux*
<*home directory*>:/intelFPGA/quartus/ip/altera	Intel Quartus Prime Standard Edition	Linux

Note: The Intel Quartus Prime software does not support spaces in the installation path.

3.1.1. Intel FPGA IP Evaluation Mode

The free Intel FPGA IP Evaluation Mode allows you to evaluate licensed Intel FPGA IP cores in simulation and hardware before purchase. Intel FPGA IP Evaluation Mode supports the following evaluations without additional license:

- Simulate the behavior of a licensed Intel FPGA IP core in your system.
- Verify the functionality, size, and speed of the IP core quickly and easily.
- Generate time-limited device programming files for designs that include IP cores.
- Program a device with your IP core and verify your design in hardware.

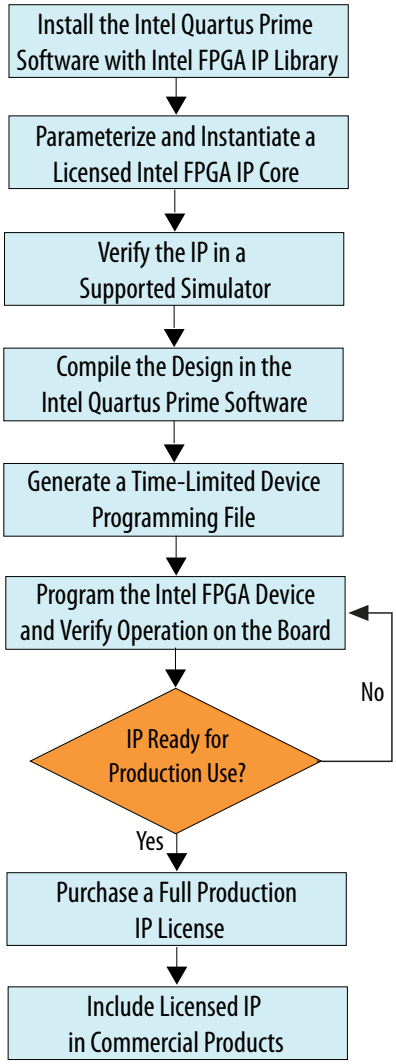
Intel FPGA IP Evaluation Mode supports the following operation modes:

- **Tethered**—Allows running the design containing the licensed Intel FPGA IP indefinitely with a connection between your board and the host computer. Tethered mode requires a serial joint test action group (JTAG) cable connected between the JTAG port on your board and the host computer, which is running the Intel Quartus Prime Programmer for the duration of the hardware evaluation period. The Programmer only requires a minimum installation of the Intel Quartus Prime software, and requires no Intel Quartus Prime license. The host computer controls the evaluation time by sending a periodic signal to the device via the JTAG port. If all licensed IP cores in the design support tethered mode, the evaluation time runs until any IP core evaluation expires. If all of the IP cores support unlimited evaluation time, the device does not time-out.
- **Untethered**—Allows running the design containing the licensed IP for a limited time. The IP core reverts to untethered mode if the device disconnects from the host computer running the Intel Quartus Prime software. The IP core also reverts to untethered mode if any other licensed IP core in the design does not support tethered mode.

When the evaluation time expires for any licensed Intel FPGA IP in the design, the design stops functioning. All IP cores that use the Intel FPGA IP Evaluation Mode time out simultaneously when any IP core in the design times out. When the evaluation time expires, you must reprogram the FPGA device before continuing hardware verification. To extend use of the IP core for production, purchase a full production license for the IP core.

You must purchase the license and generate a full production license key before you can generate an unrestricted device programming file. During Intel FPGA IP Evaluation Mode, the Compiler only generates a time-limited device programming file (<*project name*>_time_limited.sof) that expires at the time limit.

Figure 7. Intel FPGA IP Evaluation Mode Flow



Note: Refer to each IP core's user guide for parameterization steps and implementation details.

Intel licenses IP cores on a per-seat, perpetual basis. The license fee includes first-year maintenance and support. You must renew the maintenance contract to receive updates, bug fixes, and technical support beyond the first year. You must purchase a full production license for Intel FPGA IP cores that require a production license, before generating programming files that you may use for an unlimited time. During Intel FPGA IP Evaluation Mode, the Compiler only generates a time-limited device programming file (`<project name>_time_limited.sof`) that expires at the time limit. To obtain your production license keys, visit the Self-Service Licensing Center.

The Intel FPGA Software License Agreements govern the installation and use of licensed IP cores, the Intel Quartus Prime design software, and all unlicensed IP cores.

Related Information

- Intel Quartus Prime Licensing Site
- Introduction to Intel FPGA Software Installation and Licensing

3.2. Specifying IP Core Parameters and Options

Follow these steps to specify the HDMI Intel FPGA IP core parameters and options.

1. Create a Intel Quartus Prime project using the **New Project Wizard** available from the File menu.
2. On the **Tools** menu, click **IP Catalog**.
3. Under **Installed IP**, double-click **Library ➤ Interface ➤ Protocols ➤ Audio&Video ➤ HDMI Intel FPGA**.
   The parameter editor appears.
4. Specify a top-level name for your custom IP variation. This name identifies the IP core variation files in your project. If prompted, also specify the targeted FPGA device family and output file HDL preference. Click **OK**.
5. Specify parameters and options in the HDMI parameter editor:
   - Optionally select preset parameter values. Presets specify all initial parameter values for specific applications (where provided).
   - Specify parameters defining the IP core functionality, port configurations, and device-specific features.
   - Specify options for generation of a timing netlist, simulation model, testbench, or example design (where applicable).
   - Specify options for processing the IP core files in other EDA tools.
6. Click **Generate** to generate the IP core and supporting files, including simulation models.
7. Click **Close** when file generation completes.
8. Click **Finish**.
9. If you generate the HDMI Intel FPGA IP core instance in a Intel Quartus Prime project, you are prompted to add `Intel Quartus Prime IP File (.qip)` and `Intel Quartus Prime Simulation IP File (.sip)` to the current Intel Quartus Prime project.

4. HDMI Hardware Design Examples

Intel offers design examples that you can simulate, compile, and test in hardware.

The implementation of the HDMI Intel FPGA IP on hardware requires additional components specific to the targeted device.

4.1. HDMI Hardware Design Examples for Intel Arria 10, Intel Cyclone 10 GX, and Intel Stratix 10 Devices

The HDMI Intel FPGA IP core offers design examples that you can generate through the IP catalog in the Intel Quartus Prime Pro Edition software.

Related Information

- HDMI Intel Arria 10 FPGA IP Design Example User Guide
  For more information about the Intel Arria 10 design examples.
- HDMI Intel Cyclone 10 GX FPGA IP Design Example User Guide
  For more information about the Intel Cyclone 10 GX design examples.
- HDMI Intel Stratix 10 FPGA IP Design Example User Guide
  For more information about the Intel Stratix 10 design examples.

4.2. HDCP Over HDMI Design Example for Intel Arria 10 Devices

The HDCP over HDMI hardware design example helps you to evaluate the functionality of the HDCP feature and enables you to use the feature in your Intel Arria 10 designs.

Note: The HDCP feature is not included in the 19.3 version of the Intel Quartus Prime Pro Edition software. To access the HDCP feature, contact Intel at https://www.intel.com/content/www/us/en/broadcast/products/programmable/applications/connectivity-solutions.html.

4.2.1. High-bandwidth Digital Content Protection (HDCP)

High-bandwidth Digital Content Protection (HDCP) is a form of digital rights protection to create a secure connection between the source to the display.

Intel created the original technology, which is licensed by the Digital Content Protection LLC group. HDCP is a copy protection method where the audio/video stream is encrypted between the transmitter and the receiver, protecting it against illegal copying.

The HDCP features adheres to *HDCP Specification version 1.4* and *HDCP Specification version 2.3*.

The HDCP 1.4 and HDCP 2.3 IPs perform all computation within the hardware core logic with no confidential values (such as private key and session key) being accessible from outside the encrypted IP.

Table 10. HDCP IP Functions

HDCP IP	Functions
HDCP 1.4 IP	• Authentication exchange — Computation of master key (Km) — Generation of random An — Computation of session key (Ks), M0 and R0. • Authentication with repeater — Computation and verification of V and V' • Link integrity verification — Computation of frame key (Ki), Mi and Ri. • All cipher modes including hdcpBlockCipher, hdcpStreamCipher, hdcpRekeyCipher, and hdcpRngCipher • Original encryption status signaling (DVI) and enhanced encryption status signaling (HDMI) • True random number generator (TRNG) — Hardware based, full digital implementation and non-deterministic random number generator
HDCP 2.3 IP	• Master Key (km), Session Key (ks) and nonce (rn, riv) generation — Compliant to NIST.SP800-90A random number generation • Authentication and key exchange — Generation of random numbers for rtx and rrx compliant to NIST.SP800-90A random number generation — Signature verification of receiver certificate (certrx) using DCP public key (kpubdcp) — 3072 bits RSASSA-PKCS#1 v1.5 — RSAES-OAEP (PKCS#1 v2.1) encryption and decryption of Master Key (km) — Derivation of kd (dkey0, dkey1) using AES-CTR mode — Computation and verification of H and H' — Computation of Ekh(km) and km (pairing) • Authentication with repeater — Computation and verification of V and V' — Computation and verification of M and M' • System renewability (SRM) — SRM signature verification using kpubdcp — 3072 bits RSASSA-PKCS#1 v1.5 • Session Key exchange • Generation and computation of Edkey(ks) and riv. • Derivation of dkey2 using AES-CTR mode • Locality Check — Computation and verification of L and L' — Generation of nonce (rn)

continued...

HDCP IP	Functions
	• Data stream management — AES-CTR mode based key stream generation • Asymmetric crypto algorithms — RSA with modulus length of 1024 (kpubrx) and 3072 (kpubdcp) bits — RSA-CRT (Chinese Remainder Theorem) with modulus length of 512 (kprivrx) bits and exponent length of 512 (kprivrx) bits • Low-level cryptographic function — Symmetric crypto algorithms • AES-CTR mode with a key length of 128 bits — Hash, MGF and HMAC algorithms • SHA256 • HMAC-SHA256 • MGF1-SHA256 — True random number generator (TRNG) • NIST.SP800-90A compliant • Hardware based, full digital implementation and non-deterministic random number generator

4.2.2. HDCP Over HDMI Design Example Architecture

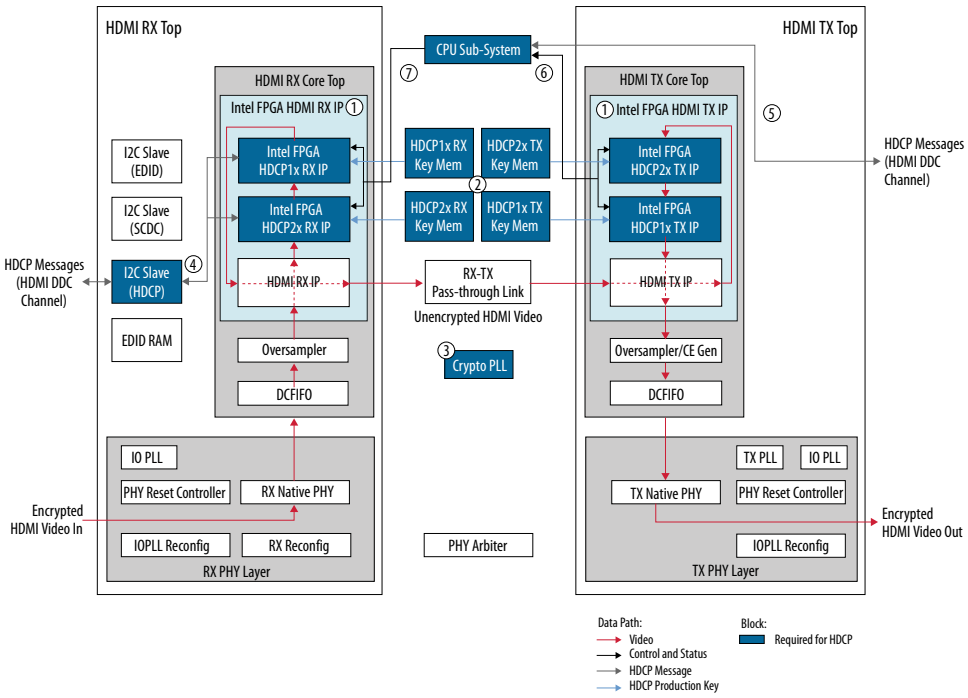
The HDCP feature protects data as the data is transmitted between devices connected through an HDMI or other HDCP-protected digital interfaces.

The HDCP-protected systems include three types of devices:

- Sources (TX)
- Sinks (RX)
- Repeaters

This design example demonstrates the HDCP system in a repeater device where it accepts data, decrypts, then re-encrypts the data, and finally retransmits data. Repeaters have both HDMI inputs and outputs. It instantiates the FIFO buffers to perform a direct HDMI video stream pass-through between the HDMI sink and source. It may perform some signal processing, such as converting videos into a higher resolution format by replacing the FIFO buffers with the Video and Image Processing (VIP) Suite IP cores.

Figure 8. HDCP Over HDMI Design Example Block Diagram



The following descriptions about the architecture of the design example correspond to the HDCP over HDMI design example block diagram.

1. The HDCP1x and HDCP2x are IPs that are available through the HDMI Intel FPGA IP parameter editor. When you configure the HDMI IP in the parameter editor, you can enable and include either HDCP1x or HDCP2x or both IPs as part of the subsystem. With both HDCP IPs enabled, the HDMI IP configures itself in the cascade topology where the HDCP2x and HDCP1x IPs are connected back-to-back.
   - The HDCP egress interface of the HDMI TX sends unencrypted audio video data.
   - The unencrypted data gets encrypted by the active HDCP block and sent back into the HDMI TX over the HDCP Ingress interface for transmission over the link.
   - The CPU subsystem as the authentication master controller ensures that only one of the HDCP TX IPs is active at any given time and the other one is passive.
   - Similarly, the HDCP RX also decrypts data received over the link from an external HDCP TX.
2. You need to program the HDCP IPs with Digital Content Protection (DCP) issued production keys. Load the following keys:

Table 11. DCP-issued Production Keys

HDCP	TX/RX	Keys
HDCP2x	TX	16 bytes: Global Constant (lc128)
	RX	• 16 bytes (same as TX): Global Constant (lc128) • 320 bytes: RSA Private Key (kprivrx) • 522 bytes: RSA Public Key Certificate (certrx)
HDCP1x	TX	• 5 bytes: TX Key Selection Vector (Aksv) • 280 bytes: TX Private Device Keys (Akeys)
	RX	• 5 bytes: RX Key Selection Vector (Bksv) • 280 bytes: RX Private Device Keys (Bkeys)

The design example implements the key memories as simple dual-port, dual-clock synchronous RAM. For small key size like HDCP2x TX, the IP implements the key memory using registers in regular logic.

Note: Intel does not provide the HDCP production keys with the design example or Intel FPGA IPs under any circumstances. To use the HDCP IPs or the design example, you must become an HDCP adopter and acquire the production keys directly from the Digital Content Protection LLC (DCP).

To run the design example, you either edit the key memory files at compile time to include the production keys or implement logic blocks to securely read the production keys from an external storage device and write them into the key memories at run time.

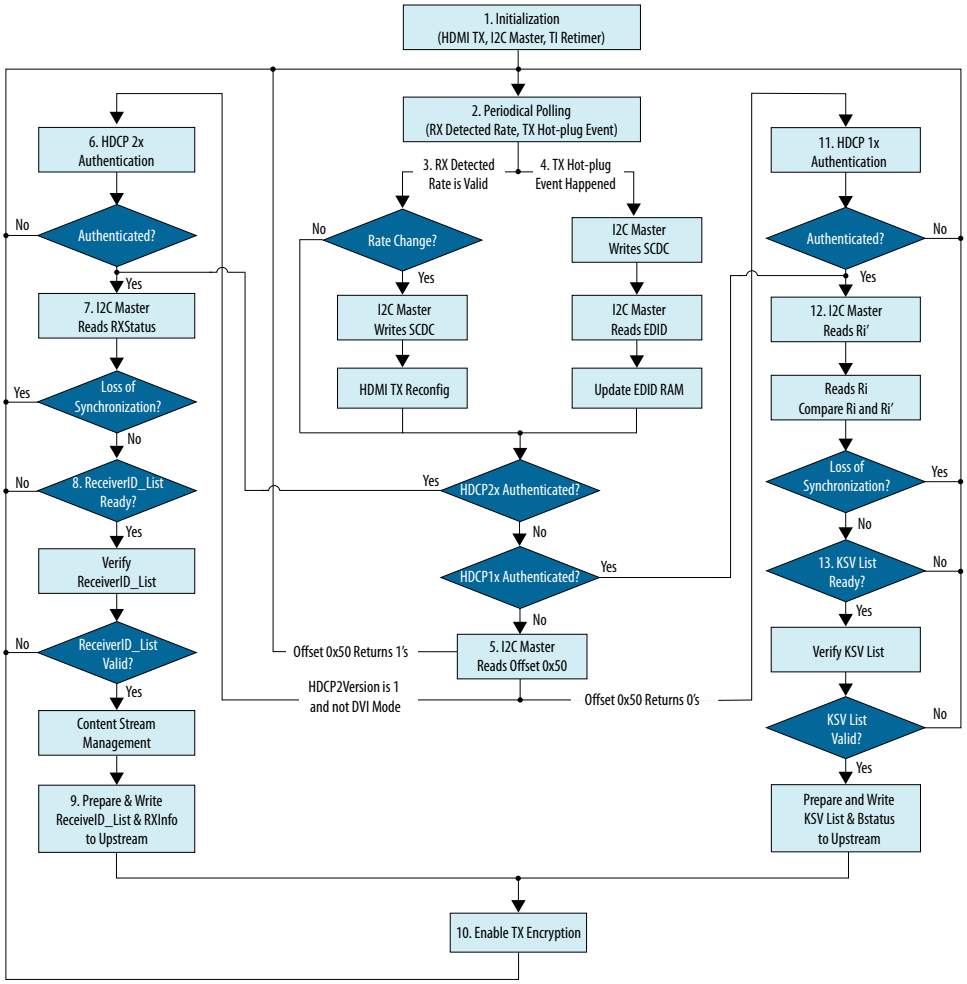
3. You can clock the cryptographic functions implemented in the HDCP2x IP with any frequency up to 200 MHz. The frequency of this clock determines how quickly the HDCP2x authentication operates. You can opt to share the 100 MHz clock used for Nios II processor but the authentication latency would be doubled compared to using a 200 MHz clock.
4. The values that must be exchanged between the HDCP TX and the HDCP RX are communicated over the HDMI DDC interface (I^2C serial interface) of the HDCP-protected interface. The HDCP RX must present a logical device on the I^2C bus for each link that it supports. The I^2C slave is duplicated for HDCP port with device address of 0x74. It drives the HDCP register port (Avalon-MM) of both the HDCP2x and HDCP1x RX IPs.
5. The HDMI TX uses the I^2C master to read the EDID from RX and transfer the SCDC data that is required for HDMI 2.0 operation to RX. The same I^2C master that is driven by the Nios II processor is also used to transfer the HDCP messages between TX and RX. The I^2C master is embedded in the CPU subsystem.
6. The Nios II processor acts as the master in the authentication protocol and drives the control and status registers (Avalon-MM) of both the HDCP2x and HDCP1x TX IPs. The software drivers implements the authentication protocol state machine including certificate signature verification, master key exchange, locality check, session key exchange, pairing, link integrity check (HDCP1x), and authentication with repeaters, such as topology information propagation and stream management information propagation. The software drivers do not implement any of the cryptographic functions required by the authentication protocol. Instead, the HDCP IP hardware implements all the cryptographic functions ensuring no confidential values can be accessed.
7. In a true repeater demonstration where propagating topology information upstream is required, the Nios II processor drives the Repeater Message Port (Avalon-MM) of both HDCP2x and HDCP1x RX IPs. The Nios II processor clears the

RX REPEATER bit to 0 when it detects the connected downstream is not HDCP-capable or when no downstream is connected. Without downstream connection, the RX system is now an end-point receiver, rather than a repeater. Conversely, the Nios II processor sets the RX REPEATER bit to 1 upon detecting the downstream is HDCP-capable.

4.2.3. Nios II Processor Software Flow

The Nios II software flowchart includes the HDCP authentication controls over HDMI application.

Figure 9. Nios II Processor Software Flowchart



1. The Nios II software initializes and resets the HDMI TX PLL, TX transceiver PHY, I^2C master and the external TI retimer.
2. The Nios II software polls periodic rate detection valid signal from RX rate detection circuit to determine whether video resolution has changed and if TX reconfiguration is required. The software also polls the TX hot-plug detect signal to determine whether a TX hot-plug event has occurred.
3. When a valid signal received from RX rate detection circuit, the Nios II software reads the SCDC and clock depth values from the HDMI RX and retrieves the clock frequency band based on the detected rate to determine whether HDMI TX PLL and transceiver PHY reconfiguration are required. If TX reconfiguration is required, the Nios II software commands the I^2C master to send the SCDC value over to external RX. It then commands to reconfigure the HDMI TX PLL and TX transceiver PHY, followed by device recalibration, and reset sequence. If the rate does not change, neither TX reconfiguration nor HDCP re-authentication is required.
4. When a TX hot-plug event has occurred, the Nios II software commands the I^2C master to send the SCDC value over to external RX, and then read EDID from RX and update the internal EDID RAM. The software then propagates the EDID information to the upstream.
5. The Nios II software starts the HDCP activity by commanding the I^2C master to read offset 0x50 from external RX to detect if the downstream is HDCP-capable, or otherwise:
   - If the returned `HDCP2Version` value is 1, the downstream is HDCP2x-capable.
   - If the returned value of the entire 0x50 reads are 0's, the downstream is HDCP1x-capable.
   - If the returned value of the entire 0x50 reads are 1's, the downstream is either not HDCP-capable or inactive.
   - If the downstream is previously not HDCP-capable or inactive but is currently HDCP-capable, the software sets the REPEATER bit of the repeater upstream (RX) to 1 to indicate the RX is now a repeater.
   - If the downstream is previously HDCP-capable but is currently not HDCP-capable or inactive, the software sets the REPEATER bit of to 0 to indicate the RX is now an endpoint receiver.
6. The software initiates the HDCP2x authentication protocol that includes RX certificate signature verification, master key exchange, locality check, session key exchange, pairing, authentication with repeaters such as topology information propagation.
7. When in authenticated state, the Nios II software commands the I^2C master to poll the `RxStatus` register from external RX, and if the software detects the `REAUTH_REQ` bit is set, it initiates re-authentication and disables TX encryption.
8. When the downstream is a repeater and the READY bit of the `RxStatus` register is set to 1, this usually indicates the downstream topology has changed. So, the Nios II software commands the I^2C master to read the `ReceiverID_List` from downstream and verify the list. If the list is valid and no topology error is detected, the software proceeds to the Content Stream Management module. Otherwise, it initiates re-authentication and disables TX encryption.
9. The Nios II software prepares the `ReceiverID_List` and `RxInfo` values and then writes to the Avalon-MM Repeater Message port of the repeater upstream (RX). The RX then propagates the list to external TX (upstream).

10. Authentication is complete at this point. The software enables TX encryption.
11. The software initiates the HDCP1x authentication protocol that includes key exchange and authentication with repeaters.
12. The Nios II software performs link integrity check by reading and comparing Ri' and Ri from external RX (downstream) and HDCP1x TX respectively. If the values do not match, this indicates loss of synchronization and the software initiates re-authentication and disables TX encryption.
13. If the downstream is a repeater and the READY bit of the `Bcaps` register is set to 1, this usually indicates that the downstream topology has changed. So, the Nios II software commands the I^2C master to read the KSV `list` value from downstream and verify the list. If the list is valid and no topology error is detected, the software prepares the KSV `list` and `Bstatus` value and writes to the Avalon-MM Repeater Message port of the repeater upstream (RX). The RX then propagates the list to external TX (upstream). Otherwise, it initiates re-authentication and disables TX encryption.

4.2.4. Design Walkthrough

Setting up and running the HDCP over HDMI design example consists of four stages.

1. Set up the hardware.
2. Edit the HDCP key memory files to include your HDCP production keys.
3. Build and compile the design.
4. View the results.

4.2.4.1. Set Up the Hardware

The first stage of the demonstration is to set up the hardware.

To set up the hardware for the demonstration:

1. Connect the Bitec HDMI 2.0 FMC daughter card (revision 11) to the Arria 10 development kit at FMC port B.
2. Connect the Arria 10 development kid to your PC using a USB cable.
3. Connect an HDMI cable from the HDMI RX connector on the Bitec HDMI 2.0 FMC daughter card to an HDCP-enabled HDMI device, such as a graphic card with HDMI output.
4. Connect another HDMI cable from the HDMI TX connector on the Bitec HDMI 2.0 FMC daughter card to an HDCP-enabled HDMI device, such as a television with HDMI input.

4.2.4.2. Include HDCP Production Keys

After setting up the hardware, you need to edit the HDCP key memory files to include your production keys.

To include the production keys, follow these steps.

1. Locate the following key memory files in the `hdcp2x/hw_demo/arria10/rtl/hdcp/` directory:

- `hdcp2x_tx_kmem.v`
- `hdcp2x_rx_kmem.v`
- `hdcp1x_tx_kmem.v`
- `hdcp1x_rx_kmem.v`

2. Open the `hdcp2x_rx_kmem.v` file and locate the predefined facsimile key R1 for Receiver Public Certificate and RX Private Key and Global Constant as shown in the examples below.

Figure 10. Wire Array of Facsimile Key R1 for Receiver Public Certificate

```
// Facsimile key R1
wire [4175:0] cert_rx_r1 =
]{
    128'h74_5b_b8_bd_04_af_b5_c5_c6_7b_c5_3a_34_90_a9_54,
    128'hc0_8f_b7_eb_a1_54_d2_4f_22_de_83_f5_03_a6_c6_68,
    128'h46_9b_c0_b8_c8_6c_db_26_f9_3c_49_2f_02_e1_71_df,
    128'h4e_f3_0e_c8_bf_22_9d_04_cf_bf_a9_0d_ff_68_ab_05,
    128'h6f_1f_12_8a_68_62_eb_fe_c9_ea_9f_a7_fb_8c_ba_b1,
    128'hbd_65_ac_35_9c_a0_33_b1_dd_a6_05_36_af_00_a2_7f,
    128'hbc_07_b2_dd_b5_cc_57_5c_dc_c0_95_50_e5_ff_1f_20,
    128'hdb_59_46_fa_47_c4_ed_12_2e_9e_22_bd_95_a9_85_59,
    128'ha1_59_3c_c7_83_01_00_01_10_00_0b_a3_73_77_dd_03,
    128'h18_03_8a_91_63_29_1e_a2_95_74_42_90_78_d0_67_25,
    128'hb6_32_2f_cc_23_2b_ad_21_39_3d_14_ba_37_a3_65_14,
    128'h6b_9c_cf_61_20_44_a1_07_bb_cf_c3_4e_95_5b_10_cf,
    128'hc7_6f_f1_c3_53_7c_63_a1_8c_b2_e8_ab_2e_96_97_c3,
    128'h83_99_70_d3_dc_21_41_f6_0a_d1_1a_ee_f4_cc_eb_fb,
    128'ha6_aa_b6_9a_af_1d_16_5e_e2_83_a0_4a_41_f6_7b_07,
    128'hbf_47_85_28_6c_a0_77_a6_a3_d7_85_a5_c4_a7_e7_6e,
    128'hb5_1f_40_72_97_fe_c4_81_23_a0_c2_90_b3_49_24_f5,
    128'hb7_90_2c_bf_fe_04_2e_00_a9_5f_86_04_ca_c5_3a_cc,
    128'h26_d9_39_7e_a9_2d_28_6d_c0_cc_6e_81_9f_b9_b7_11,
    128'h33_32_23_47_98_43_0d_a5_1c_59_f3_cd_d2_4a_b7_3e,
    128'h69_d9_21_53_9a_f2_6e_77_62_ae_50_da_85_c6_aa_c4,
    128'hb5_1c_cd_a8_a5_dd_6e_62_73_ff_5f_7b_d7_3c_17_ba,
    128'h47_0c_89_0e_62_79_43_94_aa_a8_47_f4_4c_38_89_a8,
    128'h81_ad_23_13_27_0c_17_cf_3d_83_84_57_36_e7_22_26,
    128'h2e_76_fd_56_80_83_f6_70_d4_5c_91_48_84_7b_18_db,
    128'h0e_15_3b_49_26_23_e6_a3_e2_c6_3a_23_57_66_b0_72,
    128'hb8_12_17_4f_86_fe_48_0d_53_ea_fe_31_48_7d_86_de,
    128'heb_82_86_1e_62_03_98_59_00_37_eb_61_e9_f9_7a_40,
    128'h78_1c_ba_bc_0b_88_fb_fd_9d_d5_01_11_94_e0_35_be,
    128'h33_e8_e5_36_fb_9c_45_cb_75_af_d6_35_ff_78_92_7f,
    128'ha1_7c_a8_fc_b7_f7_a8_52_a9_c6_84_72_3d_1c_c9_df,
    128'h35_c6_e6_00_e1_48_72_ce_83_1b_cc_f8_33_2d_4f_98,
     80'h75_00_3c_41_df_7a_ed_38_53_b1
};
```

Figure 11. Wire Array of Facsimile Key R1 for RX Private Key and Global Constant

```
wire [511:0] kprivrx_qinv_r1 =
{
    128'h3e_53_0a_f4_8e_75_e1_52_c6_24_e9_f7_bb_ac_3f_22,
    128'h5f_e8_e0_79_35_ff_91_ee_22_56_d2_00_68_32_c4_e1,
    128'h5f_ff_f8_b1_1d_ee_dc_57_81_d1_ab_8b_37_22_e3_9f,
    128'hd0_a1_c1_ce_1d_d0_24_23_a0_0e_f7_a6_db_a3_ea_d3
};

wire [511:0] kprivrx_dq_r1 =
{
    128'h10_0e_2e_18_ad_5d_e4_43_fe_81_1e_17_aa_d0_52_31,
    128'h5e_10_76_a2_35_d9_37_43_b0_f5_0c_04_81_e3_45_24,
    128'h6d_53_be_59_b6_81_58_c4_49_3e_d5_31_89_5d_2e_a2,
    128'h62_a9_0f_47_5e_8f_51_19_27_4e_66_4b_8a_72_89_bd
};

wire [511:0] kprivrx_dp_r1 =
{
    128'h60_71_9b_e9_e8_f3_97_1f_fe_13_d4_bf_7a_a2_0d_f6,
    128'h7b_cf_3e_aa_17_47_75_c3_7f_ec_d9_44_9e_c9_6a_02,
    128'he9_e4_af_56_51_d5_47_a9_09_b2_c5_16_a7_8b_2b_34,
    128'ha0_33_6e_2f_3d_95_7b_e8_ef_02_e4_14_bf_44_28_d9
};

wire [511:0] kprivrx_q_r1 =
{
    128'hbe_00_19_76_c6_b4_ba_19_d4_69_fa_4d_e2_f8_30_27,
    128'h36_2b_4c_c4_34_ab_d3_d9_8c_d6_b8_0d_37_5e_59_4b,
    128'h76_70_68_2b_1f_4c_3d_47_5f_a5_b1_cd_74_56_88_fe,
    128'h7c_f8_3b_30_6f_fd_c3_ed_87_3c_a1_53_84_c3_d2_7f
};

wire [511:0] kprivrx_p_r1 =
{
    128'hec_be_e5_5b_9e_7a_50_8a_96_80_c8_db_b0_ed_44_f2,
    128'hba_1d_5d_80_c1_c8_b3_c2_74_de_ee_28_ec_dc_78_c8,
    128'h67_53_07_f2_f8_75_9c_4c_a5_6c_48_94_c8_eb_ad_d7,
    128'h7d_d2_ea_df_74_20_62_c9_81_a8_3c_36_b9_ea_40_fd
};

wire [127:0] lc128_r1 =
{
    128'h93_ce_5a_56_a0_a1_f4_f7_3c_65_8a_1b_d2_ae_f0_f7
};
```

3. Locate the placeholder for the production keys and replace with your own production keys in their respective wire array in big endian format.

Figure 12. Wire Array of HDCP Production Keys (Placeholder)

```
// Production key (placeholder)
wire [4175:0] cert_rx_prod =
{
   4176'd0
};

wire [511:0] kprivrx_qinv_prod =
{
   512'd0
};

wire [511:0] kprivrx_dq_prod =
{
   512'd0
};

wire [511:0] kprivrx_dp_prod =
{
   512'd0
};

wire [511:0] kprivrx_q_prod =
{
   512'd0
};

wire [511:0] kprivrx_p_prod =
{
   512'd0
};

wire [127:0] lc128_prod =
{
   128'd0
};
```

4. Repeat Step 3 for all other key memory files. When you have finished including your production keys in all the key memory files, ensure that the USE_FACSIMILE parameter is set to 0 at the design example top level file (`a10_hdmi2_demo.v`)

4.2.4.3. Build and Compile the Design

After you include your own production keys, you can now build the design.

You can use the provided Tcl script to build and compile the design.

1. Install Intel Quartus Prime Pro Edition version 19.3.
2. Open a Nios II Command Shell.
3. Change the directory to your working directory, specifically the `hdcp2x/hw_demo/arria10/script/` directory.
4. Run `./runall.tcl`.
   This script executes the following commands:

- Generate IP catalog files
- Generate the Platform Designer system
- Create an Intel Quartus Prime project
- Create a software work space and build the software
- Compile the Intel Quartus Prime project
- Perform a full compilation

4.2.4.4. View the Results

At the end of the demonstration, you will be able to view the results on the HDCP-enabled HDMI external sink.

To view the results of the demonstration, follow these steps:

1. Power up the Intel FPGA board.
2. Change the directory to `hdcp2x/hw_demo/arria10/quartus/` directory.
3. Type the following command on the Nios II Command Shell to download the Software Object File (`.sof`) to the FPGA.

   `nios2-configure-sof output_files/`<Quartus project name>.sof
4. Power up the HDCP-enabled HDMI external source and sink (if you haven't done so). The HDMI external sink displays the output of your HDMI external source.

4.2.4.4.1. LED Functions

The LEDs on the board indicates the demonstration status.

Table 12. LED Indicators

LED	Functions
user_led[0]	RX HDMI PLL lock status. • 0: Unlocked • 1: Locked
user_led[1]	RX HDMI core lock status • 0: At least 1 channel unlocked • 1: All 3 channels locked
user_led[2]	RX HDCP1x IP decryption status. • 0: Inactive • 1: Active
user_led[3]	RX HDCP2x IP decryption status. • 0: Inactive • 1: Active
user_led[4]	TX HDMI PLL lock status. • 0: Unlocked • 1: Locked
user_led[5]	TX transceiver PLL lock status. • 0: Unlocked • 1: Locked
user_led[6]	TX HDCP1x IP encryption status.

continued...

LED	Functions
	• 0: Inactive • 1: Active
user_led[7]	TX HDCP2x IP encryption status. • 0: Inactive • 1: Active

4.2.5. Security Considerations

When using the HDCP feature, be mindful of the following security considerations.

- When designing a repeater system, you must block the received video from entering the TX IP in the following conditions:
  - If the received video is HDCP-encrypted (i.e. encryption status `hdcp1_enabled` or `hdcp2_enabled` from the RX IP is asserted) and the transmitted video is not HDCP-encrypted (i.e. encryption status `hdcp1_enabled` or `hdcp2_enabled` from the TX IP is not asserted).
  - If the received video is HDCP TYPE 1 (i.e. `streamid_type` from the RX IP is asserted) and the transmitted video is HDCP 1.4 encrypted (i.e. encryption status `hdcp1_enabled` from the TX IP is asserted)
- You should maintain the confidentiality and integrity of your HDCP production keys, and any user encryption keys.
- Intel strongly advices you to develop any Intel Quartus Prime projects and design source files that contain encryption keys in a secure compute environment to protect the keys.
- Intel strongly advices you to design security features in FPGAs to protect the design, including any embedded encryption keys, from unauthorized copying, reverse engineering, and tampering.

Related Information

AN 556: Using the Design Security Features in Intel FPGAs

4.3. HDMI Hardware Design Examples for Arria V and Stratix V Devices

The HDMI hardware design example helps you evaluate the functionality of the HDMI Intel FPGA IP core and provides a starting point for you to create your own design for Arria V and Stratix V devices.

The design example runs on the following device kits:

- Arria V GX starter kit
- Stratix V GX development kit
- Bitec HDMI HSMC 2.0 Daughter Card Revision 8

Related Information

AN 837: Design Guidelines for Intel FPGA HDMI

4.3.1. HDMI Hardware Design Components

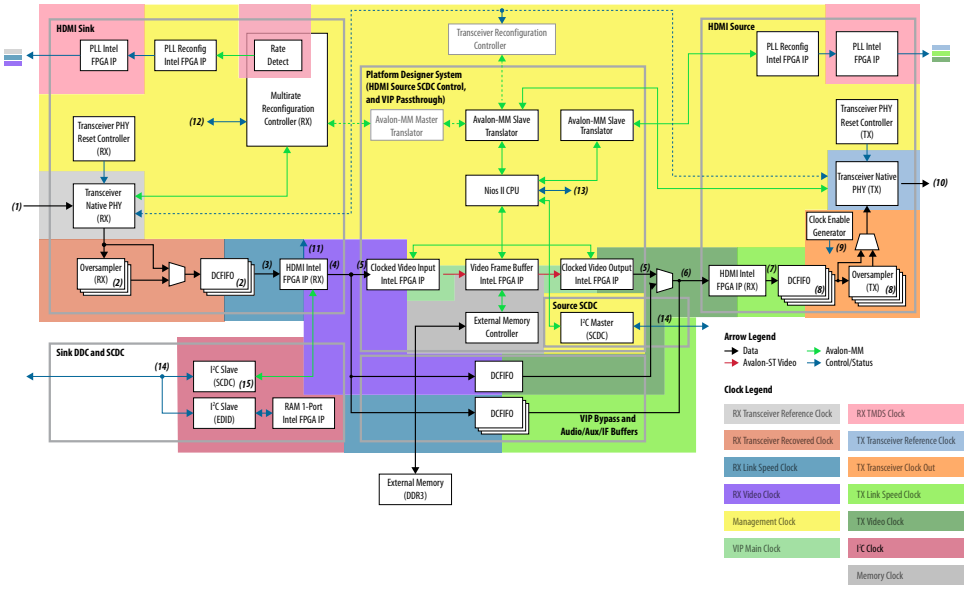
The demonstration designs instantiate the Video and Image Processing (VIP) Suite IP cores or FIFO buffers to perform a direct HDMI video stream passthrough between the HDMI sink and source.

The hardware demonstration design comprises the following components:

- HDMI sink
  - Transceiver Native PHY (RX)
  - Transceiver PHY Reset Controller (RX)
  - PLL
  - PLL Reconfiguration
  - Multirate Reconfiguration Controller (RX)
  - Oversampler (RX)
  - DCFIFO
- Sink Display Data Channel (DDC) and Status and Control Data Channel (SCDC)
- Transceiver Reconfiguration Controller
- VIP bypass and Audio, Auxiliary and InfoFrame buffers
- Platform Designer system
  - VIP passthrough for HDMI video stream
  - Source SCDC controller
  - HDMI source reconfiguration controller
- HDMI source
  - Transceiver Native PHY (TX)
  - Transceiver fPLL
  - Transceiver PHY Reset Controller (TX)
  - PLL
  - PLL Reconfiguration
  - Oversampler (TX)
  - DCFIFO
  - Clock Enable Generator

Figure 13. HDMI Hardware Design Example Block Diagram

The figure below shows a high level architecture of the design.



The following details of the design example architecture correspond to the numbers in the block diagram.

1. The sink TMDS data has three channels: data channel 0 (blue), data channel 1 (green), and data channel 2 (red).
2. The Oversampler (RX) and dual-clock FIFO (DCFIFO) instances are duplicated for each TMDS data channel (0,1,2).
3. The video data input width for each color channel of the HDMI RX core is equivalent to RX transceiver PCS-PLD parallel data width per channel.
4. Each color channel is fixed at 16 bpc. The video data output width of the HDMI RX core is equivalent to the value of symbols per clock*16*3.
5. The video data input width of the Clocked Video Input (CVI) and Clocked Video Output (CVO) IP cores are equivalent to the value of NUMBER_OF_PIXELS_IN_PARALLEL * BITS_PER_PIXEL_PER_COLOR_PLANE * NUMBER_OF_COLOR_PLANES. To interface with the HDMI core, the values of NUMBER_OF_PIXELS_IN_PARALLEL, BITS_PER_PIXEL_PER_COLOR_PLANE, and NUMBER_OF_COLOR_PLANES must match the symbols per clock, 16 and 3 respectively.
6. The video data input width of the HDMI TX core is equivalent to the value of symbols per clock*16*3. You can use the user switch to select the video data from the CVO IP core (VIP passthrough) or DCFIFO (VIP bypass).
7. The video data output width for each color channel of the HDMI TX core is equivalent to TX transceiver PCS-PLD parallel data width per channel.
8. The DCFIFO and the Oversampler (TX) instances are duplicated for each TMDS data channel (0,1,2) and clock channel.
9. The Oversampler (TX) uses the clock enable signal to read data from the DCFIFO.
10. The source TMDS data has four channels: data channel 0 (blue), data channel 1 (green), data channel 2 (red), and clock channel.

11. The RX Multirate Reconfiguration Controller requires the status of `TMDS_Bit_clock_Ratio` port to perform appropriate RX reconfiguration between the TMDS character rates below 340 Mcsc (HDMI 1.4b) and above 340 Mcsc (HDMI 2.0b). The status of the port is also required by the Nios II processor and the HDMI TX core to perform appropriate TX reconfiguration and scrambling.
12. The reset control and lock status signals from HDMI PLL, RX Transceiver Reset Controller and HDMI RX core.
13. The reset and oversampling control signals for HDMI PLL, TX Transceiver Reset Controller, and HDMI TX core. The lock status and rate detection measure valid signals from the HDMI sink initiate the TX reconfiguration process.
14. The I^2C SCL and SDA lines with tristate buffer for bidirectional configuration. Use the ALTIOBUF IP core for Arria V and Stratix V devices.
15. The SCDC is mainly designed for the source to update the `TMDS_Bit_Clock_Ratio` and `Scrambler_Enable` bits of the sink TMDS Configuration register. .

4.3.1.1. Transceiver Native PHY (RX)

- Transceiver Native PHY in Arria V devices
  - To operate the TMDS bit rate up to 3,400 Mbps, configure the Transceiver Native PHY at 20 bits at PCS – PLD interface with the HDMI RX core at 2 symbols per clock. When the PCS – PLD interface width is 20 bits, the minimum link rate is 611 Mbps.
  - To operate the TMDS bit rate up to 6,000 Mbps, configure the Transceiver Native PHY at 40 bits with the HDMI RX core at 4 symbols per clock. When the PCS – PLD interface width is 40 bits, the minimum link rate is 1,000 Mbps.
  - Oversampling is required for TMDS bit rate which is below the minimum link rate.
- Transceiver Native PHY in Stratix V devices
  - To operate the TMDS bit rate up to 6,000 Mbps, configure the Transceiver Native PHY at 20 bits at PCS – PLD interface with the HDMI RX core at 2 symbols per clock. When the PCS – PLD interface width is 20 bits, the minimum link rate is 611 Mbps.

Table 13. Arria V and Stratix V Transceiver Native PHY (RX) Configuration Settings (6,000 Mbps)

This table shows an example of Arria V and Stratix V Transceiver Native PHY (RX) configuration settings for TMDS bit rate of 6,000 Mbps.

Parameters	Settings
Datapath Options	
Enable TX datapath	Off
Enable RX datapath	On
Enable Standard PCS	On
Initial PCS datapath selection	Standard
Number of data channels	3
Enable simplified data interface	On

RX PMA	
Data rate	6,000 Mbps
Enable CDR dynamic reconfiguration	On
Number of CDR reference clocks	2 (2)
Selected CDR reference clock	0 (2)
Selected CDR reference clock frequency	600 MHz
PPM detector threshold	1,000 PPM
Enable rx_pma_clkout port	On
Enable rx_is_lockedtodata port	On
Enable rx_is_lockedtoref port	On
Enable rx_set_locktodata and rx_set_locktoref ports	On

Standard PCS	
Standard PCS protocol	Basic
Standard PCS/PMA interface width	• 10 (for 1 symbol per clock) • 20 (for 2 and 4 symbols per clock)
Enable RX byte deserializer	• Off (for 1 and 2 symbols per clock) • On (for 4 symbols per clock)

(2) The Bitec HDMI HSMC 2.0 daughter card routes the TMDS clock pin to the transceiver serial data pin. To use the TMDS clock to drive the HDMI PLL, the TMDS clock must also drive the transceiver dedicated reference clock pin. The number of CDR reference clocks is 2 with reference clock 1 (unused) driven by the TMDS clock and reference clock 0 driven by the HDMI PLL output clock. The selected CDR reference clock will be fixed at 0.

Table 14. Arria V and Stratix V Transceiver Native PHY (RX) Common Interface Ports

This table describes the Arria V and Stratix V Transceiver Native PHY (RX) common interface ports.

Signals	Direction	Description
Clocks		
`rx_cdr_refclk[1:0]`	Input	Input reference clock for the RX CDR circuitry. • To support arbitrary wide data rate range from 250 Mbps to 6,000 Mbps, you need a generic core PLL to obtain a higher clock frequency from the TMDS clock. You need a higher clock frequency to create oversampled stream for data rates below the minimum transceiver data rate—for example, 611 Mbps or 1,000 Mbps). • If the TMDS clock pin is routed to the transceiver dedicated reference clock pin, you only need to create one transceiver reference clock input. You can use the TMDS clock as reference clock for a generic core PLL to drive the transceiver. • If you use Bitec HDMI HSMC 2.0 daughter card, the TMDS clock pin is routed to the transceiver serial data pin. In this case, to use the TMDS clock as a reference clock for a generic core PLL, the clock must also drive the transceiver dedicated reference clock. Connect bit 0 to the generic core PLL output and bit 1 to the TMDS clock and set the selected CDR reference clock at 0.
`rx_std_clkout[2:0]`	Output	RX parallel clock output. • The CDR circuitry recovers the RX parallel clock from the RX data stream when the CDR is configured at lock-to-data mode. • The RX parallel clock is a mirror of the CDR reference clock when the CDR is configured at lock-to-reference mode.
`rx_std_coreclkin[2:0]`	Input	RX parallel clock that drives the read side of the RX phase compensation FIFO. Connect to `rx_std_clkout` ports.
`rx_pma_clkout[2:0]`	Output	RX parallel clock (recovered clock) output from PMA. Leave unconnected.
Resets		
`rx_analogreset[2:0]`	Input	Active-high, edge-sensitive, asynchronous reset signal. When asserted, resets the RX CDR circuit, deserializer. Connect to Transceiver PHY Reset Controller IP core.
`rx_digitalreset[2:0]`	Input	Active-high, edge-sensitive, asynchronous reset signal. When asserted, resets the digital component of the RX data path. Connect to the Transceiver PHY Reset Controller IP core.
PMA Ports		
`rx_set_locktoref[2:0]`	Input	When asserted, programs the RX CDR to lock to reference mode manually. The lock to reference mode enables you to control the reset sequence using `rx_set_locktoref` and `rx_set_locktodata`. The Multirate Reconfiguration Controller (RX) sets this port to 1 if oversampling mode is required. Otherwise, this port is set to 0.

continued...

PMA Ports		
		Refer "Transceiver Reset Sequence" in Transceiver Reset Control in Arria V/Stratix V Devices for more information about manual control of the reset sequence.
rx_set_locktodata[2:0]	Input	Always driven to 0. When rx_set_locktoref is driven to 1, the CDR is configured to lock-to-reference mode. Otherwise, the CDR is configured to lock-to-data mode.
rx_is_lockedtoref[2:0]	Output	When asserted, the CDR is locked to the incoming reference clock. Connect this port to rx_is_lockedtodata port of the Transceiver PHY Reset Controller IP core when rx_set_locktoref is 1.
rx_is_lockedtodata[2:0]	Output	When asserted, the CDR is locked to the incoming data. Connect this port to rx_is_lockedtodata port of Transceiver PHY Reset Controller IP core when rx_set_locktoref is 0.
rx_serial_data[2:0]	Input	RX differential serial input data.

PCS Ports		
unused_rx_parallel_data	Output	Leave unconnected.
rx_parallel_data[S*3*10-1:0]	Output	PCS RX parallel data. *Note:* S=Symbols per clock.

Calibration Status Port		
rx_cal_busy[2:0]	Output	When asserted, indicates that the initial RX calibration is in progress. This port is also asserted if the reconfiguration controller is reset. Connect to the Transceiver PHY Reset Controller IP core.

Reconfiguration Ports		
reconfig_to_xcvr[209:0]	Input	Reconfiguration signals from the Transceiver Reconfiguration Controller.
reconfig_from_xcvr[137:0]	Output	Reconfiguration signals to the Transceiver Reconfiguration Controller.

4.3.1.2. PLL Intel FPGA IP Cores

Use the PLL Intel FPGA IP core as the HDMI PLL to generate reference clock for RX or TX transceiver, link speed, and video clocks for the HDMI RX or TX IP core.

The HDMI PLL is referenced by the arbitrary TMDS clock. For HDMI source, you can reference the HDMI PLL by a separate clock source in the VIP passthrough design, which contains frame buffer. The HDMI PLL for TX has the same desired output frequencies as RX across symbols per clock and color depth.

- For TMDS bit rates ranging from 3,400 Mbps to 6,000 Mbps (HDMI 2.0), the TMDS clock rate is 1/40 of the TMDS bit rate. The HDMI PLL generates reference clock for RX/TX transceiver at 4 times the TMDS clock.
- For TMDS bit rates below 3,400 Mbps (HDMI 1.4b), the TMDS clock rate is 1/10 of the TMDS bit rate. The HDMI PLL generates reference clock for RX/TX transceiver at identical rate as the TMDS clock.

If the TMDS link operates at TMDS bit rates below the minimum RX/TX transceiver link rate, your design requires oversampling and a factor of 5 is chosen. The minimum link rate of the RX/TX transceiver vary across device families and symbols per clock. The HDMI PLL generates reference clock for RX/TX transceiver at 5 times the TMDS clock.

Note: Place the PLL Intel FPGA block on the transmit path (`pll_hdmi_tx`) in the physical location next to the transceiver PLL.

Table 15. HDMI PLL Desired Output Frequencies for 8-bpc Video

This table shows an example of HDMI PLL desired output frequencies across various TMDS clock rates and symbols per clock for all supported device families using 8-bpc video.

Device Family	Symbols Per Clock	Minimum Link Rate (Mbps)	TMDS Bit Rate (Mbps)	Oversampling (5x) Required	TMDS Clock Rate (MHz)	RX/TX Transceiver Refclk (MHz)	RX/TX Link Speed Clock (MHz)	RX/TX Video Clock (MHz)
Arria V	2	611	270	Yes	27	135	13.5	13.5
			742.5	No	74.25	74.25	37.125	37.125
			1,485	No	148.5	148.5	74.25	74.25
			2,970	No	297	297	148.5	148.5
	4	1,000	270	Yes	27	135	6.75	6.75
			742.5	Yes	74.25	371.25	18.5625	18.5625
			1,485	No	148.5	148.5	37.125	37.125
			5,940	No	148.5	594	148.5	148.5
Stratix V	2	611	540	Yes	54	270	27	27
			1,620	No	162	162	81	81
			5,934	No	296.7	593.4	296.7	296.7

The color depths greater than 8 bpc or 24 bpp are defined to be deep color. For a color depth of 8 bpc, the core carries the pixels at a rate of one pixel per TMDS clock. At deeper color depths, the TMDS clock runs faster than the source pixel clock to provide the extra bandwidth for the additional bits.

The TMDS clock rate is increased by the ratio of the pixel size to 8 bits:

- 8 bits mode—TMDS clock = 1.0 × pixel or video clock (1:1)
- 10 bits mode—TMDS clock = 1.25 × pixel or video clock (5:4)
- 12 bits mode—TMDS clock = 1.5 × pixel or video clock (3:2)
- 16 bits mode—TMDS clock = 2 × pixel or video clock (2:1)

Table 16. HDMI PLL Desired Output Frequencies for Deep Color Video

This table shows an example of HDMI PLL desired output frequencies across symbols per clock and color depths.

Symbols Per Clock	Oversampling (5x) Required	Bits Per Component	TMDS Bit Rate (Mbps)	TMDS Clock Rate (MHz)	RX/TX Transceiver Refclk (MHz)	RX/TX Link Speed Clock (MHz)	RX/TX Video Clock (MHz)
2	Yes	8	270	27	135	13.5	13.5
		10 (3)	337.5	33.75	168.75	16.875	13.5

continued...

Symbols Per Clock	Oversampling (5x) Required	Bits Per Component	TMDS Bit Rate (Mbps)	TMDS Clock Rate (MHz)	RX/TX Transceiver Refclk (MHz)	RX/TX Link Speed Clock (MHz)	RX/TX Video Clock (MHz)
		12 (3)	405	40.5	202.5	20.25	13.5
		16 (3)	540	54	270	27	13.5
4	No	8	1,485	148.5	148.5	37.125	37.125
		10 (3)	1,856.25	185.625	185.625	46.40625	37.125
		12 (3)	2,227.5	222.75	222.75	55.6875	37.125
		16 (3)	2,970	297	297	74.25	37.125

The default frequency setting of the HDMI PLL is fixed at possible maximum value for each clock for appropriate timing analysis.

Note: This default combination is not valid for any HDMI resolution. The core will reconfigure to the appropriate settings upon power up.

4.3.1.3. PLL Reconfig Intel FPGA IP Core

The PLL Reconfig Intel FPGA IP core facilitates dynamic real-time reconfiguration of PLLs in Intel FPGAs.

Use the IP core to update the output clock frequency, PLL bandwidth in real-time, without reconfiguring the entire FPGA.

You can run this IP core at 100 MHz in Stratix V devices. In Arria V devices, you need to run at 75 MHz for timing closure. To simplify clocking in Arria V devices, the entire management clock domain is capped at 75 MHz.

4.3.1.4. Multirate Reconfig Controller (RX)

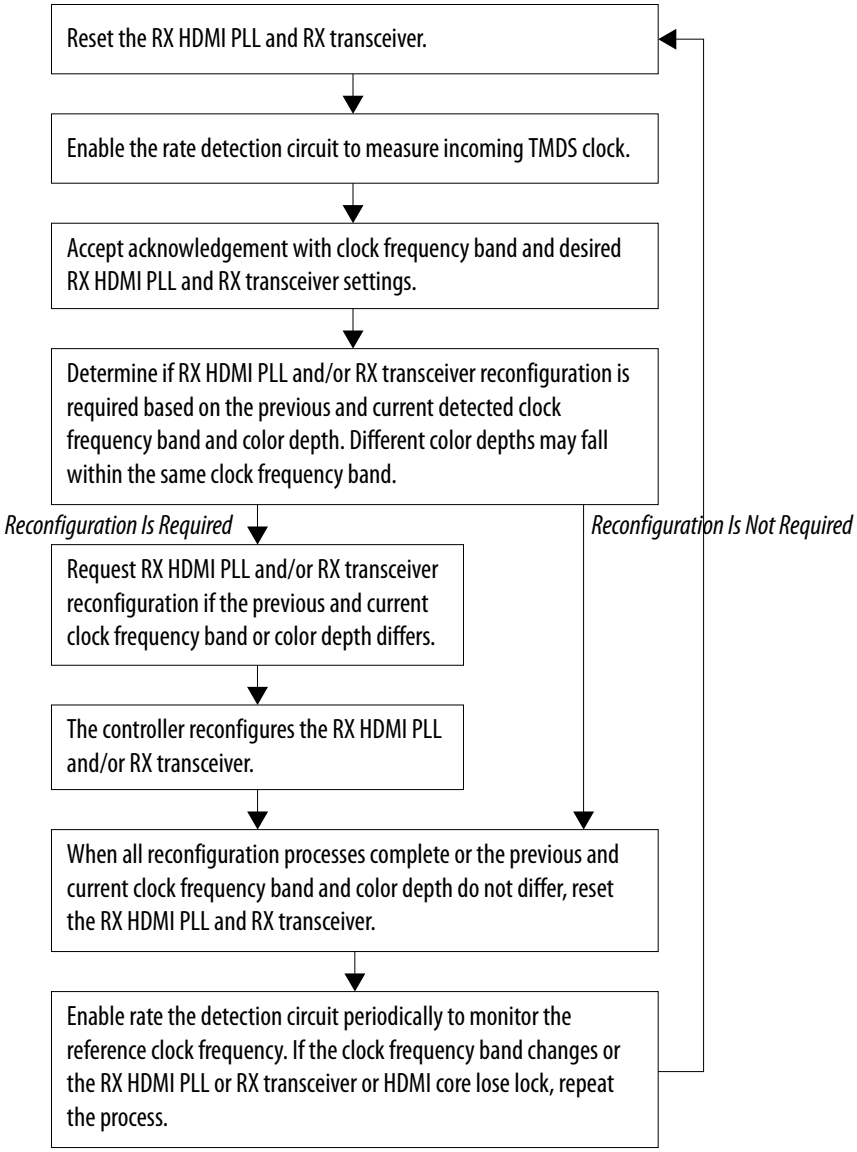
The Multirate Reconfig Controller implements rate detection circuitry with the HDMI PLL to drive the RX transceiver to operate at any arbitrary link rates ranging from 250 Mbps to 6,000 Mbps. Link rate of 6,000 Mbps is not the absolute maximum but the intention is to support HDMI 2.0b link rate.

The Multirate Reconfig Controller performs rate detection on the HDMI PLL arbitrary reference clock, which is also the TMDS clock, to determine the clock frequency band. Based on the detected clock frequency band, the circuitry dynamically reconfigures the HDMI PLL and transceiver settings to accommodate for the link rate change.

(3) For this release, deep color video is only demonstrated in VIP bypass mode. It is not available in VIP passthrough mode.

Figure 14. Multirate Reconfiguration Sequence Flow

This figure illustrates the multirate reconfiguration sequence flow of the controller when it receives input data stream and reference clock frequency, or when the transceiver is unlocked.



4.3.1.5. Oversampler (RX)

The Oversampler (RX) extracts data from the oversampled incoming data stream when the detected clock frequency band is below the transceiver minimum link rate.

The oversampling factor is fixed at 5 and you can program the data width to support different number of symbols. The supported data width is 20 bit for 2 symbols per clock and 40 bits for 4 symbols per clock. The extracted bit will be accompanied by data valid pulse which asserts every 5 clock cycles.

4.3.1.6. DCFIFO

The DCFIFO transfers data from the RX transceiver recovered clock domain to the RX link speed clock domain. The DCFIFO transfers data from the TX link speed clock domain to the TX transceiver parallel clock out domain.

- Sink
  - When the Multirate Reconfig Controller (RX) detects an incoming input stream that is below the transceiver minimum link rate, the DCFIFO accepts the data from the Oversampler with data valid pulse as write request asserted every 5 clock cycles.
  - Otherwise, it accepts data directly from the transceiver with write request asserted at all times.
- Source
  - When Nios II processor determines the outgoing data stream is below the TX transceiver minimum link rate, the TX transceiver accepts the data from the Oversampler (TX).
  - Otherwise, the TX transceiver reads data directly from the DCFIFO with read request asserted at all times.

4.3.1.7. Sink Display Data Channel (DDC) & Status and Control Data Channel (SCDC)

The HDMI source uses the DDC to determine the capabilities and characteristics of the sink by reading the Enhanced Extended Display Identification Data (E-EDID) data structure.

The E-EDID memory is stored using the RAM 1-Port IP core. A standard two-wire (clock and data) serial data bus protocol (I2C slave-only controller) is used to transfer CEA-861-D compliant E-EDID data structure.

The 8-bit I^2C slave addresses for the E-EDID are 0xA0/0xA1. The LSB indicates the access type: 1 for read and 0 for write. When an HPD event occurs, the I^2C slave responds to E-EDID data by reading from the RAM.

The I^2C slave-only controller is also used to support SCDC for HDMI 2.0b operation. The 8-bit I^2C slave addresses for the SCDC are 0xA8/0xA9. When an HPD event occurs, the I^2C slave performs write/read transaction to/from SCDC interface of HDMI RX core. This I^2C slave-only controller for SCDC is not required if HDMI 2.0b is not intended.

4.3.1.8. Transceiver Reconfiguration Controller

You can use the Transceiver Reconfiguration Controller IP core to change the device transceiver settings at any time.

You can selectively reconfigure any portion of the transceiver. The reconfiguration of each portion requires a read-modify-write operation (read first, then write). The read-modify-write operation modifies only the appropriate bits in a register and does not affect the other bits.

The Transceiver Reconfiguration Controller is only available and required in Arria V and Stratix V devices. Because the RX and TX transceivers share a single controller, the controller requires Platform Designer interconnects, such as Avalon-MM Master Translator and Avalon-MM Slave Translator, in the Platform Designer system.

- The Avalon-MM Master Translator provides an interface between this controller and the RX Multirate Reconfig Controller.
- The Avalon-MM Slave Translator arbitrates the RX and TX reconfiguration event for this controller.

4.3.1.9. VIP Bypass and Audio, Auxiliary and InfoFrame Buffers

The video data output and synchronization signals from HDMI RX core is looped through a DCFIFO across RX and TX video clock domains. The General Control Packet (GCP), InfoFrames (AVI, VSI, and AI), auxiliary data and audio data are looped through DCFIFOs across RX and TX link speed clock domains.

The auxiliary data port of the HDMI TX core controls the auxiliary data that flow through DCFIFO through backpressure. The backpressure ensures there is no incomplete auxiliary packet on the auxiliary data port. This block also performs external filtering on the audio data and audio clock regeneration packet from the auxiliary data stream before sending to the HDMI TX core auxiliary data port.

4.3.1.10. Transceiver Native PHY (TX)

The Arria V and Stratix V Transceiver Native PHY (TX) configuration settings are typically the same as RX.

Table 17. Arria V and Stratix V Transceiver Native PHY (TX) Configuration Settings (6,000 Mbps)

This table shows an example of Arria V and Stratix V Transceiver Native PHY (TX) configuration settings for TMDS bit rate of 6,000 Mbps.

Parameters	Settings
Datapath Options	
Enable TX datapath	On
Enable RX datapath	Off
Enable Standard PCS	On
Initial PCS datapath selection	Standard
Number of data channels	4
Bonding mode	xN
Enable simplified data interface	On

TX PMA	
Data rate	6,000 Mbps
TX local clock division factor	1
Enable TX PLL dynamic reconfiguration	On
Use external TX PLL	Off
Number of TX PLLs	1
Main TX PLL logical index	0
Number of TX PLL reference clocks	1
PLL type	CMU

continued...

TX PMA	
Reference clock frequency	600 MHz
Selected reference clock source	0
Selected clock network	xN

Standard PCS	
Standard PCS protocol	Basic
Standard PCS/PMA interface width	• 10 (for 1 symbol per clock) • 20 (for 2 and 4 symbols per clock)
Enable TX byte serializer	• Off (for 1 and 2 symbols per clock) • On (for 4 symbols per clock)

Table 18. Arria V and Stratix V Transceiver Native PHY (TX) Common Interface Ports

This table describes the Arria V and Stratix V Transceiver Native PHY (TX) common interface ports.

Signals	Direction	Description
Clocks		
`tx_pll_refclk`	Input	The reference clock input to the TX PLL.
`tx_std_clkout[3:0]`	Output	TX parallel clock output.
`tx_std_coreclkin[3:0]`	Input	TX parallel clock that drives the write side of the TX phase compensation FIFO. Connect to `tx_std_clkout[0]` ports.
Resets		
`tx_analogreset[3:0]`	Input	When asserted, resets all the blocks in TX PMA. Connect to Transceiver PHY Reset Controller (TX) IP core.
`tx_digitalreset[3:0]`	Input	When asserted, resets all the blocks in TX PCS. Connect to the Transceiver PHY Reset Controller (TX) IP core.
TX PLL		
`pll_powerdown`	Input	When asserted, resets the TX PLL. Connect to the Transceiver PHY Reset Controller (TX) IP core.
`pll_locked`	Output	When asserted, indicates that the TX PLL is locked. Connect to the Transceiver PHY Reset Controller (TX) IP core.
PCS Ports		
`unused_tx_parallel_data`	Input	Leave unconnected.
`tx_parallel_data[S*4*10-1:0]`	Input	PCS TX parallel data. *Note:* S=Symbols per clock.
PMA Port		
`tx_serial_data[3:0]`	Output	TX differential serial output data.

Calibration Status Port		
`tx_cal_busy[3:0]`	Output	When asserted, indicates that the initial TX calibration is in progress. This port is also asserted if the reconfiguration controller is reset. Connect to the Transceiver PHY Reset Controller (TX) IP core.

Reconfiguration Ports		
`reconfig_to_xcvr[349:0]`	Input	Reconfiguration signals from the Transceiver Reconfiguration Controller.
`reconfig_from_xcvr[229:0]`	Output	Reconfiguration signals to the Transceiver Reconfiguration Controller.

4.3.1.11. Transceiver PHY Reset Controller

The Transceiver PHY Reset Controller IP core ensures a reliable initialization of the RX and TX transceivers.

The reset controller has separate reset controls per channel to handle synchronization of reset inputs, lagging of PLL locked status, and automatic or manual reset recovery mode.

4.3.1.12. Oversampler (TX)

The Oversampler (TX) transmits data by repeating each bit of the input word a given number of times and constructs the output words.

The oversampling factor is fixed at 5. The Oversampler (TX) assumes that the input word is only valid every 5 clock cycles. This block enables when the outgoing data stream is determined to be below the TX transceiver minimum link rate by reading once from the DCFIFO every 5 clock cycles.

4.3.1.13. Clock Enable Generator

The Clock Enable Generator is a logic that generates a clock enable pulse.

This clock enable pulse asserts every 5 clock cycles and serves as a read request signal to clock the data out from DCFIFO.

4.3.1.14. Platform Designer System

The Platform Designer system consists of the VIP passthrough for HDMI video stream, source SDC controller, and source reconfiguration controller blocks.

4.3.1.14.1. VIP Passthrough for HDMI Video Stream

For certain example designs, you can loop the video data output and synchronization signals from HDMI RX core through the VIP data path.

The Clocked Video Input II (CVI II) Intel FPGA IP core converts clocked video formats to Avalon-ST video by stripping incoming clocked video of horizontal and vertical blanking, leaving only active picture data.

- The IP core provides clock crossing capabilities to allow video formats running at different frequencies to enter the system.
- The IP core also detects the format of the incoming clocked video and provides this information in a set of registers.
- The Nios II processor uses this information to reconfigure the video frame mode registers of the CVO IP core in the VIP passthrough design.

The Video Frame Buffer II Intel FPGA IP core buffers video frames into external RAM.

- The IP core supports double and triple buffering with a range of options for frame dropping and repeating.
- You can use the buffering options to solve throughput issues in the data path and perform simple frame rate conversion.

In a VIP passthrough design, you can reference the HDMI source PLL and sink PLL using separate clock sources. However, in a VIP bypass design, you must reference the HDMI source PLL and sink PLL using the same clock source.

The Clocked Video Output II (CVO II) Intel FPGA IP core converts data from the flow-controlled Avalon-ST video protocol to clocked video.

- The IP core provides clock crossing capabilities to allow video formats running at different frequencies to be created from the system.
- It formats the Avalon-ST video into clocked video by inserting horizontal and vertical blanking and generating horizontal and vertical synchronization information using the Avalon-ST video control and active picture packets.
- The video frame is described using the mode registers that are accessed through the Avalon-MM control port.

Table 19. Difference between VIP Passthrough Design and VIP Bypass Design

VIP Passthrough Design	VIP Bypass Design
• Can reference the HDMI source PLL and sink PLL using separate clock sources • Demonstrates only certain video formats—640×480p60, 720×480p60, 1280×720p60, 1920×1080p60, and 3840×2160p24	• Must reference the HDMI source PLL and sink PLL using the same clock source • Demonstrates all video formats.

Table 20. VIP Passthrough and VIP Bypass Options for the Supported Devices

Device Family	Symbols Per Clock	HDMI Specification Support	Bitec HDMI HSMC 2.0 Daughter Card	Directory	VIP Passthrough	VIP Bypass
Arria V	2	1.4b	HSMC (Rev8)	av_sk	Supported	Supported
	4	2.0b	HSMC (Rev8)	av_sk_hdmi2	Not supported	Supported
Stratix V	2	2.0b	HSMC (Rev8)	sv_hdmi2	Not supported	Supported

4.3.1.14.2. Source SCDC Controller

The source SCDC Controller contains the I^2C master controller. The I^2C master controller transfers the SCDC data structure from the FPGA source to the external sink for HDMI 2.0b operation.

For example, if the outgoing data stream is 6,000 Mbps, the Nios II processor commands the I^2C master controller to update the `TMDS_Bit_Clock_Ratio` and `Scrambler_Enable` bits of the sink TMDS configuration register to 1. The same I^2C master can also transfer the DDC data structure (E-EDID) between the HDMI source and external sink.

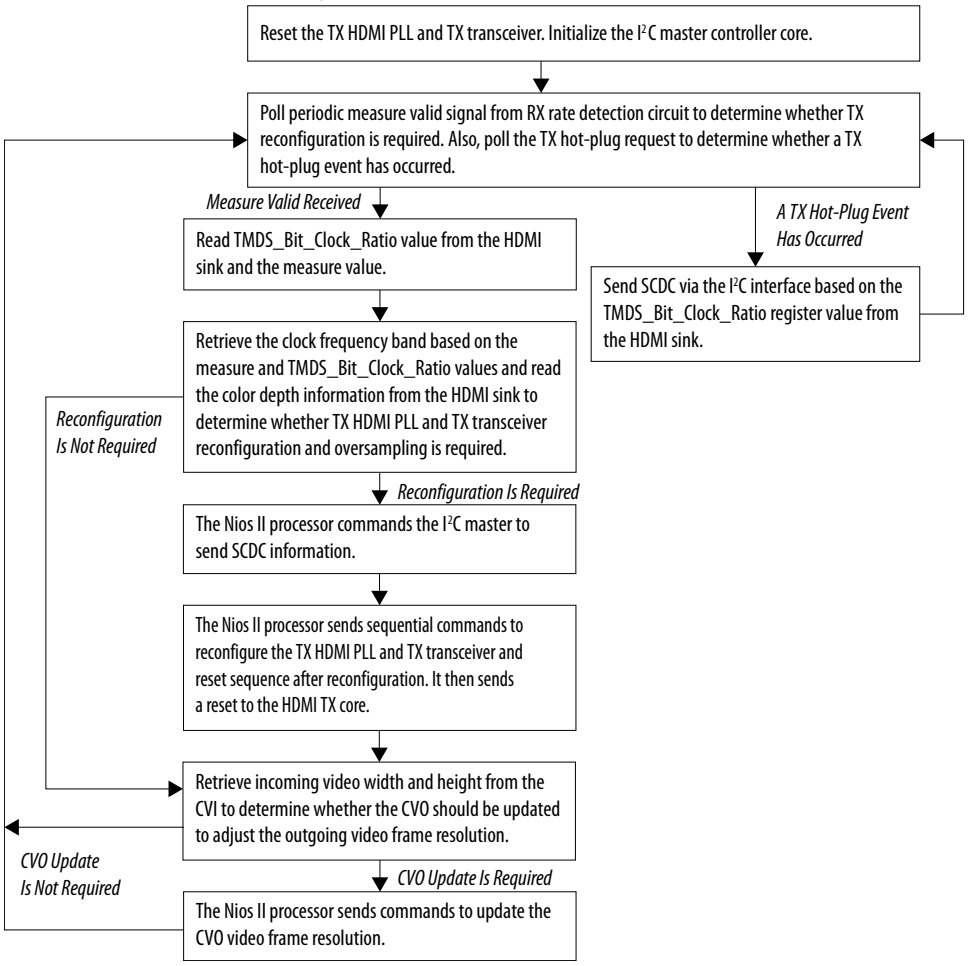
4.3.1.14.3. Source Reconfiguration Controller

The Nios II CPU acts as the multirate reconfiguration controller for the HDMI source.

The CPU relies on the periodic rate detection from the Multirate Reconfig Controller (RX) to determine if TX requires reconfiguration. The Avalon-MM slave translator provides the interface between the Nios II processor Avalon-MM master interface and the Avalon-MM slave interfaces of the externally instantiated HDMI source's PLL Reconfig Intel FPGA IP and Transceiver Native PHY (TX).

Figure 15. Nios II Software Flow

The reconfiguration sequence flow for TX is the same as RX, except that the PLL and transceiver reconfiguration, and the reset sequence is performed sequentially. The figure illustrates the Nios II software flow that involves the controls for CVO, I²C master and HDMI source.



4.3.2. HDMI Hardware Design Requirements

The HDMI design requires an Intel FPGA board and supporting hardware.

- Intel FPGA board
- Bitec HDMI HSMC 2.0 daughter card
- Standard HDMI source—for example, PC with a graphic card and HDMI output
- Standard HDMI sink—for example, monitor with HDMI input
- 2 HDMI cables
  - A cable to connect the graphics card to the Bitec daughter card RX connector.
  - A cable to connect the Bitec daughter card TX connector to the monitor.

Table 21. Intel FPGA Boards and Bitec HDMI HSMC 2.0 Daughter Cards Supported for the Design

Design Example	Intel FPGA Board	Bitec HDMI HSMC 2.0 Daughter Card
Arria V (av_sk)	Arria V GX FPGA Starter Kit	HSMC (Rev8)
Arria V (av_sk_hdmi2)	Arria V GX FPGA Starter Kit	HSMC (Rev8)
Stratix V (sv_hdmi2)	Stratix V GX FPGA Development Kit	HSMC (Rev8)

Related Information

- Arria V GX Starter Kit User Guide
- Stratix V GX FPGA Development Kit User Guide

4.3.3. Design Walkthrough

Setting up and running the HDMI hardware design consists of four stages.

You can use the Intel-provided scripts to automate these stages.

1. Set up the hardware.
2. Copy the design files to your working directory.
3. Build and compile the design.
4. View the results.

4.3.3.1. Set Up the Hardware

The first stage of the demonstration is to set up the hardware.

To set up the hardware for the demonstration:

1. Connect the Bitec HDMI HSMC 2.0 daughter card to the FPGA development board.
2. Connect the FPGA board to your PC using a USB cable.

   Note: The Arria V GX FPGA Starter Kit and Stratix V GX FPGA Development Kit have an On-Board Intel FPGA Download Cable II connector. If your version of the board does not have this connector, you can use an external Intel FPGA Download Cable cable.
3. Connect an HDMI cable from the HDMI RX connector on the Bitec HDMI HSMC 2.0 daughter card to a standard HDMI source, in this case a PC with a graphic card and HDMI output.
4. Connect another HDMI cable from the HDMI TX connector on the Bitec HDMI HSMC 2.0 daughter card to a standard HDMI sink, in this case a monitor with HDMI input.

4.3.3.2. Copy the Design Files

After you set up the hardware, you copy the design files. Copy the hardware demonstration files from one of the following paths to your working directory:

- Arria V
  - 2 symbols per clock (HDMI 1.4b) demonstration: `<IP root directory>/altera_hdmi/hw_demo/av_sk`
  - 4 symbols per clock (HDMI 2.0b) demonstration: `<IP root directory>/altera_hdmi/hw_demo/av_sk_hdmi2`
- Stratix V
  - 2 symbols per clock (HDMI 2.0b) demonstration: `<IP root directory>/altera_hdmi/hw_demo/sv_hdmi2`

4.3.3.3. Build and Compile the Design

After you copy the design files, you can build the design.

You can use the provided Tcl script to build and compile the FPGA design.

1. Open a Nios II Command Shell.
2. Change the directory to your working directory.
3. Type the command and enter `source runall.tcl`.
   This script executes the following commands:
   - Generate IP catalog files
   - Generate the Platform Designer system
   - Create an Intel Quartus Prime project
   - Create a software work space and build the software
   - Compile the Intel Quartus Prime project
   - Run Analysis & Synthesis to generate a post-map netlist for DDR assignments —for VIP passthrough design only
   - Perform a full compilation

   Note: If you are a Linux user, you will get a message `cygpath: command not found`. You can safely ignore this message; the script will proceed to generate the next commands.

4.3.3.4. View the Results

At the end of the demonstration, you will be able to view the results on the standard HDMI sink (monitor).

To view the results of the demonstration, follow these steps:

1. Power up the Intel FPGA board.
2. Type the following command on the Nios II Command Shell to download the Software Object File (`.sof`) to the FPGA.

   `nios2-configure-sof output_files/`<Quartus project name>.sof
3. Power up the standard HDMI source and sink (if you haven't done so).
   The design displays the output of your video source (PC).

Note: If the output does not appear, press `cpu_resetn` to reinitialize the system or perform HPD by unplugging the cable from the standard source and plug it back again.

4. Open the graphic card control utility (if you are using a PC as source). Using the control panel, you can switch between various video resolutions.

   The av_hdmi2 and sv_hdmi2 demonstration designs allow any video resolutions up to 4Kp60. The av_sk design allows *640×480p60*, *720×480p60*, *1280×720p60*, *1920×1080p60*, and *3840×2160p24* when you select the VIP passthrough mode (`user_dipsw[0] = 0`). If you select the VIP bypass mode (`user_dipsw[0] = 1`, the design allows any video resolutions up to 4Kp60.

4.3.3.4.1. Push Buttons, DIP Switches and LED Functions

Use the push buttons, DIP switches, and LED functions on the board to control your demonstration.

Table 22. Push Buttons, DIP Switches and LEDs Functions

Push Button/ DIP Switch/LED	Pins		Functions
	av_sk/av_sk_hdmi2	sv_hdmi2	
cpu_resetn	D5	AM34	Press once to perform system reset.
user_pb[0]	A14	A7	Press once to turn on and turn off HPD signal to the standard HDMI source.
user_pb[1]	B15	B7	Press and hold to instruct the TX to send DVI encoded signal and release to send HDMI encoded signal.
user_pb[2]	B14	C7	Press and hold to instruct the TX to stop sending InfoFrames and release to resume sending.
user_dipsw[0]	D15	Unused	Only used in av_sk design which demonstrates the VIP passthrough feature. • 0: VIP passthrough • 1: VIP bypass
user_led[0]	F17	J11	RX HDMI PLL lock status. • 0: Unlocked • 1: Locked
user_led[1]	G15	U10	RX transceiver ready status. • 0: Not ready • 1: Ready
user_led[2]	G16	U9	RX HDMI core lock status • 0: At least 1 channel unlocked • 1: All 3 channels locked
user_led[3]	G17	AU24	RX oversampling status. • 0: Non-oversampled (more than 611 Mbps for av_sk and sv_hdmi2, more than 1,000 Mbps for av_sk_hdmi2) • 1: Oversampled (less than 611 Mbps for av_sk and sv_hdmi2, less than 1,000 Mbps for av_sk_hdmi2)
user_led[4]	D16	AF28	TX HDMI PLL lock status.

continued...

Push Button/ DIP Switch/LED	Pins		Functions
	av_sk/av_sk_hdmi2	sv_hdmi2	
			• 0: Unlocked • 1: Locked
user_led[5]	C13	AE29	TX transceiver ready status. • 0: Not ready • 1: Ready
user_led[6]	C14	AR7	TX transceiver PLL lock status. • 0: Unlocked • 1: Locked
user_led[7]	C16	AV10	TX oversampling status. • 0: Non-oversampled (more than 611 Mbps for av_sk and sv_hdmi2, more than 1,000 Mbps for av_sk_hdmi2) • 1: Oversampled (less than 611 Mbps for av_sk and sv_hdmi2, less than 1,000 Mbps for av_sk_hdmi2)

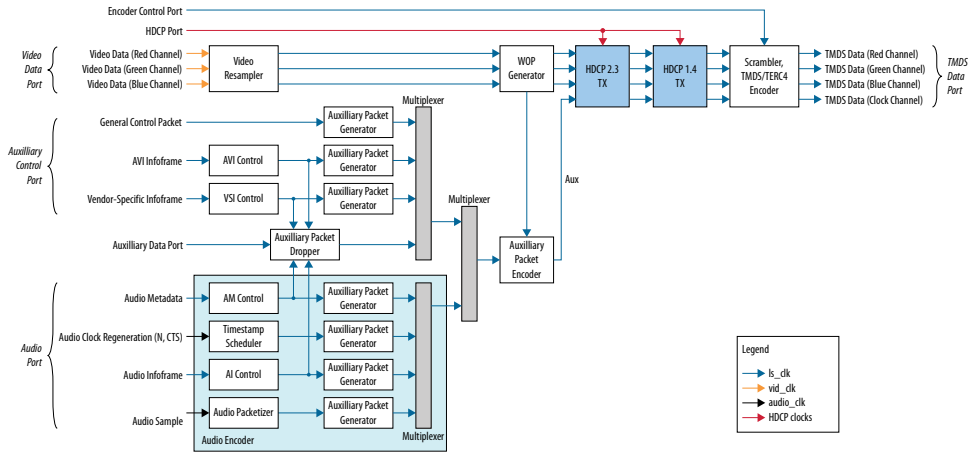
5. HDMI Source

5.1. Source Functional Description

The HDMI source core provides direct connection to the Transceiver Native PHY through a 10-bit, 20-bit or 40-bit parallel data path.

Figure 16. HDMI Source Signal Flow Diagram

The figure below shows the flow of the HDMI source signals. The figure shows the various clocking domains used within the core.



The source core provides four 10-bit, 20-bit or 40-bit parallel data paths corresponding to the 3 color channels and the clock channel.

The source core accepts video, audio, and auxiliary channel data streams. The core produces a scrambled and TMDS/TERC4 encoded data stream that would typically connect to the high-speed transceiver parallel data inputs.

Note: The scrambled data only applies for HDMI 2.0b stream with TMDS Bit Rate higher than 3.4 Gbps.

Central to the core is the Scrambler, TMDS/TERC4 Encoder. The encoder processes either video or auxiliary data.

5.1.1. Source Scrambler, TMDS/TERC4 Encoder

The TMDS/TERC4 encoder implements 8-bit to 10-bit and 4-bit to 10-bit algorithms as defined in the *HDMI 1.4b Specification Section 5.4*. Each data channel, with exception of the clock channel, has its own encoder. You can configure the core to enable scrambling, as defined in the *HDMI 1.4b Specification Section 6.1.2*, before TMDS/TERC4 encoding.

The encoder processes symbol data at 1, 2, or 4 symbols per clock. When the encoder operates in 2 or 4 symbols per clock, it also produces the output in the form of two or four encoded symbols per clock.

The TMDS/TERC4 encoder also produces digital visual interface (DVI) signaling when you deassert the `mode` input signal. DVI signaling is identical to HDMI signaling, except for the absence of data and video islands and TERC4 auxiliary data.

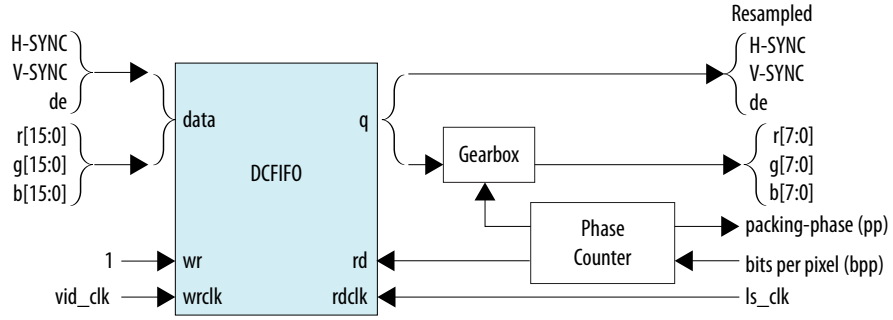
5.1.2. Source Video Resampler

The video resampler consists of a dual-clock FIFO (DCFIFO) and a gearbox.

The gearbox converts data of 8, 10, 12, or 16 bits per component to 8-bit per component data based on the current color depth. The General Control Packet (GCP) conveys the color depth information.

Figure 17. Source Video Resampler Signal Flow Diagram

The figure below shows the components of the video resampler and the signal flow between these components.



The resampler adheres to the recommended phase encoding method described in *HDMI 1.4b Specification Section 6.5*.

- The phase counter must register the last pixel packing-phase (pp) of the last pixel of the last active line.
- The core then transmits the pp value to the attached sink device in the GCP for packing synchronization.

The HDMI cable may send across four different pixel encodings: RGB 4:4:4, YCbCr 4:4:4, and YCbCr 4:2:2 (as described in *HDMI 1.4b Specification Section 6.5*), and YCbCr 4:2:0 (as described in *HDMI 2.0b Specification Section 7.1*).

Figure 18. Pixel Data Input Format RGB/YCbCr 4:4:4

The figure below shows the RGB/YCbCr 4:4:4 color space pixel bit-field mappings per symbol. When the actual color depth is below 16 bpc, the unused LSBs are set to zero.

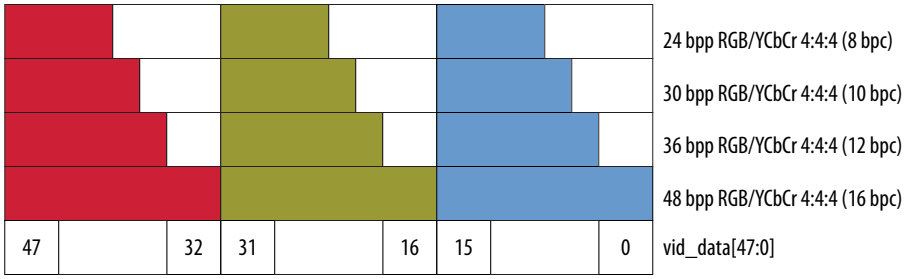
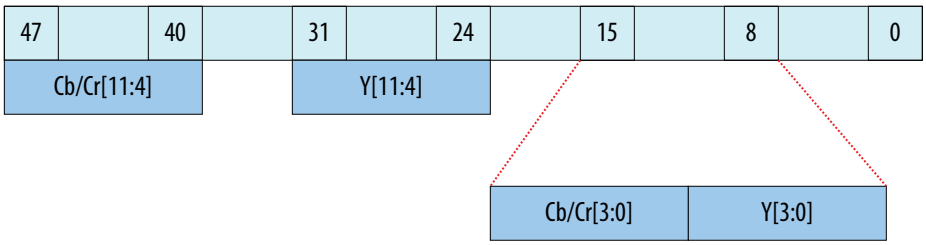


Figure 19. Pixel Data Input Format YCbCr 4:2:2 (12 bpc)

The figure below shows the YCbCr 4:2:2 color space pixel bit-field mappings per symbol. As with 4:4:4 color space, the unused LSBs are set to zero.

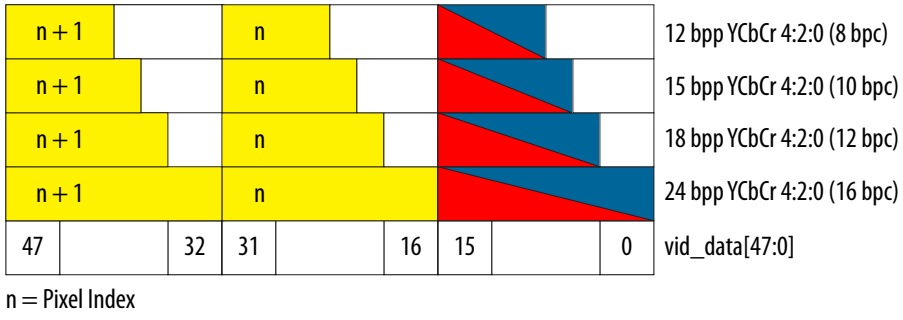


The higher order 8 bits of the Y samples are mapped to the 8 bits of Channel 1 and the lower order 4 bits are mapped to the lower order 4 bits of Channel 0.

The first pixel transmitted within a Video Data Period contains three components, Y0, Cb0 and Cr0. The Y0 and Cb0 components are transmitted during the first pixel period while Cr0 is transmitted during the second pixel period. This second pixel period also contains the only component for the second pixel, Y1. In this way, the link carries one Cb sample for every two pixels and one Cr sample for every two pixels. These two components (Cb and Cr) are multiplexed onto the same signal paths on the link.

Figure 20. Pixel Data Input Format YCbCr 4:2:0

The figure shows the YCbCr 4:2:0 color space pixel bit-field mappings. As with 4:4:4 color space, the unused LSBs are set to zero.



The two horizontally successive 8-bit Y components are transmitted in TMDS Channels 1 and 2, in that order. The 8-bit Cb or Cr components are transmitted alternately in TMDS Channel 0, line by line.

For even lines starting with line 0:

- `vid_data[47:32]` always transfer the Yn+1 component
- `vid_data[31:16]` always transfer the Yn component
- `vid_data[15:0]` always transfer the Cbn component

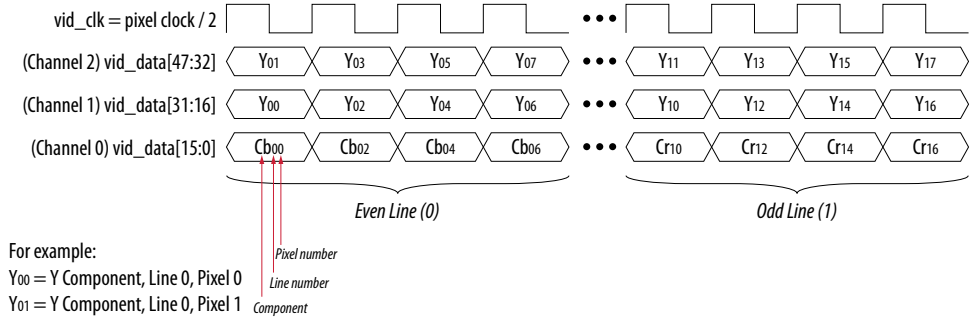
For odd lines:

- `vid_data[47:32]` always transfer the Yn+1 component
- `vid_data[31:16]` always transfer the Yn component
- `vid_data[15:0]` always transfer the Crn component

The frequency of `vid_clk` must be halved when YCbCr 4:2:0 is used, because two pixels are fed into a single clock cycle.

Figure 21. YCbCr 4:2:0 Transport Using 1 Symbol Per Clock Mode

The figure below shows the YCbCr 4:2:0 transmission when the core operates in 1 symbol per clock mode.



5.1.3. Source Window of Opportunity Generator

The source Window of Opportunity (WOP) generator creates valid data islands within the blanking regions.

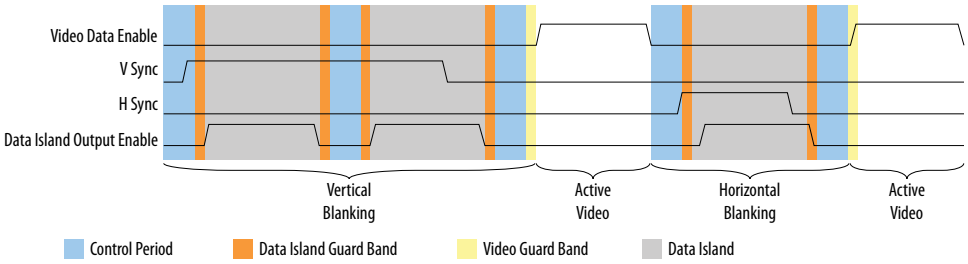
During horizontal blanking region, the WOP generator creates a leading region to hold at least 12 period symbols that include eight preamble symbols. The generator also creates a trailing region to hold two data island trailing guard band symbols, at least 12 control period symbols that include eight preamble symbols and two video leading guard band symbols.

During vertical blanking region, the source cannot send more than 18 auxiliary packets consecutively. The WOP generator deasserts the data island output enable (`aux_wop`) line after every 18th auxiliary packet for 32-symbol clocks.

The WOP generator also has an integral number of auxiliary packet cycles: 24 clocks when processing in 1-symbol mode, 16 clocks when processing in 2-symbol mode, and 8 clocks when processing in 4-symbol mode.

Figure 22. Typical Window of Opportunity

The figure below shows a typical output from the WOP generator.



5.1.4. Source Auxiliary Packet Encoder

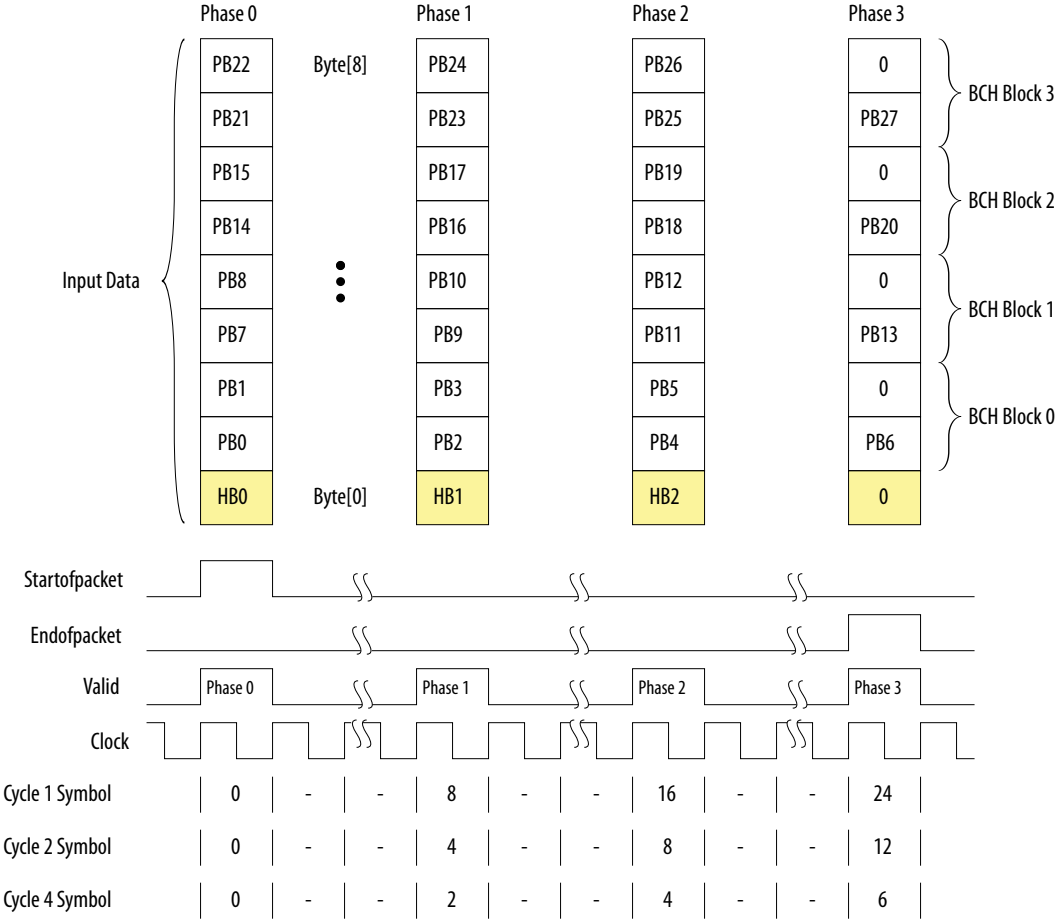
Auxiliary packets are encoded by the source auxiliary packet encoder.

The auxiliary packets originate from several sources, which are multiplexed into the auxiliary packet encoder in a round-robin schedule. The auxiliary packet encoder converts a standard stream into the channel data format required by the TERC4 encoder.

The auxiliary packet encoder also calculates and inserts the Bose-Chaudhuri-Hocquenghem (BCH) error correction code.

Figure 23. Auxiliary Packet Encoder Input

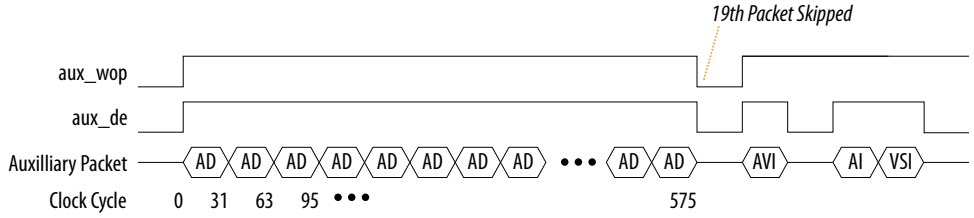
The figure below shows the auxiliary packet encoder input from a 72-bit input data.



The encoder assumes the data valid input will remain asserted for the duration of a packet to complete. A packet is always 24 clocks (in 1-symbol mode), 12 clocks (in 2-symbol mode), or 6 clocks (in 4-symbol mode).

Figure 24. Typical Auxiliary Packet Stream During Blanking Interval

The figure below shows a typical auxiliary packet stream in 1-symbol per clock mode, where 0 denotes a null packet.



AD: Audio Data
AVI: Auxilliary Video Infoframe
AI: Audio Information Infoframe
VSI: Vendor Specific Infoframe

5.1.5. Source Auxiliary Packet Generators

The source core uses various auxiliary packet generators. The packet generators convert the packet field inputs to the auxiliary packet stream format.

The packet generator propagates backpressure from the output ready signal to the input ready signal. The generator asserts the input valid signal when a packet is ready to be transmitted. The input valid signal remains asserted until the end of the packet and the generator receives a ready acknowledgment.

5.1.6. Source Auxiliary Data Path Multiplexers

The auxiliary data path multiplexers provide paths for the various auxiliary packet generators.

The various auxiliary packet generators traverse a multiplexed routing path to the auxiliary packet encoder. The multiplexers obey a round-robin schedule and propagate backpressure.

5.1.7. Source Auxiliary Control Port

To simplify the user logic, the source core has control ports to send the most common auxiliary control packets.

These packets are: General Control Packet, Auxiliary Video Information (AVI) InfoFrame, and HDMI Vendor Specific InfoFrame (VSI).

The core sends the default values in the auxiliary packets. The default values allow the core to send video data compatible with the *HDMI 1.4b Specification* with minimum description.

You can also override the generators using the customized input values. The override values replace the default values when the input checksum is non-zero.

Table 23. Insertion and Filtration

Auxiliary Packets		Insertion/Filtration	Frequency of Insertion
General Control Packet (GCP)	–	The core always inserts GCP packets from the GCP sideband upon the rising edge of `vsync`. The core always removes the GCP in the Auxiliary Data Port. You must provide the pixel packing and color depth information through the `gcp` port.	Once per frame.
Auxiliary Video Information (AVI) InfoFrame	info_avi[112]=1'b0	The core inserts `info_avi[111:0]` when there is a non-zero bit upon the rising edge of `vsync`. The core send default values when all bits are zero. The core filters the AVI InfoFrame packet on the Auxiliary Data Port.	Once per frame.
	info_avi[112]=1'b1	The core does not insert `info_avi[111:0]`. The AVI InfoFrame packet on the Auxiliary Data Port passes through.	

continued...

Auxiliary Packets		Insertion/Filtration	Frequency of Insertion
Vendor Specific InfoFrame (VSI)	info_vsi[61]=1'b0	The core inserts `info_vsi[60:0]` when there is a non-zero bit upon the rising edge of `vsync`. The core sends default values when all bits are zero. The core filters the VSI InfoFrame packet on the Auxiliary Data Port.	Once per frame.
	info_vsi[61]=1'b1	The core does not insert `info_vsi[60:0]`. The VSI InfoFrame packet on the Auxiliary Data Port passes through.	
Audio Metadata (AM)	audio_metadata[165]=1'b0	The core inserts audio_metadata[164:0] when `audio_format[3:0]` is 3D audio or MST audio upon the rising edge of `vsync`. The core filters the AM packet on the Auxiliary Data Port.	Once per frame.
	audio_metadata[165]=1'b1	The core does not insert `audio_metadata[164:0]`. The AM packet on the Auxiliary Data Port passes through.	
Audio InfoFrame (AI)	audio_info_ai[48]=1'b0	The core inserts `audio_info_ai[47:0]` when there is a non-zero bit upon the rising edge of `vsync`. The core sends default values when all bits are zero. The core filters the AI packet on the Auxiliary Data Port.	Once per frame.
	audio_info_ai[48]=1'b1	The core does not insert `audio_info_ai[47:0]`. The AI packet on the Auxiliary Data Port passes through.	
Audio Control Regeneration (ACR)	–	The core always inserts the `audio_N` and `audio_CTS`. The core does not filter the ACR packet in the auxiliary. If there is ACR packet in the Auxiliary Data Port, you must remove it before passing into the Auxiliary Data Port.	Every 1 ms.
Audio Sample	–	The core always inserts `audio_data`. The core does not filter the audio sample packet in the Auxiliary Data Port. If there is audio sample packet in the Auxiliary Data Port, you must remove it before passing into the Auxiliary Data Port.	Based on audio sample rate.

5.1.7.1. Source General Control Packet (GCP)

Table 24. Source GCP Bit-Fields

This table lists the controllable bit-fields for the Source `gcp[5:0]` port.

Bit Field	Name	Value				Comment
gcp[3:0]	Color Depth (CD)	CD3	CD2	CD1	CD0	Color depth
		0	0	0	0	Color depth not indicated
		0	1	0	0	8 bpc or 24 bits per pixel (bpp)

continued...

Bit Field	Name	Value				Comment
		0	1	0	1	10 bpc or 30 bpp
		0	1	1	0	12 bpc or 36 bpp
		0	1	1	1	16 bpc or 48 bpp
		Others				Reserved
gcp[4]	Set_AVMUTE	Refer to *HDMI 1.4b Specification Section 5.3.6*.				
gcp[5]	Clear_AVMUTE	Refer to *HDMI 1.4b Specification Section 5.3.6*.				

All other fields for the source GCP, (for example, Pixel Packing Phase and Default Phase as described in *HDMI 1.4b Specification Section 5.3.6*) are calculated automatically inside the core. You must provide the bit-field values in the table above through the source `gcp[5:0]` port. The GCP on the Auxiliary Data Port will always be filtered.

5.1.7.2. Source Auxiliary Video Information (AVI) InfoFrame Bit-Fields

Table 25. Source Auxiliary Video Information (AVI) InfoFrame

The table below lists the bit-fields for the AVI InfoFrame port bundle (as described in *HDMI 1.4b Specification Section 8.2.1*).

The signal bundle is clocked by `ls_clk`.

Bit-field	Name	Description	Default Value
7:0	Checksum	Checksum	8'h67
9:8	S	Scan information	2'h0
11:10	B	Bar info data valid	2'h0
12	A0	Active information present	1'h0
14:13	Y	RGB or YCbCr indicator	2'h0
15	Reserved	Returns 0	1'h0
19:16	R	Active format aspect ratio	4'h8
21:20	M	Picture aspect ratio	2'h0
23:22	C	Colorimetry (for example: ITU BT.601, BT.709)	2'h0
25:24	SC	Non-uniform picture scaling	2'h0
27:26	Q	Quantization range	2'h0
30:28	EC	Extended colorimetry	3'h0
31	ITC	IT content	1'h0
38:32	VIC	Video format identification code	7'h00
39	Reserved	Returns 0	1'h0
43:40	PR	Picture repetition factor	4'h0
45:44	CN	Content type	2'h0
47:46	YQ	YCC quantization range	2'h0

continued...

Bit-field	Name	Description	Default Value
63:48	ETB	Line number of end of top bar	16'h0000
79:64	SBB	Line number of start of bottom bar	16'h0000
95:80	ELB	Pixel number of end of left bar	16'h0000
111:96	SRB	Pixel number of start of right bar	16'h0000
112	Control	Disables the core from inserting the InfoFrame packet. • 1: The core does not insert `info_avi[111:0]`. The AVI InfoFrame packet on the Auxiliary Data Port passes through. • 0: The core inserts `info_avi[111:0]` when there is a non-zero bit. The core sends default values when all bits are zero. The core filters the AVI InfoFrame packet on the Auxiliary Data Port.	–

5.1.7.3. Source HDMI Vendor Specific InfoFrame (VSI)

Table 26. Source HDMI Vendor Specific InfoFrame Bit-Fields

The table below lists the bit-fields for VSI (as described in *HDMI 1.4b Specification Section 8.2.3*).

The signal bundle is clocked by `ls_clk`.

Bit-field	Name	Description	Default Value
4:0	Length	Length of HDMI VSI payload	5'h06
12:5	Checksum	Checksum	8'h69
36:13	IEEE	24-bit IEEE registration identifier (0×000C03)	24'h000C03
41:37	Reserved	Reserved (0)	5'h00
44:42	HDMI_Video_Format	Structure of extended video formats exclusively defined in *HDMI 1.4b Specification*	3'h0
52:45	HDMI_VIC or 3D_Structure	• If HDMI_Video_Format = 3'h1, [52:45] = HDMI proprietary video format identification code • If HDMI_Video_Format = 3'h2, [52:49] = 3D_Structure and [48:45] = Reserved (0)	8'h00
57:53	Reserved	Reserved (0)	5'h00
60:58	3D_Ext_Data	3D extended data	3'h0
61	Control	Disables the core from inserting the InfoFrame packet. • 1: The core does not insert `info_vsi[60:0]`. The VSI InfoFrame packet on the Auxiliary Data Port passes through. • 0: The core inserts `info_vsi[60:0]` when there is a non-zero bit. The core sends default values when all bits are zero. The core filters the VSI InfoFrame packet on the Auxiliary Data Port.	–

5.1.8. Source Audio Encoder

Audio transport allows four packet types:

- Audio Clock Regeneration
- Audio InfoFrame
- Audio Metadata
- Audio Sample

The Audio Clock Regeneration packet contains the CTS and N values. You need to provide these values as recommended in *HDMI 1.4b Specification Section 7.2.1 through 7.2.3* and *HDMI 2.0b Specification Section 9.2.1*. The core schedules this packet to be sent every ms. The timestamp scheduler uses the `audio_clk` and *N* value to determine a 1-ms interval.

The audio data queues on a DCFIFO. The core also uses the DCFIFO to synchronize its clock to `ls_clk`. The Audio Packetizer packs the audio data into the Audio Sample packets according to the specified audio format (as described in *HDMI 1.4b Specification Section 5.3.4*). An Audio Sample packet can contain up to 4 audio samples, based on the required audio sample clock. The core sends the Audio Sample packets whenever there is an available slot in the auxiliary packet stream.

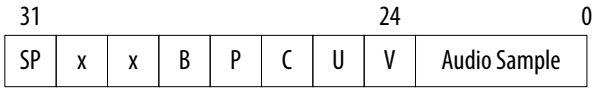
The core determines the payload data packet type from the `audio_format[3:0]` signal.

Table 27. Definition of the Supported audio_format[3:0]

Value	Name	Description
0	Linear Pulse-Code Modulation (LPCM)	Use packet type 0x02 to transport payload data
4	3D Audio (LPCM)	Use packet type 0x0B to transport payload data
6	Multi-Stream(MST) Audio for LPCM	Use packet type 0x0E to transport payload data
Others	–	Reserved

The 32-bit audio data is packed in IEC-60958 standard. The least significant word is the left channel sample.

Figure 25. Audio Data Packing



The fields are defined as:

SP : Sample Present
x : Not Used
B : Start of 192-bit IEC-60958 Channel Status
P : Parity Bit
C : Channel Status
U : User Data Bit
V : Valid Bit

The `audio_data` port is always at a fixed value of 256 bits. In the LPCM format, the core can send up to 8 channels of audio data.

- Channel 1 audio data should be present at `audio_data[31:0]`.
- Channel 2 audio data should be present at `audio_data[63:32]` and so on.

The Sample Present (SP) bit determines whether to use 2-channel or 8-channel layout. If the SP bit from Channel 3 is high, then the core uses the 8-channel layout. If otherwise, the core uses the 2-channel layout. The core ignores all other fields if the SP bit is 0.

The core requires an `audio_de` port for designs in which the `audio_clk` port frequency is higher than the actual audio sample clock. The `audio_de` port qualifies the audio data. If `audio_clk` is the actual audio sample clock, you can tie the `audio_de` signal to 1. For audio channels fewer than 8, insert 0 to the respective audio data of the unused audio channels.

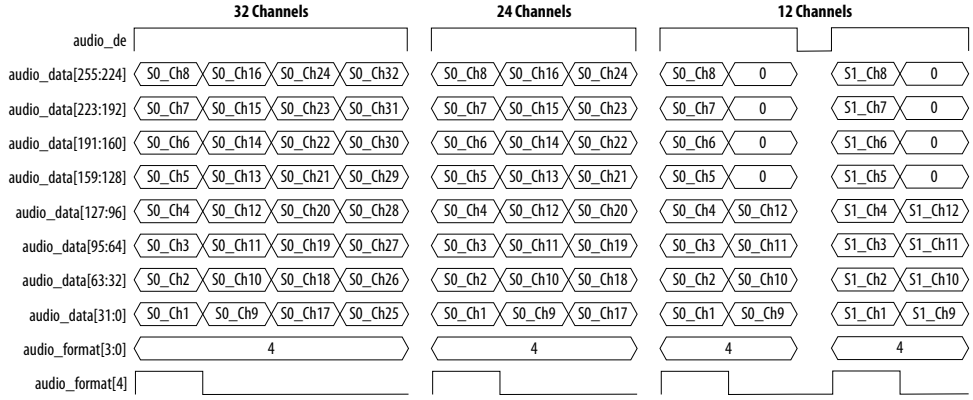
The Audio Clock Regeneration and Audio Sample packets on the Auxiliary Data Port are not filtered by the core. You must filter these packets externally if you want to loop back the auxiliary data stream from the sink.

3D Audio Format

In 3D format, the core sends up to 32 channels audio data by consuming up to 4 writes of 8 channels. Assert `audio_format[4]` to indicate the first 8 channels of each sample. For audio channels greater than 8, do not drive `audio_clk` at actual audio sample clock; instead drive `audio_clk` with `ls_clk` and qualify `audio_data` with `audio_de`.

Figure 26. 3D Audio Input Example

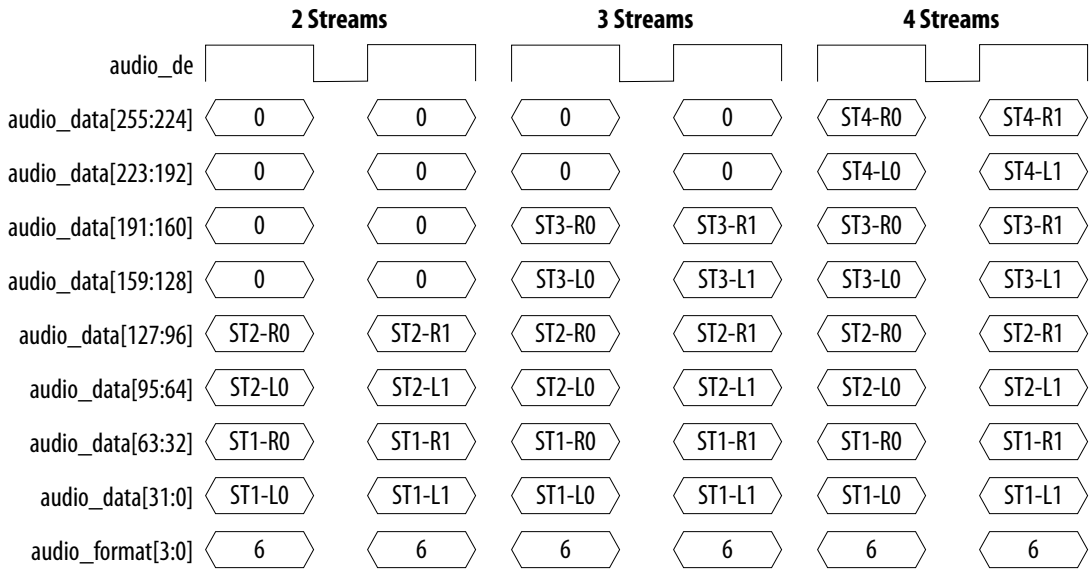
Figure below shows the three examples of 3D audio: Full 32 channels, 24 channels, and 12 channels. In the 12 channels example, the 4 most significant audio channels of the last beat are zero.



MST Audio Format

In MST format, the core sends 2, 3, or 4 streams of audio. For audio streams fewer than 4, you must set the respective audio data to zero for the unused streams as shown in the figure below.

Figure 27. MST Audio Input Example



5.1.8.1. Audio InfoFrame (AI) Bundle Bit-Fields

The core sends the AI default values in the auxiliary packets.

The default values are overridden by the customized input values (`audio_info_ai[47:0]`) when the input checksum is non-zero. The core sends the AI packet on the active edge of the V-SYNC signal to ensure that the packet is sent once per field.

Table 28. Source Audio InfoFrame Bundle Bit-Fields

Table below lists the AI signal bit-fields (as described in *HDMI 1.4b Specification Section 8.2.2*). The signal bundle is clocked by `ls_clk`.

Bit-field	Name	Description	Default Value
7:0	Checksum	Checksum	8'h71
10:8	CC	Channel count	3'h0
11	Reserved	Returns 0	1'h0
15:12	CT	Audio format type	4'h0
17:16	SS	Bits per audio sample	2'h0
20:18	SF	Sampling frequency	3'h0
23:21	Reserved	Returns 0	3'h0
31:24	CXT	Audio format type of the audio stream	8'h00
39:32	CA	Speaker location allocation FL, FR	8'h00
41:40	LFEPBL	LFE playback level information, dB	2'h0
42	Reserved	Returns 0	1'h0

continued...

Bit-field	Name	Description	Default Value
46:43	LSV	Level shift information, dB	4'h0
47	DM_INH	Down-mix inhibit flag	1'h0
48	Control	Disables the core from inserting the AI packet. • 1: The core does not insert `audio_info_ai[47:0]`. The AI packet on the Auxiliary Data Port passes through. • 0: The core inserts `audio_info_ai[47:0]` when there is a non-zero bit. The core sends default values when all bits are zero. The core filters the AI packet on the Auxiliary Data Port.	–

5.1.8.2. Audio Metadata Bundle Bit-Fields

The Audio Metadata (AM) packet carries additional information related to 3D Audio and Multi-Stream Audio (MST).

The core sends the AM packet on the active edge of the V-SYNC signal to ensure that the packet is sent once per field. The signal bundle of `audio_metadata[165:0]` is clocked by `ls_clk`.

Table 29. Audio Metadata Bundle Bit-Fields for Packet Header and Control

Table below lists the AM signal bit-fields for packet header (as described in the *HDMI 2.0b Specification Section 8.3*) and control.

Bit-field	Name	Description
0	3D_AUDIO	• 1: Transmits 3D audio • 0: Transmits MST audio
2:1	NUM_VIEWS	Number of views for an MST stream
4:3	NUM_AUDIO_STR	Number of audio streams - 1
165	Control	Disables the core from inserting the AM packet. • 1: The core does not insert `audio_metadata[164:0]`. The AM packet on the Auxiliary Data Port passes through. • 0: The core inserts `audio_metadata[164:0]` when `audio format[3:0]` is 3D audio or MST audio. The core filters the AM packet on the Auxiliary Data Port.

Table 30. Audio Metadata Bundle Bit-Fields for Packet Content when 3D_AUDIO = 1

Table below lists the AM signal bit-fields for packet content when 3D_AUDIO = 1 (as described in the *HDMI 2.0b Specification Section 8.3.1*).

Bit-field	Name	Description
9:5	3D_CC	Channel count of the transmitted 3D audio
12:10	Reserved	Reserved (0)

continued...

Bit-field	Name	Description
16:13	ACAT	Audio channel allocation standard
20:17	Reserved	Reserved (0)
28:21	3D_ACAT	Channel/Speaker allocation for 3D audio
164:29	Reserved	Reserved (0)

Table 31. Audio Metadata Bundle Bit-Fields for Packet Content when 3D_AUDIO = 0

Table below lists the AM signal bit-fields for packet content when 3D_AUDIO = 0 (as described in the *HDMI 2.0b Specification Section 8.3.2*).

Bit-field	Name	Description
5	Multiview_Left_0	Left stereoscopic picture (Subpacket 0 in MST Audio Sample Packet)
6	Multiview_Right_0	Right stereoscopic picture (Subpacket 0 in MST Audio Sample Packet)
12:7	Reserved	Reserved (0)
15:13	Suppl_A_Type_0	Supplementary audio type (Subpacket 0 in MST Audio Sample Packet)
16	Suppl_A_Mixed_0	Mix of main audio components and a supplementary audio track (Subpacket 0 in MST Audio Sample Packet)
17	Suppl_A_Valid_0	Audio stream contains a supplementary audio track (Subpacket 0 in MST Audio Sample Packet)
19:18	Reserved	Reserved (0)
20	LC_Valid_0	Validity of Language_Code (Subpacket 0 in MST Audio Sample Packet)
44:21	Language_Code_0	Audio stream language (Subpacket 0 in MST Audio Sample Packet)
45	Multiview_Left_1	Left stereoscopic picture (Subpacket 1 in MST Audio Sample Packet)
46	Multiview_Right_1	Right stereoscopic picture (Subpacket 1 in MST Audio Sample Packet)
52:47	Reserved	Reserved (0)
55:53	Suppl_A_Type_1	Supplementary audio type (Subpacket 1 in MST Audio Sample Packet)
56	Suppl_A_Mixed_1	Mix of main audio components and a supplementary audio track (Subpacket 1 in MST Audio Sample Packet)
57	Suppl_A_Valid_1	Audio stream contains a supplementary audio track (Subpacket 1 in MST Audio Sample Packet)
59:58	Reserved	Reserved (0)
60	LC_Valid_1	Validity of Language_Code (Subpacket 1 in MST Audio Sample Packet)
84:61	Language_Code_1	Audio stream language (Subpacket 1 in MST Audio Sample Packet)
85	Multiview_Left_2	Left stereoscopic picture (Subpacket 2 in MST Audio Sample Packet)
86	Multiview_Right_2	Right stereoscopic picture (Subpacket 2 in MST Audio Sample Packet)

continued...

Bit-field	Name	Description
92:87	Reserved	Reserved (0)
95:93	Suppl_A_Type_2	Supplementary audio type (Subpacket 2 in MST Audio Sample Packet)
96	Suppl_A_Mixed_2	Mix of main audio components and a supplementary audio track (Subpacket 2 in MST Audio Sample Packet)
97	Suppl_A_Valid_2	Audio stream contains a supplementary audio track (Subpacket 2 in MST Audio Sample Packet)
99:98	Reserved	Reserved (0)
100	LC_Valid_2	Validity of Language_Code (Subpacket 2 in MST Audio Sample Packet)
124:101	Language_Code_2	Audio stream language (Subpacket 2 in MST Audio Sample Packet)
125	Multiview_Left_3	Left stereoscopic picture (Subpacket 3 in MST Audio Sample Packet)
126	Multiview_Right_3	Right stereoscopic picture (Subpacket 3 in MST Audio Sample Packet)
132:127	Reserved	Reserved (0)
135:133	Suppl_A_Type_3	Supplementary audio type (Subpacket 3 in MST Audio Sample Packet)
136	Suppl_A_Mixed_3	Mix of main audio components and a supplementary audio track (Subpacket 3 in MST Audio Sample Packet)
137	Suppl_A_Valid_3	Audio stream contains a supplementary audio track (Subpacket 3 in MST Audio Sample Packet)
139:138	Reserved	Reserved (0)
140	LC_Valid_3	Validity of Language_Code (Subpacket 3 in MST Audio Sample Packet)
164:141	Language_Code_3	Audio stream language (Subpacket 3 in MST Audio Sample Packet)

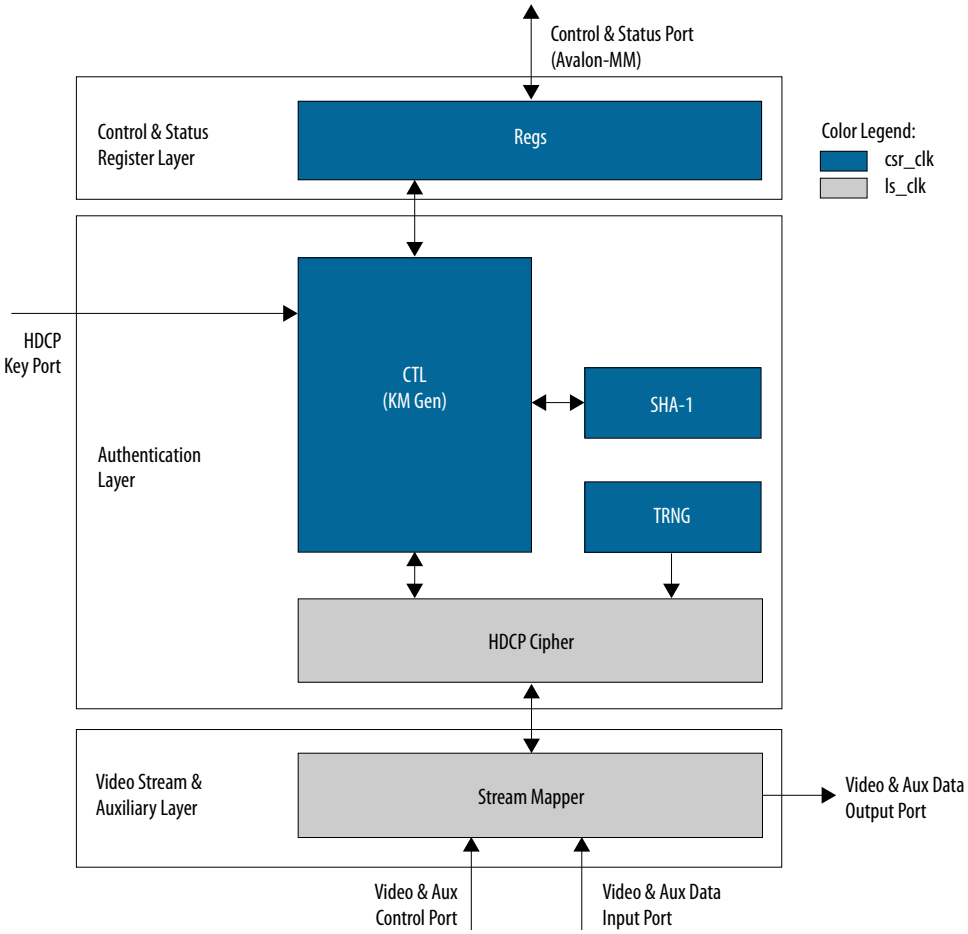
5.1.9. HDCP 1.4 TX Architecture

The HDCP 1.4 transmitter block encrypts video and auxiliary data prior to the transmission over serial link that has HDCP 1.4 device connected.

The HDCP 1.4 TX core consists of the following entities:

- Control and Status Registers Layer
- Authentication Layer
- Video Stream and Auxiliary Layer

Figure 28. Architecture Block Diagram of HDCP 1.4 TX IP



The Nios II processor typically drives the HDCP 1.4 TX core. The processor implements the authentication protocol. The processor accesses the IP through the Control and Status Port using Avalon Memory Mapped (Avalon-MM) interface.

The HDCP specifications requires the HDCP 1.4 TX core to be programmed with the DCP-issued production keys – Device Private Keys (Akeys) and Key Selection Vector (Aksv). The IP retrieves the key from the on-chip memory externally to the core through the HDCP Key Port. The on-chip memory must store the key data in the arrangement in the table below.

Table 32. HDCP 1.4 TX Key Port Addressing

Address	Content
6'h29	{16'd0, Aksv[39:0]}
6'h28	Akeys39[55:0]
6'h27	Akeys38[55:0]

continued...

Address	Content
...	...
6'h01	Akeys01[55:0]
6'h00	Akeys00[55:0]

When authenticating with the HDCP 1.4 repeater device, the HDCP 1.4 TX core must perform the second part of the authentication protocol. This second part corresponds to the computation of the SHA-1 hash digest for all downstream device KSVs which are written to the registers in Control and Status Register Layer using the Control and Status Port (Avalon-MM).

The Video Stream and Auxiliary layer receives audio and video content over its Video and Aux Data Input Port, and performs the encryption operation. The Video Stream and Auxiliary Layer detects the Encryption Status Signaling (ESS) provided by the HDMI TX core to determine when to encrypt frames.

You can use the HDCP 1.4 registers to customize your design configurations. The HDCP 1.4 TX core supports full handshaking mechanism for authentication. Every issued command should be followed by polling of the assertion of its corresponding status bit before proceeding to issuing the next command. The value of AUTH_CMD must be in one-hot format that only one bit can be set at a time.

Table 33. HDCP 1.4 TX Registers Mapping

Address	Register	R/W	Reset	Bit	Bit Name	Description
0x00	AUTH_CMD (one-hot)	WO	0x0	31:6	Reserved	Reserved.
				5	GO_V	Set to 1 to compute V and compare against V' during authentication with repeater. Self-cleared.
				4	Reserved	Reserved.
				3	GEN_RI	Set to 1 to generate and receive R0 during authentication exchange or Ri during link integrity verification. Ri-Ri' comparison should be performed by Nios II processor. Self-cleared.
				2	GO_KM	Set to 1 to compute master key (km). Self-cleared.
				1	GEN_AKSV	Set to 1 to request and receive Aksv. Self-cleared.
				0	GEN_AN	Set to 1 to generate and receive new true random An. Self-cleared.
0x01	AUTH_MSGDATAIN	WO	0x0	31:8	Reserved	Reserved.
				7:0	MSGDATAIN	Write messages (in byte) from receiver in burst mode. 1. Master key computation: Prior to setting GO_KM to 1, the BCAPS.REPEATER bit had

continued...

Address	Register	R/W	Reset	Bit	Bit Name	Description
						to be set and the following messages had to be written in this sequence: a. 5 bytes of Bksv with least significant byte (lsb) first. 2. V generation: Prior to setting GO_V to 1, the following messages had to be written in this sequence: a. 20 bytes of V' with lsb first b. Variable length of KSV list with lsb first c. 2 bytes of Bstatus with lsb first
0x02	AUTH_STATUS	RO	0x0	31	KM_OK	Asserted by the core to indicate the received Bksv is valid. Poll KM_DONE until it is set before reading KM_OK.
				30	V_OK	Asserted by the core to indicate V-V' comparison is passed. Poll V_DONE until it is set before reading V_OK.
				29:6	Reserved	Reserved.
				5	V_DONE	Asserted by the core when V is generated. Self-cleared upon next GO_V is set.
				4	Reserved	Reserved
				3	RI_DONE	Asserted by the core when Ri is generated. Self-cleared upon next GEN_RI is set.
				2	KM_DONE	Asserted by the core when Km is generated. Self-cleared upon next GO_KM is set.
				1	AKSV_DONE	Asserted by the core when Aksv is ready to be read from MSGDATAOUT. Self-cleared upon next GEN_AKSV is set.
				0	AN_DONE	Asserted by the core when new random An is generated and ready to be read from MSGDATAOUT. Self-cleared upon next GEN_AN is set.
0x03	AUTH_MSGDATAOUT	RO	0x0	31:8	Reserved	Reserved.

continued...

Address	Register	R/W	Reset	Bit	Bit Name	Description
				7:0	MSGDATAOUT	Read messages (in byte) from the IP in burst mode. 1. An generation: When AN_DONE is set to 1, reading this offset 8 times to obtain An with lsb first. 2. Aksv request: When AKSV_DONE is set to 1, reading this offset 5 times to obtain Aksv with lsb first. 3. Ri request: When RI_DONE is set to 1, reading this offset 2 times to obtain Ri with lsb first.
0x04	VID_CTL	RW	0x0	31:1	Reserved	Reserved.
				0	HDCP_ENABLE	Set to 1 to enable HDCP 1.4 encryption. Set to 0 if HDCP 1.4 encryption is not required especially when it is in unauthenticated state.
0x05	BCAPS	RW	0x0	31:2	Reserved	Reserved..
				1	REPEATER	Downstream repeater capability. Write bit 6 (REPEATER) of Bcaps received from downstream to this offset.
				0	Reserved	Reserved.

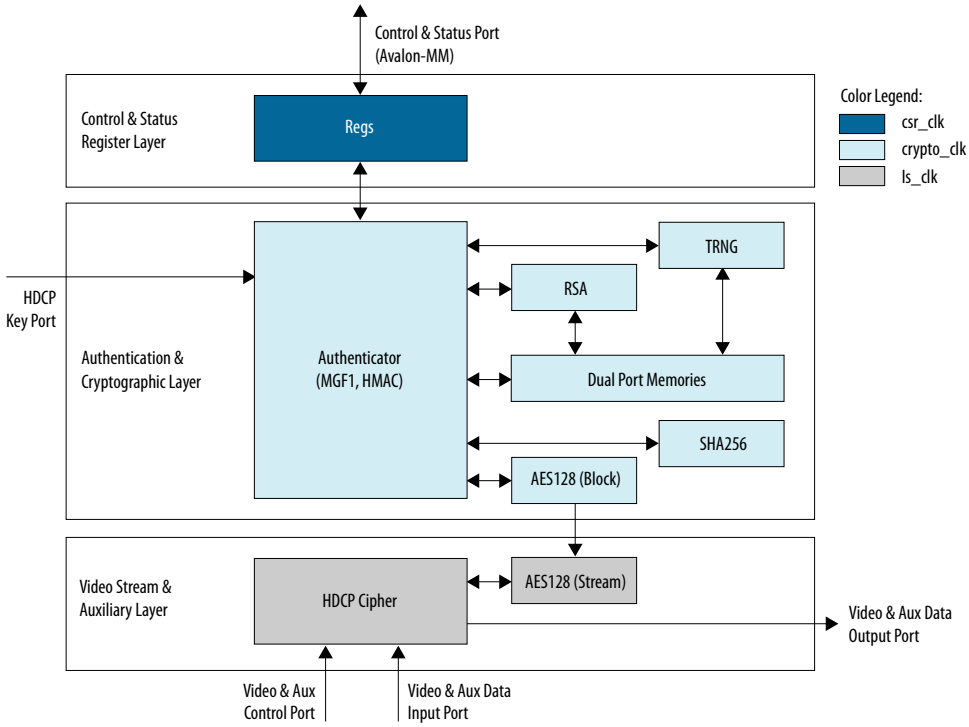
5.1.10. HDCP 2.3 TX Architecture

The HDCP 2.3 transmitter block encrypts video and auxiliary data prior to the transmission over serial link that has HDCP 2.3 device connected.

The HDCP 2.3 TX core consists of the following entities:

- Control and Status Registers Layer
- Authentication and Cryptographic Layer
- Video Stream and Auxiliary Layer

Figure 29. Architecture Block Diagram of HDCP 2.3 TX IP



The Nios II processor typically drives the HDCP 2.3 TX core. The processor implements the authentication protocol. The processor accesses the IP through the Control and Status Port using Avalon Memory Mapped (Avalon-MM) interface.

The HDCP specifications requires the HDCP 2.3 TX core to be programmed with the DCP-issued production key – Global Constant (lc128). The IP retrieves the key from the on-chip memory externally to the core through the HDCP Key Port. The on-chip memory must store the key data in the arrangement in the table below.

Table 34. HDCP 2.3 TX Key Port Addressing

Address	Content
2'h3	lc128[127:96]
2'h2	lc128[95:64]
2'h1	lc128[63:32]
2'h0	lc128[31:0]

The Video Stream and Auxiliary Layer receives audio and video content over its Video and Aux Data Input port, and performs the encryption operation. The Video Stream and Auxiliary Layer detects the Encryption Status Signaling (ESS) provided by the HDMI TX core to determine when to encrypt frames.

You can use the HDCP 2.3 registers to perform authentication. The HDCP 2.3 TX core supports full handshaking mechanism for authentication. Every issued command should be followed by polling of the assertion of its corresponding status bit before proceeding to issuing the next command. The value of CRYPTO_CMD must be in one-hot encoding format that only one bit can be set at a time.

Table 35. HDCP 2.3 TX Registers Mapping

Address	Register	R/W	Reset	Bit	Bit Name	Description
0x00	CRPYTO_CMD (one-hot)	WO	0x0	31:11	Reserved	Reserved
				10	GO_HMAC_M	Set to 1 to compute M and verify against M'. Self-cleared upon operation is busy.
				9	GO_HMAC_V	Set to 1 to compute V and verify against V'. Self-cleared upon operation is busy.
				8	GEN_RIV	Set to 1 to generate and receive new random riv. Self-cleared upon operation is busy.
				7	GEN_EDKEYKS	Set to 1 to generate and receive new random Edkey(ks). Self-cleared upon operation is busy.
				6	GO_HMAC_L	Set to 1 to compute L and verify against L'. Self-cleared upon operation is busy.
				5	GEN_RN	Set to 1 to generate and receive new random rn. Self-cleared upon operation is busy.
				4	GO_HMAC_H	Set to 1 to compute H and verify against H'. Self-cleared upon operation is busy.
				3	GO_KD	Set to 1 to compute kd (dkey0, dkey1). Self-cleared upon operation is busy.
				2	GEN_EKPUBKM	Set to 1 to generate and receive new random Ekpub(km). Self-cleared upon operation is busy.
				1	GO_SIG	Set to 1 to verify signature (certrx or SRM). Self-cleared upon operation is busy.
				0	GEN_RTX	Set to 1 to generate and receive new random rtx. Self-cleared upon operation is busy.
0x01	CRYPTO_MSGDATAIN	WO	0x0	31:8	Reserved	Reserved
				7:0	MSGDATAIN	Write messages (in byte) from receiver in burst mode. 1. Signature verification (certrx): Prior to setting GO_SIG to 1, the following messages had to be written in this sequence:

continued...

Address	Register	R/W	Reset	Bit	Bit Name	Description
						a. 384 bytes of signature with least significant byte (lsb) first b. 5 bytes of Receiver ID with most significant byte (msb) first c. 128 bytes of Receiver Public Key modulus (n) with msb first d. 3 bytes of Receiver Public Key exponent (e) with msb first e. 2 bytes of Reserved with msb first 2. Signature verification (SRM): Prior to setting GO_SIG to 1, the following messages had to be written in this sequence: a. 384 bytes of signature with lsb first b. All preceding fields of the SRM (except signature) with msb first 3. Master Key encryption: Prior to setting GEN_EKPUBKM to 1, the following messages had to be written in this sequence: a. 128 bytes of Receiver Public Key modulus (n) with msb first b. 3 bytes of Receiver Public Key exponent (e) with msb first. 4. Compute kd for HMAC: Prior to setting GO_KD to 1, the following messages had to be written in this sequence: a. 8 bytes of rrx with msb first b. 3 bytes of RxCaps with msb first 5. H-H' comparison: Prior to setting GO_HMAC_H to 1, the following messages had to be written in this sequence: a. 32 bytes of H' with msb first 6. L-L' comparison: Prior to setting GO_HMAC_L to 1, the following messages had to be written in this sequence: a. 32 bytes of L' with msb first 7. V-V' comparison: Prior to setting GO_HMAC_V to 1, the following messages had to be written in this sequence:

continued...

Address	Register	R/W	Reset	Bit	Bit Name	Description
						a. 16 bytes of V' with msb first b. Variable length of ReceiverID_List with msb first c. 2 bytes of RxInfo with msb first d. 3 bytes of seq_num_V with msb first 8. M-M' comparison: Prior to setting GO_HMAC_M to 1, the following messages had to be written in this sequence: a. 32 bytes of M' with msb first b. 2 bytes of StreamID_Type with msb first c. 3 bytes of seq_num_M with msb first
0x02	CRYPTO_STATUS	RO	0x0	31	SIG_OK	Asserted by the core to indicate signature verification is passed. Poll SIG_DONE until it is set before reading SIG_OK.
				30	H_OK	Asserted by the core to indicate H-H' comparison is passed. Poll H_DONE until it is set before reading H_OK.
				29	L_OK	Asserted by the core to indicate L-L' comparison is passed. Poll L_DONE until it is set before reading L_OK.
				28	V_OK	Asserted by the core to indicate V-V' comparison is passed. Poll V_DONE until it is set before reading V_OK.
				27	M_OK	Asserted by the core to indicate M-M' comparison is passed. Poll M_DONE until it is set before reading M_OK.
				26:11	Reserved	Reserved
				10	M_DONE	Asserted by the core when M-M' comparison is done. Self-cleared upon next GO_HMAC_M is set.
				9	V_DONE	Asserted by the core when V-V' comparison is done. Self-cleared upon next GO_HMAC_V is set.
				8	RIV_DONE	Asserted by the core when riv is generated and ready to be read from MSGDATAOUT. Self-cleared upon next GEN_RIV is set.
				7	EDKEYKS_DONE	Asserted by the core when Edkey(ks) is generated and ready to be read from MSGDATAOUT. Self-cleared upon next GEN_EDKEYKS is set.

continued...

Address	Register	R/W	Reset	Bit	Bit Name	Description
				6	L_DONE	Asserted by the core when L-L' comparison is done. Self-cleared upon next GO_HMAC_L is set.
				5	RN_DONE	Asserted by the core when rn is generated and ready to be read from MSGDATAOUT. Self-cleared upon next GEN_RN is set.
				4	H_DONE	Asserted by the core when H-H' comparison is done. Self-cleared upon next GO_HMAC_H is set.
				3	KD_DONE	Asserted by the core when kd is generated. Self-cleared upon next GO_KD is set.
				2	EKPUBKM_DONE	Asserted by the core when Ekpub(km) is generated and ready to be read from MSGDATAOUT. Self-cleared upon next GEN_EKPUBKM is set.
				1	SIG_DONE	Asserted by the core when signature verification is done. Self-cleared upon next GO_SIG is set.
				0	RTX_DONE	Asserted by the core when rtx is generated and ready to be read from MSGDATAOUT. Self-cleared upon next GEN_RTX is set.
0x03	CRYPTO_MSGDATAOUT	RO	0x0	31:8	Reserved	Reserved.
				7:0	MSGDATAOUT	Read messages (in byte) from IP core in burst mode. 1. Rtx generation: When RTX_DONE is set to 1, reading this offset 8 times to obtain rtx with msb first. 2. Master Key generation: When EKPUBKM_DONE is set to 1, reading this offset 128 times to obtain Ekpub(km) with msb first. 3. Rn generation: When RN_DONE is set to 1, reading this offset 8 times to obtain rn with msb first. 4. Session Key generation: When EDKEYKS_DONE is set to 1, reading this offset 16 times to obtain Edkey(ks) with msb first. 5. Riv generation: When RIV_DONE is set to 1, reading this offset 8 times to obtain riv with msb first.
0x04	VID_CTL	RW	0x0	31:1	Reserved	Reserved.
				0	HDCP_ENABLE	Set to 1 to enable HDCP 2.3 encryption. Set to 0 if HDCP 2.3 encryption is not required especially when it is in unauthenticated state.

5.2. Source Interfaces

The table lists the source's port interfaces.

Table 36. HDMI Source Interfaces

N is the number of symbols per clock.

Interface	Port Type	Clock Domain	Port	Direction	Description
Reset	Reset	–	reset	Input	Main asynchronous reset input.
Clock	Clock	–	ls_clk	Input	Link speed clock input. Relationship to vid_clk as a function of color depth: • 8 bpc: 1x vid_clk • 10 bpc: 1.25x vid_clk • 12 bpc: 1.5x vid_clk • 16 bpc: 2x vid_clk This signal connects to the transceiver output clock only if an application does not require low TMDS Bit Rate (below the minimum transceiver data rate), which means no oversampling is required. This signal should connect to a PLL output clock that meets the vid_clk relationship if an application requires low TMDS Bit Rate (below the minimum transceiver data rate), which means oversampling is required. Refer to Source Clock Tree on page 82 for more information.
	Clock	–	vid_clk	Input	Video data clock input. For RGB and YCbCr 4:4:4/4:2:2 transport: • 1 symbol per clock mode = pixel clock • 2 symbols per clock mode = pixel clock/2 • 4 symbols per clock mode = pixel clock/4 For YCbCr 4:2:0 transport: • 1 symbol per clock mode = pixel clock/2 • 2 symbols per clock mode = pixel clock/4 • 4 symbols per clock mode = pixel clock/8
	Clock	–	audio_clk	Input	Audio clock input. Connect this signal to ls_clk by qualifying the slower frequency of audio_data with audio_de.

continued...

Interface	Port Type	Clock Domain	Port	Direction	Description
					If you connect this signal to a clock at actual audio sample frequency, you must tie `audio_de` to 1. For audio channels greater than 8, do not drive `audio_clk` at actual audio sample clock; instead drive `audio_clk` with `ls_clk` and qualify `audio_data` with `audio_de`. *Note:* Applicable only when you turn on the **Support auxiliary** and **Support audio** parameters.
Video Data Port	Conduit	`vid_clk`	`vid_data[N*48-1:0]`	Input	Video 48-bit pixel data input port. • In 2 symbols per clock (N=2) mode, this port accepts two 48-bit pixels per clock. • In 4 symbols per clock (N=4) mode, this port accepts four 48-bit pixels per clock.
	Conduit	`vid_clk`	`vid_de[N-1:0]`	Input	Video data enable input that indicates active picture region.
	Conduit	`vid_clk`	`vid_hsync[N-1:0]`	Input	Video horizontal sync input.
	Conduit	`vid_clk`	`vid_vsync[N-1:0]`	Input	Video vertical sync input.
TMDS Data Port (4)	Conduit	`ls_clk`	`out_b[N*10-1:0]`	Output	TMDS encoded blue channel (0) output.
	Conduit	`ls_clk`	`out_g[N*10-1:0]`	Output	TMDS encoded green channel (1) output.
	Conduit	`ls_clk`	`out_r[N*10-1:0]`	Output	TMDS encoded red channel (2) output.
	Conduit	`ls_clk`	`out_c[N*10-1:0]`	Output	Clock channel output. For `out_c` values, refer to Table 37 on page 81 and Table 38 on page 81.
Encoder Control Port	Conduit	`ls_clk`	`mode`	Input	Encoding mode input. • 0: DVI • 1: HDMI

continued...

(4) Connect to the transceiver data input if no oversampling is required. If oversampling is required, the port should connect to a DCFIFO and an oversampling user logic before connecting to a transceiver data input. Refer to Source Clock Tree on page 82 for more information.

Interface	Port Type	Clock Domain	Port	Direction	Description
	Conduit	ls_clk	TMDS_Bit_clock_Ratio	Input	Indicates if TMDS Bit Rate is greater than 3.4Gbps. • 0: (TMDS Bit Rate) / (TMDS Clock Rate) ratio is 10 • 1 = (TMDS Bit Rate) / (TMDS Clock Rate) ratio is 40
	Conduit	ls_clk	Scrambler_Enable	Input	Enables scrambling. • 0: Instructs the source device not to perform scrambling • 1: Instructs the source device to perform scrambling
	Conduit	ls_clk	ctrl[N*6-1:0]	Input	DVI control side-band inputs to override the necessary control and synchronization data in the green and red channels. **Bit-Field** / **Name** N*6+5 — CTL3 N*6+4 — CTL2 N*6+3 — CTL1 N*6+2 — CTL0 N*6+1 — Reserved (0) N*6 — Reserved (0) Refer to the *HDMI 1.4b Specification* for more information.
Auxiliary Data Port (Applicable only when you enable **Support auxiliary** parameter)	Conduit	ls_clk	aux_ready	Output	Auxiliary data channel ready output. Asserted high to indicate that the core is ready to accept data.
	Conduit	ls_clk	aux_valid	Input	Auxiliary data channel valid input to qualify the data.
	Conduit	ls_clk	aux_data[71:0]	Input	Auxiliary data channel data input. For information about the bit-fields, refer to Figure 23 on page 56.
	Conduit	ls_clk	aux_sop	Input	Auxiliary data channel start-of-packet input to mark the beginning of a packet.
	Conduit	ls_clk	aux_eop	Input	Auxiliary data channel end-of-packet input to mark the end of a packet.
Auxiliary Control Port (Applicable only when	Conduit	ls_clk	gcp[5:0]	Input	General Control Packet user input.

continued...

Interface	Port Type	Clock Domain	Port	Direction	Description
you enable **Support auxiliary** parameter)					For information about the bit-fields, refer to Table 24 on page 58.
	Conduit	ls_clk	info_avi[112:0]	Input	Auxiliary Video Information InfoFrame user input. For information about the bit-fields, refer to Table 25 on page 59.
	Conduit	ls_clk	info_vsi[61:0]	Input	Vendor Specific Information InfoFrame user input. For information about the bit-fields, refer to Table 26 on page 60.
Audio Port (Applicable only when you enable **Support auxiliary** and **Support audio** parameters)	Conduit	audio_clk	audio_CTS[19:0]	Input	Audio CTS value input.
	Conduit	audio_clk	audio_N[19:0]	Input	Audio N value input.
	Conduit	audio_clk	audio_data[255:0]	Input	Audio data input. For audio channel values, refer to Table 39 on page 81.
	Conduit	audio_clk	audio_de	Input	Audio data valid input.
	Conduit	audio_clk	audio_mute	Input	Audio mute input. No audio will be transmitted when this signal is asserted high.
	Conduit	ls_clk	audio_info_ai[48:0]	Input	Audio InfoFrame user input. *Note:* If you provide audio_info_ai[48:0] using audio_clk with actual audio sample frequency, you must synchronize the clock domain to ls_clk externally. For information about the bit-fields, refer to Table 28 on page 63.
	Conduit	ls_clk	audio_metadata[165:0]	Input	Carries additional information related to 3D audio and MST audio. *Note:* If you provide audio_metadata[165:0] using audio_clk with actual audio sample frequency, you must synchronize the clock domain to ls_clk externally. For information about the bit-fields, refer to Table 29 on page 64, Table 30 on page 64, and Table 31 on page 65.

continued...

Interface	Port Type	Clock Domain	Port	Direction	Description
	Conduit	audio_clk	audio_format[4:0]	Input	Controls the transmission of the 3D audio and indicates the audio format to be transmitted. **Bit-Field** / **Description** 4 – Assert to indicate the first 8 channels of each 3D audio sample. 3:0 – For information about the bit-fields, refer to Table 27 on page 61.
HDCP Port (Applicable only when you enable **Support HDCP 2.3** or **Support HDCP 1.4** parameters)	Reset	–	hdcp_reset	Input	Main asynchronous reset.
	Clock	–	csr_clk	Input	HDCP clock for Control and Status Registers Layer. Typically, shares the Nios II processor clock (100 MHz).
		–	crypto_clk	Input	HDCP 2.3 clock for Authentication and Cryptographic Layer. You can use any clock with a frequency of up to 200 MHz. *Note:* The clock frequency determines the authentication latency.
	Avalon-MM	csr_clk	csr_addr[7:0]	Input	The Avalon-MM slave port that provides access to internal control and status register, mainly for authentication messages transfer. This interface is expected to operate at Nios II processor clock domain. Because of the extremely large bit portion of message, the IP transfers the message in burst mode with full handshaking mechanism. Write transfers always have a wait time of 0 cycle while read transfers have a wait time of 1 cycle. The addressing should be accessed as word addressing in the Platform Designer flow. For example, addressing of 4 in the Nios II software selects the address of 1 in the slave.
			csr_wr	Input	
			csr_rd	Input	
			csr_wrdata[31:0]	Input	
			csr_rddata[31:0]	Output	
	Conduit (Key)	crypto_clk (HDCP 2.3)	kmem_wait[0] (HDCP 2.3) kmem_wait[1] (HDCP 1.4)	Input	Always keep this signal asserted until the key is ready to be read.

continued...

Interface	Port Type	Clock Domain	Port	Direction	Description
		csr_clk (HDCP 1.4)	kmem_rdaddr[3:0] (HDCP 2.3) kmem_rdaddr[9:4] (HDCP 1.4)	Output	Key read address bus. [3:2] = Reserved.
			kmem_q[31:0] (HDCP 2.3) kmem_q[87:32] (HDCP 1.4)	Input	Key read data transfer. Read transfer always have a wait time of 1 cycle.
	Conduit	ls_clk	hdcp1_enabled	Output	This signal is asserted by the IP if the outgoing video and auxiliary data are HDCP 1.4 encrypted.
			hdcp2_enabled	Output	This signal is asserted by the IP if the outgoing video and auxiliary data are HDCP 2.3 encrypted.

Table 37. out_c Value for TMDS Bit Rate Less than 3.4 Gbps

TMDS_Bit_clock_Ratio = 0 and out_c value is constant.

N	out_c Value
1	10'b1111100000
2	20'b1111100000_1111100000
4	40'b1111100000_1111100000 1111100000_1111100000

Table 38. out_c Value for TMDS Bit Rate Greater than 3.4 Gbps

TMDS_Bit_clock_Ratio = 1 and out_c value is repeated indefinitely.

N	out_c Value			
	t	t+1	t+2	t+3
1	10'h000	10'h000	10'h3ff	10'h3ff
2	20'h00000	20'hfffff	20'h00000	20'hfffff
4	40'hfffff 00000	40'hfffff 00000	40'hfffff 00000	40'hfffff 00000

Table 39. Audio Channel

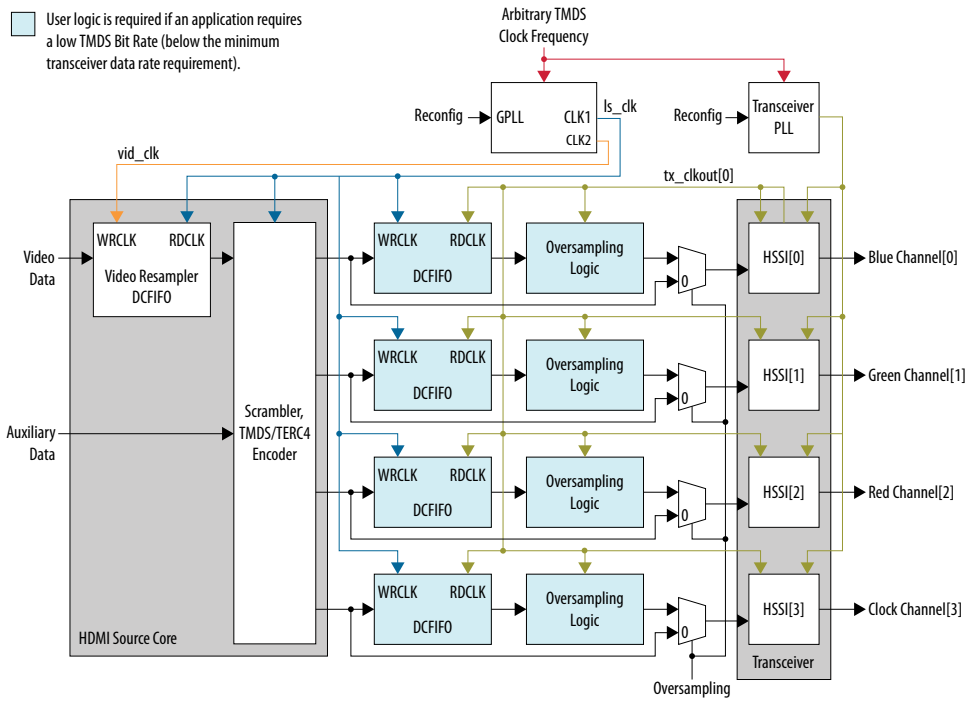
Bit-Field	Audio Channel	
	LPCM and 3D Audio (LPCM)	MST Audio (LPCM)
255:224	8 or 16 or 24 or 32	Stream 4 right channel
223:192	7 or 15 or 23 or 31	Stream 4 left channel
191:160	6 or 14 or 22 or 30	Stream 3 right channel
159:128	5 or 13 or 21 or 29	Stream 3 left channel
127:96	4 or 12 or 20 or 28	Stream 2 right channel
95:64	3 or 11 or 19 or 27	Stream 2 left channel
63:32	2 or 10 or 18 or 26	Stream 1 right channel
31:0	1 or 9 or 17 or 25	Stream 1 left channel

5.3. Source Clock Tree

The source uses various clocks.

Figure 30. Source Clock Tree

The figure shows how the different clocks connect in the source core.



For HDMI source, you must instantiate 4 transceiver channels: 3 channels to transmit data and 1 channel to transmit clock information.

The core uses a general purpose phase-locked loop (GPLL), that is referenced by an arbitrary TMDS clock frequency, to generate the link speed clock (`ls_clk`) and video clock (`vid_clk`).

- The video data clocks into the core at `vid_clk`.
- The TMDS data clocks out from the core at `ls_clk`.

The same arbitrary TMDS clock frequency is also used to drive the transceiver PLL. `ls_clk` and `tx_clkout[0]` are derived from `vid_clk` based on the color depth, TMDS_Bit_clock_Ratio, and user oversampling control bit information.

If an application requires low TMDS Bit Rate (below the transceiver minimum data rate requirement), then the application needs a user logic consisting of a DCFIFO and oversampling logic.

- The DCFIFO synchronizes the TMDS data from `ls_clk` to a faster transceiver output clock (`tx_clkout[0]`).
- The oversampling logic repeats each bit of the TMDS data a given number of times.
- When you enable the oversampling control bit, the transceiver transmits the TMDS data between the HDMI source core and the oversampling logic.
- You can use `tx_clkout[0]` across 4 channels if the transceiver is in bonding mode.

If an application does not require low TMDS Bit Rate, you can connect the core output directly to the transceiver with `tx_clkout[0]` driving the core `ls_clk`. You do not require the GPLL to generate `CLK1` (`ls_clk`).

Related Information

- HDMI Hardware Design Examples for Arria V and Stratix V Devices on page 30
- HDMI Hardware Design Examples for Intel Arria 10, Intel Cyclone 10 GX, and Intel Stratix 10 Devices on page 18

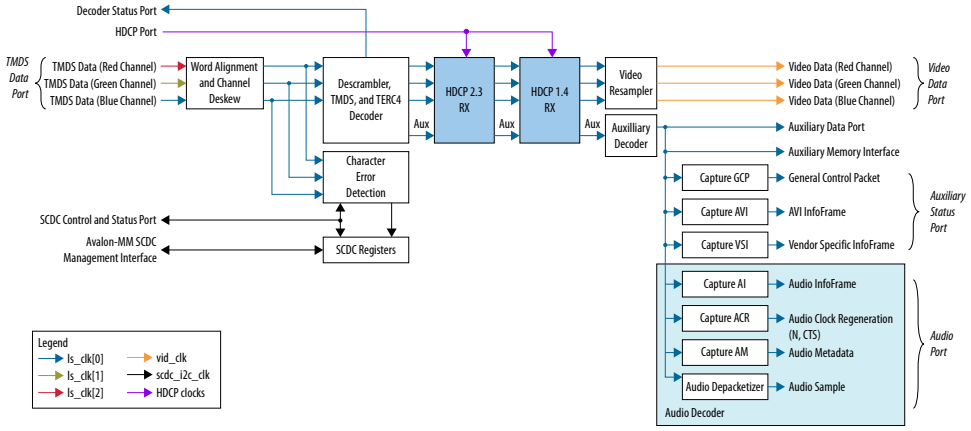
6. HDMI Sink

6.1. Sink Functional Description

The HDMI sink core provides direct connection to the Transceiver Native PHY through a 20-bit or 40-bit parallel data path.

Figure 31. HDMI Sink Signal Flow Diagram

The figure below shows the flow of the HDMI sink signals. The figure shows the various clocking domains used within the core.



The sink core provides three 10-bit, 20-bit, or 40-bit data input paths corresponding to the color channels. The sink core clocks the three 10-bit, 20-bit, or 40-bit channels from the transceiver outputs using the respective transceiver clock outputs.

- Blue channel: 0
- Green channel: 1
- Red channel: 2

6.1.1. Sink Word Alignment and Channel Deskew

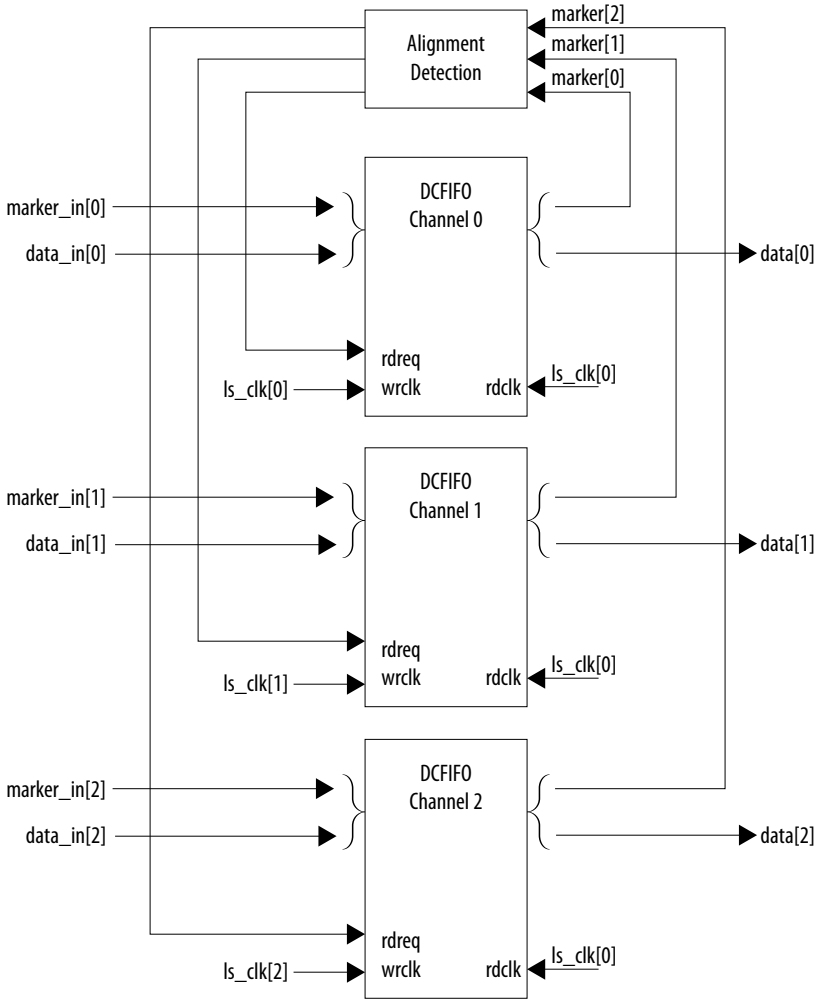
The input stage of the sink is responsible for synchronizing the incoming parallel data channels correctly. The synchronization is split to two stages: word alignment and channel deskew.

Table 40. Synchronization Stages

Stage	Description
Word Alignment	• Correctly aligns the incoming parallel data to word boundaries using bit-slip technique. • TMDS encoding does not guarantee unique control codes, but the core can still use the sequence of continuous symbols found in data and video preambles to align. • The alignment algorithm searches for 8 consecutive 0×54 or 0×ab corresponding to the data and video preambles. *Note:* The preambles are also present in Digital Video Interface (DVI) coding. • The alignment logic asserts a marker indicator when the 8 consecutive signals are detected. • Similarly, the logic infers alignment loss when 8K symbol clocks elapse without a single marker assertion. *Note:* If you are using Intel Arria 10 or Intel Cyclone 10 GX devices, soft word alignment logic in the HDMI RX core is disabled for HDMI 2.0 resolution (data rate >3.4 Gbps). Hard transceiver PCS word alignment is used with some control logic to achieve faster word alignment with more optimized resource utilization. Refer to the design example user guides for more information. *Note:* If you are using Intel Stratix 10 devices, the HDMI RX core uses a new word alignment algorithm logic to achieve fast word alignment time for HDMI 2.0 resolution (data rate >3.4Gbps).
Channel Deskew	• When the data channels are aligned, the core then attempts to deskew each channel. • The sink core deskews at the rising edge of the marker insertion. • For every correct deskewed lane, the marker insertion will appear in all three TMDS encoded streams. • The sink core deskews using three dual-clock FIFOs. • The dual-clock FIFOs also synchronize all three data streams to the blue channel clock to be used later throughout the decoder core.

Figure 32. Channel Deskew DCFIFO Arrangement

The figure below shows the signal flow diagram of the deskew logic.



The FIFO read signal of the channels is normally asserted. The sink core deasserts a particular FIFO read signal if a marker appears at its output and not in the other two FIFO outputs. By deasserting, the sink core stalls the data stream for sufficient cycles to remove the channel skew. If any of the FIFO channels overflow, the sink core asserts a reset signal which propagates backwards to the word alignment logic.

6.1.2. Sink Descrambler, TMDS/TERC4 Decoder

The sink TMDS/TERC4 decoder follows the HDMI/DVI specification. The core enable descrambling automatically when it detects the `Scramble_Enable` bit of the SCDC registers.

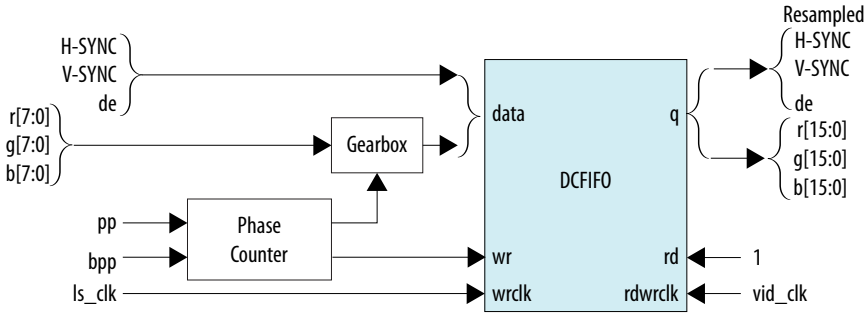
The sink core feeds the aligned channels into the TMDS/TERC4 decoder. You can parameterize the decoder to operate in 1, 2, or 4 TMDS symbols per clock. If you choose 2 or 4 TMDS symbols per clock, the decoder will produce 2 or 4 decoded symbols per clock. The decoded symbols per clock output supports high pixel clock resolutions on low-end FPGA devices.

6.1.3. Sink Video Resampler

The video resampler consists of a gearbox and a dual-clock FIFO (DCFIFO).

The gearbox converts 8-bpc data to 8-, 10-, 12- or 16-bpc data based on the current color depth. The GCP conveys the color depth (bpp) information.

Figure 33. Sink Resampler Signal Flow Diagram



The resampler adheres to the recommended phase count method described in *HDMI 1.4b Specification Section 6.5*.

- To keep the source and sink resamples synchronized, the source must send the packing-phase (pp) value to the sink during the vertical blanking phase, using the general control packet.
- The pp corresponds to the phase of the last pixel in the last active video line.
- The phase-counter logic compares its own pp value to the pp value received in the general control packet and *slips* the phase count if the two pp values do not agree.

The output from the resampler is fixed at 16 bpc. When the resampler operates in lower color depths, the low order bits are zero. The pixel data output format across color space are are described in Figure 10-12.

6.1.4. Sink Auxiliary Decoder

The sink core decodes the auxiliary data path into a 72-bit wide standard packet stream. The stream contains a valid, start-of-packet (SOP) and end-of-packet (EOP) marker.

Table 41. Auxiliary Packet Memory Map

This table lists the addresses corresponding to the captured packets.

Memory Start Address	Packet Name
0	NULL PACKET
4	Audio Clock Regeneration (N/CTS)
8	Audio Sample
12	General Control
16	ACP Packet
20	ISRC1 Packet
24	ISRC2 Packet

continued...

Memory Start Address	Packet Name
28	One Bit Audio Sample Packet 5.3.9
32	DST Audio Packet
36	High Bit rate (HBR) Audio Stream Packet
40	Gamut Metadata Packet
44	3D Audio Sample Packet
48	One Bit 3D Audio Sample Packet
52	Audio Metadata Packet
56	Multi-Stream Audio Sample Packet
60	One Bit Multi-Stream Audio Sample Packet
64	Vendor-Specific InfoFrame
68	AVI InfoFrame
72	Source Product Descriptor InfoFrame
76	Audio InfoFrame
80	MPEG Source InfoFrame
84	TSC VBI InfoFrame
88	Dynamic Range and Mastering InfoFrame

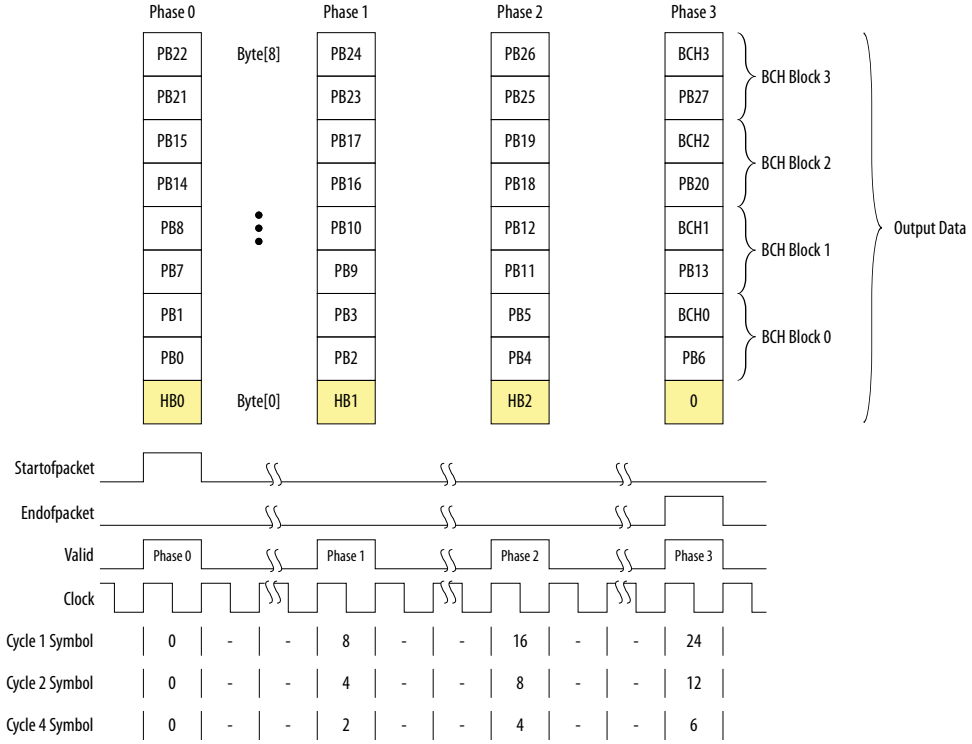
Table 42. Packet Payload Data Byte

This table shows the representation of each packet payload data byte.

Word Offset	Byte Offset								
	8	7	6	5	4	3	2	1	0
0	PB22	PB21	PB15	PB14	PB8	PB7	PB1	PB0	HB0
1	PB24	PB23	PB17	PB16	PB10	PB9	PB3	PB2	HB1
2	PB26	PB25	PB19	PB18	PB12	PB11	PB5	PB4	HB2
3	BCH3	PB27	BCH2	PB20	BCH1	PB13	BCH0	PB6	HBCH0

Figure 34. Auxiliary Data Stream Signal

The figure below shows the relationship between the data bit-field and its clock cycle based on 1-, 2-, or 4-symbol per clock mode.



The data output at EOP contains the received BCH error correcting code. The sink core does not perform any error correction within the core. The auxiliary data is available outside the core.

Note: You can find the bit-field nomenclature in the *HDMI 2.0b Specification*.

6.1.5. Sink Auxiliary Packet Capture

To simplify user applications and minimize external logic, the core captures 3 different packet types and presents the packets outside the core.

These packets are: General Control Packet (GCP), Auxiliary Video Information (AVI) InfoFrame, and HDMI Vendor Specific InfoFrame (VSI).

The GCP, AVI and VSI bit-fields (excluding control bit) are defined in Table 24 on page 58. Table 25 on page 59. and Table 26 on page 60 respectively with reserved bits return 0.

6.1.6. Sink Auxiliary Data Port

The auxiliary port is attached to external memory. This port allows you to write packets to memory for use outside the HDMI core.

The core calculates the address for the data port using the header byte of the received packet. The core writes packet types 0–15 into a contiguous memory region.

Figure 35. Typical Application of AUX Packet Register Interface

The figure below shows a typical application of the auxiliary data port.

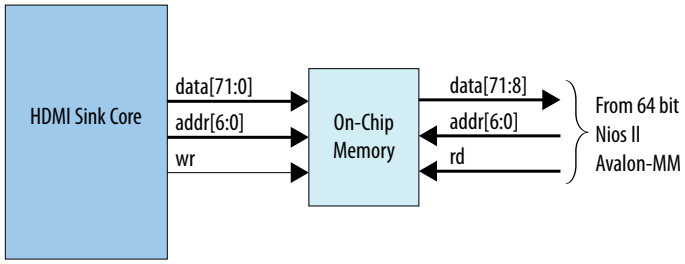


Table 43. Auxiliary Packet Memory Map

Memory Start Address	Packet Name
0	NULL PACKET
4	Audio Clock Regeneration (N/CTS)
8	Audio Sample
12	General Control
16	ACP Packet
20	ISRC1 Packet
24	ISRC2 Packet
28	One Bit Audio Sample Packet 5.3.9
32	DST Audio Packet
36	High Bitrate (HBR) Audio Stream Packet
40	Gamut Metadata Packet
44	3D Audio Sample Packet
48	One Bit 3D Audio Sample Packet
52	Audio Metadata Packet
56	Multi-Stream Audio Sample Packet
60	One Bit Multi-Stream Audio Sample Packet
64	Vendor-Specific InfoFrame
68	AVI InfoFrame
72	Source Product Descriptor InfoFrame
76	Audio InfoFrame
80	MPEG Source InfoFrame
84	TSC VBI InfoFrame
88	Dynamic Range and Mastering InfoFrame

Table 44. Packet Payload Data Byte

The table below lists the representation of each packet payload data byte.

Word Offset	Byte Offset								
	8	7	6	5	4	3	2	1	0
0	PB22	PB21	PB15	PB14	PB8	PB7	PB1	PB0	HB0
1	PB24	PB23	PB17	PB16	PB10	PB9	PB3	PB2	HB1
2	PB26	PB25	PB19	PB18	PB12	PB11	PB5	PB4	HB2
3	BCH3	PB27	BCH2	PB20	BCH1	PB13	BCH0	PB6	HBCH0

Note: The packet fields (PB0-PB26) are described in the HDMI 1.4b Specification (Chapter 8.2.1).

6.1.7. Sink Audio Decoder

The Audio Clock Regeneration packet transmits the CTS and N values required to synthesize the audio sample clock. The core also makes the CTS and N values available outside the core.

An audio clock synthesizer uses a phase-counter to recover the audio sample rate. The output from the audio clock synthesizer generates a valid pulse at the same rate as the audio sample clock from the attached source device. This valid pulse is available outside the core as an audio sample valid signal. This signal reads from a FIFO, which governs the rate of audio samples. The audio depacketizer drives the input to the FIFO.

The audio depacketizer extracts the 32-bit audio sample data from the incoming Audio Sample packets. The Audio Sample packets can hold from one to four sample data values. The audio format indicates the format of the received audio data as defined in Table 27 on page 61.

The Audio InfoFrame and Audio Metadata packets are not used within the core. The packets are captured and presented outside the core. The bit fields (excluding control bit) are defined in Table 28 on page 63, Table 29 on page 64, Table 30 on page 64, and Table 31 on page 65 with reserved bits return 0.

6.1.8. Status and Control Data Channel (SCDC) Interface

For applications using the HDMI 2.0b feature, the core provides a memory slave port to the SCDC registers.

This memory slave port connects to an I^2C slave component. The `TMDS_Bit_clock_Ratio` output from the SCDC interface indicates when the core requires the TMDS Bit Rate/TMDS Clock Rate ratio of 40. This bit is also stored in its corresponding field in the SCDC registers.

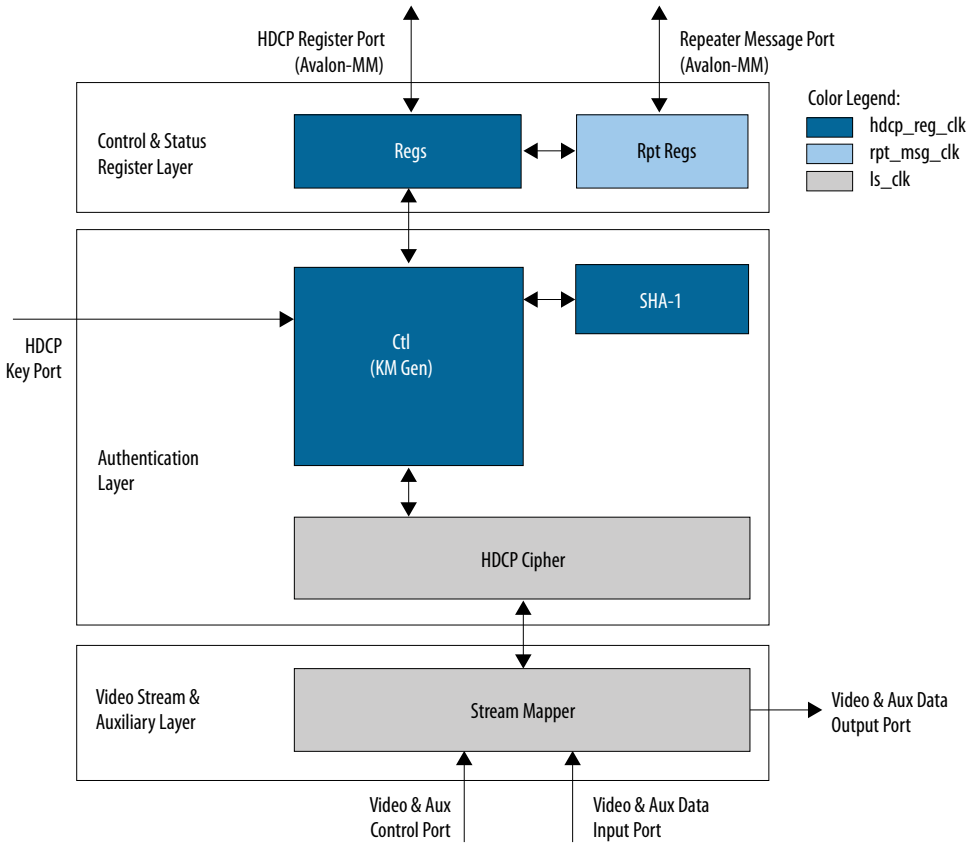
The *HDMI 2.0b Specification* requires the core to respond to the presence of the 5V input from the connector and the state of the HPD signal. The 5V input and HPD signal are used in the register mechanism updates. The signals are synchronous to the `scdc_i2c_clk` clock domain. You must create a 100-ms delay on the HPD signal externally to the core.

For more information about the Status and Control Data Channel, you may refer to *HDMI 2.0b Specification Chapter 10.4*. You can obtain the address map for the registers in the *HDMI 2.0b Specification*.

6.1.9. HDCP 1.4 RX Architecture

The HDCP 1.4 receiver block decrypts the protected video and auxiliary data from the connected HDCP 1.4 device. The HDCP 1.4 receiver block has identical structure layers as the HDCP 1.4 transmitter block.

Figure 36. Architecture Block Diagram of HDCP 1.4 RX IP



The HDCP 1.4 RX core is fully autonomous. For HDMI application, the transmitter drives the HDCP 1.4 RX core using the standard DDC interface supporting I^2C protocol. You need an I^2C slave externally to drive the IP through the HDCP Register Port (Avalon-MM).

The HDCP specifications requires the HDCP 2.3 RX core to be programmed with the DCP-issued production key – Device Private Keys (Bkeys) and Key Selection Vector (Bksv). The IP retrieves the key from the on-chip memory externally to the core through the HDCP Key Port. The on-chip memory must store the key data in the arrangement shown in the table below.

Table 45. HDCP 1.4 RX Key Port Addressing

Address	Content
6'h29	{16'd0, Bksv[39:0]}
6'h28	Bkeys39[55:0]
6'h27	Bkeys38[55:0]
...	...
6'h01	Bkeys01[55:0]
6'h00	Bkeys00[55:0]

The Video Stream and Auxiliary Layer receives audio and video content over its Video and Aux Data Input Port, and performs the decryption operation. The Video Stream and Auxiliary Layer detects the Encryption Status Signaling (ESS) provided by the HDMI IP to determine when to decrypt frames.

To implement the HDCP 1.4 RX core as a repeater upstream interface, the IP must propagate certain information such as KSV list and `Bstatus` to the upstream transmitter and to be used for SHA-1 hash digest. The repeater downstream interface (TX) must provide this information using the Repeater Message Port (Avalon-MM). You can use the same clock source to drive the clocking for the HDCP Register Port and Repeater Message Port.

The RX registers mapping defined in the following table is equivalent to the address space for HDCP 1.4 receiver defined in the HDCP specification.

Table 46. HDCP 1.4 RX Registers Mapping

Address	Register	R/W	Reset	Bit	Bit Name	Description
0x00	BKSV0	RO	-	7:0	-	Bit [7:0] of HDCP Receiver KSV.
0x01	BKSV1		-	7:0		Bit [15:8] of HDCP Receiver KSV.
0x02	BKSV2		-	7:0		Bit [23:16] of HDCP Receiver KSV.
0x03	BKSV3		-	7:0		Bit [31:24] of HDCP Receiver KSV.
0x04	BKSV4		-	7:0		Bit [39:32] of HDCP Receiver KSV.
0x05-0x07	Rsvd	RO	0x00	7:0	-	Reserved. All bytes read as 0x00.
0x08	RI_PRIME0	RO	0x00	7:0	-	Link verification response. Bit [7:0] of Ri'.
0x09	RI_PRIME1		0x00	7:0	-	Link verification response. Bit [15:8] of Ri'.
0x0A	PJ_PRIME	RO	0x00	7:0	-	Reserved. All bytes read as 0x00.
0x0B – 0x0F	Rsvd	RO	0x00	7:0	-	Reserved. All bytes read as 0x00.
0x10	AKSV0	WO	0x00	7:0	-	Bit [7:0] of HDCP Transmitter KSV.
0x11	AKSV1		0x00	7:0		Bit [15:8] of HDCP Transmitter KSV.

continued...

Address	Register	R/W	Reset	Bit	Bit Name	Description
0x12	AKSV2		0x00	7:0		Bit [23:16] of HDCP Transmitter KSV.
0x13	AKSV3		0x00	7:0		Bit [31:24] of HDCP Transmitter KSV.
0x14	AKSV4		0x00	7:0		Bit [39:32] of HDCP Transmitter KSV.
0x15	AINFO	WO	0x00	7:0	-	Reserved.
0x16 – 0x17	Rsvd	RO	0x00	7:0	-	Reserved. All bytes read as 0x00.
0x20	V_PRIME_H0_0	RO	0x00	7:0	-	H0 part of SHA-1 hash value used in the authentication protocol HDCP repeaters. Bit [7:0] of H0 value.
0x21	V_PRIME_H0_1		0x00	7:0		Bit [15:8] of H0 value.
0x22	V_PRIME_H0_2		0x00	7:0		Bit [23:16] of H0 value.
0x23	V_PRIME_H0_3		0x00	7:0		Bit [31:24] of H0 value.
0x24	V_PRIME_H1_0	RO	0x00	7:0	-	H1 part of SHA-1 hash value used in the authentication protocol HDCP repeaters. Bit [7:0] of H1 value.
0x25	V_PRIME_H1_1		0x00	7:0		Bit [15:8] of H1 value.
0x26	V_PRIME_H1_2		0x00	7:0		Bit [23:16] of H1 value.
0x27	V_PRIME_H1_3		0x00	7:0		Bit [31:24] of H1 value.
0x28	V_PRIME_H2_0	RO	0x00	7:0	-	H2 part of SHA-1 hash value used in the authentication protocol HDCP repeaters. Bit [7:0] of H2 value.
0x29	V_PRIME_H2_1		0x00	7:0		Bit [15:8] of H2 value.
0x2A	V_PRIME_H2_2		0x00	7:0		Bit [23:16] of H2 value.
0x2B	V_PRIME_H2_3		0x00	7:0		Bit [31:24] of H2 value.
0x2C	V_PRIME_H3_0	RO	0x00	7:0	-	H3 part of SHA-1 hash value used in the authentication protocol HDCP repeaters. Bit [7:0] of H3 value.
0x2D	V_PRIME_H3_1		0x00	7:0		Bit [15:8] of H3 value.
0x2E	V_PRIME_H3_2		0x00	7:0		Bit [23:16] of H3 value.
0x2F	V_PRIME_H3_3		0x00	7:0		Bit [31:24] of H3 value.
0x30	V_PRIME_H4_0	RO	0x00	7:0	-	H4 part of SHA-1 hash value used in the authentication protocol HDCP repeaters. Bit [7:0] of H4 value.
0x31	V_PRIME_H4_1		0x00	7:0		Bit [15:8] of H4 value.
0x32	V_PRIME_H4_2		0x00	7:0		Bit [23:16] of H4 value.
0x33	V_PRIME_H4_3		0x00	7:0		Bit [31:24] of H4 value.

continued...

Address	Register	R/W	Reset	Bit	Bit Name	Description
0x34 – 0x3F	Rsvd	RO	0x00	7:0	-	Reserved. All bytes read as 00.
0x40	BCAPS	RO	0x00	7	HDMI_RESERVED	0 = Receiver not capable of supporting HDMI 1 = Receiver capable of supporting HDMI
				6	REPEATER	HDCP repeater capability. 0 = Receiver is not a repeater. 1 = Receiver is a repeater.
				5	READY	KSV FIFO ready. When set to 1, the receiver has built the list of attached KSVs and computed the verification value V'. This value is always 0 during the computation of V'.
				4	FAST	This bit reads as 0.
				3:2	Reserved	These bits read as 0.
				1	FEATURES_1_1	Reserved. This bit reads as 0.
				0	FAST_REAUTHENTICATION	This bit reads as 1.
0x41	BSTATUS0	RO	0x00	7	MAX_DEVS_EXCEEDED	Topology error indicator. When set to 1, more than 127 downstream devices, or the capacity of the KSV FIFO, are attached.
				6:0	DEVICE_COUNT	Total number of attached downstream devices. Always 0 for HDCP Receivers. This count does not include the HDCP Repeater itself, but only downstream devices downstream from the HDCP Repeater.
0x42	BSTATUS1		0x00	7:6	Rsvd	These bits read as 0.
				5	HDMI_RESERVED_2	Reserved for future possible HDMI use.
				4	HDMI_MODE	HDMI mode. When set to 1, the HDCP Receiver has transitioned from DVI mode to HDMI mode.
				3	MAX_CASCADE_EXCEEDED	Topology error indicator. When set to 1, more than 7 levels of video repeater have been cascaded together.
				2:0	DEPTH	3-bit repeater cascade depth. This value gives the number of attached levels through the connection topology.

continued...

Address	Register	R/W	Reset	Bit	Bit Name	Description
0x43	KSV_FIFO	RO	0x00	7:0	-	Key selection vector FIFO. Used to pull downstream KSVs from HDCP Repeaters.
0x44 – 0xBF	Rsvd	RO	0x00	7:0	-	Reserved. All bytes read as 0x00.
0xC0 – 0x100	DBG	RW	0x00	7:0	-	Implementation-specific debug registers.

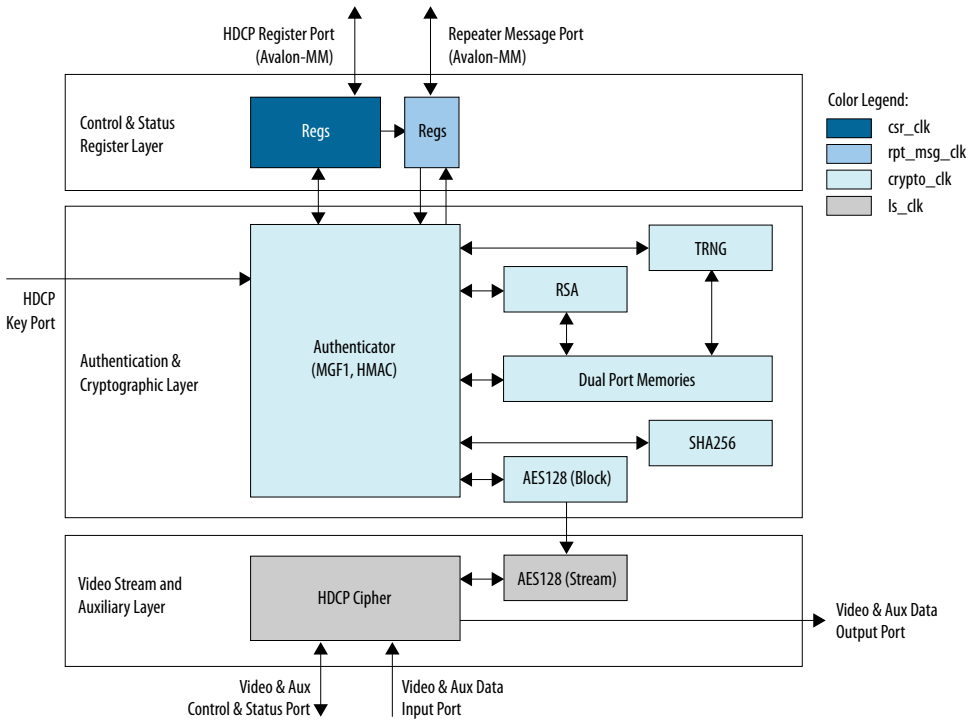
Table 47. HDCP 1.4 RX Repeater Registers Mapping

Address	Register	R/W	Reset	Bit	Bit Name	Description
0x00	RPT_KSV_LIST	WO	0x0	31:8	Reserved	Reserved
				7:0	KSV_LIST	Byte write KSV List in big endian order.
0x01	RPT_BSTATUS	RW	0x0	31:19	Reserved	Reserved
				18	REQUEST	Read-only. Asserted by the core to request for KSV_LIST and BSTATUS. This usually happens when re-authentication is triggered by the connected upstream. Note that when REQUEST is asserted, the READY should also be asserted.
				17	READY	Read-only. Asserted by the core to indicate KSV_LIST and BSTATUS are processed. Write KSV_LIST and BSTATUS after this bit is asserted.
				16	VALID	Set to 1 after KSV_LIST and BSTATUS are written. Self-cleared by the core after KSV_LIST and BSTATUS are read.
				15:0	BSTATUS	[15:12]: Reserved. [11]: MAX_CASCADE_EXCEEDED [10:8]: DEPTH [7]: MAX_DEVS_EXCEEDED [6:0]: DEVICE_COUNT
0x02	RPT_MISC	RW	-	31:1	Reserved	Reserved.
				0	REPEATER	Set to 0 if no downstream is connected or if the connected downstream is not HDCP 1.4-capable. This means the receiver IP core is an end-point receiver rather than a repeater. Set to 1 if the connected downstream is HDCP-capable.

6.1.10. HDCP 2.3 RX Architecture

The receiver block decrypts the protected video and auxiliary data from the connected HDCP 2.3 device. The HDCP 2.3 receiver block has identical structure layers as the HDCP 2.3 transmitter block.

Figure 37. Architecture Block Diagram of HDCP 2.3 RX IP



The HDCP 2.3 RX core is fully autonomous. For HDMI application, the transmitter drives the HDCP 2.3 RX core using the standard DDC interface supporting I^2C protocol.

The HDCP specifications requires the HDCP 2.3 RX core to be programmed with the DCP-issued production key – Global Constant (lc128), RSA private key (kprivrx) and RSA Public Key Certificate (certrx). The IP retrieves the key from the on-chip memory externally to the core through the HDCP Key Port. The on-chip memory must store the key data in the arrangement shown in the table below.

Table 48. HDCP 2.3 RX Key Port Addressing

Address	Content
8'hE3	lc128[127:96]
8'hE2	lc128[95:64]
8'hE1	lc128[63:32]
8'hE0	lc128[31:0]
8'hDF	kprivrx_p[511:480]
...	...
8'hD0	kprivrx_p[31:0]
8'hCF	kprivrx_q[511:480]
...	...
8'hC0	kprivrx_q[31:0]

continued...

Address	Content
8'hBF	kprivrx_dp[511:480]
...	...
8'hB0	kprivrx_dp[31:0]
8'hAF	kprivrx_dq[511:480]
...	...
8'hA0	kprivrx_dq[31:0]
8'h9F	kprivrx_qinv[511:480]
...	...
8'h90	kprivrx_qinv[31:0]
8'h83–8'h8F	Reserved
8'h82	{16'd0, certrx[4175:4160]}
8'h81	certrx[4159:4128]
...	...
8'h01	certrx[63:32]
8'h00	certrx[31:0]

The Video Stream and Auxiliary Layer receives audio and video content over its Video and Aux Data Input Port, and performs the decryption operation. The Video Stream and Auxiliary Layer detects the Encryption Status Signaling (ESS) provided by the HDMI IP to determine when to decrypt frames.

To implement the HDCP 2.3 RX core as a repeater upstream interface, the IP must propagate certain information such as `ReceiverID List` and `RxInfo` to the upstream transmitter and to be used for HMAC computation. The repeater downstream interface (TX) must provide this information using the Repeater Message Port (Avalon-MM). You can use the same clock source to drive the clocking for the HDCP Register Port and Repeater Message Port.

The RX registers mapping defined in the following table is equivalent to the address space for HDCP 2.3 receiver defined in the HDCP specification.

Table 49. HDCP 2.3 RX Registers Mapping

Address	Register	R/W	Reset	Bit	Bit Name	Description
0x44 – 0x4F	Rsvd	RO	0x00	7:0	Reserved	Reserved.
0x50	`HDCP2VERSION`	RO	0x04	7:3	Reserved	Reserved.
				2	HDCP22	When set to 1, the core supports HDCP 2.2 and above.
				1:0	Reserved	Reserved.
0x51 – 0x5F	Rsvd	RO	0x00	7:0	Reserved	Reserved.
0x60	`WRITE_MESSAGE`	WO	0x00	7:0	`WR_MSG`	Variable length message written by the transmitter as a single burst write to this address.

continued...

Address	Register	R/W	Reset	Bit	Bit Name	Description
0x61 – 0x6F	Rsvd	RO	0x00	7:0	Reserved	Reserved.
0x70	RXSTATUS0	RO	0x00	7:0	MSG_SIZE0	The lower part of message size in bytes available at the receiver for reading by the transmitter.
0x71	RXSTATUS1	RO	0x00	7:4	Reserved	Reserved
				3	REAUTH_REQ	When set to 1, indicates the link integrity check failure at the receiver (including upstream side of the repeater) or the upstream side of the repeater has transitioned into an unauthenticated state. Self-cleared by the core on every new authentication initiated by the AKE_Init message.
				2	READY	When set to 1, the repeater has built the list of downstream Receiver IDs and computed the verification value V'. Self-cleared by the core as soon as the RepeaterAuth_Send_ReceiverID_List message has been read by the transmitter or on every new authentication request by the transmitter.
				1:0	MSG_SIZE1	The upper part of message size in bytes available at the receiver for reading by the transmitter.
0x72 – 0x7F	Rsvd	RO	0x00	7:0	Reserved	Reserved.
0x80	READ_MESSAGE	RO	0x00	7:0	RD_MSG	Variable length message read by the transmitter as a single burst read from this address.
0x81 – 0xBF	Rsvd	RO	0x00	7:0	Reserved	Reserved.
0xC0 – 0xFF	DBG	RW	0x00	7:0	DBG_REGS	Implemented specific debug registers.

Table 50. HDCP 2.3 RX Repeater Registers Mapping

Address	Register	R/W	Reset	Bit	Bit Name	Description
0x00	RPT_RCVDID_LIST	WO	0x0	31:8	Reserved	Reserved
				7:0	RCVDID_LIST	Byte write ReceiverID_List in big endian order.
0x01	RPT_RXINFO	RW	0x0	31:19	Reserved	Reserved
				18	REQUEST	Read-only. Asserted by the core to request for RCVDID_LIST and RXINFO. This usually happens when re-authentication is triggered by the connected upstream. Note that when REQUEST is asserted, the READY should also be asserted.

continued...

Address	Register	R/W	Reset	Bit	Bit Name	Description
				17	READY	Read-only. Asserted by the core to indicate RCVDID_LIST and RXINFO are processed. Write RCVDID_LIST and RXINFO after this bit is asserted.
				16	VALID	Set to 1 after RCVDID_LIST and RXINFO are written. Self-cleared by the core after RCVDID_LIST and RXINFO are read.
				15:0	RXINFO	[15:12]: Reserved. [11:9]: DEPTH [8:4]: DEVICE_COUNT [3]: MAX_DEVS_EXCEEDED [2]: MAX_CASCADE_EXCEEDED [1]: HDCP2_REPEATER_DOWNSTREAM [0]: HDCP1_DEVICE_DOWNSTREAM
0x02	RPT_TYPE	RO	0x0	31:9	Reserved	Reserved
				8	VALID	Asserted by the core to indicate content stream TYPE is valid. Self-cleared by the core after TYPE is read.
				7:0	TYPE	0x00: Type 0 Content Stream 0x01: Type 1 Content Stream 0x02-0xFF: Reserved. Treated as Type 1 Content Stream.
0x03	RPT_MISC	RW	0x0	31:1	Reserved	Reserved.
				0	REPEATER	Set to 0 if no downstream is connected or if the connected downstream is not HDCP 2.3-capable. This means the receiver IP core is an end-point receiver rather than a repeater. Set to 1 if the connected downstream is HDCP- capable.

6.2. Sink Interfaces

The table lists the sink's port interfaces.

Table 51. Sink Interfaces

N is the number of symbols per clock.

Interface	Port Type	Clock Domain	Port	Direction	Description
Reset	Reset	–	reset	Input	Main asynchronous reset input. *Note:* Asserting the reset input resets the SCDC register.
Clock	Clock	–	ls_clk[2:0]	Input	Link speed clock input.

continued...

Interface	Port Type	Clock Domain	Port	Direction	Description
					These clocks correspond to the `in_r` (2), `in_g` (1), and `in_b` (0) TMDS encoded data inputs. Relationship to `vid_clk` as a function of color depth: • 8 bpc: 1x vid_clk • 10 bpc: 1.25x vid_clk • 12 bpc: 1.5x vid_clk • 16 bpc: 2x vid_clk This signal connects to the transceiver output clock only if TMDS Bit Rate is above the minimum transceiver data rate, which means no oversampling is required. This signal should connect to a PLL output clock that meets the `vid_clk` relationship if TMDS Bit Rate is below the minimum transceiver data rate, which means oversampling is required.
	Clock	–	`vid_clk`	Input	Video data clock input. For RGB and YCbCr 4:4:4/4:2:2 transport: • 1 symbol per clock mode = pixel clock • 2 symbols per clock mode = pixel clock/2 • 4 symbols per clock mode = pixel clock/4 For YCbCr 4:2:0 transport: • 1 symbol per clock mode = pixel clock/2 • 2 symbols per clock mode = pixel clock/4 • 4 symbols per clock mode = pixel clock/8
	Clock	–	`scdc_i2_clk`	Input	Avalon-MM SCDC Management Interface clock input.
Video Data Port	Conduit	`vid_clk`	`vid_data[N*48-1:0]`	Output	Video 48-bit pixel data output port. In 2 symbols per clock (N=2) mode, this port produces two 48-bit pixels per clock. In 4 symbols per clock (N=4) mode, this port produces four 48-bit pixels per clock.
	Conduit	`vid_clk`	`vid_de[N-1:0]`	Output	Video data enable output that indicates active picture region.

continued...

Interface	Port Type	Clock Domain	Port	Direction	Description
	Conduit	vid_clk	vid_hsync[N-1:0]	Output	Video horizontal sync output.
	Conduit	vid_clk	vid_vsync[N-1:0]	Output	Video vertical sync output.
	Conduit	vid_clk	locked[2:0]	Output	Indicates that the HDMI sink core is locked to the TMDS signals. Each bit represents a TMDS color channel. **Bit-Field** / **Channel** 0 / Blue (0) 1 / Green (1) 2 / Red (2)
	Conduit	vid_clk	vid_lock	Output	Asserted when the vid_de length is consistent for 3 frames. If the the vid_de length is inconsistent for 2 frames, this signal deasserts.
TMDS Data Port (5)	Conduit	ls_clk[0]	in_b[N*transceiver width-1:0]	Input	TMDS encoded blue channel (0) input.
	Conduit	ls_clk[1]	in_g[N*transceiver width-1:0]	Input	TMDS encoded green channel (1) input.
	Conduit	ls_clk[2]	in_r[N*transceiver width-1:0]	Input	TMDS encoded red channel (2) input.
	Conduit	ls_clk[2:0]	in_lock[2:0]	Input	Ready signal from the transceiver reset controller that indicates the transceivers are locked. Indicates the readiness of the up-stream block for the HDMI sink core to start operating. When this port is low, all blocks, except the SCDC registers and the character error detection block are hold in reset. Each bit represents a color channel.
Decoder Status Port	Conduit	ls_clk[0]	ctrl[N*6-1:0]	Output	DVI (mode = 0) status signals that overwrite the control and synchronization character in the green and red channels. **Bit-Field** / **Name**

continued...

(5) Connect to the transceiver data output if no oversampling is required. If oversampling is required, the port should connect to a DCFIFO and an oversampling user logic before connecting to a transceiver data output. Refer to Sink Clock Tree on page 107 for more information.

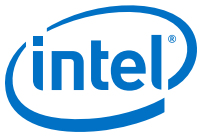
Interface	Port Type	Clock Domain	Port	Direction	Description	
					N*6+5	CTL3
					N*6+4	CTL2
					N*6+3	CTL1
					N*6+2	CTL0
					N*6+1	Reserved (0)
					N*6	Reserved (0)
					Refer to the *HDMI 1.4b Specification* for more information.	
	Conduit	ls_clk[0]	mode	Output	Indicates the encoding mode of the incoming TMDS signals. • 0: DVI • 1: HDMI	
SCDC Control Port	Conduit	scdc_i2c_clk	in_5v_power	Input	Detects the presence of 5V input voltage.	
	Conduit	scdc_i2c_clk	in_hpd	Input	Detects the Hot Plug Detect (HPD) status. This signal should be driven with the same signal to the HPD pin on the HDMI connector.	
	Conduit	scdc_i2c_clk	TMDS_Bit_clock_Ratio	Output	Indicates if the TMDS Bit Rate is greater than 3.4 Gbps • 0: (TMDS Bit Rate) / (TMDS Clock Rate) ratio is 10 • 1: (TMDS Bit Rate) / (TMDS Clock Rate) ratio is 40	
Avalon-MM SCDC Management Interface (6)	Avalon-MM	scdc_i2c_clk	scdc_i2c_addr[7:0]	Input	Address.	
	Avalon-MM	scdc_i2c_clk	scdc_i2c_r	Input	Assert to indicate a read transfer.	
	Avalon-MM	scdc_i2c_clk	scdc_i2c_rdata[7:0]	Output	Data driven from the core in response to a read transfer.	
	Avalon-MM	scdc_i2c_clk	scdc_i2c_w	Input	Assert to indicate a write transfer.	
	Avalon-MM	scdc_i2c_clk	scdc_i2c_wdata[7:0]	Input	Data for write transfers.	
Auxiliary Data Port (Applicable only when you enable **Support auxiliary**	Conduit	ls_clk[0]	aux_valid	Output	Auxiliary data channel valid output to qualify the data.	

continued...

(6) Refer to *HDMI 2.0b Specification Section 10.4* for address and data bit mapping.

Interface	Port Type	Clock Domain	Port	Direction	Description
parameter)	Conduit	ls_clk[0]	aux_data[71:0]	Output	Auxiliary data channel data output. For information about the bit-fields, refer to Figure 34 on page 89.
	Conduit	ls_clk[0]	aux_sop	Output	Auxiliary data channel start-of-packet output to mark the beginning of a packet.
	Conduit	ls_clk[0]	aux_eop	Output	Auxiliary data channel end-of-packet output to mark the end of a packet.
	Conduit	ls_clk[0]	aux_error	Output	Asserted when there is auxiliary data channel CRC error.
Auxiliary Status Port (Applicable only when you enable **Support auxiliary** parameter)	Conduit	ls_clk[0]	gcp[5:0]	Output	General Control Packet output. For information about the bit-fields, refer to Table 24 on page 58.
	Conduit	ls_clk[0]	info_avi[111:0]	Output	Auxiliary Video Information InfoFrame output. For information about the bit-fields, refer to Table 25 on page 59.
	Conduit	ls_clk[0]	info_vsi[60:0]	Output	Vendor Specific Information InfoFrame output. For information about the bit-fields, refer to Table 26 on page 60.
Auxiliary Memory Interface (Applicable only when you enable **Support auxiliary** parameter)	Conduit	ls_clk[0]	aux_pkt_addr[6:0]	Output	Auxiliary packet memory buffer address output.
	Conduit	ls_clk[0]	aux_pkt_data[71:0]	Output	Auxiliary packet memory buffer data output.
	Conduit	ls_clk[0]	aux_pkt_wr	Output	Auxiliary packet memory buffer write strobe output.
Audio Port (Applicable only when you enable **Support auxiliary** and **Support audio** parameters)	Conduit	ls_clk[0]	audio_CTS[19:0]	Output	Audio CTS value output.
	Conduit	ls_clk[0]	audio_N[19:0]	Output	Audio N value output.
	Conduit	ls_clk[0]	audio_data[255:0]	Output	Audio data output. For audio channel values, refer to Table 39 on page 81.
	Conduit	ls_clk[0]	audio_de	Output	Audio data valid output.
	Conduit	ls_clk[0]	audio_metadata[164:0]	Output	Additional information related to 3D audio and MST audio.

continued...



Interface	Port Type	Clock Domain	Port	Direction	Description
					For information about the bit-fields, refer to Table 29 on page 64, Table 30 on page 64, and Table 31 on page 65.
	Conduit	ls_clk[0]	audio_format[4:0]	Output	Indicates 3D audio status and the audio format detected. **Bit-Field** / **Description** 4 / The core asserts to indicate the first 8 channels of each 3D audio sample. 3:0 / For information about the bit-fields, refer to Table 27 on page 61.
	Conduit	ls_clk[0]	audio_info_ai[47:0]	Output	Audio InfoFrame output bundle. For information about the bit-fields, refer to Table 28 on page 63.
HDCP Port (Applicable only when you enable **Support HDCP 2.3** or **Support HDCP 1.4** parameters)	Reset	–	hdcp_reset	Input	Main asynchronous reset.
	Clock	–	hdcp_i2c_clk	Input	HDCP clock for Control and Status Registers (HDCP Registers). Typically, shares the I^2C slave clock (100 MHz).
		–	crypto_clk	Input	HDCP 2.3 clock for Authentication and Cryptographic Layer. You can use any clock with a frequency up to 200 MHz. *Note:* The clock frequency determines the authentication latency.
		–	rpt_msg_clk	Input	HDCP clock for the Repeater registers in the Control and Status Registers Layer. Typically, shares the clock (100 MHz) that drives the repeater downstream Nios II processor.

continued...

Interface	Port Type	Clock Domain	Port	Direction	Description
	Avalon-MM	hdcp_i2c_clk	hdcp_i2c_addr[7:0]	Input	The Avalon-MM slave port that provides access to HDCP registers. The I2C slave must drive this port.
			hdcp_i2c_wr	Input	
			hdcp_i2c_rd	Input	
			hdcp_i2c_wrdata[7:0]	Input	
			hdcp_i2c_rddata[7:0]	Output	
	Conduit	hdcp_i2c_clk	i2c_stop_det	Input	Assert this signal to indicate the stop condition for each I2C command.
	Avalon-MM	rpt_msg_clk	rpt_msg_addr[7:0]	Input	The Avalon-MM slave port that provides access to the Repeater registers, mainly for ReceiverID List and RxInfo. This interface is expected to operate at repeater downstream Nios II processor clock domain. Because of the extremely large bit portion of message, the IP transfers the message in burst mode with full handshaking mechanism. Write transfers always have a wait time of 0 cycle while read transfers have a wait time of 1 cycle. The addressing should be accessed as word addressing in the Platform Designer flow. For example, addressing of 4 in the Nios II software selects the address of 1 in the slave.
			rpt_msg_wr	Input	
			rpt_msg_rd	Input	
			rpt_msg_wrdata[31:0]	Input	
			rpt_msg_rddata[31:0]	Output	
	Conduit (Key)	crypto_clk (HDCP 2.3) hdcp_i2c_clk (HDCP 1.4)	kmem_wait[0] (HDCP 2.3) kmem_wait[1] (HDCP 1.4)	Input	Always keep this signal asserted until the key is ready to be read.
			kmem_rdaddr[7:0] (HDCP 2.3) kmem_rdaddr[13:8] (HDCP 1.4)	Output	Key read address bus.
			kmem_q[31:0] (HDCP 2.3) kmem_q[87:32] (HDCP 1.4)	Input	Key read address bus. Read transfer always have 1 cycle of wait time.

continued...

Interface	Port Type	Clock Domain	Port	Direction	Description
	Conduit	ls_clk	hdcp1_enabled	Output	This signal is asserted by the IP if the incoming video and auxiliary data are HDCP 1.4 encrypted.
			hdcp2_enabled	Output	This signal is asserted by the IP if the incoming video and auxiliary data are HDCP 2.3 encrypted.
			streamid_type	Output	• 0: The received stream is Type 0 • 1: The received stream is Type 1

6.3. Sink Clock Tree

The sink core uses various clocks.

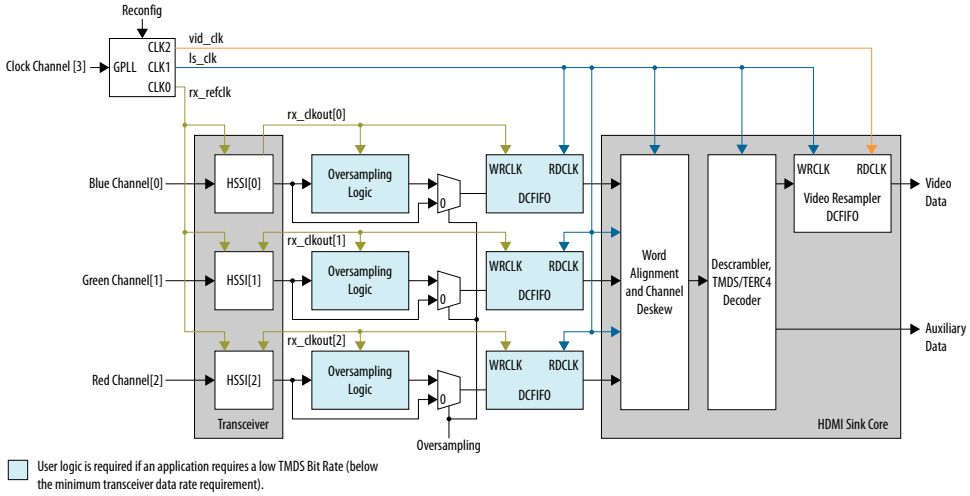
The logic clocks the transceiver data into the core using the three CDR clocks: (rx_clk[2:0]).

The TMDS and TERC4 decoding is done at the link-speed clock (ls_clk). The sink then resamples the pixel data and presents the data at the output of the core at the video pixel clock (vid_clk).

The pixel data clock depends on the video format used (within HDMI specification).

Figure 38. Sink Clock Tree

The figure shows how the different clocks connect in the sink core.



For HDMI sink, you must instantiate 3 receiver channels to receive TMS data.

The core uses a general purpose phase-locked loop (GPLL), that is referenced by the source Clock Channel, to generate the transceiver CDR reference clock (rx_refclk), link speed clock (ls_clk), and video clock (vid_clk) for the core.

Note: GPLL refers to IOPLL Intel FPGA IP for Intel Arria 10, Intel Cyclone 10 GX, and Intel Stratix 10 devices; PLL Intel FPGA IP for Arria V and Stratix V devices.

- The TMDS data clocks into the core at `ls_clk[2:0]` with all channels driven by the same clock source (`GPLL CLK1`).
- The video data clocks out from the core at `vid_clk`.

`rx_refclk`, `ls_clk`, and `vid_clk` are derived based on the color depth, TMDS_Bit_clock_Ratio, user oversampling control bit information, and the detected Clock Channel frequency band.

If an application requires low TMDS Bit Rate (below the transceiver minimum data rate requirement), then the application needs a user logic consisting of a DCFIFO and oversampling logic.

- The oversampling logic extracts the data from the oversampled incoming data stream.
- When you enable the oversampling control bit, the DCFIFO gets the TMDS data between the transceiver and the oversampling logic.
- The DCFIFO synchronizes the TMDS data from the fastest transceiver output clock (`rx_clkout[2:0]`) to the `ls_clk` domain.

If an application does not require low TMDS Bit Rate, the Clock Channel drives the transceiver `rx_refclk` directly. You can also connect the transceiver output to the core with `ls_clk[2:0]` driven by `rx_clkout[2:0]`.

Related Information

- HDMI Hardware Design Examples for Arria V and Stratix V Devices on page 30
- HDMI Hardware Design Examples for Intel Arria 10, Intel Cyclone 10 GX, and Intel Stratix 10 Devices on page 18

7. HDMI Parameters

Use the settings in the HDMI parameter editor to configure your design.

7.1. HDMI Source Parameters

Table 52. HDMI Source Parameters

Parameter	Value	Description
Device family	Intel Stratix 10 Intel Arria 10 Intel Cyclone 10 GX Arria V Stratix V	Targeted device family; matches the project device family.
Direction	Transmitter Receiver	Select HDMI transmitter.
Symbols per clock	1, 2, or 4 symbols per clock	Determines how many TMDS symbols and pixels are processed per clock. • Stratix V devices support 1 or 2 symbols per clock • Arria V devices support 1, 2, or 4 symbols per clock • Intel Arria 10, Intel Cyclone 10 GX, and Intel Stratix 10 devices support only 2 symbols per clock
Support auxiliary	On, Off	Determines if auxiliary channel encoding is included. This parameter is turned on by default.
Support deep color	On, Off	Determines if the core can encode deep color formats. This parameter is turned on by default.
Support audio	On, Off	Determines if the core can encode audio data. To enable this parameter, you must also enable the **Support auxiliary** parameter. This parameter is turned on by default.
Support HDCP 1.4	On, Off	Turn on to enable HDCP 1.4 TX support. This parameter can only be used when you enable 2 symbols per clock with Intel Arria 10 devices. *Note:* The HDCP-related parameters are not included in the 19.3 version of the Intel Quartus Prime Pro Edition software. To access the HDCP feature, contact Intel at https://www.intel.com/content/www/us/en/broadcast/products/programmable/applications/connectivity-solutions.html.
Support HDCP 2.3	On, Off	Turn on to enable HDCP 2.3 TX support. This parameter can only be used when you enable 2 symbols per clock with Intel Arria 10 devices.

continued...

Parameter	Value	Description
		Note: The HDCP-related parameters are not included in the 19.3 version of the Intel Quartus Prime Pro Edition software. To access the HDCP feature, contact Intel at https://www.intel.com/content/www/us/en/broadcast/products/programmable/applications/connectivity-solutions.html.

7.2. HDMI Sink Parameters

Table 53. HDMI Sink Parameters

Parameter	Value	Description
Device family	Intel Stratix 10 Intel Arria 10 Intel Cyclone 10 GX Arria V Stratix V	Targeted device family; matches the project device family.
Direction	Transmitter Receiver	Select HDMI receiver.
Symbols per clock	1, 2, or 4 symbols per clock	Determines how many TMDS symbols and pixels are processed per clock. • Stratix V devices support 1 or 2 symbols per clock • Arria V devices support 1, 2, or 4 symbols per clock • Intel Arria 10, Intel Cyclone 10 GX, and Intel Stratix 10 devices support only 2 symbols per clock
Support auxiliary	On. Off	Determines if auxiliary channel encoding is included. This parameter is turned on by default.
Support deep color	On. Off	Determines if the core can encode deep color formats. This parameter is turned on by default.
Support audio	On, Off	Determines if the core can encode audio data. To enable this parameter, you must also enable the **Support auxiliary** parameter. This parameter is turned on by default.
Support HDCP 1.4	On, Off	Turn on to enable HDCP 1.4 RX support. This parameter can only be used when you enable 2 symbols per clock with Intel Arria 10 devices. *Note:* The HDCP-related parameters are not included in the 19.3 version of the Intel Quartus Prime Pro Edition software. To access the HDCP feature, contact Intel at https://www.intel.com/content/www/us/en/broadcast/products/programmable/applications/connectivity-solutions.html.
Support HDCP 2.3	On, Off	Turn on to enable HDCP 2.3 RX support. This parameter can only be used when you enable 2 symbols per clock with Intel Arria 10 devices.

continued...

Parameter	Value	Description
		Note: The HDCP-related parameters are not included in the 19.3 version of the Intel Quartus Prime Pro Edition software. To access the HDCP feature, contact Intel at https://www.intel.com/content/www/us/en/broadcast/products/programmable/applications/connectivity-solutions.html.
Manufacturer OUI	—	The Manufacturer Organizationally Unique Identifier (OUI) assigned to the manufactured device to be written into the SCDC registers of address 0xD0, 0xD1, and 0xD2. Key in 3 byte hexadecimal data.
Device ID String	—	The Device Identification (ID) string to be written into the SCDC registers from addresses 0xD3 to 0xDa. Use this parameter to identify the sink device. You can key in up to eight ASCII characters. If you use less than eight characters, the unused bytes are set to 0x00.
Hardware Revision	—	Indicates the major and minor revisions of the hardware. Key in one byte of integer data. • Upper byte represents major revision. • Lower byte represents minor revision. The hardware major revision increments on a major silicon or board revision. The hardware minor revision increments on a minor silicon revision or minor board revision and resets to 0 when the major revision increments.

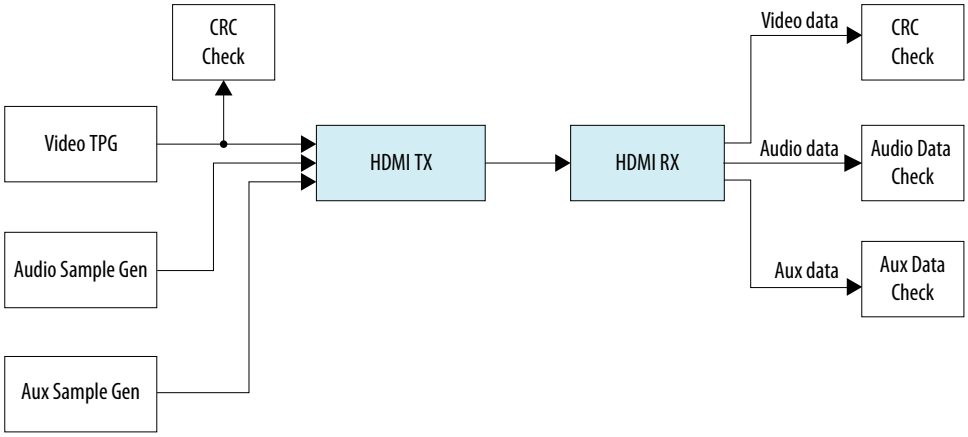
8. HDMI Simulation Example

The HDMI simulation example evaluates the functionality of the HDMI Intel FPGA IP core and provides a starting point for you to create your own simulation.

This simulation example targets the ModelSim - Intel FPGA Starter Edition simulator. The simulation covers the following core features:

- IEC-60958 audio format
- Standard H/V/DE/RGB input video format
- Support for 4 symbols per clock
- Support for HDMI 2.0b scrambled operation

Figure 39. HDMI Testbench



The Test Pattern Generator (TPG) provides the video stimulus. The IP core stimulates the HDMI TX core using an audio packet generator and aux packet generator. The output from the HDMI TX core drives the HDMI RX core.

The IP core requires a memory-mapped master stimulus to operate the testbench for HDMI 2.0b scrambling. This stimulus implements the activity normally seen across the I^2C DDC channel. At this point, the IP core asserts the scramble enable bit in the SCDC registers.

The testbench implements CRC checking on the input and output video. The testbench checks the CRC value of the transmitted data against the CRC calculated in the received video data. The testbench performs the checking after detecting 4 stable V-SYNC signals from the receiver.

The aux sample generator generates a fixed data to be transmitted from the transmitter. On the receiver side, the generator compares whether the expected aux data is received and decoded correctly.

The audio sample generator generates an incrementing test data pattern to be transmitted through the audio channel. On the receiver side, the audio data checker checks and compares whether the incrementing test data pattern is received and decoded correctly.

8.1. Simulation Walkthrough

Setting up and running the HDMI simulation example consists of two steps.

Note: This simulation flow applies only to Intel Quartus Prime Standard Edition using ModelSim - Intel FPGA Starter Edition. For Intel Quartus Prime Pro Edition flow, refer to the respective *Design Example User Guides*.

1. Copy the simulation files from `<IP root directory>/altera/altera_hdmi/sim_example` to your working directory.
2. Generate the IP simulation files and scripts, compile, and simulate.
   a. Start the Nios II Command Shell.
   b. Type the command below and enter.

```
sh runall.sh
```

This script executes the following commands:

Command	
Generate the simulation files for the HDMI cores.	• `ip-generate --project-directory=./ --component-file=./hdmi_rx_single.qsys --output-directory=./hdmi_rx_single/sim/ --file-set=SIM_VERILOG --report-file=sopcinfo:./hdmi_rx_single.sopcinfo --report-file=html:./hdmi_rx_single.html --report-file=spd:./hdmi_rx_single/sim/hdmi_rx_single.spd --report-file=qip:./hdmi_rx_single/sim/hdmi_rx_single.qip` • `ip-generate --project-directory=./ --component-file=./hdmi_rx_double.qsys --output-directory=./hdmi_rx_double/sim/ --file-set=SIM_VERILOG --report-file=sopcinfo:./hdmi_rx_double.sopcinfo --report-file=html:./hdmi_rx_double.html --report-file=spd:./hdmi_rx_double/sim/hdmi_rx_double.spd --report-file=qip:./hdmi_rx_double/sim/hdmi_rx_double.qip` • `ip-generate --project-directory=./ --component-file=./hdmi_tx_single.qsys --output-directory=./hdmi_tx_single/sim/ --file-set=SIM_VERILOG --report-file=sopcinfo:./hdmi_tx_single.sopcinfo --report-file=html:./hdmi_tx_single.html --report-file=spd:./hdmi_tx_single/sim/hdmi_tx_single.spd --report-file=qip:./hdmi_tx_single/sim/hdmi_tx_single.qip` • `ip-generate --project-directory=./ --component-file=./hdmi_tx_double.qsys --output-directory=./hdmi_tx_double/sim/ --file-set=SIM_VERILOG --report-file=sopcinfo:./hdmi_tx_double.sopcinfo --report-file=html:./hdmi_tx_double.html --report-file=spd:./hdmi_tx_double/sim/hdmi_tx_double.spd --report-file=qip:./hdmi_tx_double/sim/hdmi_tx_double.qip`
Merge the four resulting `msim_setup.tcl` scripts to create a single `mentor/msim_setup.tcl` script.	`ip-make-simscript --spd=./hdmi_tx_single/sim/hdmi_tx_single.spd --spd=./hdmi_tx_double/sim/hdmi_tx_double.spd --spd=./hdmi_rx_single/sim/hdmi_rx_single.spd --spd=./hdmi_rx_double/sim/hdmi_rx_double.spd`
Compile and simulate the design in the ModelSim software.	`vsim -c -do msim_hdmi.tcl`
Generate the simulation files for the HDMI cores.	
Merge the resulting `msim_setup.tcl` scripts to create a single `mentor/msim_setup.tcl` script.	
Compile and simulate the design in the ModelSim software.	

Example successful result:

```
# SYMBOLS_PER_CLOCK = 4
# VIC = 0
# AUDIO_CLK_DIVIDE = 800
# TEST_HDMI_6G = 1
# Simulation pass
# ** Note: $finish : bitec_hdmi_tb.v (647)
```

```
     Time: 15702552 ns Iteration: 3 Instance: /bitec_hdmi_tb
# End time: 14:39:02 on Feb 04,2016, Elapsed time: 0:03:17
# Errors: 0, Warnings: 134
```

9. HDMI Intel FPGA IP User Guide Archives

IP versions are the same as the Intel Quartus Prime Design Suite software versions up to 19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
19.1	HDMI Intel FPGA IP User Guide
18.1	HDMI Intel FPGA IP User Guide
18.0	HDMI Intel FPGA IP User Guide
17.1	HDMI IP Core User Guide
17.0	HDMI IP Core User Guide
16.1	HDMI IP Core User Guide
16.0	HDMI IP Core User Guide
15.1	HDMI IP Core User Guide
15.0	HDMI IP Core User Guide
14.1	HDMI IP Core User Guide

10. Document Revision History for the HDMI Intel FPGA IP User Guide

Document Version	Intel Quartus Prime Version	IP Version	Changes
2019.10.10	19.3	19.1.0	• Added a new section about High-bandwidth Digital Content Protection (HDCP). This feature is available only for Intel Arria 10 devices. *Note:* The HDCP feature is not included in the 19.3 version of the Intel Quartus Prime Pro Edition software. To access this feature, contact Intel at https://www.intel.com/content/www/us/en/broadcast/products/programmable/applications/connectivity-solutions.html. • Added information about the following HDCP-related parameters in the *HDMI Source Parameters* and *HDMI Sink Parameters* sections: — **Support HDCP 1.4** — **Support HDCP 2.3** • Added information about HDCP-related signals in the *Source Interfaces* and *Sink Interfaces* sections. • Added information about a new design example that demonstrates the HDCP feature for Intel Arria 10 devices in the Intel Quartus Prime Pro Edition software.
2019.04.29	19.1	–	• Added support for Intel Stratix 10 L-tile devices. Support for both Intel Stratix 10 L-tile and H-tile devices are final. • Updated the support for YCbCr 4:2:2 pixel encoding in the *Resource Utilization* section. The HDMI IP core supports 8-bit and 10-bit color depth for YCbCr 4:2:2 pixel encoding. • Added performance data for Intel Stratix 10 L-tile and H-tile devices, and updated the data for Intel Arria 10 and Intel Cyclone 10 GX devices for version 19.1. • Updated the description for the `locked[2:0]`, `in_lock[2:0]`, and `ctrl[N*6-1:0]` ports. • Added infomatition insertion and filtration for the control ports in the *Source Auxiliary Control Port* section.
2019.01.21	18.1	–	• Added a note in the *Sink Word Alignment and Channel Deskew* section that the word alignment logic in the HDMI RX core is disabled for HDMI 2.0 resolution (data rate >3.4 Gbps) in Intel Arria 10 and Intel Cyclone 10 GX devices. For Intel Stratix 10 devices, the HDMI RX core uses a new word alignment algorithm logic to achieve fast word alignment time for HDMI 2.0 resolution (data rate >3.4Gbps).

continued...

Document Version	Intel Quartus Prime Version	IP Version	Changes
			• Updated the description for the `vid_lock` port to add that the IP detects HTotal, VTotal, HSync Width, VSync Width, HSync Polarity, and VSync Polarity. and a change in these parameters across two frames will deassert the `vid_lock` signal.
2018.05.07	18.0	–	• Update the HDMI specification reference to 2.0b. The HDMI Intel FPGA IP core now supports *HDMI Specification 2.0b*. • Added preliminary support for Intel Stratix 10 (H-Tile) devices. • Updated support for Intel Cyclone 10 GX devices to final. • Clarified in the features list that HDMI IP core supports up to 32 channels in 2-channel or 8-channel layouts. • Added link to the *HDMI Intel Cyclone 10 GX FPGA IP Design Example User Guide*. • Updated all IP names as part of standardization and rebranding exercise. • Removed a note that said the HDMI RX core does not support SCDC read request feature for this release. The HDMI RX core fully supports SCDC features since version 17.1. • Added a note in the *Sink Clock Tree* section that GPLL refers to IOPLL Intel FPGA IP for Intel Arria 10, Intel Cyclone 10 GX, and Intel Stratix 10 devices; PLL Intel FPGA IP for Arria V and Stratix V devices. • Edited the recommended speed grade information forIntel Cyclone 10 GX. The recommended speed grade is -5. • Edited typo in *3D Audio Input Example* figure. • Changed the term *Video Format* to *Pixel Encoding* to be consistent with *HDMI Specification 2.0b*. • Restructured the document. Placed the *HDMI Hardware Design* chapter after the *HDMI Getting Started* chapter.

Date	Version	Changes
November 2017	2017.11.06	• Added advance support for Intel Cyclone 10 GX devices. • Added resource utilization data for Intel Cyclone 10 GX devices. • Changed *bits per color (bpc)* to *bits per component (bpc)* as stated in the *HDMI Specification 2.0*. • Renamed HDMI IP core to HDMI Intel FPGA as per Intel rebranding. • Changed the term Qsys to Platform Designer. • Reorganized and updated the *Source Functional Description* and *Source Functional Description* sections for better understanding. • Added description for the following new bit-fields: — Audio InfoFrame Bundle Bit-fields — Audio Metadata Bundle Bit-Fields for Packet Header and Control — Audio Metadata Bundle Bit-Fields for Packet Content When 3D_AUDIO = 1 — Audio Metadata Bundle Bit-Fields for Packet Content When 3D_AUDIO = 0 • Added support for up to 32 audio channels. • Added support for up to 1,536 kHz audio sample frequency.

continued...

Date	Version	Changes
		• Updated the *3D Audio Format* section and the description for `audio_clk` that for audio channels greater than 8, do not drive `audio_clk` at actual audio sample clock. Instead drive `audio_clk` with `ls_clk` and qualify `audio_data` with `audio_de` • Updated the *HDMI Intel FPGA Source Clock Tree* and *HDMI Intel FPGA Sink Clock Tree* sections. • Updated the *HDMI Intel FPGA Source Parameter* and *HDMI Intel FPGA Sink Parameter* sections. • Updated the *HDMI Intel FPGA Source Interfaces* and *HDMI Intel FPGA Sink Interfaces* sections. • Updated the description for the **Support for deep color** parameter. The parameter is now turned on by default. • Edited the HDMI Intel FPGA testbench block diagram. Removed 4 symbols/clock to avoid confusion. • Added a note in the *HDMI Intel FPGA Hardware Demonstration* section that the demonstration is only applicable for Arria V and Stratix V devices. For Intel Arria 10 devices, refer to the *HDMI Intel FPGA Design Example User Guide for Intel Arria 10 Devices*. • Added a note in the *Simulation Walkthrough* section that the walkthrough is only applicable for Intel Quartus Prime Standard Edition. For Intel Quartus Prime Pro Edition, refer to the *HDMI Intel FPGA Design Example User Guide for Intel Arria 10 Devices*. • Moved information about the HDMI Intel FPGA design example parameters to the *HDMI Intel FPGA Design Example User Guide for Intel Arria 10 Devices*.
May 2017	2017.05.08	• Rebranded as Intel. • Added recommended speed grades for Intel Arria 10 devices.
December 2016	2016.12.20	• Updated the HDMI IP core resource utilization table with 16.1 information. • Added a note for YCbCr 4:2:2 video format that 8 and 10 bits per color use the same pixel encoding as 12 bits per color, but the valid bits are left-justified with zeros padding the bits below the least significant bit. • Added information for the new Design Example parameters. • Removed all Arria 10 design example related information. For more information about Arria 10 design examples, refer to the *HDMI IP Core Design Example User Guide*. • Edited the typos in the HDMI Audio Format topic. • Added information that the HDMI IP core does not support 8-channel audio. • Added a new output port `version[31:0]` for HDMI source and sink.
May 2016	2016.05.02	• Updated the HDMI IP core resource utilization table with 16.0 information. • Added information about Audio Metadata Packet for *HDMI Specification Version 2.0*. • Added information about new HDMI source ports: — `audio_metadata[164:0]` — `audio_format[4:0]` • Added information about new HDMI sink ports: — `audio_metadata[164:0]` — `audio_format[4:0]` — `vid_lock` — `aux_error` • Provided detailed information about the HDMI source and sink `audio_de[7:0]` port. • Updated the testbench diagram and description to include audio data and auxiliary data information.

continued...

Date	Version	Changes
		• Added a note for Altera PLL to place the PLL in the transmit path (`pll_hdmi_tx`) in the physical location next to the transceiver PLL. • Updated the HDMI sideband signals (HDMI AVI and VSI bit-fields) with default values. • Added links to archived versions of the *HDMI IP Core User Guide*.
November 2015	2015.11.02	• Updated the HDMI IP core resource utilization table with 15.1 information. • Changed instances of *Quartus II* to *Intel Quartus Prime*. • Added full support for Arria 10 devices. • Added support for new features: — Deep color — 8-channel audio • Added the following parameters for HDMI source: — **Support for 8-channel audio** — **Support for deep color** • Added the following parameters for HDMI sink: — **Support for 8-channel audio** — **Support for deep color** — **Manufacturer OUI** — **Device ID String** — **Hardware Revision** • Updated the following interface ports for HDMI source: — Added `ctrl` port — Removed `gcp_Set_AVMute` and `gcp_Clear_AVMute` ports • Updated the following interface ports for HDMI sink: — Added `ctrl` , `mode`, `in_5v_power`, and `in_hpd` ports — Removed `gcp_Set_AVMute` and `gcp_Clear_AVMute` ports • Updated the HDMI sink and source block diagrams to reflect the new features. • Provided block diagrams for deep color mapping. • Generalized the HDMI hardware demonstration design for all supported device families (Arria V, Stratix V, and Arria 10) with detailed description.
May 2015	2015.05.04	• Updated the HDMI IP core resource utilization table with 15.0 information. • Added information about 4 symbols per clock mode. • Added information about Status and Control Data Channel (SCDC) for *HDMI specification version 2.0*. • Added the following interface ports for HDMI source: — `TMDS_Bit_clock_Ratio` — `Scrambler_Enable` • Added the `TMDS_Bit_clock_Ratio` interface port for HDMI sink. • Updated the HDMI hardware demonstration design with HDMI 2.0 information. • Added software process flow for the HDMI hardware demonstration.
December 2014	2014.12.15	Initial release.