HDMI Source/Sink Modules Documentation

Release 0.0.1

srivatsan-ramesh

Mar 03, 2017

Contents

1 Requirements **3**

2 Getting Started **5**

3 Running Tests **7**

4 Using Models **9**
4.1 API Documentation 10

Python Module Index **19**

Implementation of HDMI Source/Sink Modules in MyHDL (http://www.myhdl.org/). Xilinx application notes xapp460 and xapp495 were used as reference.

CHAPTER 1

Requirements

- Python 3.5
- MyHDL 1.0.dev0

CHAPTER 2

Getting Started

Clone this repository to get started.

```
git clone https://github.com/srivatsan-ramesh/HDMI-Source-Sink-Modules.git
```

The project requires MyHDL version 1.0dev. The version has not been released (yet) and needs to be installed manually.

```
cd HDMI-Source-Sink-Modules/
pip install -r requirements.txt
```

After the dependencies have been installed, install the project.

```
python setup.py install
```

CHAPTER 3

Running Tests

The tests can be run using pytest.

```
cd tests/
py.test
```

CHAPTER 4

Using Models

HDMI Source/Sink Models are non-convertible modules which can be used to simulate the behaviour of HDMI Source/Sink Modules. To initialize the interfaces -

```
# Interfaces for Tx Model

video_interface_tx = VideoInterface(clock)
aux_interface_tx = AuxInterface(clock)
hdmi_interface_tx = HDMIInterface(clock10x)

# Interfaces for Rx Model

video_interface_rx = VideoInterface(clock)
aux_interface_rx = AuxInterface(clock)
hdmi_interface_rx = HDMIInterface(clock10x)
```

Here the two signals clock signals are driven by the clock_driver() block.

```
driver = clock_driver(clock)
```

The 'clock10x' signal should have a frequency 10 times that of clock signal.

The models can be initialized as -

```
hdmi_tx_model = HDMITxModel(clock, reset,
                            video_interface_tx, aux_interface_tx, hdmi_interface_tx)
hdmi_rx_model = HDMIRxModel(video_interface_rx, aux_interface_rx, hdmi_interface_rx)
```

To simulate their process make use of the process() block of the models.

```
hdmi_tx_inst = hdmi_tx_model.process()
hdmi_rx_inst = hdmi_rx_model.process()
```

Contents:

API Documentation

Interfaces

class hdmi.interfaces.aux_interface.**AuxInterface**(*clock*, *aux_depth=(4, 4, 4)*)

disable_aux()
Makes the ADE signal 0

enable_aux()
Makes the ADE signal 1

get_ade()

Returns ADE signal value

Return type int

get_aux_data()

Returns A list of aux signal values

Return type list

read_aux()

write_aux(*aux0*, *aux1*, *aux2*)
Transactor for passing signals to audio interface

Parameters

- **aux0** – The auxiliary data(usually audio data)
- **aux1** – The auxiliary data(usually audio data)
- **aux2** – The auxiliary data(usually audio data)

Example

```
# Values passed should be non negative integers less than 2**aux_depth[i]
yield aux_interface.write_aux(2, 2, 2)
```

class hdmi.interfaces.video_interface.**VideoInterface**(*clock*, *resolution=(640, 480)*, *color_depth=(8, 8, 8)*)

disable_video()
Makes the VDE signal 0

enable_video()
Makes the VDE signal 1

get_pixel()

Returns pixel values R, G, B

Return type list

get_vde()

Returns the VDE signal value

Return type int

read_pixel()

reset_cursor()
Resets the horizontal and vertical counters to 0.

write_pixel(*pixel*)
Write transactor for passing signals to video interface

Parameters **pixel** – The pixel value is a tuple (R, G, B)

Example

```
# Values passed should be non negative integers less than 2**color_depth[i]
video_interface.write_pixel(3, 4, 5)
```

class hdmi.interfaces.hdmi_interface.**HDMIInterface**(*clock10x*)

get_TMDS_data()

read_data()

write_data(*TMDS_R*, *TMDS_G*, *TMDS_B*, *TMDS_CLK*)
Write transactor for passing signals to external HDMI interface

Parameters

- **TMDS_R** – Serialized TMDS encoded video data
- **TMDS_G** – Serialized TMDS encoded video data
- **TMDS_B** – Serialized TMDS encoded video data
- **TMDS_CLK** – Clock used by the sink to recover data

Example

```
# The values passed should be 1(or True) or 0(or False).
yield hdmi_interface.write_data(0, 0, 0, 0)
```

HDMI Models

class hdmi.models.encoder_model.**EncoderModel**(*clock*, *reset*, *video_in*, *audio_in*, *c0*, *c1*, *vde*, *ade*, *data_out*, *channel='BLUE'*)
A non convertible model to simulate the behaviour of a TMDS and TERC4 encoder.

Parameters

- **clock** – pixel clock as input
- **reset** – asynchronous reset input (active high)
- **video_in** – video input of a single channel
- **audio_in** – audio input
- **c0** – used to determine preamble
- **c1** – used to determine preamble

- **vde** – video data enable
- **ade** – audio data enable
- **data_out** – 10 bit parallel output
- **channel** – Indicates 'RED', 'GREEN' or 'BLUE' channel

Example

```
encoder_model = EncoderModel(*params)
process_inst = encoder_model.process()
process_inst.run_sim()
```

process = <myhdl._block._bound_function_wrapper object>

class hdmi.models.decoder_model.**DecoderModel**(*clock*, *data_in*, *video_preamble*, *data_island_preamble*, *c0*, *c1*, *vde*, *ade*, *video_out*, *audio_out*, *channel='BLUE'*)

A non-convertible HDMI Decoder Model which decodes the TMDS data and outputs the video and aux data. This is modelled after the xapp495 decoder module.

Parameters

- **clock** – The system clock or the pixel clock
- **data_in** – The TMDS data (10 bits width) to be decoded
- **video_preamble** – signal to detect the video preamble in the input data
- **data_island_preamble** – signal to detect the data island preamble in the input data
- **c0** – Control signal (hsync for Blue channel)
- **c1** – Control signal (vsync for Blue channel)
- **vde** – Video Data enable
- **ade** – Audio data enable
- **video_out** – Output video data
- **audio_out** – Output audio (or aux) data.
- **channel** – Color of the channel ('RED', 'GREEN' or 'BLUE'). Default value is 'BLUE'

Example

```
decoder_model = DecoderModel(*params)
process_inst = decoder_model.process()
process_inst.run_sim()
```

process = <myhdl._block._bound_function_wrapper object>

write_data(*data_in*)

Writes the given data onto the input data signal

Parameters **data_in** – 10 bit intbv value(or a 10 bit integer)

class hdmi.models.hdmi_tx_model.**HDMITxModel**(*clock*, *reset*, *video_interface*, *aux_interface*, *hdmi_interface*)

A non-convertible HDMI Transmitter Model which encodes the input video and AUX data and transmits it. This is modelled after the xapp495 HDMI Tx module.

Parameters

- **clock** – System clock or the pixel clock
- **reset** – Reset signal
- **video_interface** – An instance of the VideoInterface class
- **aux_interface** – An instance of the AUXInterface class
- **hdmi_interface** – An instance of the HDMIInterface class

Example

```
hdmi_tx_model = HDMITxModel(*params)
process_inst = hdmi_tx_model.process()
process_inst.run_sim()
```

process = <myhdl._block._bound_function_wrapper object>

class hdmi.models.hdmi_rx_model.**HDMIRxModel**(*video_interface*, *aux_interface*, *hdmi_interface*)

A non-convertible HDMI Transmitter Model which encodes the input video and AUX data and transmits it. This is modelled after the xapp495 HDMI Tx module.

Parameters

- **video_interface** – An instance of the VideoInterface class
- **aux_interface** – An instance of the AUXInterface class
- **hdmi_interface** – An instance of the HDMIInterface class

Example

```
hdmi_rx_model = HDMIRxModel(*params)
process_inst = hdmi_rx_model.process()
process_inst.run_sim()
```

process = <myhdl._block._bound_function_wrapper object>

HDMI Transmitter

hdmi.cores.transmitter.hdmi_encoder.**hdmi_encoder**

This block implements the HDMI encoder module modelled after the xapp 495 application notes.

Parameters

- **p_clock** – The pixel clock
- **p_clockx2** – The clock with twice the frequency of the pixel clock
- **p_clockx10** – The clock with ten times the frequency of the pixel clock
- **reset** – An asynchronous reset signal

- **serdes_strobe** – Serdes strobe for serdes blocks
- **video_interface** – An instance of VideoInterface
- **aux_interface** – An instance of AuxInterface
- **hdmi_interface** – An instance of HDMIInterface

Returns A list of myhdl instances.

Return type myhdl.instances()

hdmi.cores.transmitter.encode.**encode**

This module performs the TMDS encoding logic of a hdmi encoder for a particular channel. It is modelled after the xilinx application notes xapp460 and xapp495.

Parameters

- **clock** – The pixel clock
- **reset** – An asynchronous reset signal
- **video_in** – input video data
- **audio_in** – input audio data
- **c0** – control signal (hsync for BLUE channel)
- **c1** – control signal (vsync for BLUE channel)
- **vde** – video data enable
- **ade** – auxiliary data enable
- **data_out** – output encoded 10 bit data
- **channel** – The color of the channel (Default: BLUE)

Returns A list of myhdl instances.

Return type myhdl.instances()

hdmi.cores.transmitter.convert_30_to_15.**convert_30_to_15**

The block converts the 30-bit data into 15-bit data.

Parameters

- **reset** – The reset signal
- **clock** – The pixel clock
- **clockx2** – The clock with twice the frequency of pixel clock
- **data_in** – The input 30-bit data
- **tmds_data2** – 5 bits of the output data (output[15:10])
- **tmds_data1** – 5 bits of the output data (output[10:5])
- **tmds_data0** – 5 bits of the output data (output[5:0])

Returns A list of myhdl instances.

Return type myhdl.instances()

hdmi.cores.transmitter.serdes_n_to_1.**serdes_n_to_1**

The block converts the n-bit parallel data into a serial data. This block will be replaced with the xilinx primitive OSERDES2 during conversion.

Parameters

- **io_clock** – The clock used in the output side (For serial data)
- **serdes_strobe** – The signal is used in the xilinx primitive
- **reset** – The reset signal
- **g_clock** – Clock on the input side (For the parallel data)
- **data_in** – The input parallel data n-bit
- **iob_data_out** – Output serial data
- **factor** – width of the input data

Returns A list of myhdl instances.

Return type myhdl.instances()

HDMI Receiver

hdmi.cores.receiver.hdmi_decoder.**hdmi_decoder**

This block implements the HDMI decoder module modelled after the xapp 495 application notes.

Parameters

- **ext_reset** – An external reset signal
- **hdmi_interface** – An instance of HDMIInterface
- **video_interface** – An instance of VideoInterface
- **aux_interface** – An instane of AuxInterface

Returns A list of myhdl instances.

Return type myhdl.instances()

hdmi.cores.receiver.decode.**decode**

This module performs the decoding logic of a hdmi decoder for a particular channel. It is modelled after the xilinx application notes xapp460 and xapp495.

Parameters

- **reset** – An asynchronous reset signal that resets the output.
- **p_clock** – The pixel clock
- **p_clockx2** – Clock with twice the frequency of pixel clock
- **p_clockx10** – Clock with ten times the frequency of pixel clock
- **serdes_strobe** – The signal used in serdes block
- **data_in_p** – Differential signal input
- **data_in_n** – Differential signal input
- **other_ch0_valid** – Denotes the validity of other channel data
- **other_ch1_valid** – Denotes the validity of other channel data
- **other_ch0_ready** – Denotes the readiness of other channel data
- **other_ch1_ready** – Denotes the readiness of other channel data
- **video_preamble** – Denotes the video preamble region of the input data
- **data_island_preamble** – Denotes the data island preamble region of the input data

- **i_am_ready** – Denotes the readiness of this channel data
- **i_am_valid** – Denotes the validity of this channel data
- **phase_align_err** – Becomes True when there is an error in aligning the phase
- **c0** – Control signal (hsync in BLUE channel)
- **c1** – Control signal (vsync in BLUE channel)
- **vde** – Video data enable
- **ade** – Auxiliary data enable
- **s_data_out** – The raw 10 bit data obtained from TMDS channels
- **video_out** – The decoded video data
- **audio_out** – The decoded audio data
- **channel** – The color of the channel (Default: BLUE)

Returns A list of myhdl instances.

Return type myhdl.instances()

hdmi.cores.receiver.phase_aligner.**phase_aligner**

This impliments the phase alignment logic modelled after the phase aligner module from xapp495 application notes.

Parameters

- **reset** – reset signal
- **clock** – pixel clock
- **s_data** – the 10-bit TMDS data
- **bit_slip** – An input signal
- **flip_gear** – An input signal
- **phase_aligned** – denotes whether the phase is aligned

Returns A list of myhdl instances.

Return type myhdl.instances()

hdmi.cores.receiver.channel_bonding.**channel_bonding**

This module implements the channel bonding logic and TMDS channel de-skew logic. It is modelled after the xapp495 channel_bonding module

Parameters

- **clock** – The global clock or the pixel clock
- **raw_data** – The 10 bit raw data obtained by de-serializing the TMDS data
- **i_am_valid** – A signal used to denote the validity of this channel
- **other_ch0_valid** – A signal used to denote the validity of other channel
- **other_ch1_valid** – A signal used to denote the validity of other channel
- **other_ch0_ready** – A signal used to denote the readiness of other channel
- **other_ch1_ready** – A signal used to denote the readiness of other channel
- **i_am_ready** – A signal used to denote the readiness of this channel

- **s_data** – The 10-bit output data after channel de-skew

Returns A list of myhdl instances.

Return type myhdl.instances()

hdmi.cores.receiver.serdes_1_to_5.**serdes_1_to_5**

The block converts the serial data into a 5 bit parallel data. This block will be replaced with the xilinx primitives IODELAY2 and ISERDES2 during conversion.

Parameters

- **use_phase_detector** – The signal is used by the xilinx primitive
- **data_in_p** – The input differential data
- **data_in_n** – The input differential data
- **rx_io_clock** – The clock from the input side (serial data)
- **rx_serdes_strobe** – The signal is used by the xilinx primitive
- **reset** – The signal is used by the xilinx primitive
- **g_clock** – The clock on the output side (parallel data)
- **bit_slip** – The signal is used by the xilinx primitive
- **data_out** – The output parallel data (5-bit)
- **diff_term** – The parameter is used by the xilinx primitive
- **bit_slip_enable** – The parameter is used by the xilinx primitive
- **sim_tap_delay** – The parameter is used by the xilinx primitive

Returns A list of myhdl instances.

Return type myhdl.instances()

Python Module Index

h

hdmi.cores.receiver.channel_bonding, 16
hdmi.cores.receiver.decode, 15
hdmi.cores.receiver.hdmi_decoder, 15
hdmi.cores.receiver.phase_aligner, 16
hdmi.cores.receiver.serdes_1_to_5, 17
hdmi.cores.transmitter.convert_30_to_15, 14
hdmi.cores.transmitter.encode, 14
hdmi.cores.transmitter.hdmi_encoder, 13
hdmi.cores.transmitter.serdes_n_to_1, 14
hdmi.interfaces.aux_interface, 10
hdmi.interfaces.hdmi_interface, 11
hdmi.interfaces.video_interface, 10
hdmi.models.decoder_model, 12
hdmi.models.encoder_model, 11
hdmi.models.hdmi_rx_model, 13
hdmi.models.hdmi_tx_model, 12

Index

A

AuxInterface (class in hdmi.interfaces.aux_interface), 10

C

channel_bonding (in module hdmi.cores.receiver.channel_bonding), 16
convert_30_to_15 (in module hdmi.cores.transmitter.convert_30_to_15), 14

D

decode (in module hdmi.cores.receiver.decode), 15
DecoderModel (class in hdmi.models.decoder_model), 12
disable_aux() (hdmi.interfaces.aux_interface.AuxInterface method), 10
disable_video() (hdmi.interfaces.video_interface.VideoInterface method), 10

E

enable_aux() (hdmi.interfaces.aux_interface.AuxInterface method), 10
enable_video() (hdmi.interfaces.video_interface.VideoInterface method), 10
encode (in module hdmi.cores.transmitter.encode), 14
EncoderModel (class in hdmi.models.encoder_model), 11

G

get_ade() (hdmi.interfaces.aux_interface.AuxInterface method), 10
get_aux_data() (hdmi.interfaces.aux_interface.AuxInterface method), 10
get_pixel() (hdmi.interfaces.video_interface.VideoInterface method), 10
get_TMDS_data() (hdmi.interfaces.hdmi_interface.HDMIInterface method), 11
get_vde() (hdmi.interfaces.video_interface.VideoInterface method), 10

H

hdmi.cores.receiver.channel_bonding (module), 16
hdmi.cores.receiver.decode (module), 15
hdmi.cores.receiver.hdmi_decoder (module), 15
hdmi.cores.receiver.phase_aligner (module), 16
hdmi.cores.receiver.serdes_1_to_5 (module), 17
hdmi.cores.transmitter.convert_30_to_15 (module), 14
hdmi.cores.transmitter.encode (module), 14
hdmi.cores.transmitter.hdmi_encoder (module), 13
hdmi.cores.transmitter.serdes_n_to_1 (module), 14
hdmi.interfaces.aux_interface (module), 10
hdmi.interfaces.hdmi_interface (module), 11
hdmi.interfaces.video_interface (module), 10
hdmi.models.decoder_model (module), 12
hdmi.models.encoder_model (module), 11
hdmi.models.hdmi_rx_model (module), 13
hdmi.models.hdmi_tx_model (module), 12
hdmi_decoder (in module hdmi.cores.receiver.hdmi_decoder), 15
hdmi_encoder (in module hdmi.cores.transmitter.hdmi_encoder), 13
HDMIInterface (class in hdmi.interfaces.hdmi_interface), 11
HDMIRxModel (class in hdmi.models.hdmi_rx_model), 13
HDMITxModel (class in hdmi.models.hdmi_tx_model), 12

P

phase_aligner (in module hdmi.cores.receiver.phase_aligner), 16
process (hdmi.models.decoder_model.DecoderModel attribute), 12
process (hdmi.models.encoder_model.EncoderModel attribute), 12
process (hdmi.models.hdmi_rx_model.HDMIRxModel attribute), 13
process (hdmi.models.hdmi_tx_model.HDMITxModel attribute), 13

R

read_aux() (hdmi.interfaces.aux_interface.AuxInterface method), 10
read_data() (hdmi.interfaces.hdmi_interface.HDMIInterface method), 11
read_pixel() (hdmi.interfaces.video_interface.VideoInterface method), 11
reset_cursor() (hdmi.interfaces.video_interface.VideoInterface method), 11

S

serdes_1_to_5 (in module hdmi.cores.receiver.serdes_1_to_5), 17
serdes_n_to_1 (in module hdmi.cores.transmitter.serdes_n_to_1), 14

V

VideoInterface (class in hdmi.interfaces.video_interface), 10

W

write_aux() (hdmi.interfaces.aux_interface.AuxInterface method), 10
write_data() (hdmi.interfaces.hdmi_interface.HDMIInterface method), 11
write_data() (hdmi.models.decoder_model.DecoderModel method), 12
write_pixel() (hdmi.interfaces.video_interface.VideoInterface method), 11