



XAPP1287 (v1.2) July 18, 2016

HDMI 2.0 Implementation on Kintex-7 FPGA GTX Transceivers

Author: Gilbert Magnaye, Yunhai Qiao, Mujib Haider, Marco Groeneveld

Summary

This application note covers the design considerations of a High-Definition Multimedia Interface (HDMI™) 2.0 implementation on the Kintex®-7 FPGA GTX transceiver using the performance features of the following Xilinx® LogiCORE™ IP cores:

- HDMI 1.4/2.0 with HDCP 1.4/2.2 Transmitter Subsystem
- HDMI 1.4/2.0 with HDCP 1.4/2.2 Receiver Subsystem
- Video PHY Controller

The design features the transmit-only and the pass-through operation modes for the HDMI solution. In the transmit-only mode, the design displays a colorbar pattern from the LogiCORE IP Test Pattern Generator (TPG) core. In the pass-through mode, an external HDMI source is used to send video data over the HDMI design. The reference design demonstrates the use of the High-bandwidth Digital Content Protection System (HDCP) Revision 1.4/2.2 capability of the HDMI solution. HDCP is used to securely send audiovisual data from an HDCP protected transmitter to HDCP protected downstream receivers. Typically, HDCP 2.2 is used to encrypt content at Ultra High Definition (UHD) while HDCP 1.4 is used as a legacy encryption scheme for lower resolutions.

The reference design targets the Xilinx Kintex®-7 FPGA KC705 evaluation kit, which uses the Kintex-7 XC7K325T-2FFG900 FPGA and the inrevium TB-FMCH-HDMI4K FMC card.

Download the [Reference design files](#) for this application note from the Xilinx website.

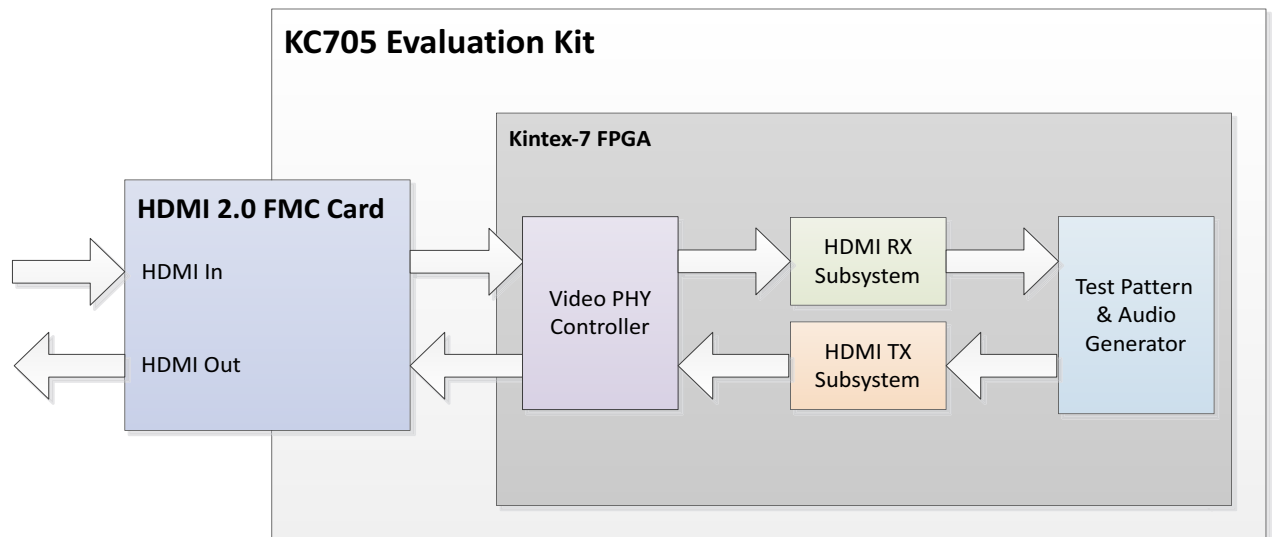
Reference Design

The reference design was created and built using the Vivado® Design Suite, System Edition 2016.2. The design also includes software built using the Xilinx Software Development Kit (SDK) 2016.2. The software runs on the MicroBlaze™ processor subsystem and implements control and status functions. Complete project files for the Vivado Design Suite and the SDK are provided with this application note to allow the examination and rebuilding of the design, or to use them as a template for starting a new design.

Hardware Requirements

Introduction

The reference design is built around the HDMI 1.4/2.0 Transmitter Subsystem (HDMI_TX_SS), HDMI 1.4/2.0 Receiver Subsystem (HDMI_RX_SS), and Video PHY (VPHY) Controller cores and leverages existing Xilinx IP cores to form the complete system. HDCP 1.4 and HDCP 2.2 support is enabled in the HDMI Subsystems to allow content protection. The input and output of the system are HDMI video streams through an HDMI 2.0 daughter card that connects to the FMC HPC connector of the Xilinx development board.



X16848-042316

Figure 1: KC705 HDMI Reference Design Block Diagram

The VPHY Controller core is configured for the HDMI application that allows transmission and reception of HDMI video/audio to and from the HDMI 2.0 daughter card.

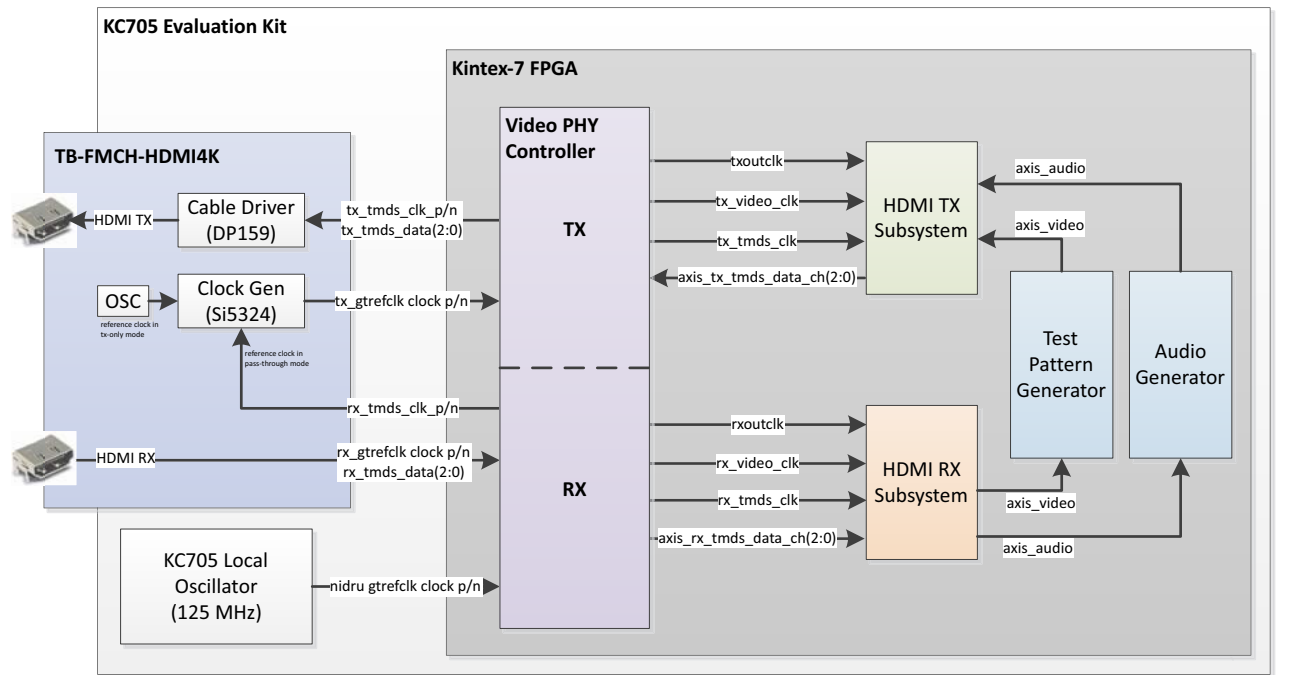


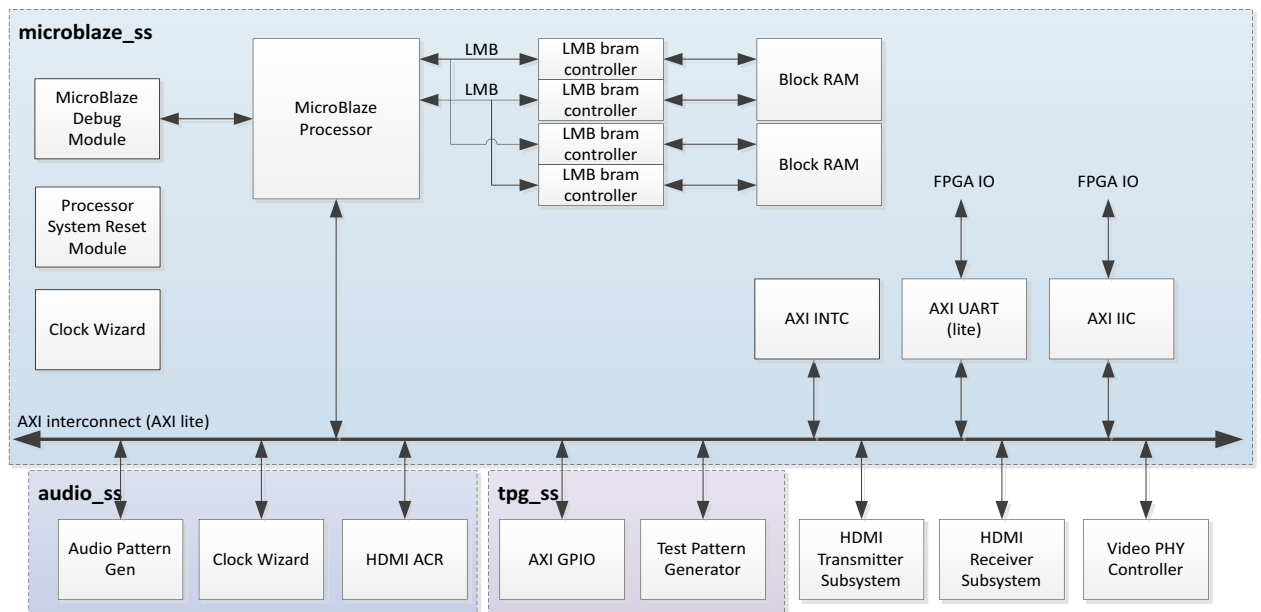
Figure 2: KC705 HDMI Reference Design Clock & Datapath Diagram

In pass-through mode, the VPHY Controller core recovers the high-speed serial video stream, converts it to parallel data streams, and forwards it to the HDMI_RX_SS core. The HDMI_RX_SS core extracts the video and audio streams from the HDMI stream and converts it to separate AXI video and audio streams. The AXI video goes through the TPG core and the AXI audio goes through a customized audio generation block. The two AXI streams eventually reach the HDMI_TX_SS core, which converts the AXI video and audio streams back to an HDMI stream before being transmitted by the VPHY Controller core as a high-speed serial data stream. The Transition Minimized Differential Signaling (TMDS) clock from the HDMI In interface is forwarded to the HDMI TX transceiver using SI5324 clock generator in the HDMI 2.0 FMC card.

In transmit only mode, the colorbar pattern is generated by the TPG core as an AXI video stream and the low frequency audio is generated by the customized audio processing block as an AXI audio stream. The two streams are forwarded to the HDMI_TX_SS for HDMI stream conversion and then to the VPHY for transmission.

High-level control of the system is provided by a simplified MicroBlaze embedded processor subsystem containing I/O peripherals, processor support IP, clock generator block, processor system reset block supply clock and reset signals for the system respectively.

Figure 3 and Table 1 displays a block diagram of the MicroBlaze processor subsystem and its address map.



X16849-0423

Figure 3: HDMI Reference Design Block Diagram

Table 1 shows the MicroBlaze subsystem address map.

Table 1: MicroBlaze Subsystem Address Map

Peripheral	Instance	Base Address	High Address
lmb_bram_if_cntlr	ilmb_bram_if_cntlr_0	0x00000000	0x0007FFFF
lmb_bram_if_cntlr	dlmb_bram_if_cntlr_0	0x00000000	0x0007FFFF
lmb_bram_if_cntlr	ilmb_bram_if_cntlr_1	0x00080000	0x0009FFFF
lmb_bram_if_cntlr	dlmb_bram_if_cntlr_1	0x00080000	0x0009FFFF
axi_intc	axi_intc_0	0x41200000	0x4120FFFF
axi_uartlite	axi_uartlite_0	0x40600000	0x4060FFFF
axi_iic	fmch_axi_iic_0	0x40800000	0x4080FFFF
aud_pat_gen	aud_pat_gen_0	0x44A60000	0x44A6FFFF
clk_wiz	clk_wiz_0	0x44A50000	0x44A5FFFF
hdmi_acr_ctrl	hdmi_acr_ctrl_0	0x44A70000	0x44A7FFFF
axi_gpio	axi_gpio_0	0x44A90000	0x44A9FFFF
v_tpg	v_tpg_0	0x44A00000	0x44A0FFFF
v_hdmi_rx_ss	v_hdmi_rx_ss_0	0x44B00000	0x44B3FFFF
v_hdmi_tx_ss	v_hdmi_tx_ss_0	0x44C00000	0x44C3FFFF
vid_phy_controller	vid_phy_controller_0	0x44A10000	0x44A1FFFF

Reference Design Specifications

In addition to the VPHY Controller, HDMI Transmitter Subsystem and HDMI Receiver Subsystem core, the reference design includes the following cores:

- MicroBlaze
- MicroBlaze Debug Module
- AXI Interconnect
- Local Memory Bus
- LMB Block RAM Controller
- Block Memory Generator
- Clocking Wizard
- Processor System Reset
- AXI UARTLite
- AXI Interrupt Controller
- AXI IIC
- AXI GPIO
- Video Test Pattern Generator
- Concat
- AXI4-Stream Register Slice
- Utility Buffer
- Utility Vector Logic
- AUD_PAT_GEN (Custom IP)
- HDMI_ACR_CTRL (Custom IP)
- ODDR (Custom IP)
- HDMI_HB (Custom IP)

Hardware System Specifications

This section describes the high-level features of the reference design including the usage of main IP blocks.

Video PHY(VPHY) Controller

The VPHY Controller core enables plug-and-play connectivity with HDMI TX and RX (Video MAC) subsystem IPs through a standardized interface. An AXI4-Lite register interface enables dynamic access to the transceiver and VPHY Controller core peripherals such as the clock detector and non-integer data recovery unit (NI-DRU).

The VPHY Controller core performs the following tasks:

- Enables the physical transmission and reception of high-speed serial data.
- Generates the clocks (link clock and video clock) required by the HDMI_TX_SS and HDMI_RX_SS cores. The respective clocks are generated as specified in the Clocking section of the HDMI 1.4/2.0 Transmitter Subsystem LogiCORE IP Product Guide (PG235) [Ref 3] and the HDMI 1.4/2.0 Receiver Subsystem LogiCORE IP Product Guide (PG236) [Ref 4].
- Generates the differential HDMI TX interface clock.
- Re-transmits the differential HDMI RX TMDS interface clock for clock forwarding in pass-through mode.
- Provides status sideband signals to the HDMI_TX_SS (vid_phy_status_sb_tx interface) and the HDMI_RX_SS (vid_phy_status_sb_rx interface) to enable the internal state machines. There are two components in the sideband signals:
 - vid_phy_status_sb_rx_tdata[0] - link clock lock indicator
 - vid_phy_status_sb_rx_tdata[1] - video clock lock indicator
- Detects the actual frequency of the TX and RX TMDS clocks through the clock detector peripheral.
- Uses NI-DRU to recover low HDMI line rates with the TMDS clock below the minimum reference clock that the receive phase-locked loop (PLL) can support.

For more information about the VPHY Controller core, see the Video PHY Controller LogiCORE IP Product Guide (PG230)[Ref 5].

7 Series GTX VPHY Implementation Specifications

The GTX transceiver in Xilinx 7 Series FPGAs has two types of PLLs, the QPLL and the CPLL. The QPLL is shared by all the four transceivers in the Quad. Each transceiver has its own CPLL. The VPHY IP allows you to choose whether the QPLL or the CPLL is used by the transmitter. The receiver should use the other PLL that is not used by the TX.

The GTX transceivers can use either the Quad PLL (QPLL) or the Channel PLL (CPLL) as the clock source for the RX and the TX. The RX and the TX can use the same PLL or different PLLs.

If both RX and TX use the same PLL, CPLL or QPLL, then they are bonded and must always run at exactly the same line rate.

Both the QPLL and CPLL have certain limitations. The QPLL has holes. Holes are certain line rates that QPLL cannot support due to the limited frequency range of its voltage controlled oscillator (VCO). However, the QPLL can support lower TMDS clock frequencies than the CPLL.

The CPLL's VCO must run in the range of 1.6 GHz to 3.3 GHz. The VCO frequency is dependent upon the TMDS clock frequency. The CPLL can apply a limited set of multipliers to the TMDS clock frequency. The GT driver measures the TMDS clock frequency and attempts to find a valid multiplier. This will result in a VCO frequency that is within the allowed range. As the largest multiplier that can be applied by the CPLL is 20, the minimum TMDS clock frequency that can be supported by the CPLL is 80 MHz. Video formats that have a TMDS clock frequency of less

than 80 MHz are not supported by the CPLL. When the GT driver detects that the TMDS clock frequency is less than 80 MHz, it enables the NI-DRU to receive these low bit rates that are less than 0.8 Gb/s. The NI-DRU runs at 2.0 Gb/s, which enables it to recover line rates that cannot be supported by the CPLL. For TMDS clock frequencies greater than 80 MHz, a multiplier of 10X or 20X is applied to keep the VCO frequency in a proper range as shown in [Table 2](#).

Table 2: 7 Series GTX CPLL Usage

TMDS Clock Frequency (MHz)	CPLL Refclk Divider	CPLL Multiplier	VCO Frequency	Notes
<80	N/A	N/A	N/A	TX: Oversampling RX: NI-DRU is used
80 to 165	1	20	1.6 to 3.3 GHz	CDR is used
165 to 330	1	10	1.65 to 3.3 GHz	CDR is used



IMPORTANT: *The HDMI RX Subsystem can receive the most valid video format using the CPLL, the NI-DRU or the native CDR, up to a maximum line rate of 6 Gb/s.*

The CPLL can support all video formats. It has no holes. However, the CPLL does require the use of the NI-DRU to cover any format that has a TMDS clock below 80 MHz. [Table 3](#) shows the standard formats with or without the DRU requirements that are supported by the CPLL.

Table 3: CPLL Support of RGB and YCbCr 4:4:4 Video Formats

Resolution	Bits Per Pixel			
	24	30	36	48
480i60	DRU	DRU	DRU	DRU
576i50	DRU	DRU	DRU	DRU
1080i50	DRU	✓	✓	✓
1080i60	DRU	✓	✓	✓
480p60	DRU	DRU	DRU	DRU
576p50	DRU	DRU	DRU	DRU
720p50	DRU	✓	✓	✓
720p60	DRU	✓	✓	✓
1080p24	✓	✓	✓	✓
1080p25	✓	✓	✓	✓
1080p30	✓	✓	✓	✓
1080p50	✓	✓	✓	✓
1080p60	✓	✓	✓	✓
2160p24	✓	✓	✓	✓
2160p25	✓	✓	✓	✓
2160p30	✓	✓	✓	✓
2160p60	✓	1 ⁽¹⁾	1 ⁽¹⁾	1 ⁽¹⁾
vgap60	DRU	DRU	DRU	DRU

Table 3: CPLL Support of RGB and YCbCr 4:4:4 Video Formats (Cont'd)

Resolution	Bits Per Pixel			
	24	30	36	48
svgap60	DRU	DRU	DRU	✓
xgap60	DRU	✓	✓	✓
sxgap60	✓	✓	✓	✓
wxgap60	DRU	✓	✓	✓
wxga+p60	✓	✓	✓	✓
uxgap60	✓	✓	✓	2 ⁽²⁾
wuxgap60	✓	✓	✓	✓
wsxgap60	✓	✓	✓	✓

Notes:

1. This format is not supported because it exceeds the maximum line rate of HDMI 2.0.
2. This format is supported for transmit, but is not currently supported by the receiver.

When the QPLL is used as the clock source for either the RX or TX, there are combinations of resolutions, color depth, and color space that cannot be supported as QPLL cannot generate the necessary clock frequencies to support those video formats. HDMI uses the lower band of QPLL. The VCO of the QPLL must run in the frequency range of 5.93 GHz to 8.0 GHz. The QPLL can apply multipliers of 20, 40, or 80 to the TMDS clock.

Table 4 displays the TMDS clock frequency interaction with the QPLL and the frequency ranges that can be supported.

Table 4: 7 Series GTX QPLL Usage

TMDS Clock Frequency (MHz)	QPLL Multiplier	Notes
<74.125	N/A	TX: Oversampling RX: NI-DRU is used
74.125 to 100	80	Supported
100 to 148.25	-	TMDS clock range cannot be supported
148.25 to 200	40	Supported
200 to 296.5	-	TMDS clock range cannot be supported
296.5 to 400	20	Supported

Note that the QPLL has two operating bands. For HDMI, the QPLL is always used in lower band since the upper band is not available in all the speed grades.

Table 5 shows the standard RGB and YCbCr 4:4:4 video formats that are supported by QPLL and the formats that are not supported by QPLL. Formats shown with a check mark are supported by the QPLL. Formats shown in pink are not supported because they fall into QPLL holes. Formats that require the DRU to receive them are so noted in the table.

Table 5: QPLL Support of RGB and YCbCr 4:4:4 Video Formats

Resolution	Bits Per Pixel			
	24	30	36	48
480i60	DRU	DRU	DRU	DRU
576i50	DRU	DRU	DRU	DRU
1080i50	✓	✓		✓
1080i60	✓	✓		✓
480p60	DRU	DRU	DRU	DRU
576p50	DRU	DRU	DRU	DRU
720p50	✓	✓		✓
720p60	✓	✓		✓
1080p24	✓	✓		✓
1080p25	✓	✓		✓
1080p30	✓	✓		✓
1080p50	✓	✓		✓
1080p60	✓	✓		✓
2160p24	✓	✓		✓
2160p25	✓	✓		✓
2160p30	✓	✓		✓
2160p60	✓	1 ⁽¹⁾	1 ⁽¹⁾	1 ⁽¹⁾
vgap60	DRU	DRU	DRU	DRU
svgap60	DRU	DRU	DRU	DRU
xgap60	DRU	✓	✓	
sxgap60			✓	
wxgap60	DRU	✓		
wxga+p60	✓			✓
uxgap60	✓			2 ⁽²⁾
wuxgap60	✓			✓
wsxgap60		✓	✓	

Notes:

1. This format is not supported because it exceeds the maximum line rate of HDMI 2.0.
2. This format is supported for transmit, but is not currently supported by the receiver.

HDMI Transmitter Subsystem

The HDMI_TX_SS includes the following AXI interfaces:

- Video

The video interface is an AXI4-Stream slave bus. It carries single, dual, or quad pixels per clock and it supports 8, 10, 12, and 16-bits per component. It uses the video protocol as defined in the Video IP: AXI Feature Adoption section of the AXI Reference Guide (UG761) [Ref 7].

- Audio

The audio interface is a 32-bit AXI4-Stream slave bus. The subsystem transports multiple channels (up to 8 channels) of uncompressed audio data.

- CPU

The CPU interface is an AXI4-Lite bus interface, which allows a processor, such as MicroBlaze, to control the subsystem by accessing its registers. This AXI4-Lite slave interface supports single beat read and write data transfers (no burst transfers).

- VPHY Controller core status sideband

The VPHY Controller core status sideband interface is a 2-bit AXI4-Stream slave bus that carries the link clock lock (bit 0) and video clock lock (bit 1) status bits.

- Three HDMI streams

The HDMI stream interface is a 40-bit AXI4-Stream master bus that is connected to the VPHY Controller core transmission.

The HDMI_TX_SS internals are fixed and cannot be modified or altered. There are two user configurable parameters that affect its internals. These parameters are the number of pixels per clock on the AXI4-Stream output and the maximum bits per component. In the reference design, these parameters are set to 2 and 12, respectively.

The HDMI_TX_SS converts the video stream and audio stream into an HDMI stream, based on the selected video format set by the processor core through the CPU interface. The subsystem then transmits the HDMI stream to the VPHY Controller core, which converts the data into electronic signals that are sent to an HDMI sink through an HDMI cable.

The HDMI_TX_SS can enable HDCP 1.4, HDCP2.2, or both protocols as part of the subsystem. The reference design instantiates both HDCP protocols and allows you to select the active protocol. When HDCP is enabled the transmitter is able to encrypt video, audio, and auxiliary packets.

For more information, see the HDMI 1.4/2.0 Transmitter Subsystem LogiCORE IP Product Guide (PG235) [Ref 3].



IMPORTANT: *HDMI 1.4/2.0 Transmitter Subsystem is configurable to support native video interface. This configuration is not covered by this application note. Refer to HDMI 1.4/2.0 Transmitter Subsystem LogiCORE IP Product Guide (PG235) [Ref 3] for more details.*

HDMI Receiver Subsystem

The HDMI_RX_SS has these AXI interfaces:

- Video

The video interface is an AXI4-Stream master bus. It carries single, dual, or quad pixels per clock and it supports 8, 10, 12, and 16-bits per component. It uses the video protocol as defined in the Video IP: AXI Feature Adoption section of the AXI Reference Guide (UG761) [Ref 7].

- Audio

The audio interface is a 32-bit AXI4-Stream master bus. The subsystem transports multiple channels (up to 8 channels) of uncompressed audio data.

- CPU

The CPU interface is an AXI4-Lite bus interface, which allows a processor, such as MicroBlaze, to control the subsystem by accessing its registers. This AXI4-Lite slave interface supports single beat read and write data transfers (no burst transfers).

- VPHY Controller core status sideband

The VPHY Controller core status sideband interface is a 2-bit AXI4-Stream slave bus that carries the link clock lock (bit 0) and video clock lock (bit 1) status bits.



IMPORTANT: *HDMI 1.4/2.0 Receiver Subsystem is configurable to support native video interface. This configuration is not covered by this application note. Refer to HDMI 1.4/2.0 Receiver Subsystem LogiCORE IP Product Guide (PG236) [Ref 4] for more details.*

- Three HDMI streams

The HDMI stream interface is a 40-bit AXI4-Stream slave bus that carries the recovered data from the VPHY Controller core.

The HDMI_RX_SS internals are fixed and cannot be modified or altered. There are three user configurable parameters that affect its internals. These parameters are the number of pixels per clock on the AXI4-Stream output, max bits per component, and the EDID RAM size. In the reference design, these parameters are set to 2, 12, and 256 respectively.

The HDMI_RX_SS extracts the video stream and audio stream from an HDMI stream from the VPHY Controller core.

The HDMI_RX_SS can enable HDCP 1.4, HDCP2.2, or both protocols as part of the subsystem. The reference design instantiates both HDCP protocols and allows you to select the active

protocol. When HDCP is enabled the receiver is able to decrypt video, audio, and auxiliary packets.

For more information, see the HDMI 1.4/2.0 Receiver Subsystem LogiCORE IP Product Guide (PG236) [Ref 4].

Utility Buffer

The Utility Buffer core is configured as an MGT differential input buffer (IBUFDSGTE) and is used to bring in the MGT reference clock outside the VPHY Quad (i.e. GTNORTHREFCLK0 or GTSOUTHREFCLK1).

Video Test Pattern Generator

The TPG core generates test patterns for video system bring up, evaluation, and debugging. The core has two main modes of operation for the reference design. These modes are pass-through and test pattern generation. In pass-through mode, the input AXI4-Stream video is passed without modification to the AXI4-Stream output interface. In generation mode, the TPG core is configured to generate a color bar pattern for demonstration purposes only.



IMPORTANT: *The TPG core provides a wide variety of test patterns for debugging and assessing video system color, quality, edge, and motion performance. The TPG core can be inserted in an AXI4-Stream video interface that allows user-selectable pass-through of system video signals or insertion of test patterns.*



IMPORTANT: *3D video stream can only be supported in pass-through mode as Test Pattern Generator is not supporting 3D video generation.*

The reference design uses these TPG core features:

- RGB, YUV 4:4:4, YUV 4:2:2, YUV 4:2:0.
- Supports 8, 10, 12, and 16-bits per color component input and output.
- Supports spatial resolutions from 64 x 64 up to 4096 x 2160.

For more information, see the Video Test Pattern Generator LogiCORE Product Guide (PG103) [Ref 6].

Custom IPs for Audio

The reference design includes two custom IPs for audio demonstration. These IPs are `aud_pat_gen` and `hdmi_acr_ctrl`. They are not official Xilinx cores. They are provided only to demonstrate the audio capabilities of the **HDMI_TX_SS** and **HDMI_RX_SS**. Xilinx does not support these cores. You as a user are expected to implement or supply your own audio processing solution.

Software Application

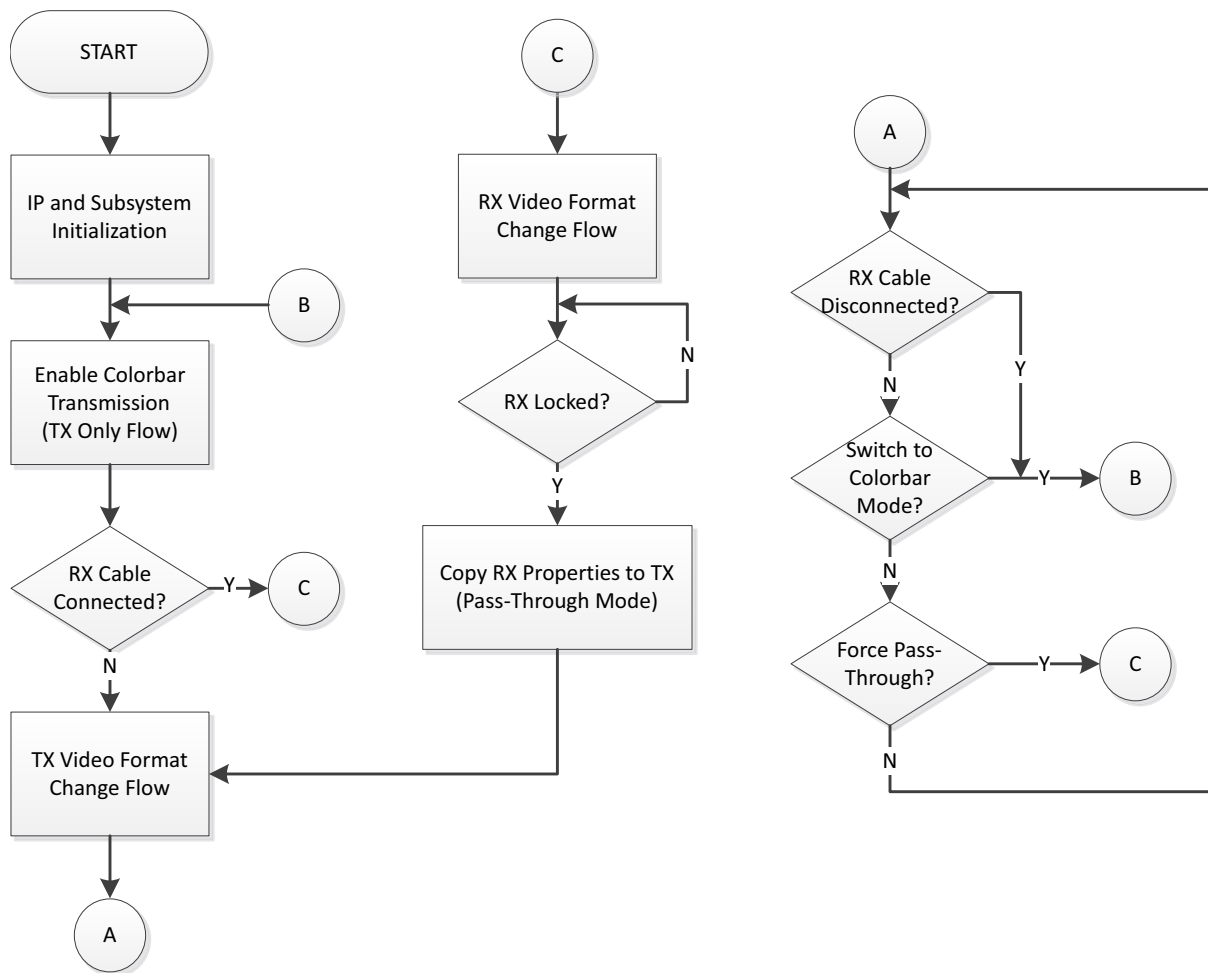
The HDMI reference design uses the built-in drivers of the VPHY, HDMI_TX_SS, and HDMI_RX_SS cores in the SDK. The SDK gives an out-of-the-box solution by providing an abstraction of the internal workings and management of the included sub-cores. The VPHY, HDMI_TX_SS, and HDMI_RX_SS drivers are bare-metal drivers that can work with multiple IP or subsystem instances, as defined in the application layer.

The VPHY Controller core driver manages the dynamic reconfiguration of the multi-gigabit transceiver and digital clock manager modules to allow seamless transmission and reception of HDMI video to and from the FPGA physical interface.

The HDMI_TX_SS and HDMI_RX_SS subsystem drivers dynamically manage the data and control flow through the processing elements, based on the input and output stream configuration set at runtime. Internally, the drivers rely on included sub-core or layer 1 drivers to configure the IP hardware block. Layer 2 or subsystem drivers provide an abstracted view of the feature set of each sub-core.

The subsystem drivers are not active drivers and rely on the application software to use the provided APIs to configure them. The application software monitors the system for external inputs, communicates changes to the input/output stream properties to the subsystem (via provided APIs), and triggers the subsystem auto-reconfiguration routine. The application software manages the HDMI TX and RX applications independently except when they are in bonded mode. In bonded mode, the HDMI_TX_SS and HDMI_RX_SS share the same GT PLL.

The main program flow is shown in [Figure 4](#). At execution, the software application initializes several IPs to activate them. The IP and subsystem initialization executes configuration initialization, interrupt handler connect, and API callback functions for HDMI_TX_SS, HDMI_RX_SS, VPHY Controller, and HDCP core IPs. Next, the color bar transmission is initiated. This process is capped by the configuration of the clock generator that triggers the TX video format change flow.



X16850-042316

Figure 4: Main Program Flow

The RX video format change is triggered as soon as a full HDMI RX cable is detected on the HDMI RX interface. Once the RX is locked to the incoming video format, the application copies the HDMI RX video properties to HDMI TX, and then proceeds to pass-through mode with the TX video format change flow.

In pass-through mode, you can force the application to TX only or color bar mode by unplugging the HDMI RX cable or by selecting color bar (Press **C**) in the UART menu. The pass-through mode can also be forced at any time by plugging in the HDMI RX cable or by pressing **P** in the application.

HDMI TX/RX Subsystem and VPHY drivers provide a mechanism for the application to register a user-defined function that is called within an interrupt context.

In the next few sections, while the various flows are illustrated, the respective callback functions are briefly discussed. For details on how to register these callback functions with provided handlers in your application code, please see the following Product Guides which also contain detail flows on how to integrate those driver APIs into your customized application:

- *LogiCORE IP Video PHY Controller v2.0 Product Guide (PG230)*[Ref 5]

- *LogiCORE IP HDMI 1.4/2.0 Transmitter Subsystem Product Guide (PG235)*[Ref 3]
- *LogiCORE IP HDMI 1.4/2.0 Receiver Subsystem Product Guide (PG236)*[Ref 4]

TX Video Format Change Flow

The TX video format change flow is triggered by a change in the TX TMDS clock frequency using an interrupt from the clock detector module in the VPHY Controller core. The VPHY Controller core driver debounces the frequency event before starting the GT TX reconfiguration based on the new video format to be transmitted. The GT reconfiguration completion is indicated by the assertion of the 0 bit of the `vid_phy_status_sb_tx_tdata`.

The VPHY Controller core driver then configures the internal TX digital clock manager (DCM), which is used to generate the video clock and TX TMDS clock. The DCM reconfiguration completion is indicated by the assertion of the 1 bit of the `vid_phy_status_sb_tx_tdata`.

The HDMI_TX_SS reconfiguration is executed after the completion of the VPHY Controller core reconfiguration and is handled by the subsystem driver. The `TxStreamUpCallback` in the application is executed when the subsystem driver is finished executing the reconfiguration process.

There are several API callback hooks that the VPHY Controller core and HDMI_TX_SS drivers execute throughout the HDMI TX operation. If necessary, these callbacks are available for inserting or adding more function calls on top of what is in the software application.

The VPHY Controller core driver TX callbacks are:

- **VPHY Controller core HDMI TX Init callback**

This callback is named `XVPHY_HDMI_HANDLER_TXINIT` in the VPHY Controller core driver and is hooked to the `VphyHdmiTxInitCallback` function in the software application. It is executed or called every time a change in the HDMI TX TMDS clock frequency occurs. In the software application, the `XV_HdmiTxSs_RefClockChangeInit` HDMI_TX_SS API is called within the hook, which resets and clears the HDMI_TX_SS settings.

- **VPHY Controller core HDMI TX Ready callback**

This callback is named `XVPHY_HDMI_HANDLER_TXREADY` in the VPHY Controller core driver and is hooked to the `VphyHdmiTxReadyCallback` function in the software application. It is executed or called every time the VPHY Controller core driver completes the initialization routine necessary for the TX video format change. The hook currently contains an empty function.

HDMI_TX_SS Driver Callbacks

The HDMI_TX_SS driver callbacks are:

- **HDMI_TX_SS TX Connect callback**

This callback is named `XV_HDMITXSS_HANDLER_CONNECT` in the HDMI_TX_SS driver and is hooked to the `TxConnectCallback` function in the software application. It is executed or called

every time an HDMI TX cable connection or disconnection occurs. The TxConnectCallback software hook detects if the HDMI sink connected is HDMI 2.0 capable through the XV_HdmiTxSs_DetectHdmi20 API. The software hook also enables or disables the differential input clock buffer depending on if cable connection or disconnection occurs, respectively.

- **HDMI_TX_SS Vsync callback**

This callback is named XV_HDMITXSS_HANDLER_VS in the HDMI_TX_SS driver and is hooked to the TxVsCallback function in the software application. It is executed or called every time a vertical sync is detected in the HDMI stream. The TxVsCallback software hook initiates the NULL auxiliary info frame transmission when in TX mode only.

- **HDMI_TX_SS Stream Up callback**

This callback is named XV_HDMITXSS_HANDLER_STREAM_UP in the HDMI_TX_SS driver and is hooked to the TxStreamUpCallback function in the software application. It is executed or called when the HDMI_TX_SS detects an assertion on video ready (TX DCM lock) signal. The TxStreamUpCallback software hook performs the video stream initialization in the HDMI_TX_SS, such as setting the oversample rate and the video property. It also enables the TX TMDS clock and programs the TX line rate in the re-timer chip on the FMC card.

- **HDMI_TX_SS Stream Down callback**

This callback is named XV_HDMITXSS_HANDLER_STREAM_DOWN in the HDMI_TX_SS driver and is unused in the software application.

- **HDMI_TX_SS HDCP Authenticated callback**

This callback is named XV_HDMITXSS_HANDLER_HDCP_AUTHENTICATE in the HDMI_TX_SS driver and is called when either the HDCP 1.4 or HDCP 2.2 transmitter has transitioned into the authenticated state. The software application uses this callback to enable encryption and print a message to the terminal indicating when authentication has successfully completed.

RX Video Format Change Flow

The RX video format change flow is triggered by a change in the RX TMDS clock frequency using an interrupt from the clock detector module of the VPHY Controller core. The VPHY Controller core driver debounces the frequency event before starting the GT RX reconfiguration based on the new video format to be transmitted. The GT reconfiguration completion is indicated by the assertion of the 0 bit of the vid_phy_status_sb_rx_tdata.

The HDMI_RX_SS reconfiguration detects the assertion of the 0 bit of the vid_phy_status_sb_rx_tdata and begins its initialization for the new video format. Part of its initialization is calling the XV_HDMIRX_STATE_STREAM_INIT callback, which configures the RX DCM inside the VPHY Controller core. The VPHY Controller core driver then configures the internal RX DCM, which is used to generate the video clock and the RX TMDS clock. The DCM reconfiguration completion is indicated by the assertion of the 1 bit of the vid_phy_status_sb_rx_tdata. The HDMI_RX_SS then detects this event and calls RxStreamUpCallback to complete the reconfiguration process.

There are several API callback hooks that the VPHY Controller core and HDMI_RX_SS drivers execute throughout the HDMI RX operation. If necessary, these callbacks are available for inserting or adding more function calls on top of what is in the software application.

VPHY Controller Core Driver RX Callbacks

The VPHY Controller core driver RX callbacks are:

- **VPHY Controller core HDMI RX Init Callback**

This callback is named `XVPHY_HDMI_HANDLER_RXINIT` in the VPHY Controller core driver and is hooked to the `VphyHdmiRxInitCallback` function in the software application. It is executed or called every time a change in the HDMI RX TMDS clock frequency occurs. In the software application, the `XV_HdmiTxSs_RefClockChangeInit` HDMI_RX_SS API is called within the hook, which gets the new TMDS clock ratio.

- **VPHY Controller core HDMI RX Ready Callback**

This callback is named `XVPHY_HDMI_HANDLER_RXREADY` in the VPHY Controller core driver and is hooked to the `VphyHdmiRxReadyCallback` function in the software application. It is executed or called every time the VPHY Controller core driver completes the initialization routine necessary for the RX video format change. The hook updates the clock and line rate parameters in the HDMI_RX_SS.

HDMI_RX_SS Driver Callbacks

The HDMI_RX_SS driver callbacks are:

- **HDMI_RX_SS RX Connect Callback**

This callback is named `XV_HDMIRXSS_HANDLER_CONNECT` in the HDMI_RX_SS driver and is hooked to the `RxConnectCallback` function in the software application. It is executed or called every time an HDMI RX cable connection or disconnection occurs. The `RxConnectCallback` software hook enables or disables the differential input clock buffer depending on if cable connection or disconnection occurs, respectively.

- **HDMI_RX_SS Auxiliary Callback**

This callback is named `XV_HDMIRXSS_HANDLER_AUX` in the HDMI_RX_SS driver and is hooked to the `RxAuxCallback` function in the software application. It is executed or called every time an auxiliary packet is detected in the HDMI RX stream. The `RxAuxCallback` software hook initiates the auxiliary info frame retransmission when in pass-through mode.

- **HDMI_RX_SS Audio Callback**

This callback is named `XV_HDMIRXSS_HANDLER_AUD` in the HDMI_RX_SS driver and is hooked to the `RxAudCallback` function in the software application. It is executed or called every time an audio packet is detected in the HDMI RX stream. The `RxAudCallback` software hook initiates the audio packet retransmission when in pass-through mode.

- **HDMI_RX_SS Link Status Callback**

This callback is named `XV_HDMIRXSS_HANDLER_LNKSTA` in the `HDMI_RX_SS` driver and is hooked to the `RxLnkStaCallback` function in the software application. It is executed or called every time a change is detected in the HDMI RX stream. The `RxLnkStaCallback` software hook issues an RX PLL (in the VPHY Controller core) reset pulse when the RX link error status has reached 255 counts.

- **HDMI_RX_SS DDC Callback**

This callback is named `XV_HDMITXSS_HANDLER_DDC` in the `HDMI_RX_SS` driver and is unused in the software application.

- **HDMI_RX_SS Stream Down Callback**

This callback is named `XV_HDMIRXSS_HANDLER_STREAM_DOWN` in the `HDMI_RX_SS` driver and is used in the software application to clear the `IsPassThrough` flag to start TX only mode.

- **HDMI_RX_SS Stream Init Callback**

This callback is named `XV_HDMIRXSS_HANDLER_STREAM_INIT` in the `HDMI_RX_SS` driver and is used in the software application to initiate RX DCM reconfiguration in the VPHY Controller core based on the incoming HDMI RX stream.

- **HDMI_RX_SS Stream Up Callback**

This callback is named `XV_HDMIRXSS_HANDLER_STREAM_UP` in the `HDMI_RX_SS` driver and is hooked to the `RxStreamUpCallback` function in the software application. It is executed or called when the `HDMI_RX_SS` detects an assertion on the video ready (RX DCM Lock) signal. The `RxStreamUpCallback` software hook copies the HDMI RX stream parameter into `HDMI_TX_SS` to prepare for pass-through mode.

- **HDMI_RX_SS HDCP Authenticated callback**

This callback is named `XV_HDMIRXSS_HANDLER_HDCP_AUTHENTICATE` in the `HDMI_RX_SS` driver and is called when either the HDCP 1.4 or HDCP 2.2 receiver has transitioned into the authenticated state. The software application uses this callback to print a message to the terminal indicating when authentication has successfully completed.

Transmitting User-defined Info Frames

User-defined info frames can be sent with the `HDMI_TX_SS` software API. The info frame data structure with a 4-byte header and 32-byte data (payload) must be constructed prior to sending the info frame API function call. You may also need to calculate their own CRC and place the CRC at the right location so that the HDMI Sink is able to decode the Info frame.

Once the info frame data structure is ready, you can call the API function to send it out:

```
void XV_HdmiTxSs_SendGenericAuxInfoframe(XV_HdmiTxSs *InstancePtr, void *AuxPtr);
```

The info frame are normally sent when a vertical sync interrupt event occurs. For details, please refer to HDMI 1.4/2.0 Transmitter Subsystem LogiCORE IP Product Guide (PG235) [Ref 3].

HDCP Authentication

The HDCP protocol establishes secure sessions between transmitters and receivers through an authentication and key exchange process. The HDMI subsystem drivers are responsible for the implementation of the HDCP authentication protocol and provides external user API's for selecting the protection scheme and initiating authentication. This section describes how the reference design system software initializes and uses the HDMI subsystem driver for HDCP.

HDCP Transmitter Authentication

The authentication flow for the HDCP transmitter is shown in [Figure 5](#). When the reference design application begins HDCP is first initialized and production keys are loaded. After initialization the poll process is started causing the HDCP state machines to execute. The transmitter waits for an HPD interrupt event (HPD Event) signaling that a receiver is present. The user input event (Protect Event) is required to set the protection scheme of the transmitter to either HDCP 1.4 or HDCP 2.2. Only after the HPD event and Protection events have occurred, the system is ready to initiate authentication. The transmitter waits for the authentication request user input (Authenticate Event) before it attempts to authenticate with a receiver. If authentication is successful the system automatically enables encryption (Encrypt Event). You also have control over enabling and disabling encryption through the Encrypt Event. In general, events are asynchronous and can potentially happen in any order, in which case the system has built-in mechanisms to recover from faults.

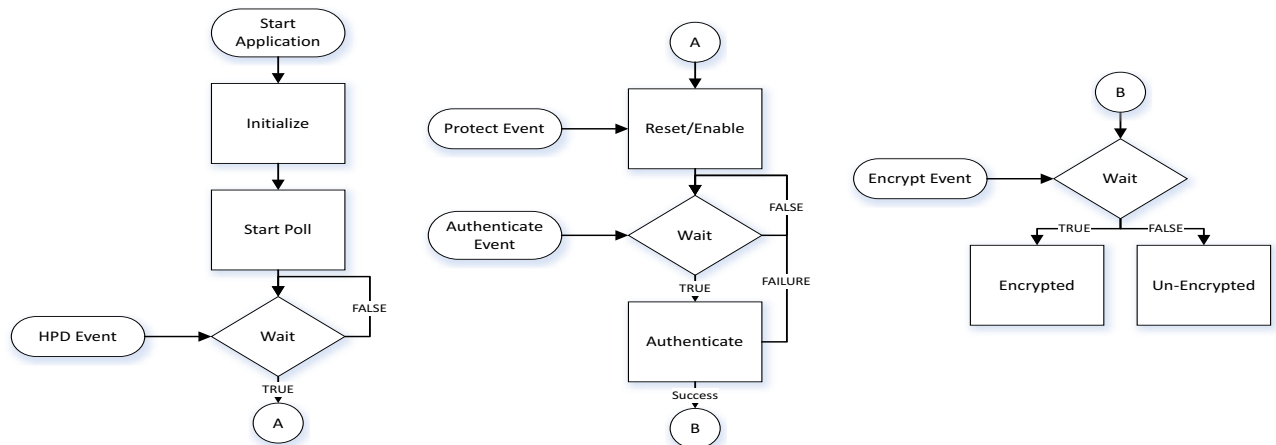
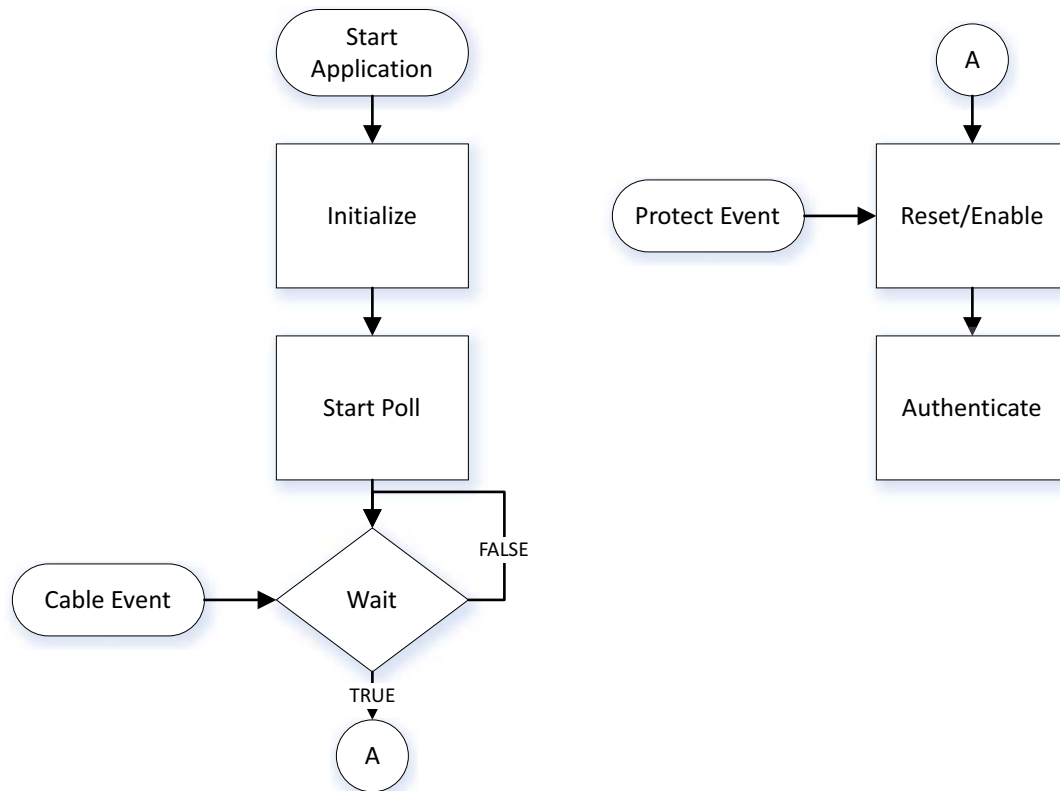


Figure 5: HDCP Transmitter Authentication Flow

X10854-042316

HDCP Receiver Authentication

The authentication flow for the HDCP receiver is shown in [Figure 6](#). After the receiver has initialized and the polling process has started to execute the receiver state machine, the system waits for a cable detect interrupt event (Cable Event) signaling that a transmitter is present. Similar to the transmitter you are required to set the protection scheme (Protect Event) to either HDCP 1.4 or HDCP 2.2. Only after cable detection and protection events have occurred can the receiver begin to service authentication requests from the transmitter. The HDCP transmitter is allowed to initiate new authentication requests at anytime.

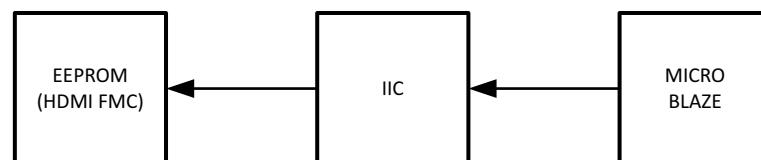


X16855-042316

Figure 6: HDCP Receiver Authentication Flow

HDCP Key Storage and Retrieval

The HDCP 1.4 and HDCP 2.2 keys are stored onboard the HDMI 2.0 FMC card EEPROM using the standalone key utility application. You are required to run the key utility application only once and henceforth the encrypted keys will remain persistently in EEPROM for use by the system application.

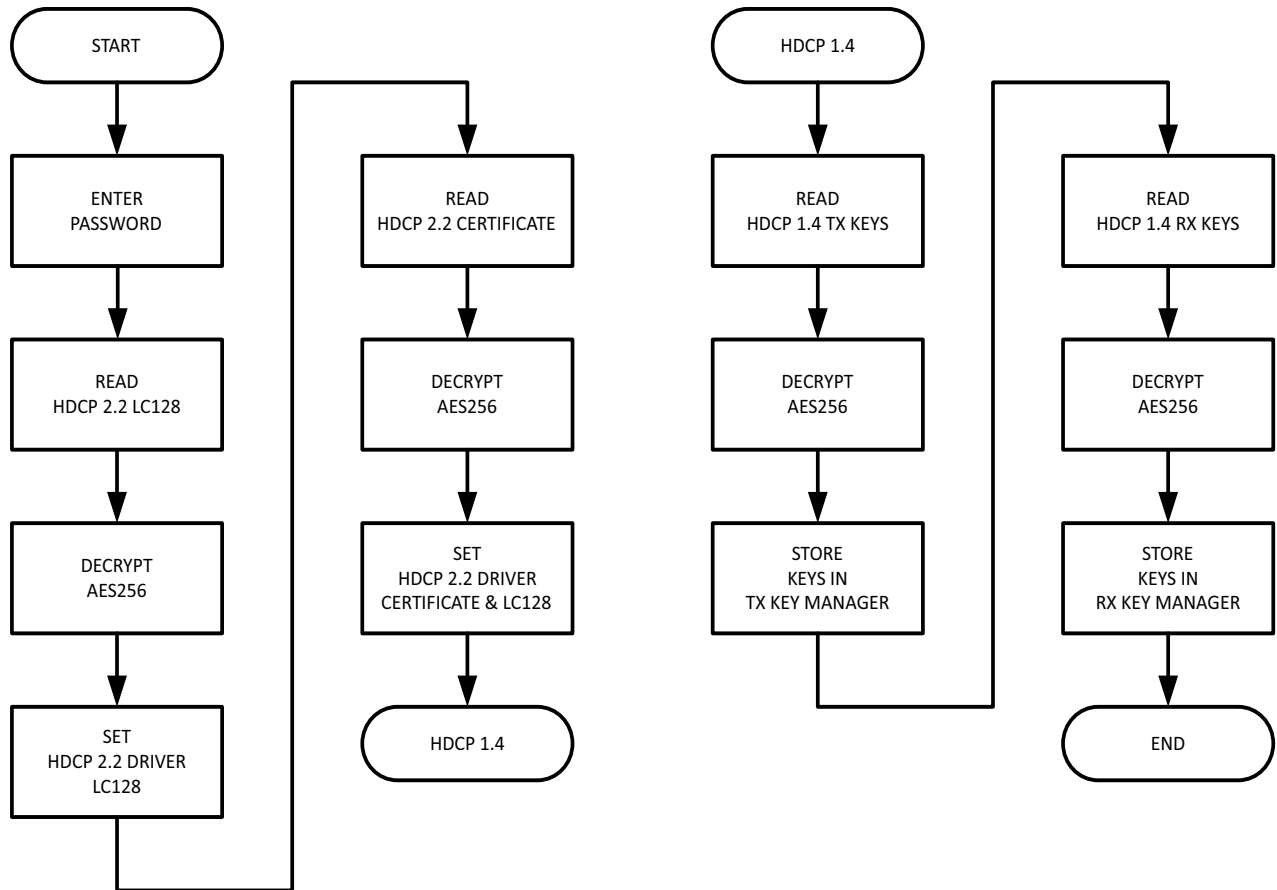


X16853-042316

Figure 7: HDCP Key Storage in EEPROM

The HDCP reference design is shipped without any HDCP keys. You are required to acquire HDCP keys through the Digital Content Protection LLC (DCP) and update the key utility application source file with the production key. Authentication is expected to fail without loading the HDCP production keys. See [Programming the HDCP Keys](#) on how to program the HDCP keys.

Once the HDCP reference design application is started, it will prompt you to enter a secret password during the boot-up process where a cryptographic hash function is applied to produce the AES cipher key used for decryption. Figure 8 shows the sequence of events after you enter the secret password. In case you do not know the password, or it is entered incorrectly the reference design still supports the HDMI functionality, but HDCP features are disabled.



X16856-042316

Figure 8: HDCP Key Description

After the HDCP keys have been retrieved from EEPROM, they are assigned to the HDMI subsystem drivers for the following key types:

- XV_HDMITXSS_KEY_HDCP22_LC128 - HDCP 2.2 transmitter global constant
- XV_HDMIRXSS_KEY_HDCP22_LC128 - HDCP 2.2 receiver global constant
- XV_HDMIRXSS_KEY_HDCP22_PRIVATE - HDCP 2.2 receiver certificate and private key
- XV_HDMITXSS_KEY_HDCP14 - HDCP 1.4 transmitter key
- XV_HDMIRXSS_KEY_HDCP14 - HDCP 1.4 receiver key

Reference Design Tool Flow and Verification

The [Reference design files](#) for this application note can be downloaded from the Xilinx website. The checklist in [Table 6](#) shows the tool flow and verification procedures used for the reference design.

Table 6: Reference Design Checklist

Parameter	Description
General	
Developer Name	Gilbert Magnaye, Yunhai Qiao
Target Devices	Kintex-7 FPGA
Source code provided?	No
Source code format (if provided)	N/A
Design uses code or IP from existing reference design, application note, 3rd party or Vivado software? If yes,	Cores generated from Vivado IP Integrator Tool
Simulation	
Functional simulation performed	N/A
Timing simulation performed?	N/A
Test bench provided for functional and timing simulation?	N/A
Test bench format	N/A
Simulator software and version	N/A
SPICE/IBIS simulations	N/A
Implementation	
Synthesis software tools/versions used	Vivado 2016.2 Synthesis
Implementation software tool(s) and version	Vivado 2016.2
Static timing analysis performed?	Yes
Hardware Verification	
Hardware verified?	Yes
Platform used for verification	Xilinx Kintex-7 FPGA KC705 Evaluation Kit

Requirements

Hardware

The following are the hardware requirements for this reference system:

- Xilinx Kintex-7 FPGA KC705 Evaluation Kit
- inrevium HDMI 2.0 Daughter Card (TB-FMCH-HDMI4K)

- Two HDMI Cables
- HDMI 2.0 Video Source (i.e. DVD Player)
- HDMI 2.0 Sink (i.e. Ultra HD TV sets)

Software

The following are the software requirements for this reference system:

- Vivado Design Suite 2016.2
- Software Development Kit (SDK) 2016.2
- Software Terminals (for example, Tera Term, HyperTerminal or PuTTY)

Reference Design Files

Figure 9 shows the reference design directory structure.



IMPORTANT: *The reference design should be unzipped close to root.*

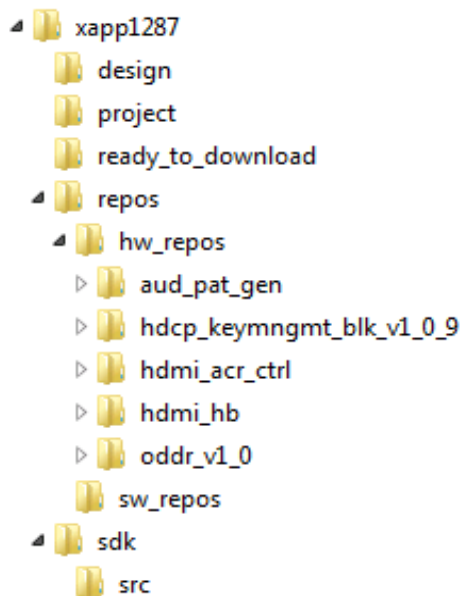


Figure 9: Reference Design Directory Structure

`all.tcl`: This is the main Tcl file to be executed in Vivado Tcl command prompt.

`design`: Contains the kc705 specific design files such as IPI TCL, top level wrapper and XDC

- `hdmi_example_kc705.tcl`: Tcl for constructing the HDMI IPI design.
- `hdmi_example_kc705.xdc`: HDMI Reference Design top level constraint file.

`project`: This is a container folder for Vivado project and generated files.

`ready_to_download`: This folder contains the bitfile and the ELF of the system.

- `hdmi_example_kc705_wrapper.bit`: This is a reference design precompiled bitfile.

`repos`: This folder contains the HW and SW repositories

`\hw_repos`: This folder is a HW repository for custom IPs

`\sw_repos`: This folder is a SW repository (empty)

`sdk`: This folder contains the SW application files.

Licensing

Ensure that the licenses for the VPHY Controller, HDMI 1.4/2.0 Transmitter Subsystem, HDMI 1.4/2.0 Receiver Subsystem, and TPG cores are installed.

Note: Xilinx issues HDCP IP licenses only to active Digital content protection (DCP) license adopters.

Reference Design Steps

This section describes the HDMI reference design setup, execution, and results.

Board Setup

The example design is targeted at the Kintex-7 KC705 development board. The example design supports the inrevium TB-FMCH-HDMI4K FMC card. The board setup is shown in [Figure 10](#).

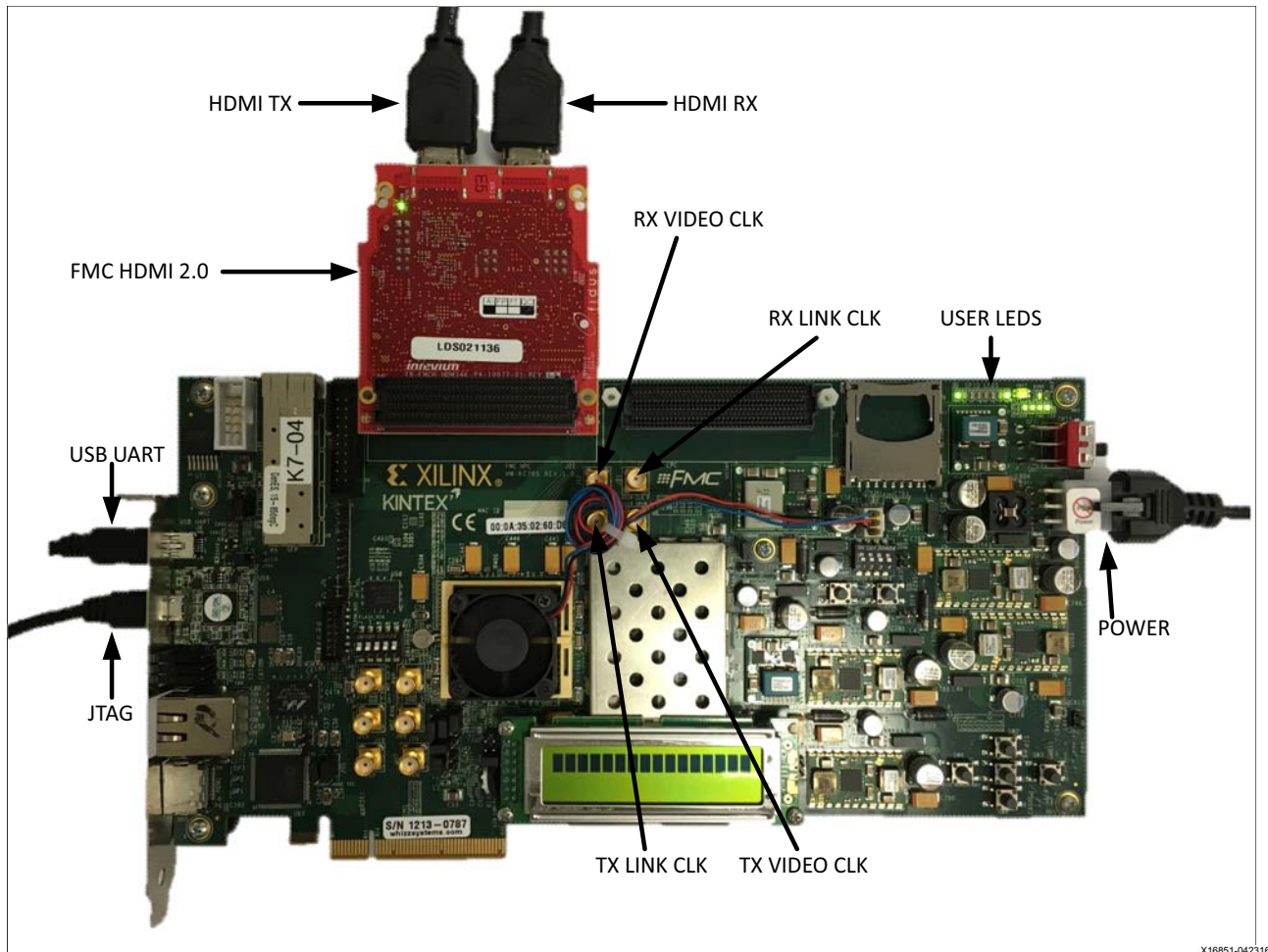


Figure 10: XAPP1287 HDMI Reference Design Setup

HW Setup

1. Connect a USB cable from the host PC to the USB JTAG port. Ensure the appropriate device drivers are installed.
2. Connect a second USB cable from the host PC to the USB UART port. Ensure that USB UART drivers described in the section Hardware Requirements have been installed.
3. Connect TB-FMCH-HDMI4K board to the HPC-FMC of KC705.
4. Connect the HDMI TX port to an HDMI 1.4/2.0 sink or monitor (i.e. Ultra HD TV set).
5. Connect the HDMI RX port to an HDMI 1.4/2.0 source (i.e. DVD Player).
6. Connect the KC705 board to a power supply slot J49.
7. Switch on the KC705 board.
8. Start a terminal program (e.g. HyperTerminal) on the host PC with these settings:
 - Baud Rate: 115200

- Data Bits: 8
- Parity: None
- Stop Bits: 1
- Flow Control: None

Table displays the onboard LEDs used in the reference design.

Table 7: Onboard LEDs

LED	Description
0	HDMI TX Subsystem lock
1	Not used
2	Not used
3	Not used
4	Not used
5	Not used
6	TX clock heartbeat
7	RX clock heartbeat

Programming the HDCP Keys

This section describes how you can program the HDCP keys into the HDMI 2.0 FMC card EEPROM using the key utility application. The keys only need to be programmed into the FMC EEPROM once.

1. Launch the Xilinx System Debugger by selecting **Start > All Programs > Xilinx Design Tools > Vivado 2016.2 > Vivado 2016.2 Tcl Shell**.
2. In the Xilinx command shell window, change to the ready_to_download directory:


```
cd <unzip_dir>/<hdcpc_key_design>/ready_to_download
```
3. Launch Xilinx System Debugger (XSDB)


```
Vivado% xsdb
```
4. Establish connections to debug targets


```
xsdb% connect
```
5. Download the bitstream for programming the HDCP keys to the FPGA:


```
xsdb% fpga -file hdcpc_key.bit
```
6. Exit the XSDB command prompt:


```
xsdb% exit
```
7. Open the file `hdcpc_key.c` in the `app_hdcpc_key/src` folder
8. The arrays `Hdcpc22Lc128`, `Hdcpc22Key`, `Hdcpc14Key1`, and `Hdcpc14Key2` hold the HDCP keys and are empty. Fill these arrays with the acquired HDCP keys. The arrays are defined in big endian byte order.

9. Save the file and compile the design.
10. Run the design.
11. The terminal will display the following output:

```

HDCP Key EEPROM v1.0
This tool encrypts and stores the HDCP 2.2 certificate and
HDCP 1.4 keys into the EEPROM on the HDMI FMC board
Enter Password ->

```

12. The HDCP keys are encrypted using this password. The same password is used in the reference design to decrypt the HDCP keys.
13. The application is terminated after completing the programming of HDCP keys.

Running the Reference Design

Use the following steps to execute the system using the files in the `ready_for_download` directory

1. Launch the Xilinx System Debugger by selecting **Start > All Programs > Xilinx Design Tools > Vivado 2016.2 > Vivado 2016.2 Tcl Shell**.
2. In the Xilinx command shell window, change to the `ready_for_download` directory:

```
Vivado% cd <unzip_dir>/kc705_hdmiss_.../ready_for_download
```

3. Invoke Xilinx System Debugger (xsdb)

```
Vivado% xsdb
```

4. Establish connections to debug targets

```
xsdb% connect
```

5. Download the bitstream to the FPGA:

```
xsdb% fpga -file hdmi_example_kc705_wrapper.bit
```

6. Exit the XSDB command prompt:

```
xsdb% exit
```



IMPORTANT: *The software application starts immediately after the completion of the FPGA configuration. The executable file (.elf) is embedded in the configuration file (download.bit).*

Results

This section describes the reference design results.

TX Only Mode

The TX only mode starts immediately after the bitstream download when no valid HDMI source is plugged in to the HDMI RX port. HyperTerminal Screen of the TX Only mode displays the following output. At the start-up, the software will ask you for the password that was used previously to load the HDCP keys by the key utility application. If the wrong password is

supplied then HDCP functionality will be disabled. The colorbar pattern displayed on the HDMI sink is shown in [Figure 11](#).

```

-----
---  HDMI SS + VPhy Example v1.0    ---
---  (c) 2015, 2016 by Xilinx, Inc. ---
-----
Build Feb  1 2016 - 08:16:36
-----
Before the HDCP functionality can be enabled,
the application will load the encrypted HDCP keys
from the HDMI FMC EEPROM.
The HDCP keys are protected with a unique password.
Please enter your password.
Enter Password ->.....
Password is valid.
Loading HDCP keys from EEPROM... done
Enabling HDCP functionality
-----
TX cable is connected
Starting colorbar

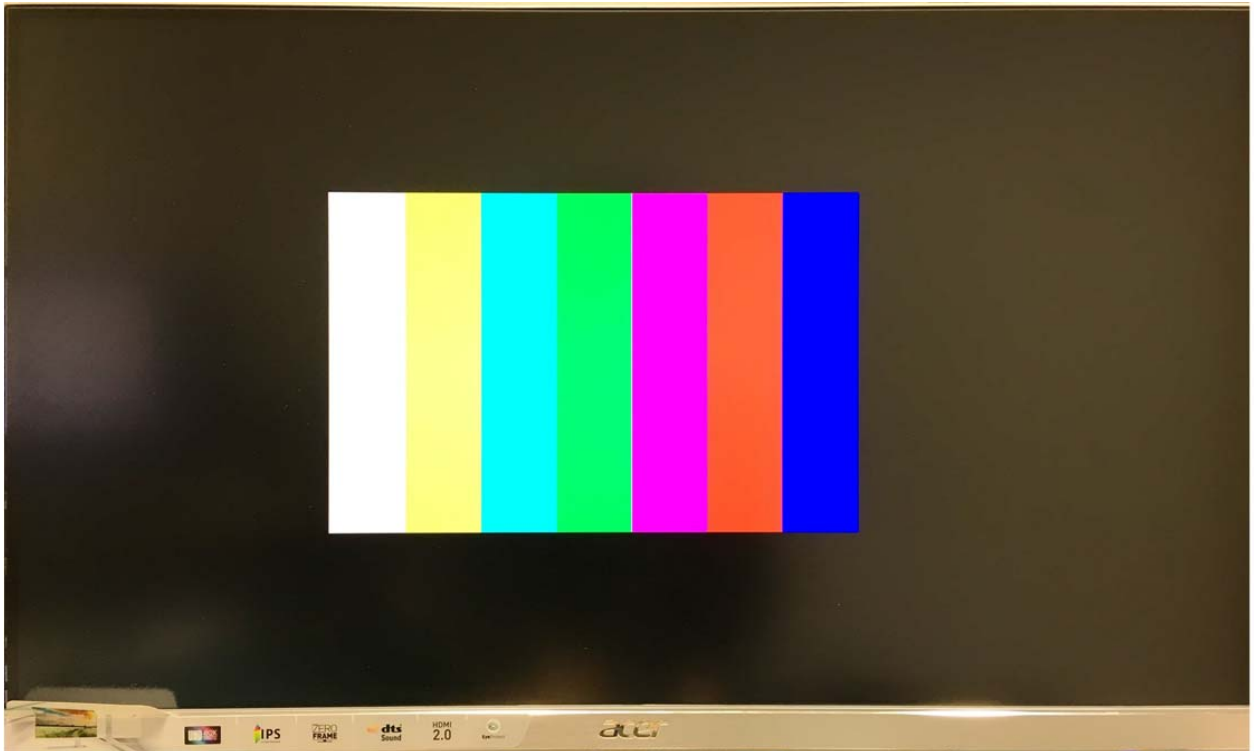
-----
---  MAIN MENU    ---
-----
i - Info
    => Shows information about the HDMI RX stream, HDMI TX stream,
        GT transceivers and PLL settings.
c - Colorbar
    => Displays the colorbar on the source output.
r - Resolution
    => Change the video resolution of the colorbar.
f - Frame rate
    => Change the frame rate of the colorbar.
d - Color depth
    => Change the color depth of the colorbar.
s - Color space
    => Change the color space of the colorbar.
p - Pass-through
    => Passes the sink input to source output.
l - GT PLL layout
    => Select GT transceiver PLL layout.
z - GT log
    => Shows the GT transceiver log information.
e - Edid
    => Display and set edid.
a - Audio
    => Audio options.
v - Video
    => Video pattern options.
h - HDCP
    => Goto HDCP menu.
Starting colorbar
TX stream is up
-----
Colorbar :
    Color Format:      RGB
    Color Depth:     8
    Pixels Per Clock: 2

```

```

Mode:           Progressive
Frame Rate:    60Hz
Resolution:    1920x1080@60Hz
Pixel Clock:   148500000
-----

```



X16882-042316

Figure 11: TX Only Mode - 1920x1080p60(default) Colorbar Pattern

Note: In Figure 11, the HDMI 2.0 sink was not set to scale up a smaller resolution. Hence, the colorbar did not occupy the whole 4K display

Pass-through Mode

The Pass-through mode starts immediately after the bitstream download when there is valid HDMI source plugged in the HDMI RX port. HyperTerminal Screen of the TX Only mode displays the following output:

```

--- HDMI SS + VPhy Example v1.0 ---
--- (c) 2015, 2016 by Xilinx, Inc. ---
-----
Build Feb  1 2016 - 08:16:36
-----
Before the HDCP functionality can be enabled,
the application will load the encrypted HDCP keys
from the HDMI FMC EEPROM.
The HDCP keys are protected with a unique password.
Please enter your password.
Enter Password ->.....
Password is valid.
Loading HDCP keys from EEPROM... done
Enabling HDCP functionality
-----
RX cable is connected

```

```

TX cable is connected
Starting colorbar

-----
---  MAIN MENU  ---
-----
i - Info
    => Shows information about the HDMI RX stream, HDMI TX stream,
        GT transceivers and PLL settings.
c - Colorbar
    => Displays the colorbar on the source output.
r - Resolution
    => Change the video resolution of the colorbar.
f - Frame rate
    => Change the frame rate of the colorbar.
d - Color depth
    => Change the color depth of the colorbar.
s - Color space
    => Change the color space of the colorbar.
p - Pass-through
    => Passes the sink input to source output.
l - GT PLL layout
    => Select GT transceiver PLL layout.
z - GT log
    => Shows the GT transceiver log information.
e - Edid
    => Display and set edid.
a - Audio
    => Audio options.
v - Video
    => Video pattern options.
h - HDCP
    => Goto HDCP menu.
Starting colorbar
Active audio channels 2
RX stream is up
TX stream is up
-----
Pass-Through :
    Color Format:      RGB
    Color Depth:     8
    Pixels Per Clock: 2
    Mode:             Progressive
    Frame Rate:      60Hz
    Resolution:      1920x1080@60Hz
    Pixel Clock:     148500000
-----

```

The display seen on the HDMI sink will depend on the content of the HDMI source connected to the HDMI RX port.

Building Hardware

This section covers rebuilding the hardware design. Before rebuilding the project, ensure that the licenses for the VPHY Controller, HDMI 1.4/2.0 Transmitter Subsystem, HDMI 1.4/2.0 Receiver Subsystem, HDCP and Test Pattern Generator are installed.

Note: To ensure that no compilation errors occur due to long file paths, unzip the project files as close to the root directory as possible. For example, with a typical Windows installation, unzip the files at C:\.

Generating Programming File in Vivado Design Suite 2016.2

1. Open Vivado Design Suite.
2. At the Tcl Console, change to the workspace directory by typing:

```
> cd <unzip dir>\xapp1287
```
3. Run the all.tcl script to create, compile and generate the project bitstream

```
> source all.tcl
```

Debugging

These user LEDs can be used to visually check the status of reference design:

LED0-HDMI TX subsystem lock

If this LED is off, it suggests that there is a mismatch in video timing information between the HDMI TX stream and what the HDMI_TX_SS is expecting. Confirm that the HDMI TX stream timing is correct, or disconnect and then reconnect the TX cable to reinitialize the HDMI TX.

LED6 - TX clock heartbeat

If this LED is not blinking, it suggests that the TX link clock is not active. Ensure that the TX cable is connected and that there is a valid HDMI stream going into the HDMI TX port.

LED7 - RX clock heartbeat

If this LED is not blinking, it suggests that the RX link clock is not active. Ensure that the RX cable is connected.

Known Issues and Limitations

1. 3D video stream can only be supported in pass-through mode as Test Pattern Generator does not support 3D video generation.

References

1. *Xilinx Kintex-7 FPGA KC705 Evaluation Kit*
(<http://www.xilinx.com/products/boards-and-kits/EK-K7-KC705-G.htm>)
2. Inrevium TB-FMCH-HDMI4K (<http://solutions.inrevium.com/products/fmc/index.html>)
3. *LogiCORE IP HDMI 1.4/2.0 Transmitter Subsystem Product Guide* ([PG235](#))
4. *LogiCORE IP HDMI 1.4/2.0 Receiver Subsystem Product Guide* ([PG236](#))
5. *LogiCORE IP Video PHY Controller Product Guide* ([PG230](#))
6. *LogiCORE IP Video Test Pattern Generator Product Guide* ([PG103](#))
7. *Vivado Design Suite AXI Reference Guide* ([UG1037](#))

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
05/20/2016	1.0	Initial Xilinx release.
06/04/2016	1.1	Added HDCP Key Utility Application in Reference Design files (No changes in the Application Note).
07/18/2016	1.2	<ul style="list-style-type: none"> • Added support for Vivado 2016.2. • Removed the following from Known Issues and Limitations as they are fixed in the current version: <ul style="list-style-type: none"> ◦ QPLL cannot be used to transmit 2160p60. ◦ Wrong luma and chroma on 720p60 at 10 BPC. ◦ Wrong luma and chroma on WXGA+ (1366x768p60) at 10 BPC. ◦ No video display on 2160p24/25/30 at 16 BPC. ◦ Wrong video output when video is in YUV420 color space, this is due to AXI4-Stream Video compatibility issues between the TPG and HDMI_TX_SS. ◦ PAL and NTSC video formats to and from HDMI_SS are not compatible with Video Processing Subsystem IP due to AXI4-Stream Video compatibility issues between them.

Please Read: Important Legal Notices

The IP/reference design(s) enables Xilinx customers to implement High-bandwidth Digital Content Protection (HDCP) cipher and authentication functions in Xilinx silicon devices using Xilinx's LogiCORE High-Definition Multimedia Interface (HDMI) IP solutions. The reference design(s) which also includes logic blocks which may be used by customers at their option to securely read HDCP device keys from an external storage device and store them on a Xilinx silicon device; alternatively, customers may independently develop such logic to perform these functions. HDCP device keys are not provided with the reference design(s) and are not available from Xilinx under any circumstances. Customers who desire to utilize the IP and reference design(s) to implement HDCP must become an HDCP Adopter and acquire device keys directly from Digital Content Protection, LLC. Failure by customers to do so will render the IP and reference design(s) incapable of successfully completing the HDCP implementation in customers' products.

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>.

© Copyright 2016 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.