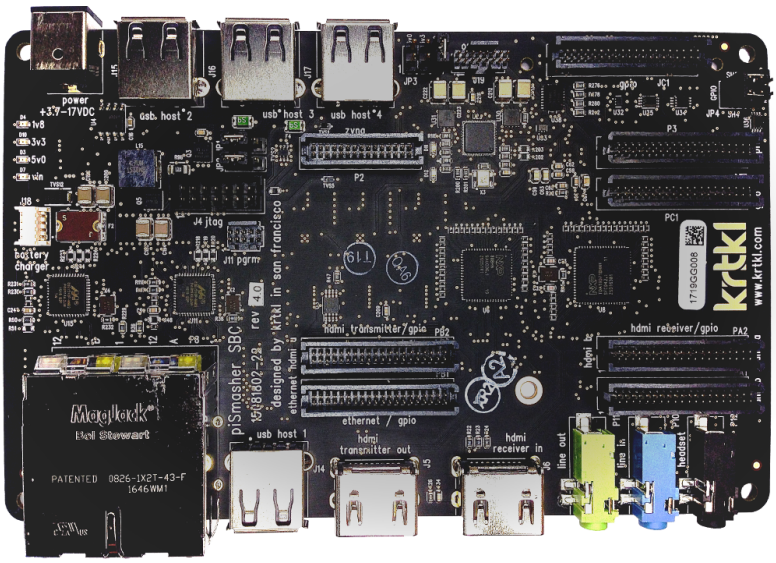
snickerdoodle



SNICKERDOODLE: PISMASHER SUPPLEMENT

krtkl inc.

snickerdoodle: piSmasher Supplement

Copyright © 2018 krtkl inc.

krtkl inc.
350 TOWNSEND STREET
SUITE 301A
SAN FRANCISCO, CA 94107
UNITED STATES

Open Source License and Reproduction

The information contained in this document is made available under a Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license please visit:
http://creativecommons.org/licenses/by-sa/4.0

Disclaimer

Use the information in this text at your own risk. The contents of this text have been researched and compiled to provide examples and understanding of the use of Snickerdoodle, without any guarantee of merchantability, durability or fitness for a particular purpose. Use of the concepts, examples, and/or other content of this document is entirely at your own risk. The contents of this text are provided "as is", with no implication of any warranty of any kind. The authors and publisher of this text should not be held liable for any loss or damage resulting directly or indirectly from the information contained herein. All copyrights are owned by their owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark. Naming of particular products or brands should not be seen as endorsements.

Trademarks and Credits

Xilinx, the Xilinx logo, Zynq, Vivado and WebPack are registered trademarks of Xilinx. All rights reserved.

ARM, Cortex, AMBA, Thumb, CoreSight and TrustZone are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. Keil is a trademark of ARM Limited. All rights reserved.

NXP and the NXP logo are trademarks of NXP B.V. All rights reserved.

Texas Instruments Incorporated and Texas Instruments are trademarks of Texas Instruments.

Microchip is a registered trademark of Microchip Technology Inc.

All other trademarks used in this text are acknowledged as belonging to their respective owners. The use of trademarks in this text does not imply any affiliation with, or endorsement of, this text by the owners of the trademarks.

Contents

1 Introduction 1
- 1.1 Dual Ethernet 2
- 1.2 Audio Codec 3
- 1.3 USB 4
- 1.4 HDMI Transmit 4
- 1.5 HDMI Receive 7

2 HDMI Transmit and Receive 10
- 2.1 Video Testing Configuration 10
- 2.2 Test Pattern Generator 11
- 2.3 Color Conversion 12

A Appendix 13
- A.1 Ethernet Connections 13
- A.2 Ethernet Constraints 15
- A.3 Audio Codec Connections 17
- A.4 Audio Codec Interface Constraints 17
- A.5 HDMI Transmit Connections 18
- A.6 HDMI Transmit Constraints 19
- A.7 HDMI Transmit Registers 20
- A.8 HDMI Receive Connections 23
- A.9 HDMI Receive Constraints 24
- A.10 HDMI Receive Video Port 25

List of Acronyms 26

List of Figures

1.1 *piSmasher Block Diagram* 1
1.2 *GMII and RGMII Timing* 2
1.3 *Ethernet GMII to RGMII on EMIO* 3
1.4 *TLV320AIC3104 Audio Codec Simplified Block Diagram* 3
1.5 *I^2S Interface Timing* 4
1.6 *TDA19988 HDMI Transmitter Simplified Block Diagram* 5
1.7 *Video Timing Signals and Frame Layout* 6
1.8 *TDA19988 HDMI Receiver Simplified Block Diagram* 7

2.1 *HDMI Video Test Configuration* 10

A.1 *Video Port Mapping Register Field* 20
A.2 *Matrix Control Register (Register Address: 0x80 Page Address: 0x00)* 21
A.3 *Matrix Control Register (Register Address: 0x80 Page Address: 0x00)* 21
A.4 *Video Port Mapping Register* 25

List of Tables

1.1 *TDA19988 Video Port Mapping* 5
1.2 *TDA19971 Video Port Mapping* 8

A.1 *Ethernet Connector and Package Pins* 13
A.2 *Ethernet Connector and Package Pins* 14
A.3 *Audio Connector and Package Pins* 17
A.4 *HDMI Transmitter Connector and Package Pins* 18
A.5 *Video Port Pin Swap Configuration Registers (Page 0x00)* 20
A.6 *Matrix Conversion Registers (Page 0x00)* 21
A.7 *Matrix Conversion Registers (Page 0x00)* 22
A.8 *Matrix Conversion Registers (Page 0x00)* 22
A.9 *HDMI Receiver Connector and Package Pins* 23

List of Code Listings

A.1 *piSmasher Ethernet 0 RGMII I/O and Timing Constraints* 15
A.2 *piSmasher Ethernet 1 RGMII I/O and Timing Constraints* 16
A.3 *piSmasher Audio Codec Constraints* 17
A.4 *piSmasher HDMI Transmit I/O Pin Constraints* 19
A.5 *piSmasher HDMI Receive I/O Pin Constraints* 24

1

Introduction

piSmasher is an single board computer baseboard for snickerdoodle. It has a number of standard computing interfaces with an additional interface layer through the programmable logic. The ubiquity of USB is available through four USB 2.0 ports for connecting common devices. Audio capabilities are provided by a low-power audio codec. This includes a line input and output as well as a headset output with microphone. Higher performance interfaces such as the HDMI transmitter and receiver and dual Gigabit Ethernet are routed through the programmable logic to leverage the parallel computing and processing ability of the FPGA independent of the processing subsystem.

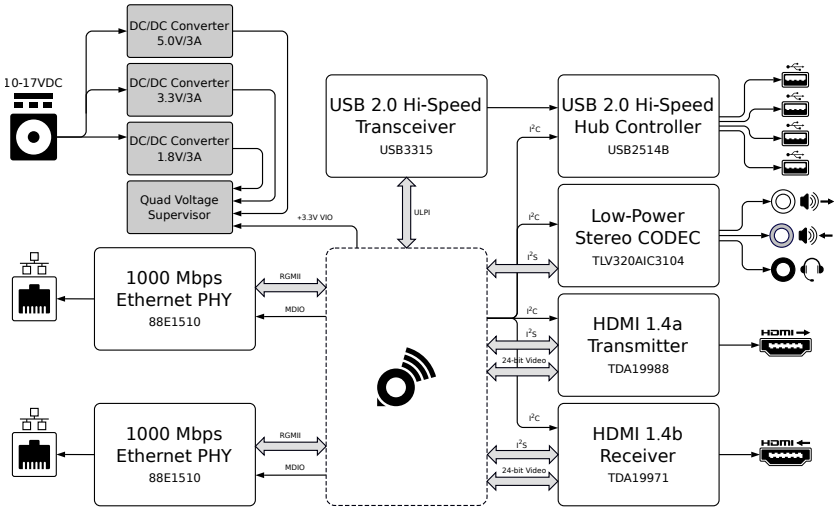


Figure 1.1: piSmasher Block Diagram

1.1 Dual Ethernet

The piSmasher is equipped with two Gigabit Ethernet interfaces (designated ETH0 and ETH1) that are connected to an attached snickerdoodle through the programmable logic. Each ethernet PHY I/O is routed to programmable logic connected pins. The attached snickerdoodle will require a loaded bitstream to route the MIO and MDIO signals to the processing subsystem. The PHY uses a RGMII interface, while the processing subsystem uses a GMII interface for ethernet peripherals connected to EMIO. These interfaces can be converted directly using GMII to RGMII IP and, as any signals routed through programmable logic, are usable in any custom logic designs.

1.1.1 *reduced gigabit media independent interface (RGMII)*

CAUTION The RGMII interface connection to each ethernet PHY is incompatible with the GMII port that appears for any ethernet interfaces configured for EMIO.

When routed to the pins on J3 (MIO), either Gigabit Ethernet MAC can be connected to a PHY using a RGMII interface. However, the MAC block uses a GMII interface when routed through the programmable logic using extended multiplexed input/output. While both of these interfaces operate at 1Gbps, they do not use the same data width. To achieve 1Gbps, a GMII interface uses eight data signals that are clocked at 125MHz. A RGMII is also clocked at 125MHZ but only uses four data signals. To operate with half as many data signals, the interface uses *double-data rate (DDR)*. Data is clocked from the transmit to the receive side of the interface using both rising and falling edges. Figure 1.2 shows the differences in sampling between an 8-bit data bus and a 4-bit data bus operating at DDR. Programmable logic designs more naturally utilize data ports that have wider buses with a single transition clock. This reduces sensitivity to disruptive timing issues such as jitter. After conditioning in the programmable logic, the interface can by converted using the GMII to RGMII IP before being routed to the pins connecting to the PHY.

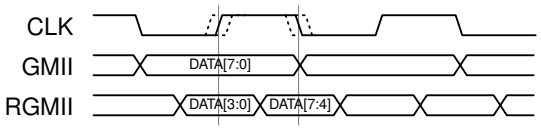


Figure 1.2: GMII and RGMII Timing

Figure 1.3 shows the connections between the PS connected GMII and optional user logic before the output signals are connected to the GMII to RGMII conversion IP. The conversion IP requires an input clock of 200MHz which is used internally to clock the logic required for operating at double-data rate. The IP also requires a connection to the MDIO signals which are used to arbitrate the data rate configuration with the PHY. A PHY address is assigned to the IP which allows the processing system to communicate to and configure the block without requiring an AXI connection. This address should be set to ensure it does not conflict with the attached PHY address.

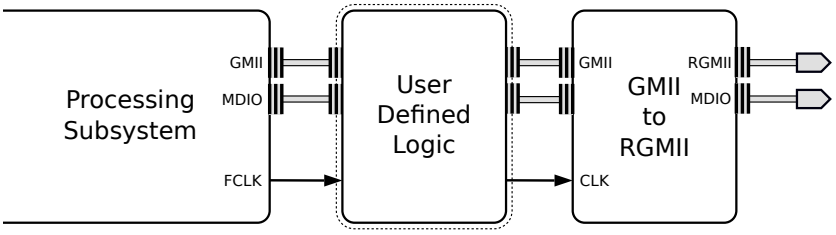


Figure 1.3: Ethernet GMII to RGMII on EMIO

1.2 Audio Codec

The audio codec interface is connected to programmable logic pins. The codec can be configured to use a number of bus standards for the interface I/O. Typically, the audio codec is configured for *inter-IC sound (I^2S)*. The user inputs and outputs to the codec include a stereo line in and out and headset. The headset has high-power stereo headphone outputs and a single (mono) microphone input. There are a number of audio processing features and control settings available on the audio codec. Control of these settings and features is made possible through an I^2C interface that allows access to the codec registers.

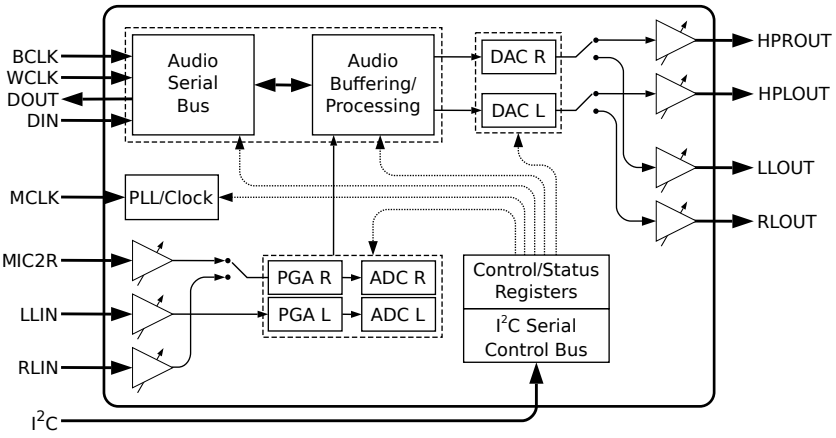


Figure 1.4: TLV320AIC3104 Audio Codec Simplified Block Diagram

1.2.1 I^2S

The default bus configuration for the audio codec is a 16-bit per channel I^2S interface. The codec can be configured to run in either master or slave mode. The interface can be configured for a variety of data rates (typically 44.1kHz or 48kHz) and the PLL can be configured to produce data rates using a variety of input clock frequencies.

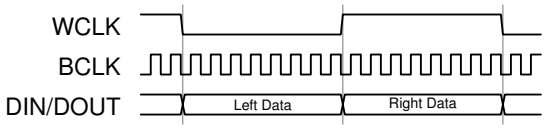


Figure 1.5: I^{2}S Interface Timing

1.3 USB

piSmasher is loaded with a USB 2.0 transceiver and 4-port hub controller. USB provides a standardized and ubiquitous interface for connecting devices providing mass storage, networking, human interfaces and a variety of other common capabilities. The hub controller, responsible for bridging connections between the transceiver and the port connectors, is capable of providing current to devices that require a power supply or for charging battery-powered devices.

1.3.1 USB 2.0 Transceiver

A Microchip USB3315 provides the physical layer for the piSmasher USB.

1.3.2 USB 2.0 Hub Controller

The USB hub controller enables the power output for each port. The +V$_{BUS}$ output to each port is provided by a set of dual-channel current-limited power switches. These switches comply with USB standards to provide switching for hot-swap devices on the ports.

1.4 HDMI Transmit

1.4.1 Configuration Port (I^{2}C)

An I^{2}C interface is used to configure the HDMI transmitter including configuring video input and output formats.

1.4.2 Video Port

Six sets of signals, arranged into four signal quartets that make up the 24-bit video interface can be configured to drive a similar set of six 4-bit nibbles of an internal video port. This means that each set of four pins can be configured to represent a specific component of the video data. Once configured, each

set of signals then represents the assigned portion of the video data as it is clocked over the interface into the transmitter.

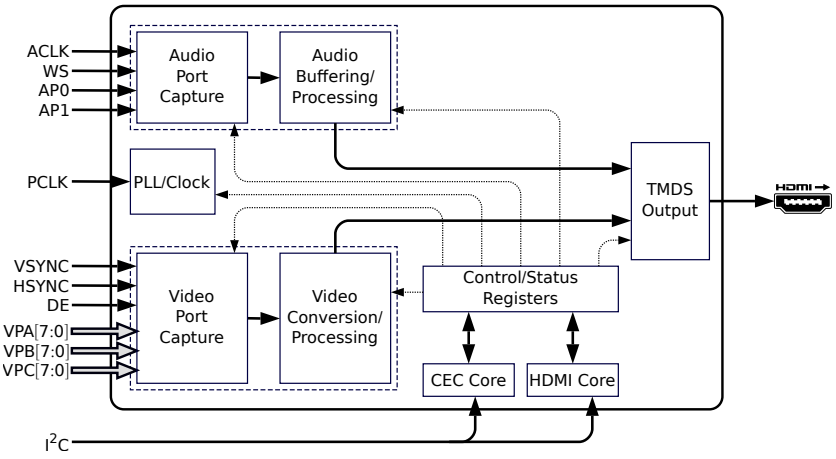


Figure 1.6: TDA19988 HDMI Transmitter Simplified Block Diagram

Table 1.1: TDA19988 Video Port Mapping

Video Port	RGB	YCbCr 4:4:4	YCbCr 4:2:2 (Semi-planar)
VP[23:20]	G[7:4]	Y[7:4]	Y[11:8]
VP[19:16]	G[3:0]	Y[3:0]	Y[7:4]
VP[15:12]	B[7:4]	Cb[7:4]	Y[3:0]
VP[11:8]	B[3:0]	Cb[3:0]	CbCr[11:8]
VP[7:4]	R[7:4]	Cr[7:4]	CbCr[7:4]
VP[3:0]	R[3:0]	Cr[3:0]	CbCr[3:0]

The swap codes in the video port mapping registers determine which color components are driven by which set of pins. The pins are bundled in groups of four which matches the internal video port bit groups.

1.4.3 Video Timing

The HDMI transmitter requires a set of video timing signals to determine the beginning and end to each line within a frame and the beginning and end of each frame. For this purpose, the HDMI transmitter uses three signals: *data enable (DE)*, *horizontal synchronization (HSYNC)* and *vertical synchronization (VSYNC)*. The HSYNC signal is used to set the end of a horizontal line. After a full frame of lines have been transmitted, the VSYNC signal asserts the end of the frame. The DE is used to enable transmission of frame data. When the DE is unasserted, the video is blanked. Assertion of the DE signal starts transmission of a new frame.

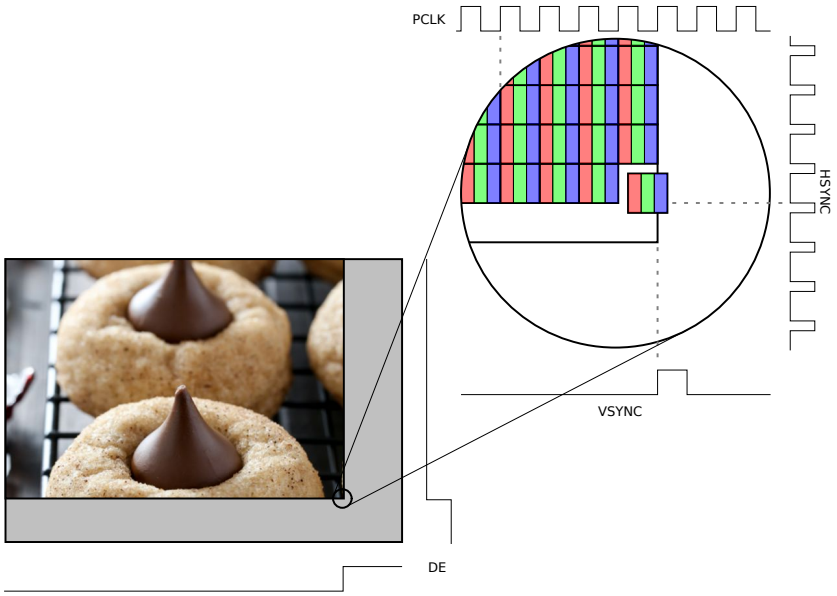


Figure 1.7: Video Timing Signals and Frame Layout

Figure 1.7 shows the pixel clock rising edge event with each pixel, horizontal synchronization event with each row and vertical synchronization event with each frame. The gray area represents the blanked area of the frame. The pixel clock and HSYNC signals will continue to toggle while blanking pixels are transmitted. The blanked area contains pixels that are clocked to/from the HDMI transceiver and must be included in calculations to determine the appropriate pixel clock and HSYNC frequencies for a particular frame rate.

$$f_{hsync} = f_{vsync} \cdot V_{frame} \tag{1.1}$$

VSYNC events will occur at the frame rate as one rising/falling edge of the VSYNC signal will be asserted per frame.Equation 1.1 shows the relationship of HSYNC and VSYNC frequency while Equation 1.2 shows the pixel clock rate as a function of HSYNC frequency.

$$f_{pclk} = f_{hsync} \cdot H_{frame} \tag{1.2}$$

The following is an example calculation of the HSYNC rate for a 720p frame operating at 60Hz.

$$f_{hsync} = f_{vsync} \cdot V_{frame}$$
$$f_{hsync} = (60Hz) \cdot (750)$$
$$f_{hsync} = 45kHz$$

The required pixel clock rate to transmit a single line of a frame can found by multiplying the calculated 45 kHz HSYNC rate by the frame width.

$$f_{pclk} = f_{hsync} \cdot H_{frame}$$
$$f_{pclk} = (45kHz) \cdot (1650)$$
$$f_{pclk} = 74.25MHz$$

1.5 HDMI Receive

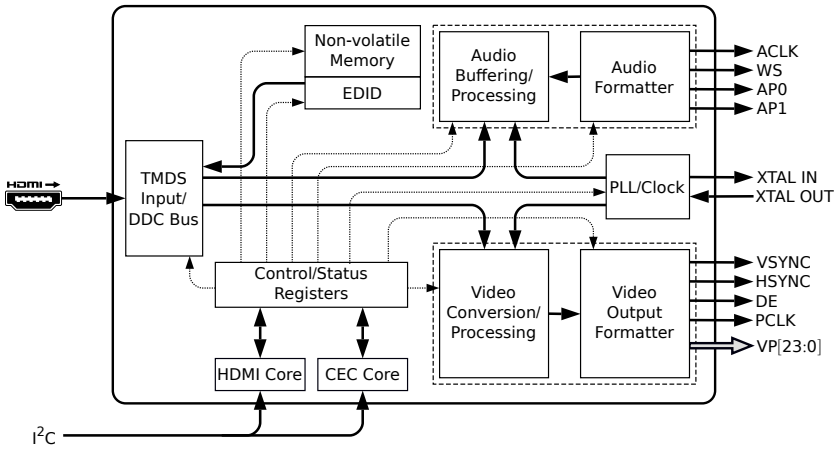


Figure 1.8: TDA19988 HDMI Receiver Simplified Block Diagram

1.5.1 Configuration Port (I^2C)

Control of the HDMI and CEC cores is possible using the I^2C port. The I^2C port is used to configure the receiver and read the device state with the control and status registers. Operation of the receiver requires enabling the device subsystems (*e.g.*, TMDS).

1.5.2 Extended Display Identification Data (EDID)

The EDID is used by HDMI sources to determine supported video formats and other capabilities of the receiver. This allows the source to automatically select the preferred operating settings, permit configuration of output within supported formats and provide other information about the receiver such as manufacturer and product descriptions. The EDID is configurable through 128-byte block of registers followed by an extension block.

1.5.3 Non-Volatile Memory

CAUTION The non-volatile memory can be written to a maximum of 200 cycles.

The HDMI receiver contains a non-volatile memory that can be used to set a default configuration for the control registers when powered on. This allows the receiver to be configured without writing the registers using the I^2C port. This is particularly useful for configuring the receiver before the I^2C interface is available and to configure parameters that do not require changes during normal operation such as EDID.

1.5.4 Video Port Configuration

The video output data from the HDMI receiver is carried on a 24-bit parallel bus. The video format represented by the data on this bus is configurable through the HDMI core accessible via the I^2C port. Similar to the HDMI transmitter, the video port mapping registers allows six 4-bit nibbles from the video port to be mapped to six groups of four signals on the physical interface. Internally, the receiver uses a 36-bit video port. To map the physical port to the internal video port, the eight most significant bits for each color encoding. For example, when mapping the video port for RGB encoding, G[3:0], B[3:0], and R[3:0 are unused and the high bits for each color are mapped to the port pins.

Table 1.2: TDA19971 Video Port Mapping

Internal Port	RGB	YCbCr 4:4:4	YCbCr 4:2:2 (Semi-planar)
VP_i[35:32]	G[11:8]	Y[11:8]	Y[11:8]
VP_i[31:28]	G[7:4]	Y[7:4]	Y[7:4]
VP_i[27:24]	G[3:0]	Y[3:0]	Y[3:0]
VP_i[23:20]	B[11:8]	Cb[11:8]	-
VP_i[19:16]	B[7:4]	Cb[7:4]	-
VP_i[15:12]	B[3:0]	Cb[3:0]	-
VP_i[11:8]	R[11:8]	Cr[11:8]	CbCr[11:8]
VP_i[7:4]	R[7:4]	Cr[7:4]	CbCr[7:4]
VP_i[3:0]	R[3:0]	Cr[3:0]	CbCr[3:0]

1.5.5 Video Timing

The pixel clock from the HDMI transmitter drives a multi-region clock capable input pin. This allows the HDMI receiver to drive any logic that should handle the video input to programmable logic. The other timing signals are DE, HSYNC and VSYNC.

2

HDMI Transmit and Receive

Both HDMI receive and HDMI transmit subsystems are connected to a mounted snickerdoodle using a 24-bit video port. The video port, along with the video timing signals, can be used by programmable logic to capture and process the incoming video stream from the HDMI receiver as well as drive the output video to the HDMI transmitter.

2.0.1 Configuration in Linux

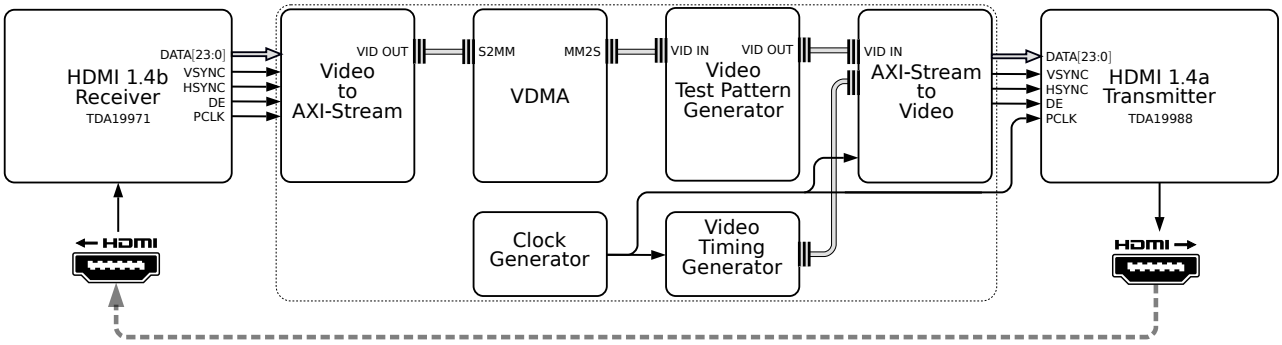


Figure 2.1: HDMI Video Test Configuration

2.1 Video Testing Configuration

2.1.1 Video Pass-Through

The video testing pipeline can be used to pass buffered input from the receiver to the transmitter. To do this, the VDMA is used to capture video data

from the receiver to frame buffer memory in DDR RAM. The frame buffer data is subsequently output by the VDMA to the transmitter.

VDMA GenLock

GenLock is a mechanism that can be used to share frame buffers between the receive and transmit channels. The linking of these channels involves the sharing of the current frame buffer address internally between a master and slave channel. For a pass-through configuration, meaning that the output will be buffered from the input, the receive channel is configured as a dynamic master and the transmit channel is configured as a dynamic slave. The receive channel will write to the shared set of frame buffers while skipping the frame buffer that is currently being read by the transmit channel. If configured for cyclic operation, the receive channel will continually cycle through and fill the frame buffers. When it encounters a frame buffer that is currently being used by the transmit channel, it skips to the next frame buffer address.

2.1.2 Video Loop-Back

A video loop-back configuration is a great way to test the performance and integrity of the system. Two things are required to complete a full loop-back design: methods for generating and capturing a video data stream. With the video in loop-back configuration, the captured data can be validated against the generation source data. With the configuration shown in Figure 2.1, the generation source can be either the test pattern generator or video frame buffer data as output by the VDMA.

2.2 Test Pattern Generator

The test pattern generator IP is capable of both passing an incoming video stream through to a video output sink as well as inserting or overlaying a predefined test pattern. The test pattern generator is capable of outputting 19 different pattern output stream backgrounds in addition to allowing the background to use the input video stream. The test pattern generator is also capable of generating a foreground box or crosshair pattern to overlay on the background video. The foreground and background patterns as well as the parameters used to define the size and motion speed of the overlay image are configured using a register set through the AXI4-Lite interface.

2.3 Color Conversion

To convert from one color space to another, a vector representing the color components can be operated on to yield the desired output color space components. To do this, each color components is organized into a vector with each element representing the Y, Cr, and Cb or R, G, and B components. The HDMI transmitter and receiver are both capable of performing this color conversion directly by configuring a set of registers. This conversion can also be bypassed altogether. The color conversion registers hold the values of a set of input and output offset vectors and a matrix used to perform the conversion. The equation representing an arbitrary color conversion is shown below.

$$\mathbf{b} = \mathbf{P}\,(\mathbf{a} + \mathbf{o}_{in}) + \mathbf{o}_{out} \tag{2.1}$$

2.3.1 Color Conversion Matrix

By setting the components of vector **a** to the input color space values, an arbitrary color conversion can be performed by setting the correct values for the conversion matrix **P** and the input and output offset vectors $\mathbf{o}_{in}$ and $\mathbf{o}_{out}$, respectively. In some cases the offset values are not used which simplifies the equation to simple matrix multiplication. For common color space conversions these component values are well known and can be referenced directly for setting the values of the matrix conversion registers. These registers are documented in Table A.8.

$$\begin{bmatrix} y/g \\ cr/r \\ cb/b \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \left(\begin{bmatrix} g/y \\ r/cr \\ b/cb \end{bmatrix} + \begin{bmatrix} o_{in,1} \\ o_{in,2} \\ o_{in,3} \end{bmatrix} \right) + \begin{bmatrix} o_{out,1} \\ o_{out,2} \\ o_{out,3} \end{bmatrix} \tag{2.2}$$

A

Appendix

A.1 Ethernet Connections

Signal	Connector Pin	Package Pin	Direction
ETH2_TXD3	JB1.5	T11	Out
ETH2_TXD2	JB1.7	T10	Out
ETH2_TXD1	JB1.11	P14	Out
ETH2_TXD0	JB1.13	R14	Out
ETH2_TXCLK	JB1.19	V13	Out
ETH2_TXEN	JB1.20	T14	Out
ETH2_RXD3	JB1.23	T16	In
ETH2_RXD2	JB1.25	U17	In
ETH2_RXD1	JB1.29	W14	In
ETH2_RXD0	JB1.31	Y14	In
ETH2_RXCLK	JB1.35	U14	In
ETH2_RXDV	JB1.37	U15	In
ETH1_PTP_E-T	JA1.29	J20	In/Out
ETH1_CLK125	JA2.35	L16	In

Table A.1: Ethernet Connector and Package Pins

Table A.2: Ethernet Connector and Package Pins

Signal	Connector Pin	Package Pin	Direction
ETH1_TXD3	JB1.6	U12	Out
ETH1_TXD2	JB1.8	T12	Out
ETH1_TXD1	JB1.12	W13	Out
ETH1_TXD0	JB1.14	V12	Out
ETH1_TXCLK	JB1.17	U13	Out
ETH1_TXEN	JB1.18	T15	Out
ETH1_RXD3	JB1.24	Y17	In
ETH1_RXD2	JB1.26	Y16	In
ETH1_RXD1	JB1.30	W15	In
ETH1_RXD0	JB1.32	V15	In
ETH1_RXDV	JB1.36	U19	In
ETH1_RXCLK	JB1.38	U18	In
ETH2_PTP_E-T	JA1.31	H20	In/Out
ETH2_CLK125	JA2.38	K17	In

A.2 Ethernet Constraints

```
# Clock
set_property PACKAGE_PIN L16 [get_ports ETH0_CLK125]
set_property IOSTANDARD LVCMOS33 [get_ports ETH0_CLK125]

create_clock -add -period 8.000 -name eth0_clk125 [get_ports
    ETH0_CLK125]

# MDIO
set_property PACKAGE_PIN U10 [get_ports ETH0_MDIO_mdc]
set_property PACKAGE_PIN T9 [get_ports ETH0_MDIO_mdio_io]

set_property IOSTANDARD LVCMOS33 [get_ports [list ETH0_MDIO_mdc
    ETH0_MDIO_mdio_io]]

# RGMII
set_property PACKAGE_PIN V15 [get_ports {ETH0_RGMII_rd[0]}]
set_property PACKAGE_PIN W15 [get_ports {ETH0_RGMII_rd[1]}]
set_property PACKAGE_PIN Y16 [get_ports {ETH0_RGMII_rd[2]}]
set_property PACKAGE_PIN Y17 [get_ports {ETH0_RGMII_rd[3]}]
set_property PACKAGE_PIN U19 [get_ports ETH0_RGMII_rx_ctl]
set_property PACKAGE_PIN U18 [get_ports ETH0_RGMII_rxc]

set_property IOSTANDARD LVCMOS33 [get_ports [list
    {ETH0_RGMII_rd[*]} ETH0_RGMII_rx_ctl ETH0_RGMII_rxc]]

create_clock -add -period 8.000 -name eth0_rgmii_rxclk [get_ports
    ETH0_RGMII_rxc]

set_property PACKAGE_PIN V12 [get_ports {ETH0_RGMII_td[0]}]
set_property PACKAGE_PIN W13 [get_ports {ETH0_RGMII_td[1]}]
set_property PACKAGE_PIN T12 [get_ports {ETH0_RGMII_td[2]}]
set_property PACKAGE_PIN U12 [get_ports {ETH0_RGMII_td[3]}]
set_property PACKAGE_PIN T15 [get_ports ETH0_RGMII_tx_ctl]
set_property PACKAGE_PIN U13 [get_ports ETH0_RGMII_txc]

set_property IOSTANDARD LVCMOS33 [get_ports [list
    {ETH0_RGMII_td[*]} ETH0_RGMII_tx_ctl ETH0_RGMII_txc]]
set_property SLEW FAST [get_ports [list {ETH0_RGMII_td[*]}
    ETH0_RGMII_tx_ctl ETH0_RGMII_txc]]
```

Code Listing A.1: piSmasher Ethernet 0 RGMII I/O and Timing Constraints

Code Listing A.2: piSmasher Ethernet 1 RGMII I/O and Timing Constraints

```
# Clock
set_property PACKAGE_PIN K17 [get_ports ETH1_CLK125]
set_property IOSTANDARD LVCMOS33 [get_ports ETH1_CLK125]

create_clock -add -period 8.000 -name eth1_clk125 [get_ports
    ETH1_CLK125]

# MDIO
set_property PACKAGE_PIN V7 [get_ports ETH1_MDIO_mdc]
set_property PACKAGE_PIN U7 [get_ports ETH1_MDIO_mdio_io]

set_property IOSTANDARD LVCMOS33 [get_ports [list ETH1_MDIO_mdc
    ETH1_MDIO_mdio_io]]

# RGMII
set_property PACKAGE_PIN Y14 [get_ports {ETH1_RGMII_rd[0]}]
set_property PACKAGE_PIN W14 [get_ports {ETH1_RGMII_rd[1]}]
set_property PACKAGE_PIN U17 [get_ports {ETH1_RGMII_rd[2]}]
set_property PACKAGE_PIN T16 [get_ports {ETH1_RGMII_rd[3]}]
set_property PACKAGE_PIN U15 [get_ports ETH1_RGMII_rx_ctl]
set_property PACKAGE_PIN U14 [get_ports ETH1_RGMII_rxc]

set_property IOSTANDARD LVCMOS33 [get_ports [list
    {ETH1_RGMII_rd[*]} ETH1_RGMII_rx_ctl ETH1_RGMII_rxc]]

create_clock -add -period 8.000 -name eth1_rgmii_rxclk [get_ports
    ETH1_RGMII_rxc]

set_property PACKAGE_PIN R14 [get_ports {ETH1_RGMII_td[0]}]
set_property PACKAGE_PIN P14 [get_ports {ETH1_RGMII_td[1]}]
set_property PACKAGE_PIN T10 [get_ports {ETH1_RGMII_td[2]}]
set_property PACKAGE_PIN T11 [get_ports {ETH1_RGMII_td[3]}]
set_property PACKAGE_PIN T14 [get_ports ETH1_RGMII_tx_ctl]
set_property PACKAGE_PIN V13 [get_ports ETH1_RGMII_txc]

set_property IOSTANDARD LVCMOS33 [get_ports [list
    {ETH1_RGMII_td[*]} ETH1_RGMII_tx_ctl ETH1_RGMII_txc]]
set_property SLEW FAST [get_ports [list {ETH1_RGMII_td[*]}
    ETH1_RGMII_tx_ctl ETH1_RGMII_txc]]
```

A.3 Audio Codec Connections

Signal	Connector/Pin	Package Pin
AC_WCLK	JA1.4	G14
AC_DIN	JB2.29	V20
AC_DOUT	JB2.31	W20
AC_MCLK	JB2.35	N20
AC_BLK	JB2.37	P20

Table A.3: Audio Connector and Package Pins

A.4 Audio Codec Interface Constraints

```
set_property PACKAGE_PIN V20 [get_ports AC_DIN]
set_property PACKAGE_PIN W20 [get_ports AC_DOUT]
set_property PACKAGE_PIN N20 [get_ports AC_MCLK]
set_property PACKAGE_PIN P20 [get_ports AC_BCLK]
set_property PACKAGE_PIN G14 [get_ports AC_WCLK]

set_property IOSTANDARD LVCMOS33 [get_ports [list AC_DIN AC_DOUT
    AC_MCLK AC_BCLK AC_WCLK]]
```

Code Listing A.3: piSmasher Audio Codec Constraints

A.5 HDMI Transmit Connections

Table A.4: HDMI Transmitter Connector and Package Pins

Signal	Connector Pin	Package Pin
HDMI_TX_DATA00	JA2.23	L19
HDMI_TX_DATA01	JA2.25	L20
HDMI_TX_DATA02	JA2.26	K19
HDMI_TX_DATA03	JB2.5	N17
HDMI_TX_DATA04	JB2.7	P18
HDMI_TX_DATA05	JB2.8	P15
HDMI_TX_DATA06	JB2.6	P16
HDMI_TX_DATA07	JB2.11	T17
HDMI_TX_DATA08	JA2.24	J19
HDMI_TX_DATA09	JA2.29	M17
HDMI_TX_DATA10	JB2.13	R18
HDMI_TX_DATA11	JB2.14	R16
HDMI_TX_DATA12	JB2.12	R17
HDMI_TX_DATA13	JB2.17	V17
HDMI_TX_DATA14	JB2.19	V18
HDMI_TX_DATA15	JB2.20	W18
HDMI_TX_DATA16	JA2.31	M18
HDMI_TX_DATA17	JA2.32	M19
HDMI_TX_DATA18	JA2.30	M20
HDMI_TX_DATA19	JB2.18	W19
HDMI_TX_DATA20	JB2.23	T20
HDMI_TX_DATA21	JB2.25	U20
HDMI_TX_DATA22	JB2.26	V16
HDMI_TX_DATA23	JB2.24	W16
HDMI_TX_DE	JB2.36	P19
HDMI_TX_VS	JB2.30	Y19
HDMI_TX_HS	JB2.32	Y18
HDMI_TX_LLC	JB2.38	N18
HDMI_TX_I2S0	JB2.4	R19
HDMI_TX_I2S1	JA2.4	J15
HDMI_TX_SCLK	JA2.17	N15
HDMI_TX_LRCLK	JA2.19	N16

A.6 HDMI Transmit Constraints

```
# Data
set_property PACKAGE_PIN L19 [get_ports {HDMI_TX_DATA[0]}]
set_property PACKAGE_PIN L20 [get_ports {HDMI_TX_DATA[1]}]
set_property PACKAGE_PIN K19 [get_ports {HDMI_TX_DATA[2]}]
set_property PACKAGE_PIN N17 [get_ports {HDMI_TX_DATA[3]}]
set_property PACKAGE_PIN P18 [get_ports {HDMI_TX_DATA[4]}]
set_property PACKAGE_PIN P15 [get_ports {HDMI_TX_DATA[5]}]
set_property PACKAGE_PIN P16 [get_ports {HDMI_TX_DATA[6]}]
set_property PACKAGE_PIN T17 [get_ports {HDMI_TX_DATA[7]}]
set_property PACKAGE_PIN J19 [get_ports {HDMI_TX_DATA[8]}]
set_property PACKAGE_PIN M17 [get_ports {HDMI_TX_DATA[9]}]
set_property PACKAGE_PIN R18 [get_ports {HDMI_TX_DATA[10]}]
set_property PACKAGE_PIN R16 [get_ports {HDMI_TX_DATA[11]}]
set_property PACKAGE_PIN R17 [get_ports {HDMI_TX_DATA[12]}]
set_property PACKAGE_PIN V17 [get_ports {HDMI_TX_DATA[13]}]
set_property PACKAGE_PIN V18 [get_ports {HDMI_TX_DATA[14]}]
set_property PACKAGE_PIN W18 [get_ports {HDMI_TX_DATA[15]}]
set_property PACKAGE_PIN M18 [get_ports {HDMI_TX_DATA[16]}]
set_property PACKAGE_PIN M19 [get_ports {HDMI_TX_DATA[17]}]
set_property PACKAGE_PIN M20 [get_ports {HDMI_TX_DATA[18]}]
set_property PACKAGE_PIN W19 [get_ports {HDMI_TX_DATA[19]}]
set_property PACKAGE_PIN T20 [get_ports {HDMI_TX_DATA[20]}]
set_property PACKAGE_PIN U20 [get_ports {HDMI_TX_DATA[21]}]
set_property PACKAGE_PIN V16 [get_ports {HDMI_TX_DATA[22]}]
set_property PACKAGE_PIN W16 [get_ports {HDMI_TX_DATA[23]}]

# Timing
set_property PACKAGE_PIN P19 [get_ports HDMI_TX_DE]
set_property PACKAGE_PIN Y18 [get_ports HDMI_TX_HS]
set_property PACKAGE_PIN Y19 [get_ports HDMI_TX_VS]
set_property PACKAGE_PIN N18 [get_ports HDMI_TX_PCLK]

set_property IOSTANDARD LVCMOS33 [get_ports [list
    {HDMI_TX_DATA[*]} HDMI_TX_DE HDMI_TX_HS HDMI_TX_VS
    HDMI_TX_PCLK]]
set_property SLEW FAST [get_ports [list {HDMI_TX_DATA[*]}
    HDMI_TX_DE HDMI_TX_HS HDMI_TX_VS HDMI_TX_PCLK]]

# Audio
set_property PACKAGE_PIN R19 [get_ports HDMI_TX_I2S0]
set_property PACKAGE_PIN J15 [get_ports HDMI_TX_I2S1]
set_property PACKAGE_PIN N15 [get_ports HDMI_TX_SCLK]
set_property PACKAGE_PIN N16 [get_ports HDMI_TX_LRCLK]

set_property IOSTANDARD LVCMOS33 [get_ports [list HDMI_TX_I2S0
    HDMI_TX_I2S1 HDMI_TX_SCLK HDMI_TX_LRCLK]]
```

Code Listing A.4: piSmasher HDMI Transmit I/O Pin Constraints

A.7 HDMI Transmit Registers

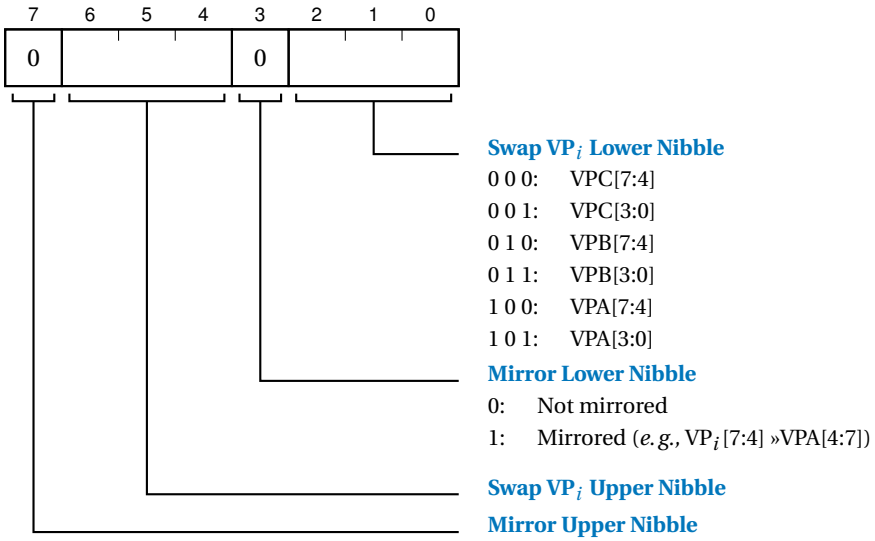


Figure A.1: Video Port Mapping Register Field

Table A.5: Video Port Pin Swap Configuration Registers (Page 0x00)

Address	Name	Default Value
0x20	VIP_CNTRL_0	`0000 0001` (0x01)
0x21	VIP_CNTRL_1	`0010 0100` (0x24)
0x22	VIP_CNTRL_2	`0101 0110` (0x56)

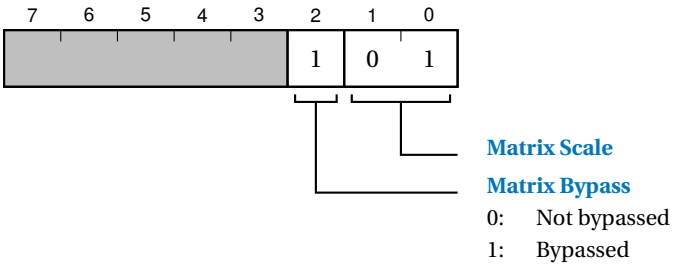


Figure A.2: Matrix Control Register (Register Address: 0x80 Page Address: 0x00)

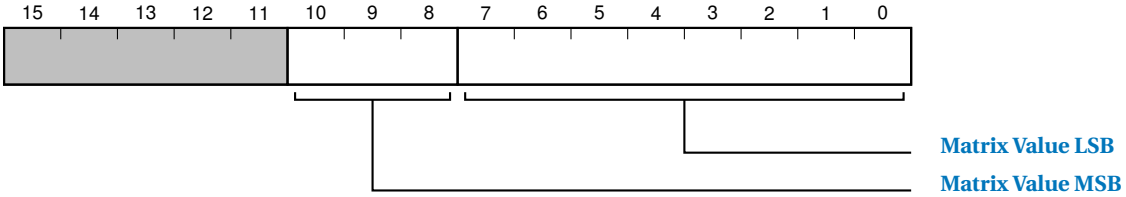


Figure A.3: Matrix Control Register (Register Address: 0x80 Page Address: 0x00)

Default matrix values are configured to perform YCbCr to RGB conversion.

Table A.6: Matrix Conversion Registers (Page 0x00)

Address	Name	Default Value
0x81	MAT_OI1_MSB	`0000 0000` (0x00)
0x82	MAT_OI1_LSB	`0000 0000` (0x00)
0x83	MAT_OI2_MSB	`0000 0110` (0x06)
0x84	MAT_OI2_LSB	`0000 0000` (0x00)
0x85	MAT_OI3_MSB	`0000 0110` (0x06)
0x86	MAT_OI3_LSB	`0000 0000` (0x00)

Table A.7: Matrix Conversion Registers (Page 0x00)

Address	Name	Default Value
0x87	MAT_P11_MSB	0000 0010 (0x02)
0x88	MAT_P11_LSB	0000 0000 (0x00)
0x89	MAT_P12_MSB	0000 0110 (0x06)
0x8A	MAT_P12_LSB	1001 0010 (0x92)
0x8B	MAT_P13_MSB	0000 0111 (0x07)
0x8C	MAT_P13_LSB	0101 0000 (0x50)
0x8D	MAT_P21_MSB	0000 0010 (0x02)
0x8E	MAT_P21_LSB	0000 0000 (0x00)
0x8F	MAT_P22_MSB	0000 0010 (0x02)
0x90	MAT_P22_LSB	1100 1110 (0xCE)
0x91	MAT_P23_MSB	0000 0110 (0x00)
0x92	MAT_P23_LSB	0000 0000 (0x00)
0x93	MAT_P31_MSB	0000 0010 (0x02)
0x94	MAT_P31_LSB	0000 0000 (0x00)
0x95	MAT_P32_MSB	0000 0000 (0x00)
0x96	MAT_P32_LSB	0000 0000 (0x00)
0x97	MAT_P33_MSB	0000 0011 (0x03)
0x98	MAT_P33_LSB	1000 1100 (0x8C)

Table A.8: Matrix Conversion Registers (Page 0x00)

Address	Name	Default Value
0x99	MAT_OO1_MSB	0000 0000 (0x00)
0x9A	MAT_OO1_LSB	0000 0000 (0x00)
0x9B	MAT_OO2_MSB	0000 0000 (0x00)
0x9C	MAT_OO2_LSB	0000 0000 (0x00)
0x9D	MAT_OO3_MSB	0000 0000 (0x00)
0x9E	MAT_OO3_LSB	0000 0000 (0x00)

A.8 HDMI Receive Connections

Signal	Connector Pin	Package Pin
HDMI_RX_DATA00	JA2.5	L14
HDMI_RX_DATA01	JA2.7	L15
HDMI_RX_DATA02	JA2.8	K16
HDMI_RX_DATA03	JA1.5	E18
HDMI_RX_DATA04	JA1.7	E19
HDMI_RX_DATA05	JA1.8	D19
HDMI_RX_DATA06	JA1.6	D20
HDMI_RX_DATA07	JA1.11	F16
HDMI_RX_DATA08	JA2.6	J16
HDMI_RX_DATA09	JA2.11	M14
HDMI_RX_DATA10	JA1.13	F17
HDMI_RX_DATA11	JA1.14	C20
HDMI_RX_DATA12	JA1.12	B20
HDMI_RX_DATA13	JA1.17	E17
HDMI_RX_DATA14	JA1.19	D18
HDMI_RX_DATA15	JA1.20	B19
HDMI_RX_DATA16	JA2.13	M15
HDMI_RX_DATA17	JA2.14	H15
HDMI_RX_DATA18	JA2.12	G15
HDMI_RX_DATA19	JA1.18	A20
HDMI_RX_DATA20	JA1.23	F19
HDMI_RX_DATA21	JA1.25	F20
HDMI_RX_DATA22	JA1.26	G19
HDMI_RX_DATA23	JA1.24	G20
HDMI_RX_DE	JA1.36	H17
HDMI_RX_VS	JA1.30	G18
HDMI_RX_HS	JA1.32	G17
HDMI_RX_PCLK	JA1.38	H16
HDMI_RX_AP0	JA1.37	H18
HDMI_RX_AP1	JA1.35	J18
HDMI_RX_SCLK	JA2.20	K14
HDMI_RX_LRCLK	JA2.18	J15

Table A.9: HDMI Receiver Connector and Package Pins

A.9 HDMI Receive Constraints

Code Listing A.5: piSmasher HDMI Receive I/O Pin Constraints

```
# Data
set_property PACKAGE_PIN L14 [get_ports {HDMI_RX_DATA[0]}]
set_property PACKAGE_PIN L15 [get_ports {HDMI_RX_DATA[1]}]
set_property PACKAGE_PIN K16 [get_ports {HDMI_RX_DATA[2]}]
set_property PACKAGE_PIN E18 [get_ports {HDMI_RX_DATA[3]}]
set_property PACKAGE_PIN E19 [get_ports {HDMI_RX_DATA[4]}]
set_property PACKAGE_PIN D19 [get_ports {HDMI_RX_DATA[5]}]
set_property PACKAGE_PIN D20 [get_ports {HDMI_RX_DATA[6]}]
set_property PACKAGE_PIN F16 [get_ports {HDMI_RX_DATA[7]}]
set_property PACKAGE_PIN J16 [get_ports {HDMI_RX_DATA[8]}]
set_property PACKAGE_PIN M14 [get_ports {HDMI_RX_DATA[9]}]
set_property PACKAGE_PIN F17 [get_ports {HDMI_RX_DATA[10]}]
set_property PACKAGE_PIN C20 [get_ports {HDMI_RX_DATA[11]}]
set_property PACKAGE_PIN B20 [get_ports {HDMI_RX_DATA[12]}]
set_property PACKAGE_PIN E17 [get_ports {HDMI_RX_DATA[13]}]
set_property PACKAGE_PIN D18 [get_ports {HDMI_RX_DATA[14]}]
set_property PACKAGE_PIN B19 [get_ports {HDMI_RX_DATA[15]}]
set_property PACKAGE_PIN M15 [get_ports {HDMI_RX_DATA[16]}]
set_property PACKAGE_PIN H15 [get_ports {HDMI_RX_DATA[17]}]
set_property PACKAGE_PIN G15 [get_ports {HDMI_RX_DATA[18]}]
set_property PACKAGE_PIN A20 [get_ports {HDMI_RX_DATA[19]}]
set_property PACKAGE_PIN F19 [get_ports {HDMI_RX_DATA[20]}]
set_property PACKAGE_PIN F20 [get_ports {HDMI_RX_DATA[21]}]
set_property PACKAGE_PIN G19 [get_ports {HDMI_RX_DATA[22]}]
set_property PACKAGE_PIN G20 [get_ports {HDMI_RX_DATA[23]}]

# Timing
set_property PACKAGE_PIN H17 [get_ports HDMI_RX_DE]
set_property PACKAGE_PIN G17 [get_ports HDMI_RX_HS]
set_property PACKAGE_PIN G18 [get_ports HDMI_RX_VS]
set_property PACKAGE_PIN H16 [get_ports HDMI_RX_PCLK]

set_property IOSTANDARD LVCMOS33 [get_ports [list
    {HDMI_RX_DATA[*]} HDMI_RX_DE HDMI_RX_HS HDMI_RX_VS
    HDMI_RX_PCLK]]

# Audio
set_property PACKAGE_PIN H18 [get_ports HDMI_RX_I2S0]
set_property PACKAGE_PIN J18 [get_ports HDMI_RX_I2S1]
set_property PACKAGE_PIN J14 [get_ports HDMI_RX_LRCLK]
set_property PACKAGE_PIN K14 [get_ports HDMI_RX_SCLK]

set_property IOSTANDARD LVCMOS33 [get_ports [list HDMI_RX_I2S0
    HDMI_RX_I2S1 HDMI_RX_LRCLK HDMI_RX_SCLK]]
```

A.10 HDMI Receive Video Port

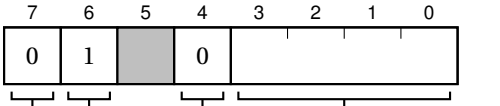


Figure A.4: Video Port Mapping Register

Video Port Map

0 0 0 0: VP_i [3:0]
0 0 0 1: VP_i [7:4]
0 0 1 0: VP_i [11:8]
0 0 1 1: VP_i [15:12]
0 1 0 0: VP_i [19:16]
0 1 0 1: VP_i [23:20]
0 1 1 0: VP_i [27:24]
0 1 1 1: VP_i [31:28]
1 0 0 0: VP_i [35:32]

Swap Bit Allocated

0: Disabled
1: Active

Disabled Output Mode

0: Pins pulled to gnd
1: High impedance (Hi-Z)

Enable Video Port Nibble

0: Disabled
1: Active

List of Acronyms

DE data enable

DMA direct memory access

EDID extended display identification data

HDMI high-definition multimedia interface

HSYNC horizontal synchronization

I²C inter-integrated circuit

I²S inter-IC sound

MDIO management data input/output

PHY physical layer

RGMII reduced gigabit media independent interface

TMDS transition-minimized differential signaling

ULPI UTMI+ low pin interface

USB universal serial bus

UTMI+ USB 2.0 transceiver macrocell interface

VDMA video direct memory access

VSYNC vertical synchronization