The Andrew and Erna Viterbi Faculty of ELECTRICAL & COMPUTER ENGINEERING

TECHNION Israel Institute of Technology

Technion – Israel Institute of Technology

Electrical & Computer Engineering

VLSI Lab

BSc Project Final Report

Hardware Implementation of a Video Enhancement Algorithm

Tamir Shriki 206924862

Tom Eitan Melamed 206029639

Supervisor Pavel Gushpan

Lab Engineer Goel Samuel

Table of Contents

1. Abstract

With the ever-increasing availability of high-quality cameras and video recording devices, the demand for enhanced video quality has also increased significantly in recent years. Enhancement algorithms are a crucial component of modern video processing techniques that help to improve the visual quality of videos, particularly useful in situations where the video quality is compromised due to factors such as low lighting, camera shake, or compression artifacts.

In this project we designed and verified a hardware module for 24bit RGB digital video enhancement, to fulfill that demand.
The module implements brightness, contrast and LUT mapping operations (intended for gamma correction) on continuous digital video input with constant delay. IO interfaces are AXI-Stream for video, with pixel per transaction transfers, and AXI-Lite for control and parameters; potentially operating at different clocks. Parameters and control change is supported in-runtime, with constant delay from transaction to video output. The data path is synchronized with AXI-Stream video pipeline without buffering, thus varied resolutions and FPS are supported.

Functional simulation of operating scenarios was performed, and results verified vs reference model.

Support for at least HD 30FPS video bitrate was achieved with total power of ~60mW and total area of ~6.3mm^2 in 180nm technology.

2. Background

2.1. Digital Image

A digital image or specifically a raster image is a 2D representation of a picture as a matrix of square pixels. It is characterized by resolution (Width x Height), color space and color depth.

A pixel is the smallest unit of an image that can be displayed on a digital device. It stands for "picture element". Each pixel has its own color, brightness value and refers to a certain channel / base vector in color space.

Color space refers to a system of organizing colors in a way that can be accurately represented on digital devices. The most common color spaces used in digital imaging is RGB, comprised of 3 channels – red, green, blue – with a color depth of 24bits total or 8bits per channel (uint8 is used to minimally store channel data). RGB is an additive color space, which means that it adds light to produce colors. It's used primarily for digital displays, such as computer monitors and TVs.

However, there are alternative representations for color modeling which aren't additive such as: HSV, HSL, and HSB (where H = Hue, represents the color itself , S = saturation, represents the intensity or purity of the color, B = brightness , L = lightness, V = value).

<u>Mathematical representation of a 24bit RGB image frame:</u>

Where m, n are spatial coordinates of a pixel, starting from top-left corner of frame.

$$f[m,n,ch] \in [0,255] = channel\ of\ pixel\ in\ 8bit\ RGB\ space\ (uint8\ representation)$$

Therefore, all operations results on 8bit RGB pixels must be bounded per channel:

$$f[m,n,ch]_{out} = \begin{cases} 0 & f[m,n,ch]_{out'} \leq 0 \\ 255 & f[m,n,ch]_{out'} \geq 255 \\ f[m,n,ch]_{out'} & else \end{cases}$$

The RGB pixel is then comprised of the 3 channels combined in the following order:

$$f[m,n] = \left[f[m,n,R], f[m,n,G], f[m,n,B]\right] = pixel\ of\ frame, in\ RGB\ space$$

2.2. Digital Video

Digital video is comprised of a series of images, called frames, quickly changing from one to the next. This rapidly progressing sequence of images produces the illusion of motion that we perceive and call video.

$$video = \{frame_t\}_t$$

FPS – Frames Per Second - is the rate of frame change. The more frames per second, the smoother the video appears.

Resolution of video refers to resolution of each frame. Higher resolution video provides more detail and clarity, but also requires more storage and processing power.

Bitrate refers to the amount of data that is transferred per second in a video. It's measured in kilobits per second (Kbps) or megabits per second (Mbps). The higher the bitrate, the higher the quality of the video. However, higher bitrates also require more storage and bandwidth.

2.3. Digital Video Enhancement Algorithms

The goal of these algorithms is to improve the visual quality of videos by correcting for various issues such as low lighting, poor contrast, and color distortion. Those issues can arise from sub-optimal conditions during the video shoot or from internal characteristics of the capturing and reproducing equipment. The most basic ones are gamma correction, contrast and brightness.
With the advent of digital video and the proliferation of video cameras and devices, the need for effective video enhancement algorithms has grown.

2.3.1. Gamma correction

Gamma correction is a pixel-wise technique used to adjust the brightness of an image or video to correct for non-linearity in the display of an image or video.
When an image or video is captured or transmitted, it may be affected by non-linearity in the display, which can cause the image or video to appear darker or brighter than it should.
Main reason for using gamma correction derived from the fact that the human eye does not perceive changes in brightness in a linear fashion; instead, it has a logarithmic response to changes in light intensity.

Therefor gamma correction can compensate for this non-linear response by adjusting the brightness levels of digital images or videos.
The gamma parameter value used for correction represents the relationship between the brightness of the input signal and the brightness of the output display.
For example, if an image has a gamma value of 2.2, it means that the image's brightness values have been adjusted to match the non-linear response of the human eye.
This helps to ensure that the image appears natural and balanced to the viewer.

$$\gamma \in \mathbb{R}^{+} = gamma\ correction\ parameter$$

$$f[m,n,ch]_{out} = round\left\{255\left(\frac{f[m,n,ch]_{in}}{255}\right)^{\gamma}\right\}$$

Equation 1 - Gamma correction on single 8bit channel

For different values of γ, the relation of x and y in figure 1 shows that:

1) When γ = 1 , the corrected gray-scale value is consistent with the gray-scale value before correction.

2) When γ < 1 , the gray-scale value of the corrected image is higher than that before correction, and it increases obviously in the low gray-scale value regions;.

3) When γ > 1 , the gray-scale value after correction is smaller than that before correction, and it decreases obviously in high gray-scale value regions.

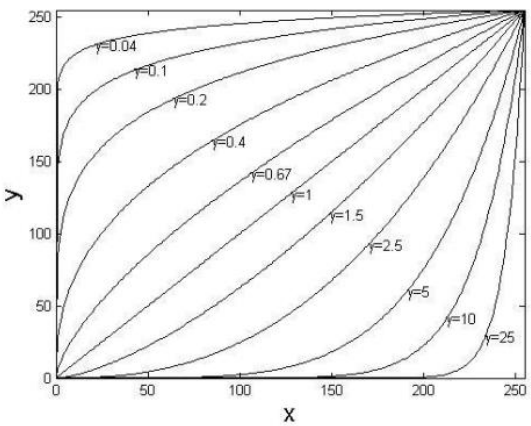

Figure 1 - The relationship between y and x for γ

2.3.2. Contrast & Brightness

Brightness and contrast are pixel-wise enhancement algorithms [Equation 2], used to adjust the overall brightness and contrast of a video. This can be useful for improving the visibility of objects in low-light situations, or for making colors in a video more vibrant. These algorithms can be used to adjust the overall brightness and contrast of a video, or to adjust the brightness and contrast of specific objects or regions within a video.

$$\mathrm{C} = contrast\ paramter$$

$$B = Brightness\ paramter$$

$$f[m,n,ch]_{out} = \mathrm{C} \cdot f[m,n,ch] + B$$

Equation 2 - contrast and brightness

2.4. AXI protocol

The AXI (Advanced eXtensible Interface) protocol is a high-performance, flexible interface for connecting various components of a system-on-a-chip (SoC) design developed by ARM. The AXI protocol is designed to support a wide range of data transfer scenarios, from simple memory-mapped access to complex DMA (Direct Memory Access) operations. Overall, the AXI protocol is characterized by its scalability, high performance, interoperability, flexibility, and reliability. These characteristics make it a popular choice for use in digital circuits, particularly in the design of complex systems such as SoCs and DSPs.

AXI has sub-protocols with different channels, tailored for specific use-cases.
All AXI channels share the same common control signals and handshake procedure [Table 1][1].

Signal	Source	Description
ACLK	Clock source	The global clock signal. All signals are sampled on the rising edge of **ACLK**.
ARESETn	Reset source	The global reset signal.
TVALID	Master	TVALID indicates that the master is driving a valid transfer. A transfer takes place when both **TVALID** and **TREADY** are asserted.
TREADY	Slave	TREADY indicates that the slave can accept a transfer in the current cycle.

Table 1 - AXI common control signals

Any payload transfer process starts with the master waiting driving the data signals and setting TVALID to high. The slave then reads the data and drives TREADY high upon finishing. Thus, a handshake occurs and the master sets TVALID to low and stops driving the data signals [Figure 2] or initiates a new payload transfer.

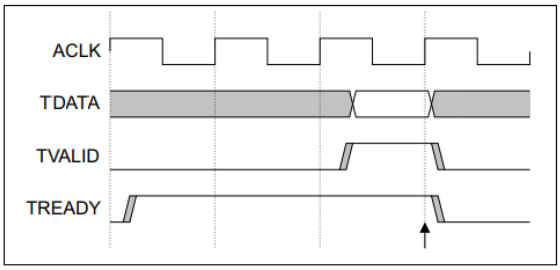

Figure 2 - AXI payload transfer & handshake

2.4.1. AXI - Stream

The AXI Stream protocol is a sub-protocol of the AXI protocol that is specifically designed for streaming data. The AXI Stream protocol uses a simple, FIFO (first-in, first-out) based interface for data transfer, making it well-suited for high-bandwidth data transfer scenarios, where memory-addressing is not required. The AXI Stream protocol is often used in applications such as video processing, where high-bandwidth data transfer is required, and uses a single data channel (TDATA) for data transfer.

The signals used in the sub-protocol are specified in [Table 2 - AXI-Stream signals].

Signal	Source	Description	Required
ACLK	Clock source	The global clock signal. All signals are sampled on the rising edge of ACLK.	Yes
ARESETn	Reset source	The global reset signal.	Yes
TVALID	Master	TVALID indicates that the master is driving a valid transfer. A transfer takes place when both **TVALID** and **TREADY** are asserted.	Yes
TREADY	Slave	TREADY indicates that the slave can accept a transfer in the current cycle.	Yes
TDATA[(8n-1):0]	Master	TDATA is the primary payload transfer. The width of the data payload is an integer number of bytes.	Yes
TSTRB[(n-1):0]	Master	TSTRB is the byte qualifier that indicates whether the content of the associated byte of **TDATA** is processed as a data byte or a position byte.	No
TKEEP[(n-1):0]	Master	TKEEP is the byte qualifier that indicates whether the content of the associated byte of TDATA is processed as part of the data stream. Associated bytes that have the **TKEEP** byte qualifier deasserted are null bytes and can be removed from the data stream.	No
TLAST	Master	TLAST indicates the boundary of a packet.	No
TID[(i-1):0]	Master	TID is the data stream identifier that indicates different streams of data.	No
TDEST[(d-1):0]	Master	TDEST provides routing information for the data stream.	No
TUSER[(u-1):0]	Master	TUSER is user defined sideband information that can be transmitted alongside the data stream.	No

Table 2 - AXI-Stream signals

2.4.2. AXI - Lite

The AXI Lite protocol is another sub-protocol of the AXI protocol that is designed for simple, memory-mapped access to registers. The AXI Lite protocol uses a smaller, simpler interface than the full AXI protocol, making it more suitable for simple register access operations. The AXI Lite protocol is often used in control and configuration scenarios.

The signals used in the sub-protocol are specified in [Table 3 - AXI-Lite signals][2].

Signal	Source	Description	Required
ACLK	Clock source	The global clock signal. All signals are sampled on the rising edge of **ACLK**.	Yes
ARESETn	Reset source	The global reset signal.	Yes
Write Address Channel			
AWVALID	Master	Write address valid. This signal indicates that the channel is signaling valid write address and control information.	Yes
AWADDR[w-1:0]	Master	Write address. The write address gives the address of the first transfer in a write burst transaction.	Yes
AWREADY	Slave	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals.	Yes
AWPROT	Master	Protection type. This signal indicates the privilege and security level of the transaction, and whether the transaction is a data access or an instruction access.	No
Write Data Channel			
WVALID	Master	Write valid. This signal indicates that valid write data and strobes are available.	Yes
WDATA[8n-1:0]	Master	Write data.	Yes
WREADY	Slave	Write ready. This signal indicates that the slave can accept the write data.	Yes
WSTRB	Master	Write strobes. This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.	No
Write Response Channel			
BREADY	Master	Response ready. This signal indicates that the master can accept a write response.	Yes
BRESP[1:0]	Slave	Write response. This signal indicates the status of the write transaction. [Table 4 - AXI RESP codes]	Yes
BVALID	Slave	Write response valid. This signal indicates that the channel is signaling a valid write response.	Yes
Read Address Channel			
ARVALID	Master	Read address valid. This signal indicates that the channel is signaling valid read address and control information.	Yes
ARADDR[w-1:0]	Master	Read address. The read address gives the address of the first transfer in a read burst transaction.	Yes
ARREADY	Slave	Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals.	Yes
ARPROT	Master	Protection type. This signal indicates the privilege and security level of the transaction, and whether the transaction is a data access or an instruction access.	No

Read Data Channel			
RREADY	Master	Read ready. This signal indicates that the master can accept the read data and response Information.	Yes
RVALID	Slave	Read valid. This signal indicates that the channel is signaling the required read data.	Yes
RDATA[8n-1:0]	Slave	Read data.	Yes
RRESP[1:0]	Slave	Read response. This signal indicates the status of the read transfer. [Table 4 - AXI RESP codes]	Yes

Table 3 - AXI-Lite signals

The slave response codes defined in AXI [Table 4 - AXI RESP codes]:

BRESP [1:0] / **RRESP**[1:0]	**Condition**	**Description**	**Required**
00	OKAY	Normal access success. Indicates that a normal access has been successful. Can also indicate an exclusive access has failed.	Yes
01	EXOKAY	Exclusive access okay. Indicates that either the read or write portion of an exclusive access has been successful.	No
10	SLVERR	Slave error. Used when the access has reached the slave successfully, but the slave wishes to return an error condition to the originating master.	Yes
11	DECERR	Decode error. Generated, typically by an interconnect component, to indicate that there is no slave at the transaction address.	No

Table 4 - AXI RESP codes

3. Project Definitions

Motivation

Video and images are often recorded in sub-optimal conditions, resulting in a poor-quality reproduction. Moreover, recording and displaying devices have different characteristics that need to be compensated for. Atop of that, people have varying individual preferences of brightness and contrast.
Many applications such as forensic sciences and medical imaging, often require high visual quality to increase performance of analysis and interpretation.
Video/image enhancement algorithms fill out those needs.
An efficient HW implementation of those algorithms in an interoperable device that can be inserted into an existing data path is thus desired.

Goals

Design a digital video enhancement block which:

- Implements brightness, contrast and gamma correction (by LUT);
- Is controllable via AXI-Lite protocol;
- Has AXI-Stream protocol video IO;
- Allows for user set parameters to be changed in-operation;
- Supports different resolutions and frame rates (FPS);
- Operates on continuous video with constant delay.

Requirements

- HDL implementation of the project goals.
- Creation of a reference model.
- Simulation of the design with video I/O.
- Validation of the design vs model, and comparison by error criteria.

4. Description of the Algorithm

All the video enhancements discussed in the background chapter are performed using point-wise operations on each pixel's RGB channels, therefore:

1. The current pixel is independent from the rest of the frame.
2. The current pixels' RGB channels are independent of each other.

This allows us to operate on the 3 channels simultaneously per pixel.

Figure X describes the algorithm mathematically, where for 8bits per channel:

x_{ch} - A pixels' channel value, when $ch \in \{R, G, B\}$

C – contrast parameter, selected from range [0,6] (specific values discussed in [

fixed_point	value	cp_param
00000000	0.000	000000000
00000001	0.125	000000111
00000010	0.250	000000110
00000011	0.375	001110110
00000100	0.500	000000101
00000101	0.625	001110101
00000110	0.750	001100101
00000111	0.875	101110100
00001000	1.000	000000100
00001001	1.125	001000111
00001010	1.250	001000110
00001100	1.500	001000101
00001110	1.750	101101001
00001111	1.875	101111001
00010000	2.000	000001001
00010001	2.125	010010111
00010010	2.250	010010110
00010100	2.500	010010101
00011000	3.000	001001001
00011100	3.500	101011010
00011110	3.750	101101010
00011111	3.875	101111010
00100000	4.000	000001010

00100001	4.125	010100111
00100010	4.250	010100110
00100100	4.500	010100101
00101000	5.000	010100100
00110000	6.000	010101001

Table 17 - contrast parameter valid set as read from contrast LUT)

B - Brightness parameter, in range $[-255, 255]$

$f(x) = y$ – mapping from/to channel value, functioning as gamma correction.

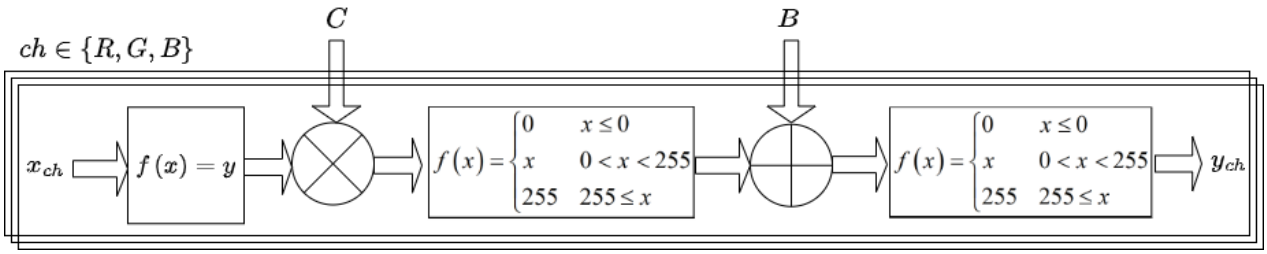

Figure 3 - mathematical description of the algorithm

5. Software Implementation

MATLAB was used to implement a software model and auxiliary scripts for design and simulation. It was chosen as it has prebuilt image/video editing functions and is a convenient tool to apply calculations on matrices.

"Image enhancement" – Script shows results of applying contrast brightness and gamma correction on a reference image according to user's parameters input. The purpose was to roughly examine the effect of different algorithm parameters in the output, and also to find their ranges of values that create significant change in the output that is noticeable for the human eye.

Based on the results we obtained from it, we chose contrast parameter values to be base-2 numbers in [0,6] range (examples at Figure 5 - MATLAB results for different contrast parameter values ,detailed discrete values [

fixed_point	value	cp_param
00000000	0.000	000000000
00000001	0.125	000000111
00000010	0.250	000000110
00000011	0.375	001110110
00000100	0.500	000000101
00000101	0.625	001110101
00000110	0.750	001100101
00000111	0.875	101110100
00001000	1.000	000000100
00001001	1.125	001000111
00001010	1.250	001000110
00001100	1.500	001000101
00001110	1.750	101101001
00001111	1.875	101111001
00010000	2.000	000001001
00010001	2.125	010010111
00010010	2.250	010010110
00010100	2.500	010010101
00011000	3.000	001001001
00011100	3.500	101011010
00011110	3.750	101101010
00011111	3.875	101111010
00100000	4.000	000001010
00100001	4.125	010100111
00100010	4.250	010100110

00100100	4.500	010100101
00101000	5.000	010100100
00110000	6.000	010101001

Table 17 - contrast parameter valid set as read from contrast LUT]). That allowed us to optimize the multiplier design using shift operations.

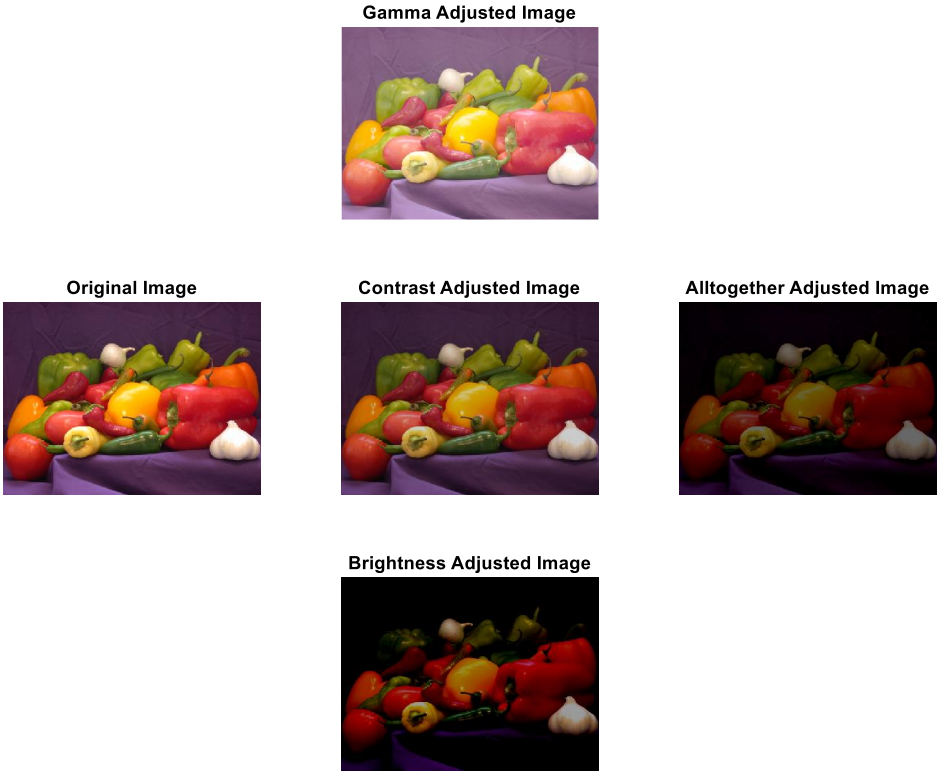

Figure 4 – Algorithms.m examples

Figure 5 - MATLAB results for different contrast parameter values

To simulate the design's functionality, we opted to drive it through a test bench with an input data stream of pixels in 24bit RGB and collect output data stream results in same format.

"Image to vector" – script facilitates simple conversion from available digital image/video formats (jpeg, bmp…) to a flat-vector text-based representation of pixels in hexadecimal notation, that can be read by a System Verilog test bench and driven to the design simulation and vice versa. Thus, allowing us to display the simulation results as an image.

"DUT vs reference model" – Script extends Prep_image for gif files to simulate video – converts gif images to same format as mentioned before, which the test bench treats as a single stream of pixels.
It applies enhancement algorithms with chosen parameters, thus allowing comparison of DUT simulation results to a reference model by calculation of error criteria. Lastly, it reconstructs the individual frames from DUT simulation results to a gif object and allows reproduction as video.

6. Architecture

6.1. Top

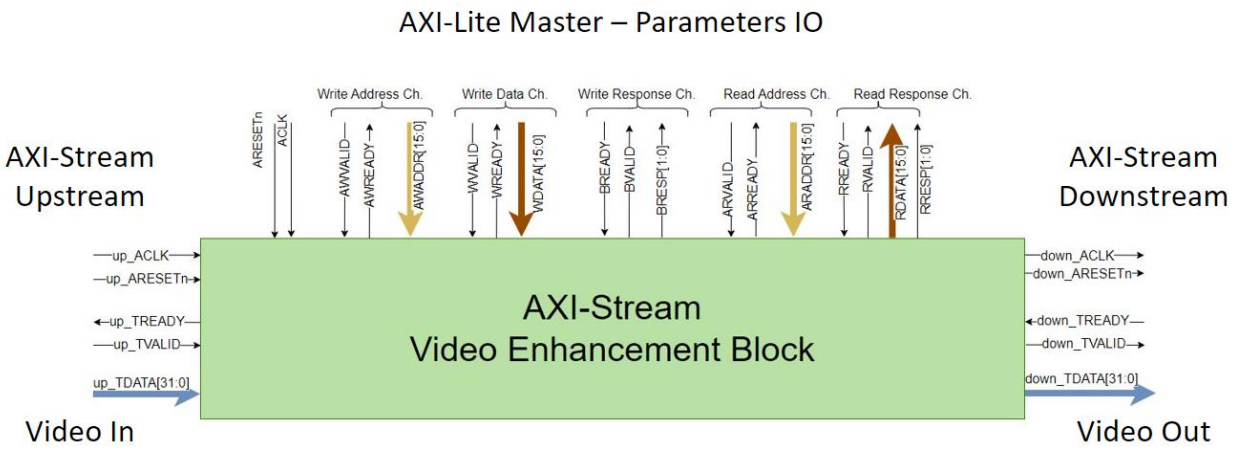

Figure 6 – Top block schematic draw

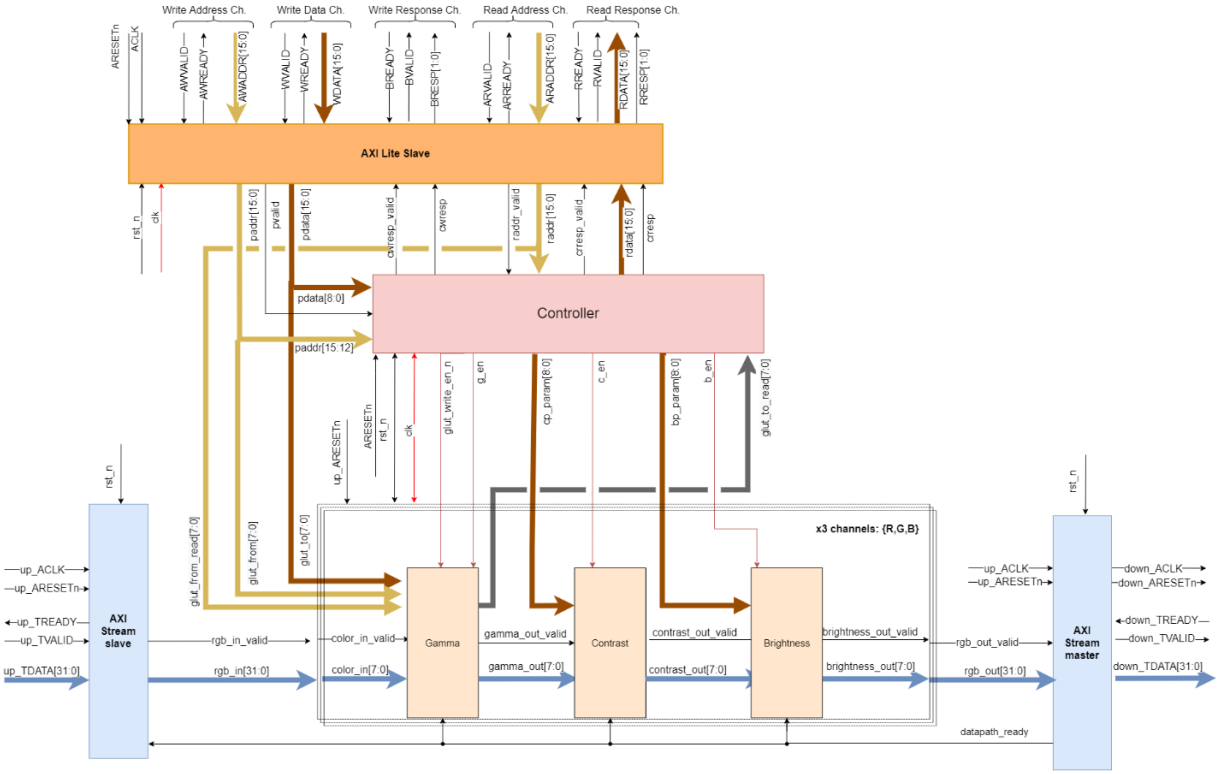

Figure 7 – Top block schematic draw overview

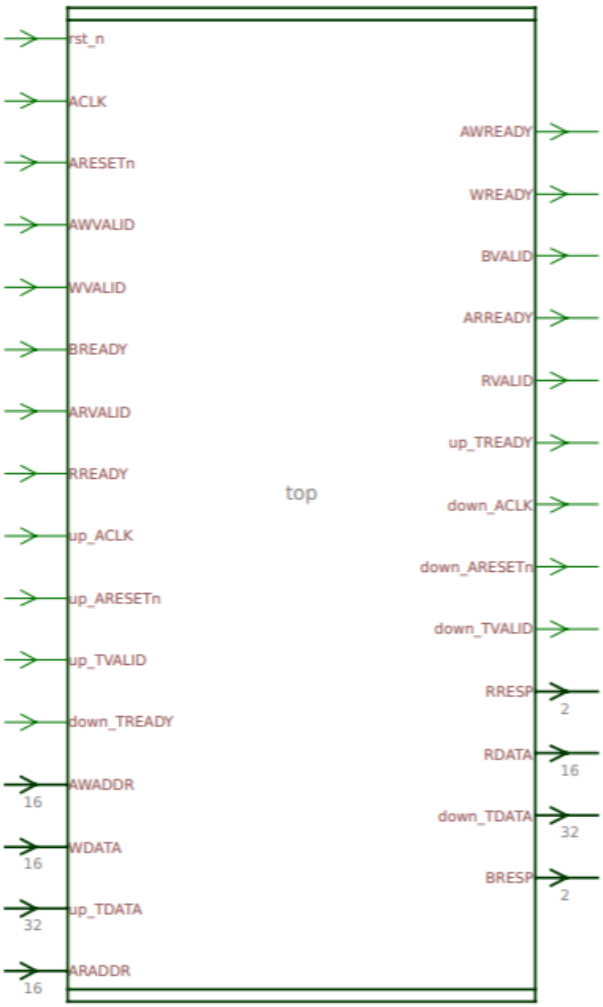

Figure 8 - Top block

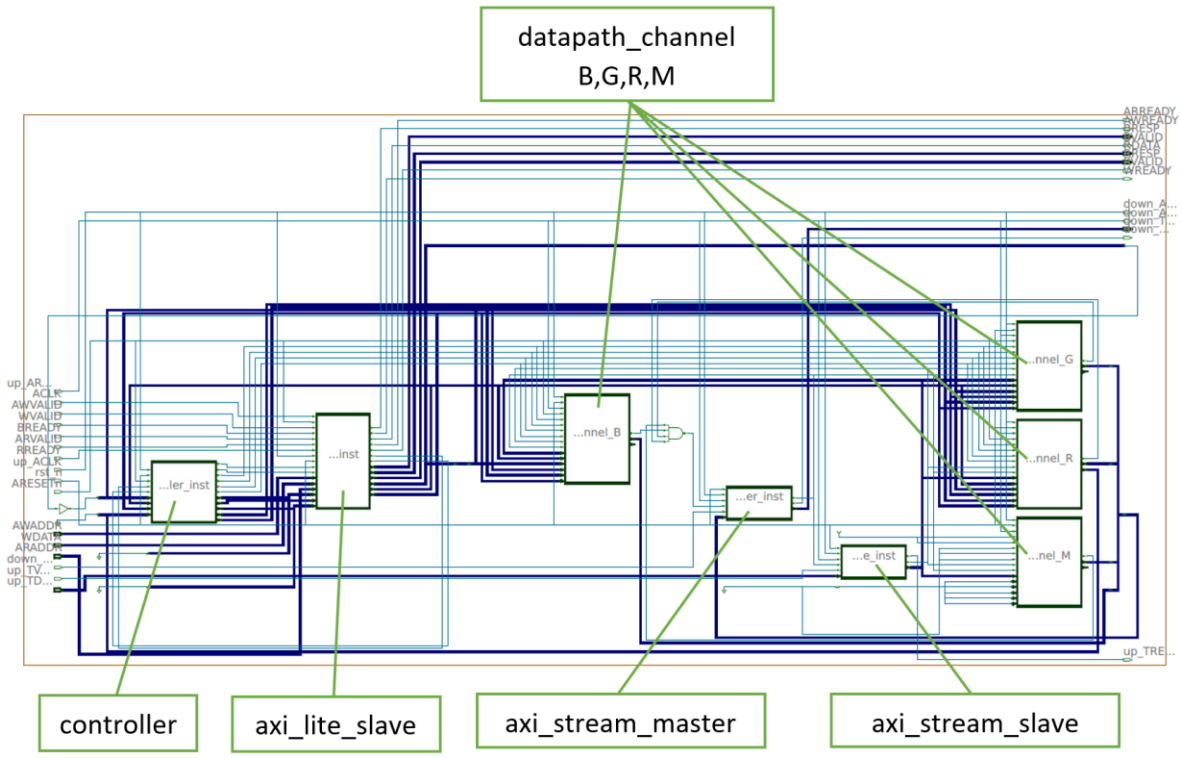

Figure 9 - Top block overview

Signal	Type	Description
rst_n	input	Full reset for block, asynchronous to clocks.
AXI – Lite		
ACLK	input	AXI-Lite clock
ARESETn	input	AXI-Lite reset for registered algorithms parameters and control bits, asynchronous to ACLK
Write Address Channel		
AWVALID	input	**Master** write address valid signal, indicates that the channel is signaling valid write address and control information
AWADDR[15:0]	input	**Master** write address. The write address gives the address of the first transfer in a write burst transaction
AWREADY	output	**Slave** write address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
Write Data Channel		
WVALID	input	**Master** write valid signal, indicates that valid write data and strobes are available
WDATA[15:0]	input	**Master** write data
WREADY	output	**Slave** write ready signal, indicates that the slave can accept the write data.
Write Response Channel		

BREADY	input	**Master** response ready signal, indicates that the master can accept a write response
BRESP[1:0]	output	**Slave** write response signal, indicates the status of the write transaction (Table 4 - AXI RESP codes).
BVALID	output	**Slave** write response valid signal, indicates that the channel is signaling a valid write response
Read Address Channel		
ARVALID	input	**Master** read address valid signal, indicates that the channel is signaling valid read address and control information
ARADDR[15:0]	input	**Master** read address. The read address gives the address of the first transfer in a read burst transaction
ARREADY	output	**Slave** read address ready signal, indicates that the slave is ready to accept an address and associated control signals.
Read Data Channel		
RREADY	input	**Master** read ready signal, indicates that the master can accept the read data and response Information
RVALID	output	**Slave** read valid signal, indicates that the channel is signaling the required read data.
RDATA[15:0]	output	**Slave** read data
RRESP[1:0]	output	**Slave** read response signal, indicates the status of the read transfer (Table 4 - AXI RESP codes).
AXI – Stream Upstream Slave		
up_ACLK	input	AXI-stream Upstream Slave clock
up_ARESETn	input	AXI-stream reset for data flows upstream, asynchronous to up_ACLK
up_DATA[31:0]	input	Data driven by master to slave
up_TVALID	input	Indicates that the master is driving a valid transfer. A transfer takes place when both TVALID and TREADY are asserted
up_TREADY	output	Indicates that the slave can accept a transfer in the current cycle
AXI – Stream Downstream Master		
down_ACLK	output	AXI-stream Downstream Master clock
down_ARESETn	output	AXI-stream reset for data flows downstream, asynchronous to down_ACLK
down_TDATA[31:0]	output	Data driven by master to slave
down_TVALID	output	Indicates that the master is driving a valid transfer. A transfer takes place when both TVALID and TREADY are asserted
down_TREADY	input	Indicates that the slave can accept a transfer in the current cycle

Table 5 - Top block signals description

Top block defines internal signals and connects relevant outputs to inputs between subblocks [which appear in Figure 7 – Top block schematic draw].

It contains 4 instances of datapath_channel block [section datapath_channel],

as 3 channels assigned for RGB; each applies enhancement on different 8 bits (taken from up_TDATA), and the fourth channel is responsible for passing meta data for the last 8 bits of up_TDATA as it is without applying enhancement.

Channels' output bits are combined back together to 32 bits at the end of the pipe (down_TDATA).

It also assigns AXI-Stream Master / Slave and datapath to have the same clock, which differs from AXI-Lite's clock, and assigns the same reset for the whole block (rst_n).

6.2. datapath_channel

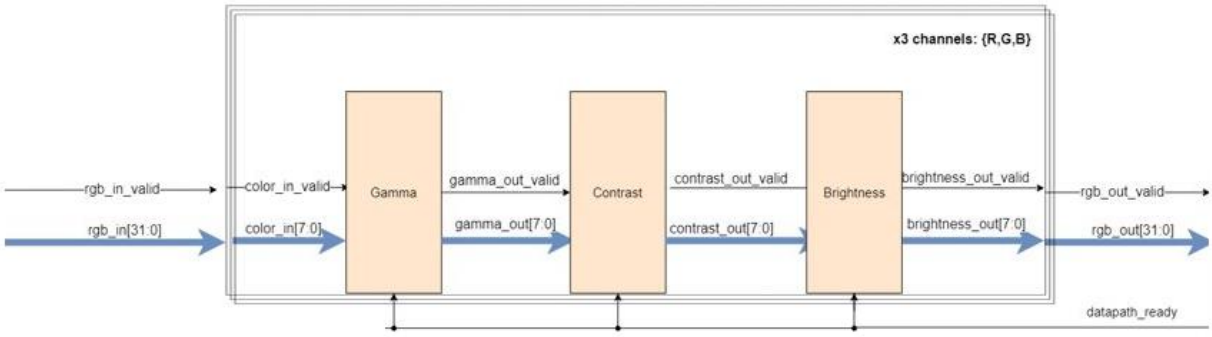

Figure 10 - Datapath channels block schematic draw overview

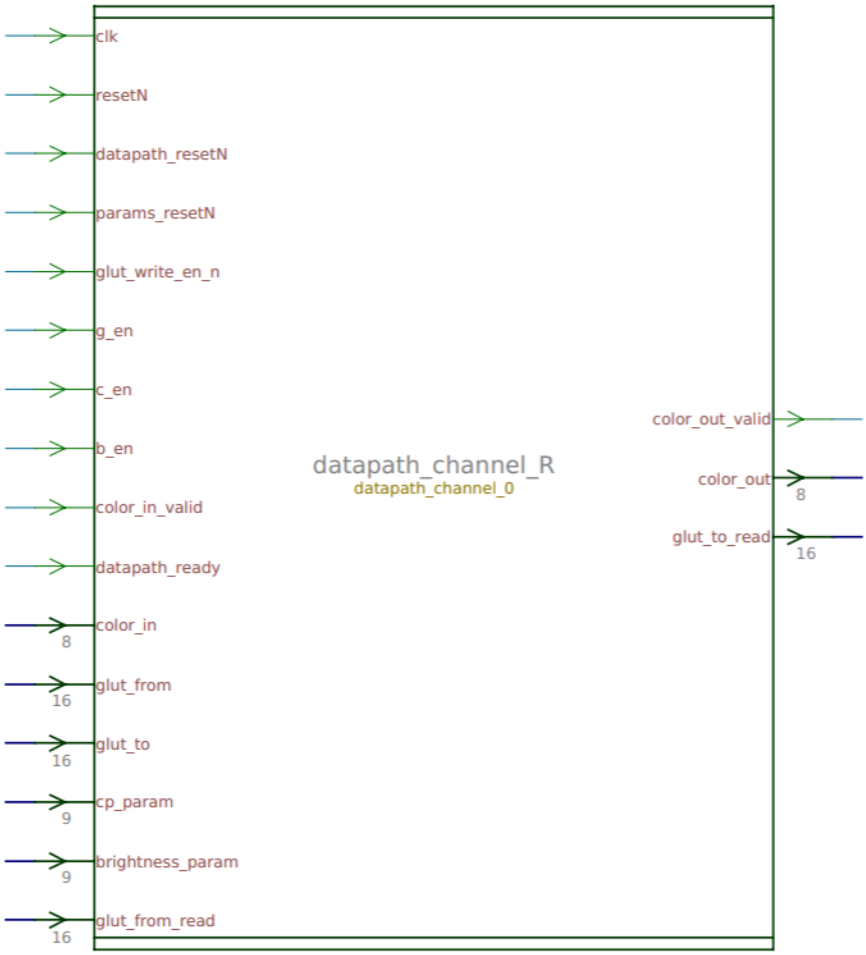

Figure 11 – Datapath R channel block

Signal	Type	Description
clk	input	Datapath clock
resetN	input	reset for data and registered algorithms parameters, asynchronous
datapath_resetN	input	resets data, asynchronous
params_resetN	input	reset for registered algorithms parameters, asynchronous
glut_write_en_n	input	enables write to gamma correction LUT
g_en	input	enables gamma correction for channel data bits
c_en	input	enables contrast algorithm for channel data bits
b_en	input	enables brightness algorithm for channel data bits
color_in_valid	input	Indicates that channel data bits are valid
datapath_ready	input	Indicates that pipe is ready to except data
color_in[7:0]	input	Channel data bits
glut_from[15:0]	input	Gamma correction LUT mapping value
glut_to[15:0]	input	Gamma correction LUT value mapped
cp_param[8:0]	input	contrast parameter value as mapped from contrast LUT [section contrast_LUT]
brightness_param[8:0]	input	brightness parameter value (2's complement)
glut_from_read[15:0]	input	Value read from gamma correction LUT
color_out_valid	output	Indicates that output channel data bits are valid
color_out[7:0]	output	output channel data bits (after enhancement algorithms)
glut_to_read[15:0]	output	gamma correction LUT mapping value required to read from

Table 6 - Datapath R channel block signals description

Block contains instances of enhancement algorithm blocks: gamma_LUT, contrast and brightness.

It connects relevant inputs and outputs for subblocks, such that color_in enhanced by algorithms' blocks in following order: gamma, contrast, brightness.

It also concatenates color_in_valid to be color_out_valid within algorithms' blocks, for AXI-Stream master to know if enhanced pixel holds valid data at the end of the pipe.

6.2.1. gamma_LUT

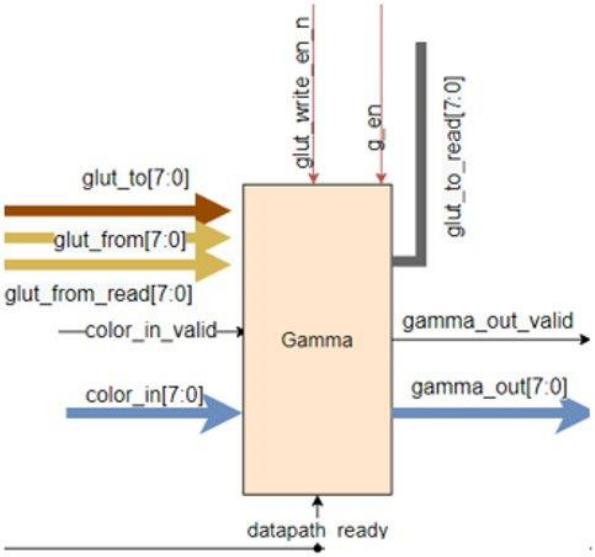

Figure 12 - Gamma LUT block schematic draw

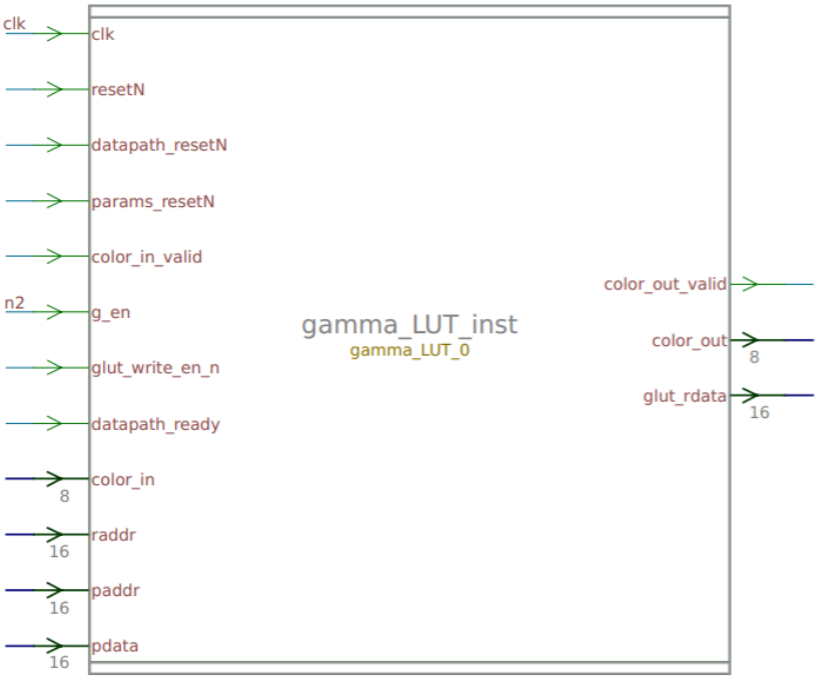

Figure 13 – Gamma LUT block

Signal	Type	Description
clk	input	Datapath clock
resetN	input	reset for data and registered algorithms parameters, asynchronous
datapath_resetN	input	resets data, asynchronous
params_resetN	input	reset for registered algorithms parameters, asynchronous
color_in_valid	input	Indicates that channel data bits are valid
g_en	input	enables gamma correction for channel data bits
glut_write_en_n	input	enables write to gamma correction LUT
datapath_ready	input	Indicates that pipe is ready to except data
color_in[7:0]	input	Channel data bits
raddr[15:0]	input	gamma correction LUT mapping value required to read from
paddr[15:0]	input	Gamma correction LUT mapping value
pdata[15:0]	input	Gamma correction LUT value mapped
color_out_valid	output	Indicates that output channel data bits are valid
color_out[7:0]	output	output channel data bits (after enhancement algorithms)
glut_rdata[15:0]	output	Value read from gamma correction LUT

Table 7 - Gamma LUT block signals description

Block contains instance for a Synchronous Write-Port, Asynchronous Dual Read-Port RAM Flip-Flop Based (DW_ram_2r_w_s_dff) and uses it as a LUT to simulate gamma correction algorithm.

LUT key-value assignment made by paddr – for the key , pdata – for corresponding value.

LUT mapping values can be read asynchronous from glut_rdata signal, according to key assigned to raddr.

As pipe ready (datapath_ready = ‘1’), and gamma algorithm is enabled (g_en=1), it maps color_in according to LUT to color_out.

It also concatenates color_in_valid to be color_out_valid.

6.2.2. contrast

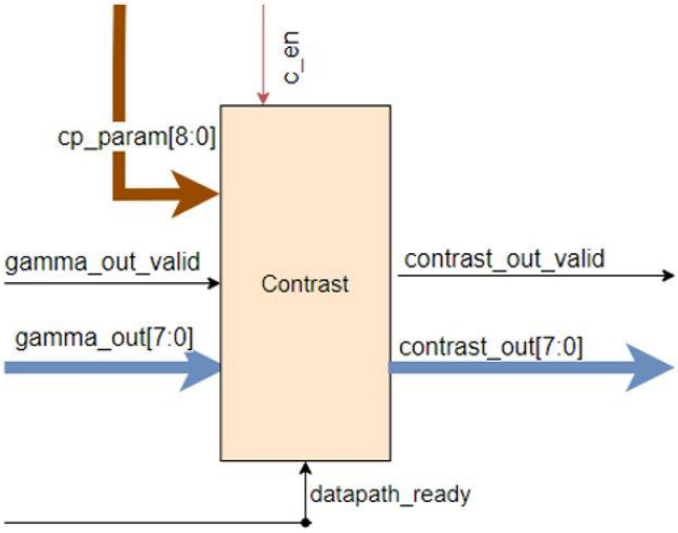

Figure 14 - Contrast block schematic draw

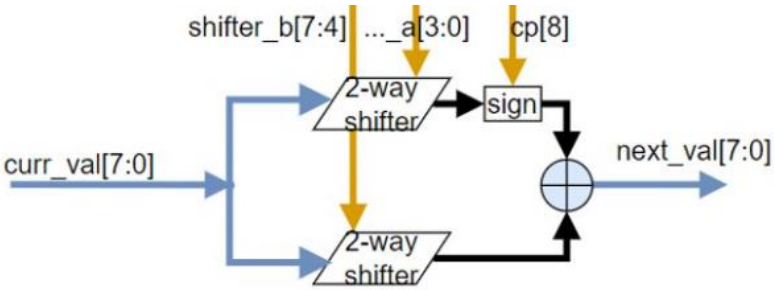

Figure 15 – Contrast block schematic draw overview

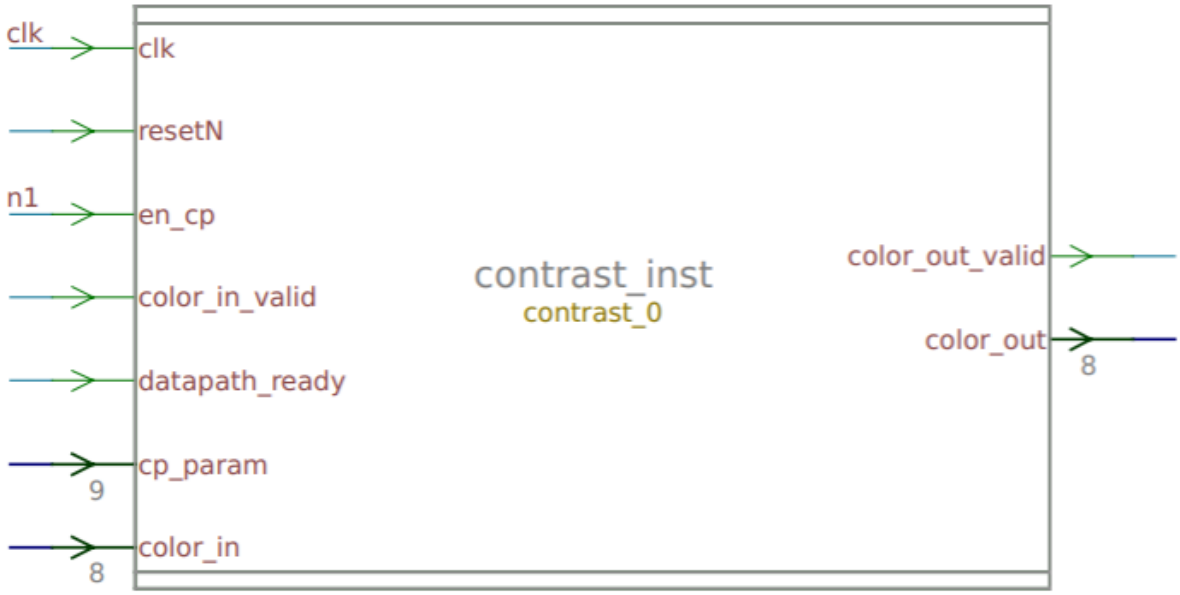

Figure 16 - Contrast block

Signal	Type	Description
clk	input	Datapath clock
resetN	input	reset for data and registered algorithms parameters, asynchronous
en_cp	input	Enables contrast algorithm
color_in_valid	input	Indicates that channel data bits are valid
datapath_ready	input	Indicates that pipe is ready to except data
cp_param[8:0]	input	Mapped contrast parameter value from contrast LUT
color_in[7:0]	input	Channel data bits
color_out_valid	output	Indicates that output channel data bits are valid
color_out[7:0]	output	Output channel data bits (after enhancement algorithms)

Table 8 - Contrast block signals description

cp_param is contrast parameter as read from contrast LUT [see section contrast_LUT], and its bits dictate which operations to make between 2 shifters in order the obtain the required initial fixed-point factor [see section shifter] .

cp_param[3:0] assigned to shifter A, and cp_param[7:4] to shifter B.

cp_param[8] determines if shifted results will be summed or subtracted [schematic draw in Figure 15 – Contrast block schematic draw overview] .

As pipe is ready (datapath_ready = ‘1’), multiplication of color_in by cp_param bounded to [0, 255] is passed as a result to color_out, and color_in_valid is concatenated to color_out_valid.

6.2.2.1. shifter

Figure 17 - Shifter block

Signal	Type	Description
shift_en	input	Enables shift operation
color_in[7:0]	input	Channel data bits
shifter_param[3:0]	input	Shifter_param[1:0] = represents shift value Shifter_param[3:2] = represents shift direction : 01 = >> 10 = << 00 = output is set to zero, regardless of shift value
color_out[9:0]	output	Shifted output channel data bits

Table 9 - Shifter block signals description

Shifter block is a combinational logic block.

If shifer is enabled, color_in is shifted according to shifter_param value bits and direction bits.

If shifer isn't enabled, bypass is performed.

6.2.3. Brightness

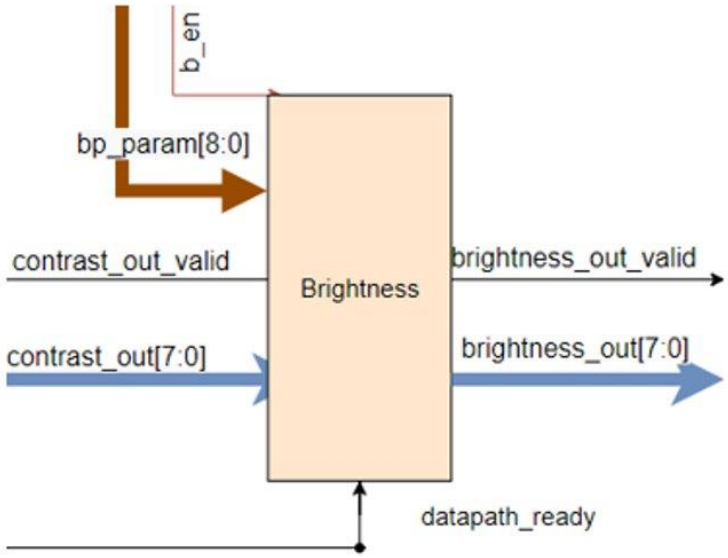

Figure 18 – Brightness block schematic draw

Figure 19 - Brightness block

Signal	Type	Description
clk	input	Datapath clock
resetN	input	reset for data and registered algorithms parameters, asynchronous
en_bp	input	Enables brightness algorithm for color_in bits
color_in_valid	input	Indicates that color_in is valid
datapath_ready	input	Indicates next block is ready to accept the data. Otherwise, stalls the pipe
brightness_param[8:0]	input	Brightness parameter, 2's complement
color_in[7:0]	input	Channel data bits
color_out_valid	output	Signaling for next block that color_out is valid
color_out[7:0]	output	Channel data bits after brightness enhancement

Table 10 - Brightness block signals description

As brightness algorithm enabled (en_bp = '1'), brightness block sums/subtracts brightnes_parameter to color_in and thresholds it to RGB valid gray scale value [255, 0].

As the pipe ready (datapath_ready = '1'), result passes to color_out, and color_in_valid passes to color_out_valid.

6.3. axi_lite

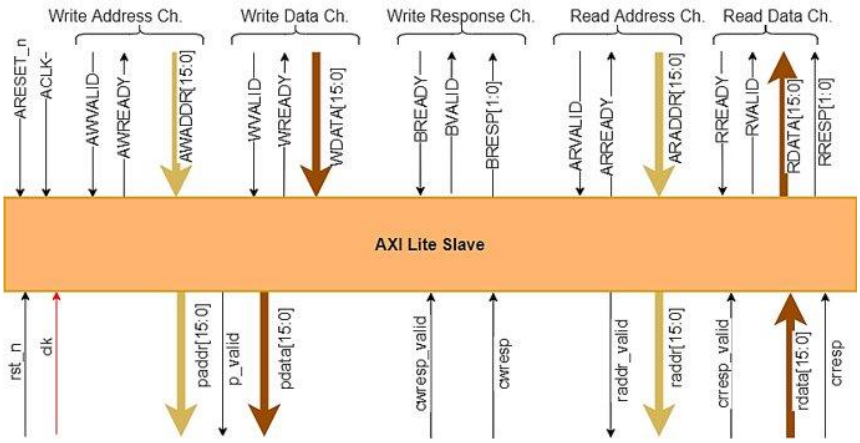

Figure 20 - AXI lite interface schematic draw

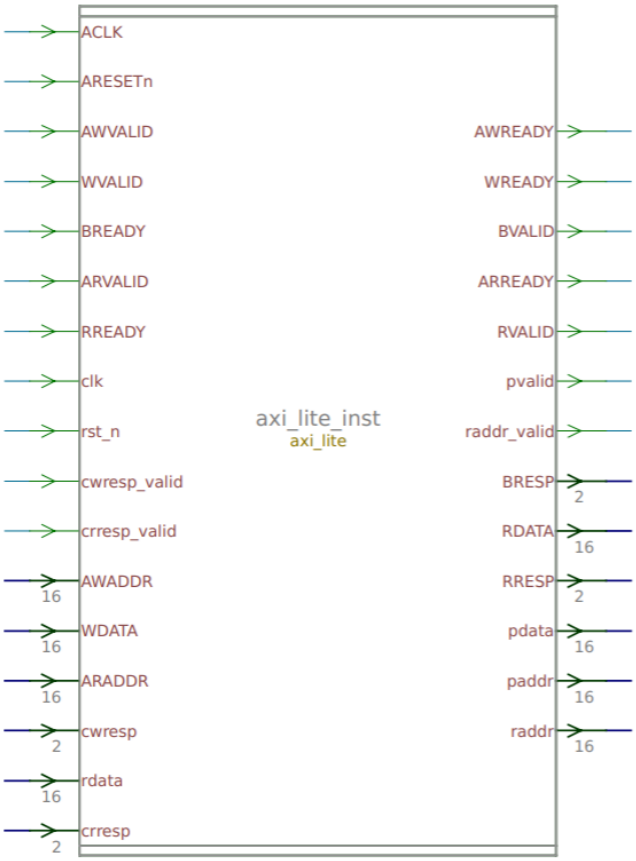

Figure 21 – AXI – Lite interface block

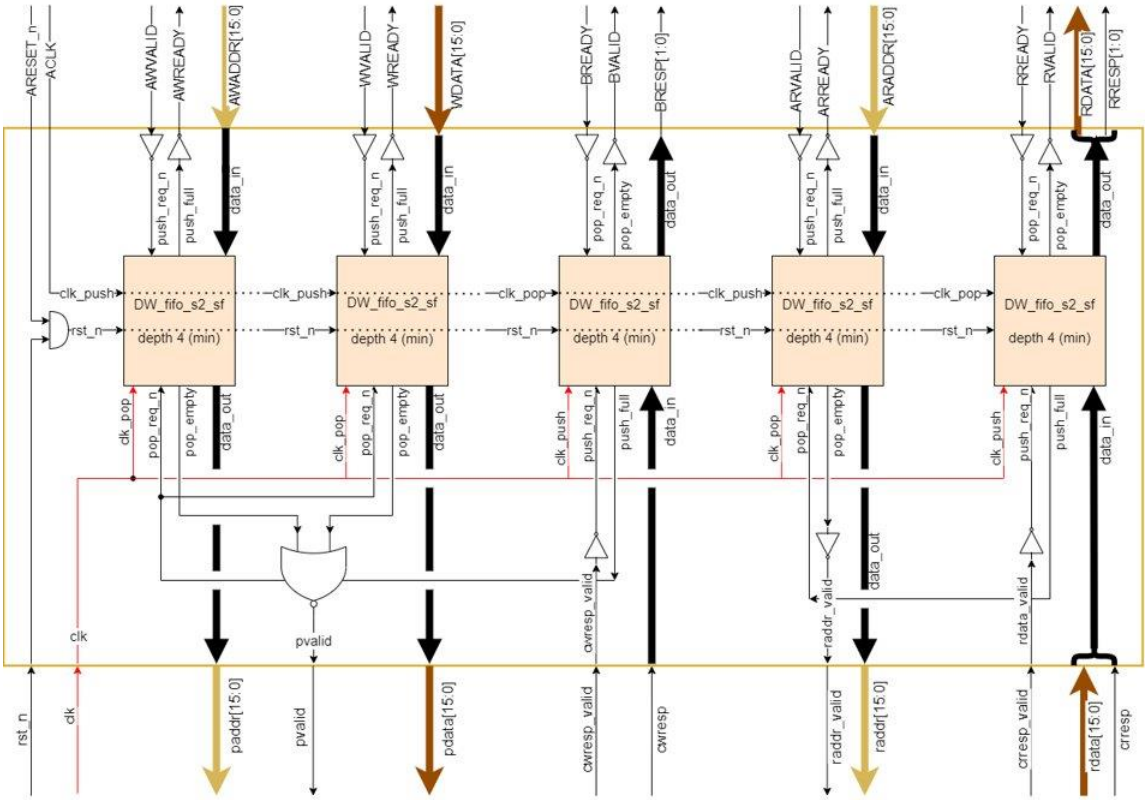

Figure 22 - AXI – Lite interface block schematic draw overview

Signal	Type	Description
rst_n	input	Full reset for block, asynchronous to clocks.
clk	input	Datapath clock
ACLK	input	AXI-Lite clock
ARESETn	input	AXI-Lite reset for registered algorithms parameters and control bits, asynchronous to ACLK
cwresp_valid	input	Indicates if controller write response is valid
crresp_valid	input	Indicates if controller read response is valid
cwresp	input	controller write response [values in Table 4 - AXI RESP codes]
rdata	input	Data read from controller - enables / brightness / contrast / gamma LUT upon raddr
crresp	input	controller read response [values in Table 4 - AXI RESP codes]
pvalid	output	Indicates if pdata is valid
raddr_valid	output	Indicates if raddr is valid
raddr[15:0]	output	Address register for read control parameters, raddr[15:12] = 0001: enables for algorithms

		0010: brightness parameter register 0100: contrast parameter register 1000: Gamma LUT register, as paddr[7:0] = bits for Gamma LUT mapping value
paddr[15:0]	output	Address register for write control parameters, paddr[15:12] = 0001: enables for algorithms 0010: brightness parameter register 0100: contrast parameter register 1000: Gamma LUT register, as paddr[7:0] = bits for Gamma LUT mapping value
pdata[15:0]	output	Data from AXI-Lite, written to enables / brightness / contrast register upon paddr
Write Address Channel		
AWVALID	input	**Master** write address valid signal, indicates that the channel is signaling valid write address and control information
AWADDR[15:0]	input	**Master** write address. The write address gives the address of the first transfer in a write burst transaction
AWREADY	output	**Slave** write address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
Write Data Channel		
WVALID	input	**Master** write valid signal, indicates that valid write data and strobes are available
WDATA[15:0]	input	**Master** write data
WREADY	output	**Slave** write ready signal, indicates that the slave can accept the write data.
Write Response Channel		
BREADY	input	**Master** response ready signal, indicates that the master can accept a write response
BRESP[1:0]	output	**Slave** write response signal, indicates the status of the write transaction (Table 4 - AXI RESP codes).
BVALID	output	**Slave** write response valid signal, indicates that the channel is signaling a valid write response
Read Address Channel		
ARVALID	input	**Master** read address valid signal, indicates that the channel is signaling valid read address and control information
ARADDR[15:0]	input	**Master** read address. The read address gives the address of the first transfer in a read burst transaction
ARREADY	output	**Slave** read address ready signal, indicates that the slave is ready to accept an address and associated control signals.
Read Data Channel		
RREADY	input	**Master** read ready signal, indicates that the master can accept the read data and response Information
RVALID	output	**Slave** read valid signal, indicates that the channel is signaling the required read data.

RDATA[15:0]	output	**Slave** read data
RRESP[1:0]	output	**Slave** read response signal, indicates the status of the read transfer (Table 4 - AXI RESP codes).

Table 11 - AXI – Lite interface block signals description

Block performs AXI-Lite protocol.

Block implementation is based on 5 instances of type DW_fifo_s2_sf [Figure 22 - AXI – Lite interface block schematic draw overview], asynchronous FIFOs with 2 clocks. One designed for pushing data, and one for popping data, and it is done by push_req_n, pop_req_n.

push_empty and push_full signal if the queue is full or empty, while queue size is 4 words.

a. Handle clock domain crossing – by synchronizing between slow external AXI-Lite clock – “ACLK”, and fast internal datapath and AXI-Stream clock – “clk”
b. Synchronize write address with write data.

6.4. axi_stream_slave

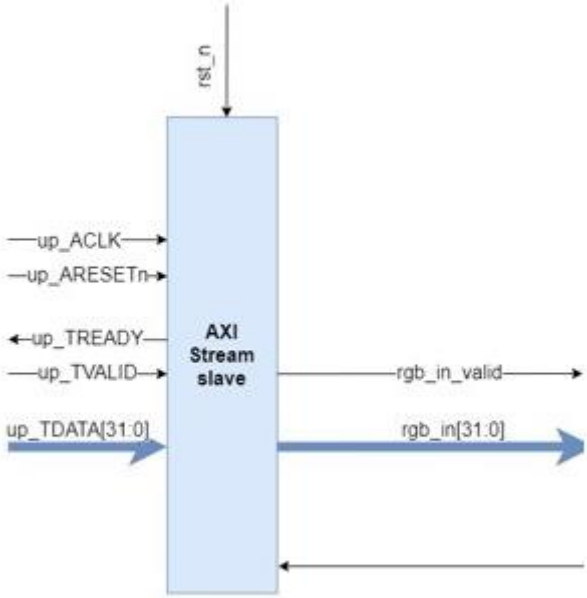

Figure 23 – AXI-Stream slave interface block schematic draw

Figure 24 – AXI-Stream slave interface block

Signal	Type	Description
resetN	input	reset for data and registered algorithms parameters, asynchronous
ACLK	input	AXI-stream Upstream Slave clock
ARESETn	input	AXI-Lite reset for registered algorithms parameters and control bits, asynchronous to ACLK
datapath_ready	input	Indicates next block is ready to accept the data. Otherwise, stalls the pipe
TVALID	input	Indicates that the master is driving a valid transfer. A transfer takes place when both TVALID and TREADY are asserted
TDATA[31:0]	input	Data driven by master to slave
TREADY	output	Indicates that the slave can accept a transfer in the current cycle
rgb_valid	output	Indicates if TDATA is valid
rgb_out[31:0]	output	TDATA value

Table 12 - AXI-Stream slave interface block signals description

Block performs AXI-Stream protocol.

As handshake occurs (TVALID and TREADY are both high), TDATA is driven into rgb_out.

In addition, rgb_valid is driven by TVALID, and TREADY is driven by datapath_ready.

6.5. axi_stream_master

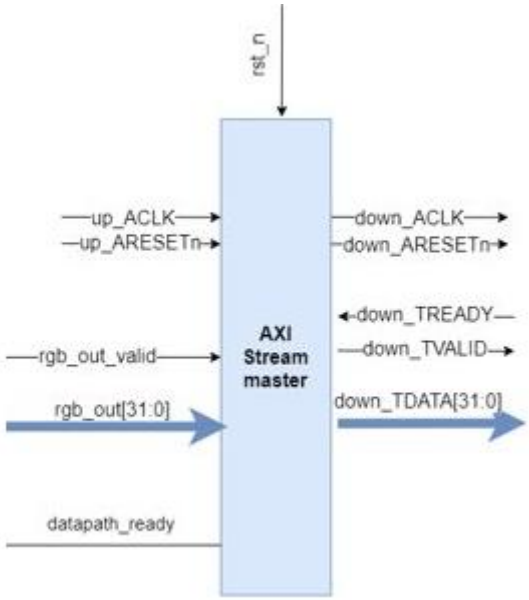

Figure 25 – AXI-Stream master interface block schematic draw

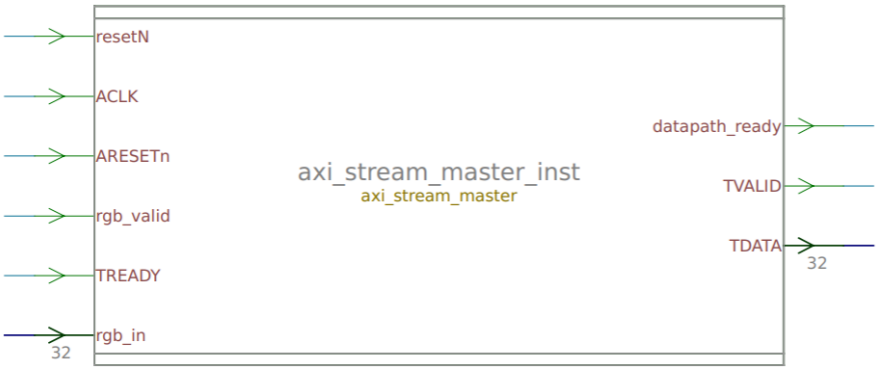

Figure 26 – AXI-Stream master interface block

Signal	Type	Description
resetN	input	reset for data and registered algorithms parameters, asynchronous
ACLK	input	AXI-stream Upstream Slave clock
ARESETn	input	AXI-Lite reset for registered algorithms parameters and control bits, asynchronous to ACLK
rgb_valid	input	Indicates if rgb_in is valid
TREADY	input	Indicates that the master is driving a valid transfer. A transfer takes place when both TVALID and TREADY are asserted
rgb_in [31:0]	input	Data driven by master to slave
datapath_ready	output	Indicates that the slave can accept a transfer in the current cycle
TVALID	output	Indicates next block is ready to accept the data. Otherwise, stalls the pipe
TDATA[31:0]	output	rgb_in value

Table 13 - AXI-Stream master interface block signals description

Block performs AXI-Stream protocol.

As handshake occurs (TVALID and TREADY are both high), rgb_in is driven into TDATA.

In addition, TVALID is driven by rgb_valid, and datapath_ready is driven by TREADY.

6.6. controller

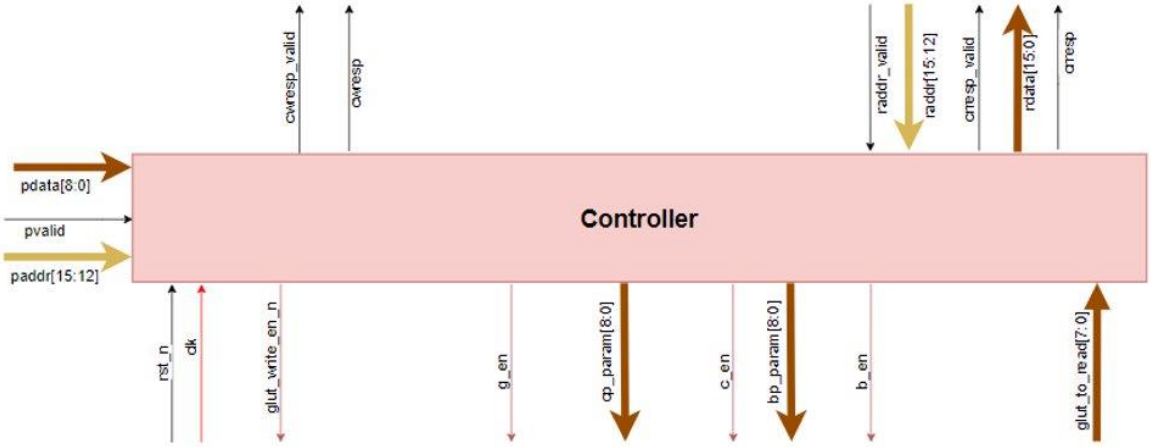

Figure 27 - Controller block schematic draw

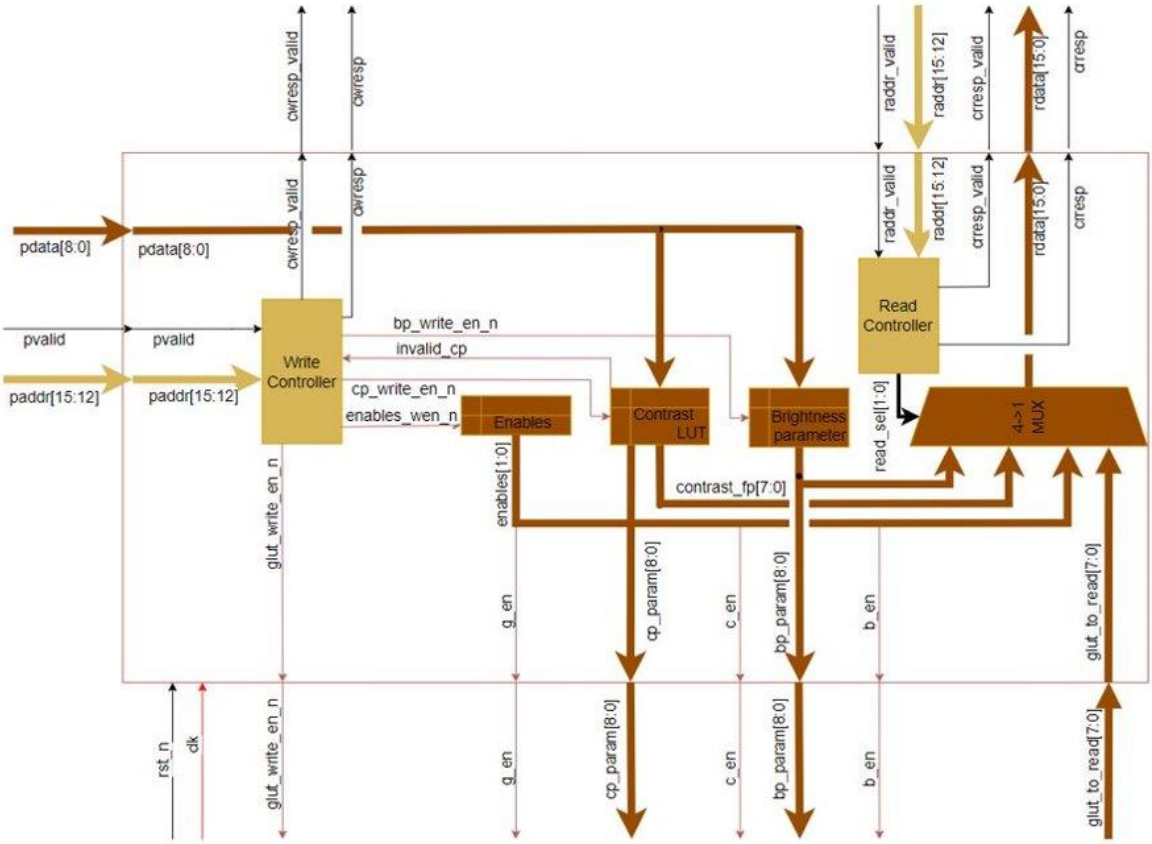

Figure 28 - Controller block schematic draw overview

Figure 29 - Controller block

Signal	Type	Description
clk	input	Datapath clock
rst_n	input	Full reset for block, asynchronous to clocks.
params_resetN	input	reset for registered algorithms parameters, asynchronous
pvalid	input	Indicates if data is valid
raddr_valid	input	Indicates if raddr is valid
pdata[15:0]	input	Data from AXI-Lite, written to enables / brightness / contrast register upon paddr
paddr[15:0]	input	Address register for write control parameters, paddr[15:12] = 0001: enables for algorithms 0010: brightness parameter register 0100: contrast parameter register 1000: Gamma LUT register, as paddr[7:0] = bits for Gamma LUT mapping value
raddr[15:0]	input	Address register for read control parameters, raddr[15:12] = 0001: enables for algorithms 0010: brightness parameter register 0100: contrast parameter register 1000: Gamma LUT register, as paddr[7:0] = bits for Gamma LUT mapping value
glut_to_read[15:0]	input	gamma correction LUT value was read
cwresp_valid	output	Indicates if controller write response is valid

crresp_valid	output	Indicates if controller read response is valid
glut_write_en_n	output	enables write to gamma correction LUT
g_en	output	enables gamma correction for channel data bits
c_en	output	enables contrast algorithm for channel data bits
b_en	output	enables brightness algorithm for channel data bits
cwresp[1:0]	output	controller write response [values in Table 4 - AXI RESP codes]
rdata[15:0]	output	Reading data from control register - enables / brightness / contrast / gamma LUT upon raddr
crresp[1:0]	output	controller read response [values in Table 4 - AXI RESP codes]
cp_param[8:0]	output	Mapped contrast parameter value from contrast LUT
brightness_param[8:0]	output	Brightness parameter, 2's complement

Table 14 - Controller block signals description

The controller houses the control and parameter memory (registers):

1. Numerical parameters that are passed as inputs to datapath blocks:
   brightness_param, cp_param
2. Control signals that enable datapath blocks operation:
   b_en, c_en, g_en
3. Write control signal for external parameter memory:
   glut_write_en_n

It handles and generates AXI-Lite packets from/to axi_lite_slave block:

1. Write requests:
   a. Upon receiving pvalid – verifies paddr holds an address that is in memory space.
   b. If it is, writes to the addressed register or enables writing to external memory (glut_write_en_n).
2. Read requests:
   a. Upon receiving raddr_valid – verifies raddr holds an address that is in memory space
   b. If it is, reads to rdata the requested addressed register (brightness parameter / contrast parameter / enables) or external memory LUT mapping value (glut_to_read)

In both functions, the controller listens to incoming AXI-Lite packets originating from the axi_lite_slave block and upon receiving a high valid signals, checks address for validity, reports in case of error, and writes / reads addressed registered.

Our design does not require a State Machine - modes of operation are determined by registers, which are written to from outside via AXI-Lite interface.

6.7. contrast_LUT

Figure 30 - Contrast LUT block

Register:	Pdata[7]	Pdata[6]	Pdata[5]	Pdata[4]	Pdata[3]	.	Pdata[2]	Pdata[1]	Pdata[0]
Weight:	2^4	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}

Table 15 - pdata fixed point read

Signal	Type	Description
clk	input	Datapath clock
resetN	input	reset for data and registered algorithms parameters, asynchronous
cp_write_en	input	Enables LUT mapping
pdata[15:0]	input	Contrast paramter fixed point value
invalid	output	Indicates if contrast_fp belongs to valid set
contrast_fp[7:0]	output	Contrast paramter fixed point value (pdata[7:0])
cp_param[8:0]	output	Contrast parameter mapped

Table 16 - Contrast LUT block signals description

pdata[7:0] is read as a fixed point number [Table 15 - pdata fixed point read] and mapped by LUT to binary number - contrast_fp, that its bits dictate which operations to make between 2 shifters [Figure 15 – Contrast block schematic draw] in contrast block (direction, shift value, and sign) in order the obtain the required initial fixed point number [Table 9 - Shifter block signals description].

In case pdata[7:0] isn't belong to the valid discrete values group [valid set in

fixed_point	value	cp_param
00000000	0.000	000000000
00000001	0.125	000000111
00000010	0.250	000000110
00000011	0.375	001110110
00000100	0.500	000000101

00000101	0.625	001110101
00000110	0.750	001100101
00000111	0.875	101110100
00001000	1.000	000000100
00001001	1.125	001000111
00001010	1.250	001000110
00001100	1.500	001000101
00001110	1.750	101101001
00001111	1.875	101111001
00010000	2.000	000001001
00010001	2.125	010010111
00010010	2.250	010010110
00010100	2.500	010010101
00011000	3.000	001001001
00011100	3.500	101011010
00011110	3.750	101101010
00011111	3.875	101111010
00100000	4.000	000001010
00100001	4.125	010100111
00100010	4.250	010100110
00100100	4.500	010100101
00101000	5.000	010100100
00110000	6.000	010101001

Table 17 - contrast parameter valid set as read from contrast LUT] controller will get an error message.

fixed_point	**value**	**cp_param**
00000000	0.000	000000000
00000001	0.125	000000111
00000010	0.250	000000110
00000011	0.375	001110110
00000100	0.500	000000101
00000101	0.625	001110101
00000110	0.750	001100101

00000111	0.875	101110100
00001000	1.000	000000100
00001001	1.125	001000111
00001010	1.250	001000110
00001100	1.500	001000101
00001110	1.750	101101001
00001111	1.875	101111001
00010000	2.000	000001001
00010001	2.125	010010111
00010010	2.250	010010110
00010100	2.500	010010101
00011000	3.000	001001001
00011100	3.500	101011010
00011110	3.750	101101010
00011111	3.875	101111010
00100000	4.000	000001010
00100001	4.125	010100111
00100010	4.250	010100110
00100100	4.500	010100101
00101000	5.000	010100100
00110000	6.000	010101001

Table 17 - contrast parameter valid set as read from contrast LUT and how it is mapped to cp_param

7. Alternative Solutions

In the development of this project different approaches were considered in key aspects.

7.1. Datapath synchronization and memory

The project started with an idea that a whole frame should be read and stored in buffer memory to be processed asynchronously from the AXI-Stream transmission rate. Thus, requiring an IO buffer and synchronization logic to prevent delay-skew between frames, but allowing for a constant operating frequency of the data path and processing between them. This approach was not required, because:

1. All the operations are pixel-wise; thus, frame memory is not required for processing.
2. The video IO (AXI-Stream) payload is a single pixel.
3. The video IO (AXI-Stream) has a clock signal (ACLK) that can be used to synchronize processing and avoid delay-skew problems altogether.
4. We have an upper bound based on resolution and FPS to design and close timing with (as resolution x FPS = pixels per second = AXI-Stream transmission rate).
   E.g. 24bit RGB HD 1920x1080 with 30 FPS requires a pixel rate of ~63MHz, which is an achievable frequency in the ancient 180nm process we are designing for.

7.2. Datapath architecture

Further analysis of the required enhancement algorithms showed that in RGB color space (which we selected for our design) they do not have cross-channel dependency. Effectively, brightness, contrast and gamma correction application are pixel-channel-wise. Therefore, they can be applied on a pixel's channels in-series or on all 3 in-parallel.
We considered serial computation at first, that would have resulted in a smaller area of the processing data path at the cost of a lower bitrate (x3 clocks for pixel than in-parallel). Yet, an additional logic and memory would have been required to buffer the 3 channels and feed them in sequence to the processing data path. HD video with 30 FPS would have required a frequency of ~186MHz, more challenging to implement on the 180nm we are designing for, or a more complex pipelining of basic arithmetic operations.
We opted to use the parallel approach to reduce complexity and achieve a higher possible pixel/sec rate, thus allowing support of higher resolutions and FPS rates.

7.3. Data type – color space

Although we chose RGB color coding, there are various color representations, such as HSB, HSL and HSV [2.1].

However, the other alternatives aren't in base-2 nor additive. Therefore, mathematical operations been made in the algorithms such multiplication and subtraction/addition will be much more expensive, in terms of hardware, latency and complexity.

7.4. Arithmetical modules micro-architecture - multiplier

Brightness and contrast operations require an arithmetical module to perform addition, subtraction, multiplication, and division with upper/lower bound keeping (overflow protection).
At first, we considered using an ALU IP. Experimenting with contrast enhancement, we noted that there is no need to support very large or very small multipliers and that it is sufficient to work with a closed set of factors. Thus, we could optimize the costly multiplier module for a uint8 operand and a multiplication factor from a closed set of fixed-point powers of 2.
We opted to use a synthesized ADD/SUB module and a shifter-based multiplier [Figure 15 – Contrast block schematic draw].

7.5. Arithmetical modules micro-architecture - identity operation vs bypass

One way to perform identity operations through algorithms is to rely on user's appropriate input parameters ($C = 1,\ B = 0\ , \gamma = 1$) without any additional hardware.

However, we chose to allow bypass by using power gating, which means turning off inactive algorithms' logic blocks, to avoid power waste.

8. Simulation & Verification

A functional verification flow was implemented using RTL simulation using System Verilog testbench and MATLAB script [Figure 31]. Its purpose is to prove general functionality of our design in working conditions, i.e. with IO signals being driven as in real usage cases and transferring real video frames.

The MATLAB script gif_tb.m [page 17] is used in this flow. It is divided into sections for parameter input, GIF loading, intermediate file creation, reference model application and comparison of System Verilog simulation results with reference model.

The reference model is implemented using MATLAB imadjust for gamma correction and arithmetic operations for contrast and brightness. Color depth of 24bits / 8bits per channel considered, values of pixel channels are capped after each calculation.

Comparison of simulation to model values is done pixel to pixel by location in frame. An "error" is counted if a pixel has a deviation of 1 uint8 or more on one or more channels. E.g.

$$reference_{x=1,y=1} = (0,0,1) \neq (0,0,0) = simulation_{x=1,y=1}$$
$$\rightarrow error = error + 1$$

As the test bench drives AXI-Lite (parameters and control) and AXI-Stream (video IO) in parallel from the start of simulation, we expect 1st frame to have a batch of pixels on which the correct parameters were not applied, since parameter loading to design is not immediate. Therefore, some "error" count on 1st frame is expected. (The minimal time from parameter loading until application to video output is 6 clks of AXI-Stream [page 62])

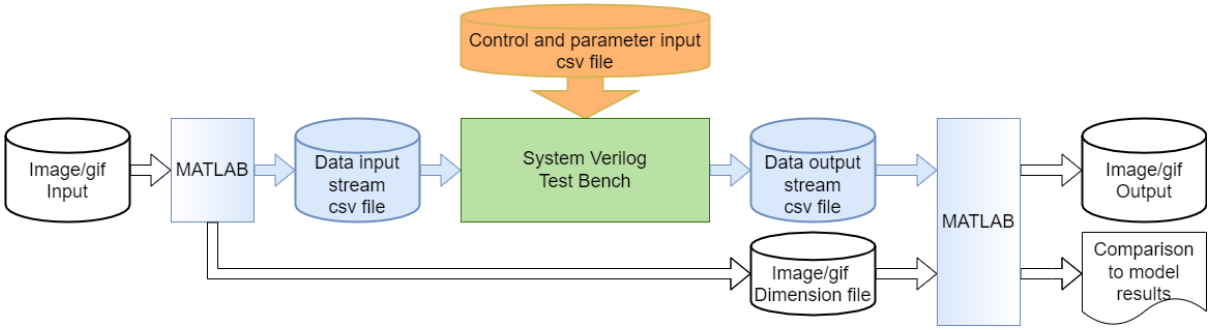

Figure 31 - Verification Flow

Verification inputs:

1. GIF image – multi frame image similar to video. (See in [Figure 32]).
2. Control parameters from the test table, inserted as:
    a. Variables to MATLAB script.
    b. CSV file with AXI-Lite control and parameter input transactions to System Verilog testbench. (See file structure in [**Error! Reference source not found.**]).

Verification outputs:

3. Graph of selected frames from model and testbench with error count of pixels that have differing values.

Verification intermediate results:

4. CSV file with pixel values of original GIF frames, as input to testbench.
5. Text file with size/resolution of GIF frames, as input to MATLAB 2nd stage.
6. CSV file with pixel values of enhanced GIF frames, as input to MATLAB 2nd stage.

Verification flow stages:

1. Load MATLAB script and set path to GIF (1) and contrast, brightness, gamma factor parameters (2.a).
2. Run MATLAB script to create intermediate files (4, 5).
3. Create/change control and parameter input CSV file with AXI-Lite transactions (2.b).
4. Load files (4, 2.b) to System Verilog testbench environment and set paths in testbench top file.
5. Run simulation to create file (6).
    a. In test #1 take waveforms of simulated events.
6. Load file (6) to MATLAB environment and set path in script.
7. Run last section of MATLAB script to create output graph (3), save if needed.

8.1. Verification – Test Table and Inputs

All the tests we implemented are described in table below.
In this document we detail tests #1, #4, #9 (in **bold**).

Test	Purpose / Comment	Gamma	Contrast	Brightness	Invalid address
AXI-Lite, control, memory					
1	**Parameters read write**	**Write to all LUT according to 0.5** **Read 1** **Read 100**	**Write 0.5** **Read**	**Write -30** **Read**	**Write** **Read**
AXI-Stream, datapath only					
2	Bypass				
3	Gamma	On 0.5			
4	**Contrast (minimal delay from param change to output)**		**On** **0.5**		
5	Brightness			On -30	
6	Combined datapath	On 0.5	On 0.5		
7	Combined datapath		On 0.5	On -30	
8	Combined datapath	On 0.5		On -30	
9	**Full datapath**	**On** **0.5**	**On** **0.5**	**On** **-30**	

Table 18 - Simulation tests

Input GIF file (1) used in current verification flow

Playable in Word version of this document.
(We note this project hit the verification phase during the 2022 FIFA World Cup.)

Figure 32 - Verification input GIF file

Input control and parameters file (2.b) – AXI-Lite transactions

For general reference to structure of file, see detailed explanation in Test #4 [page 62].

Consists of channel signals of AXI-Lite interface originating in master, that are pushed by System Verilog testbench to slave at each ACLK, simulating functional operation.
Each transaction includes a comment string, which can be displayed in simulation environment (e.g. in waveforms of Test #1 [page 55]).

		Write Address			Write Data		Write Response	Read Data Address			Read Response	
wait_ACLKS	ARESETn	AWADDR[15:12]_-_bin	AWADDR[7:0]_-_dec	AWVALID	WDATA	WVALID	BREADY	ARADDR[15:12]_-_bin	ARADDR[7:0]_-_dec	ARVALID	RREADY	comment
0	1	0	0	0	0	0	0	0	0	0	0	nothing
0	0	0	0	0	0	0	0	0	0	0	0	**reset_params**
0	1	0	0	0	0	0	0	0	0	0	0	nothing
3	1	0010	0	1	-30	1	1	0	0	0	0	**load_brigthness_-_factor_-30**
3	1	0	0	0	0	0	1	0	0	0	0	load_brigthness_-_write_response?
0	1	0	0	0	0	0	0	0	0	0	0	nothing
3	1	0100	0	1	4	1	1	0	0	0	0	**load_contrast_-_factor_0.5**
3	1	0	0	0	0	0	1	0	0	0	0	load_contrast_-_write_response?
0	1	0	0	0	0	0	0	0	0	0	0	nothing
3	1	0	0	0	0	0	0	0110	0	1	1	read_invalid_addr
3	1	0	0	0	0	0	1	0	0	0	1	read_invalid_read_response
0	1	0	0	0	0	0	0	0	0	0	0	nothing
3	1	0001	0	1	6	1	1	0	0	0	0	**load_enables_-_ON,ON,OFF**
3	1	0	0	0	0	0	1	0	0	0	0	load_enables_-_write_response?

Table 19 - AXI-Lite transactions input

8.2. Verification – Test #1

Test #1 verifies the AXI-Lite interface, internal memory addresses to registers mapping and read/write operations. Therefore, waveforms are used, and output results of MATLAB reference model are ignored.

Incoming AXI-Lite transactions from external master can be either write or read requests referencing a valid or invalid address. Our design returns a relevant slave response in all those cases, as detailed in AXI-Lite background [Table 4 - AXI RESP codes].

All tested operations were successful and in accordance with AXI-Lite specifications.

Invalid write and read request responses: [Figure 33, Figure 34].

Valid write and read request pairs to same registers:

- Register “enables” – controls bypass of gamma, contrast, brightness.
  [Figure 35, Figure 36]

- Register “contrast” – factor of contrast operation
  (as received from user/AXI-Lite master).
  Note: “brightness” is identical in operation.
  [Figure 37, Figure 38]

- RAM LUT “gamma” – mapping of pixel channel values.
  [Figure 39, Figure 40, Figure 41]

Flow input (1) – GIF file, as in [Figure 32].

Flow input (2.a) – Parameters for MATLAB reference model.

None, reference model not used in this test.

Flow input (2.b) – Control and parameters file for testbench (AXI-Lite transactions)
Note: comments correspond to AXI_Lite_comment in waveforms on following pages.

Wait_ACLKS	ARESETn	AWADDR[15:12]-bin	AWADDR[7:0]-dec	AWVALID	WDATA	WVALID	BREADY	ARADDR[15:12]-bin	ARADDR[7:0]-dec	ARVALID	RREADY	comment
3	1	**100**	**0**	**1**	**4**	1	1	0	0	0	0	**load_contrast_-_factor_0.5**
3	1	0	0	0	0	0	**1**	0	0	0	0	load_contrast_-
0	1	**1000**	**1**	**1**	**16**	1	**1**	0	0	0	0	**load_gamma_lut**
3	1	**1000**	**100**	**1**	**160**	1	**1**	0	0	0	0	load_gamma_lut
3	1	0	0	0	0	0	0	**1000**	**1**	**1**	**1**	**read_gamma_lut**
3	1	0	0	0	0	0	0	**1000**	**100**	**1**	**1**	**read_gamma_lut**
3	1	0	0	0	0	0	0	**100**	**0**	**1**	**1**	**read_contrast**
3	1	**1**	**0**	**1**	**4**	1	1	0	0	0	0	load_enables_-_ON,OFF,OFF
3	1	0	0	0	0	0	**1**	0	0	0	0	load_enables_-_write_response?
3	1	0	0	0	0	0	0	**1**	**0**	**1**	**1**	read_enables
3	1	0	0	0	0	0	1	0	0	0	**1**	read_enables_-_read_response?
3	1	**1111**	**0**	**1**	4	1	1	0	0	0	0	**load_invalid_addr**
3	1	0	0	0	0	0	**1**	0	0	0	0	load_invalid_addr_-
3	1	0	0	0	0	0	0	**1100**	**0**	**1**	**1**	**read_invalid**
3	1	0	0	0	0	0	1	0	0	0	**1**	read_invalid_-_read_response?

Table 20 - Input (2.b) of test #1

Waveform outputs - Invalid read/write requests

A write/load request to an invalid address – BRESP SLVERR returned with BVALID handshake.

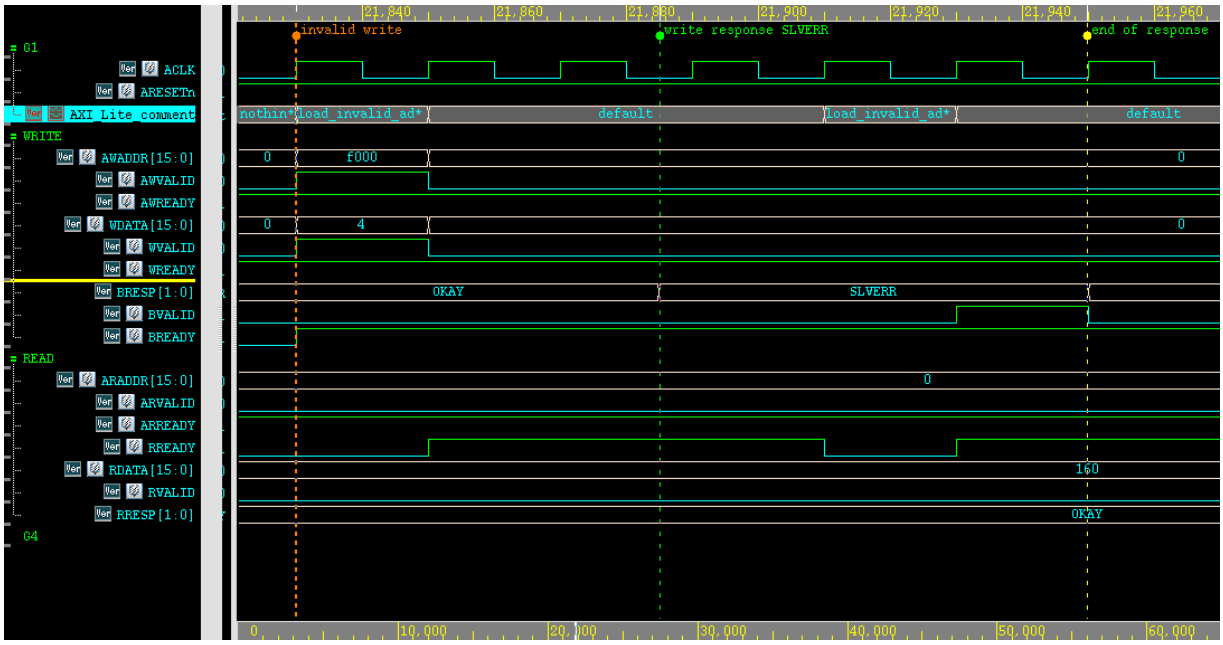

Figure 33 - AXI-Lite invalid write request

A read request from an invalid address – RRESP SLVERR returned with RVALID handshake.

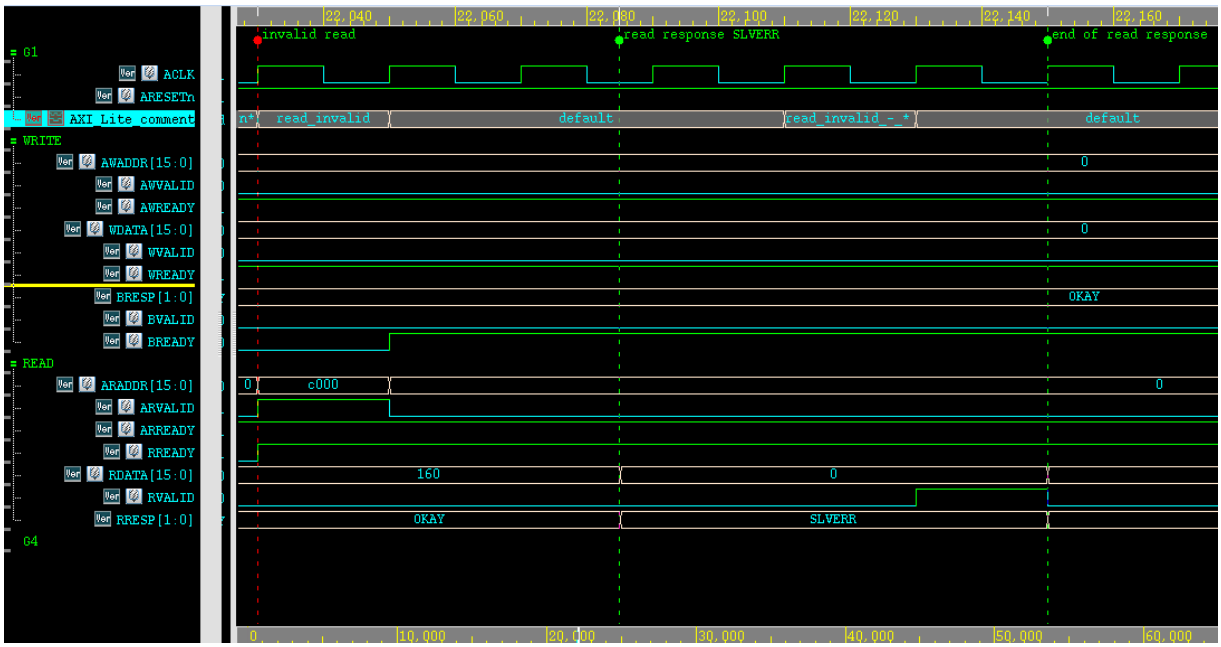

Figure 34 - AXI-Lite invalid read response

Waveform outputs - Valid read/write requests to control register “enables”

A valid write request to control register “enables” of value “4” – BRESP OKAY returned with BVALID handshake.

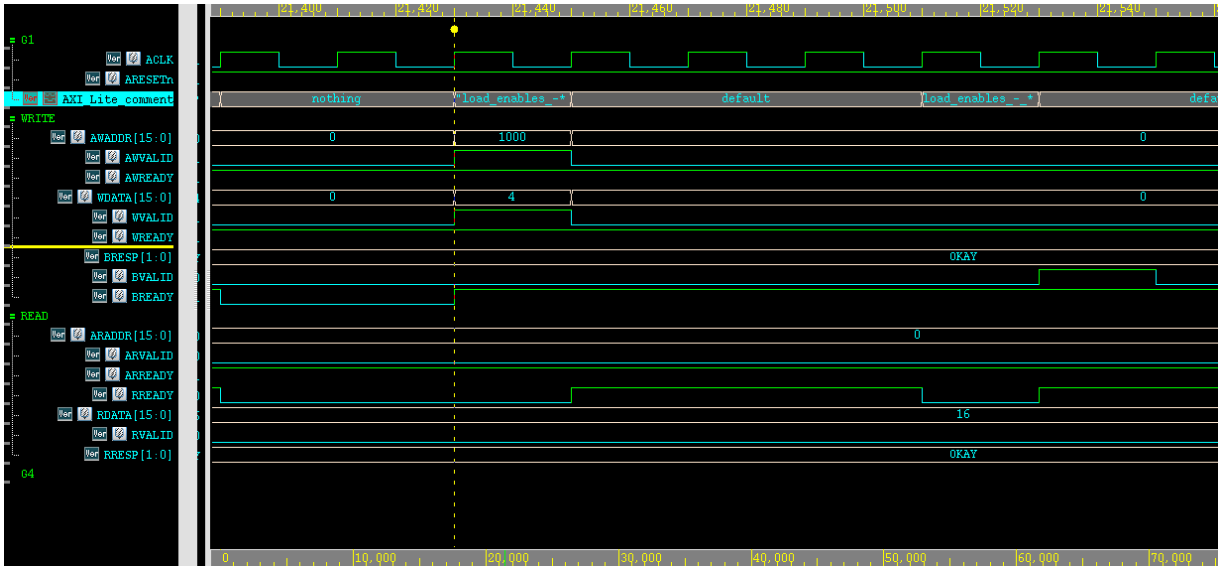

Figure 35 - AXI-Lite write 4 to enables

A valid read request from control register “enables” – previously written value “4” returned on RDATA with RRESP OKAY and RVALID handshake.

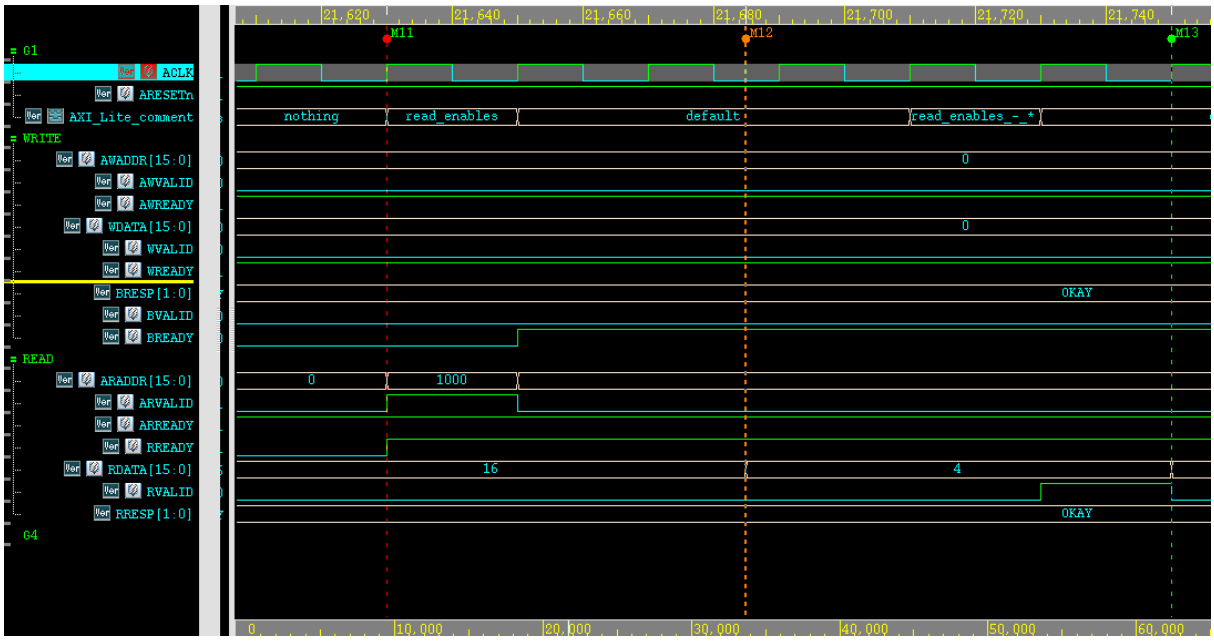

Figure 36 - AXI-Lite read 4 from enables

Waveform outputs - Valid read/write requests to parameter register “contrast”

A valid write request to parameter register “contrast” of value “4” (that is x0.5 in fixed-point format used) – BRESP OKAY returned with BVALID handshake.

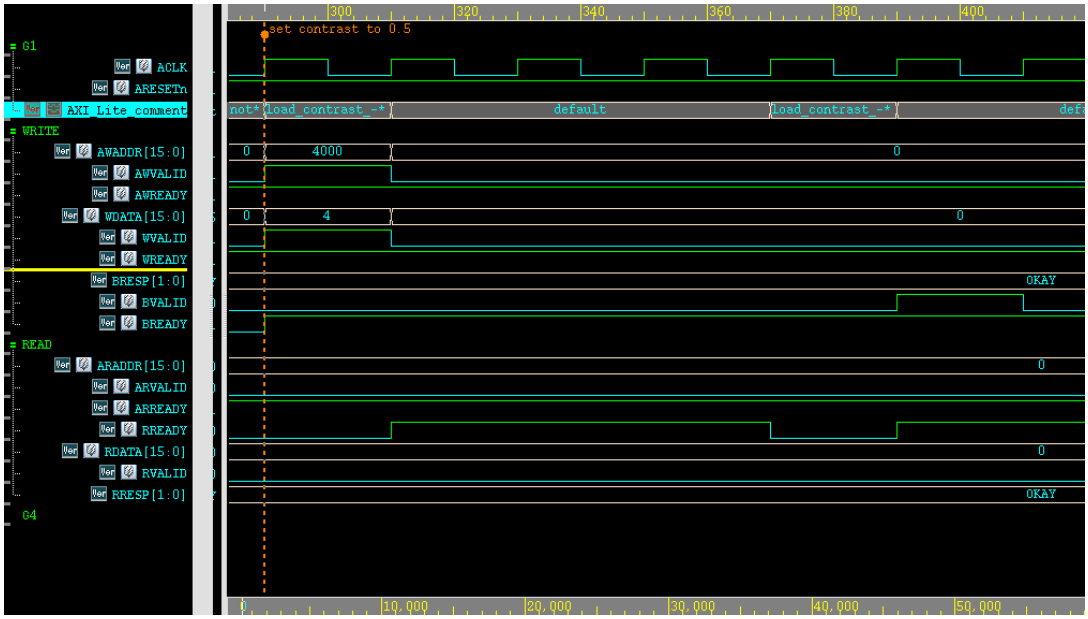

Figure 37 - AXI-Lite write 4 to contrast

A valid read request from parameter register “contrast” – previously written value “4” returned on RDATA with RRESP OKAY and RVALID handshake.

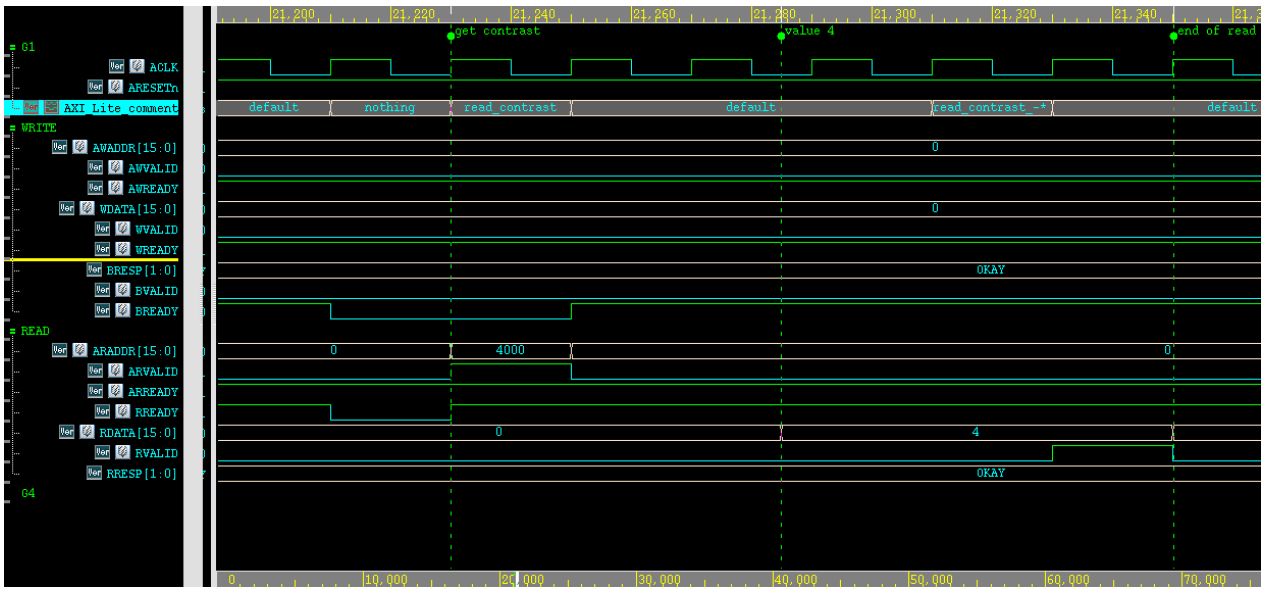

Figure 38 - AXI-Lite read 4 from contrast

Waveform outputs - Valid read/write requests to LUT “gamma” (internal RAM)

A valid write request to internal RAM LUT “gamma” of value “160” to address “64” (in hex, 100 in decimal) (that is mapping 64 to 160) – BRESP OKAY returned with BVALID handshake.

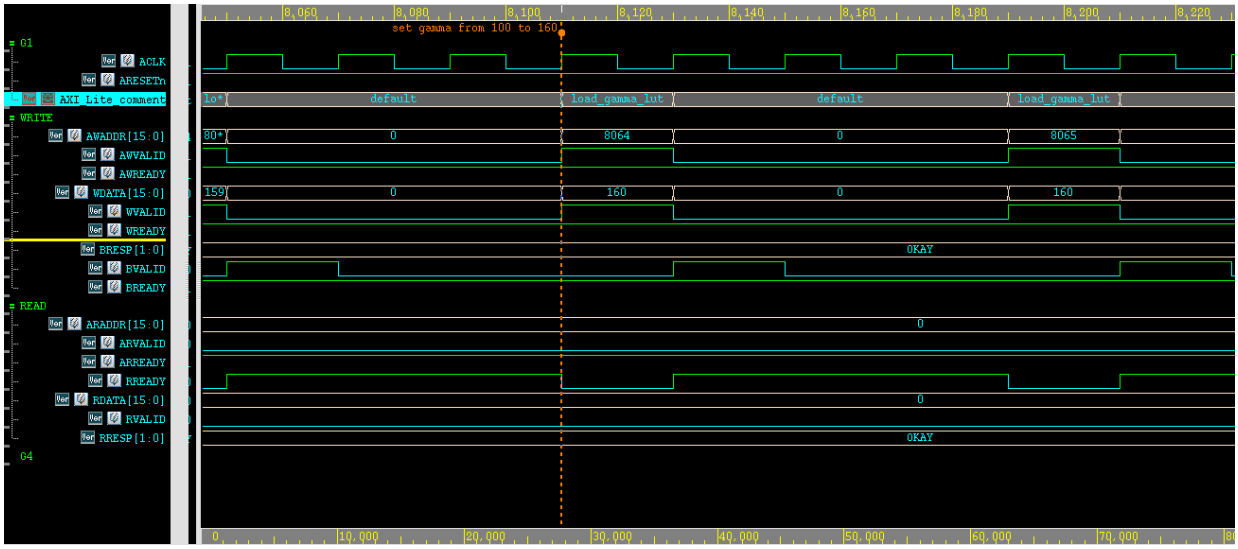

Figure 39 - AXI-Lite write “160” to addr “64” in gamma LUT

Valid read requestes from to internal RAM LUT “gamma”– previously written mappings “1” to “16” and “64” (in hex) to “160” returned on RDATA with RRESP OKAY and RVALID handshakes.

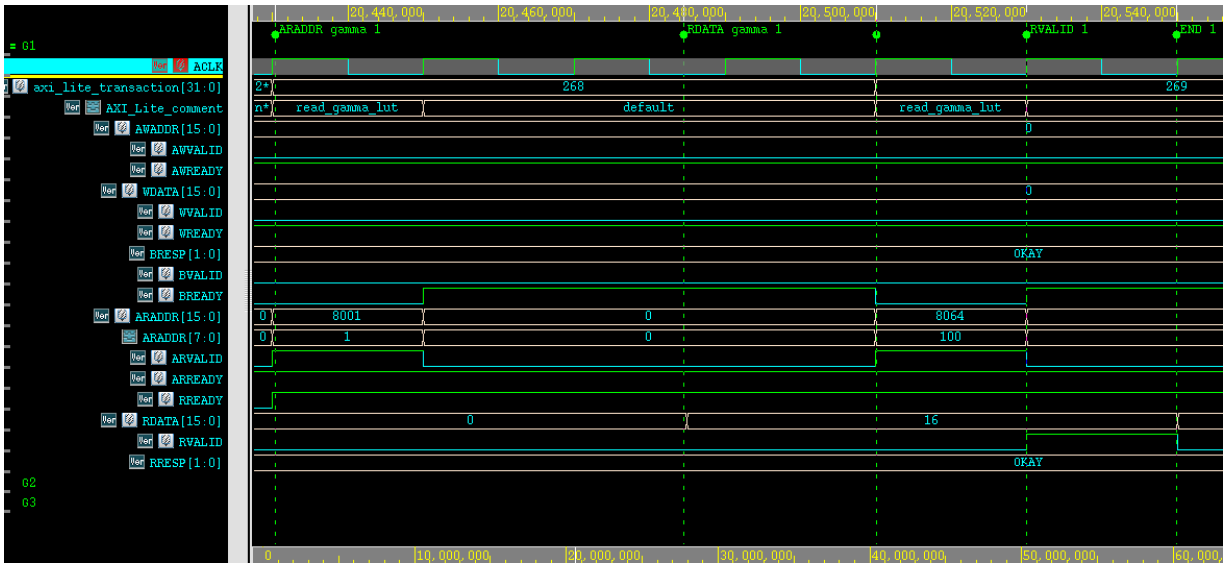

Figure 40 - AXI-Lite read “16” from addr “1” in gamma LUT

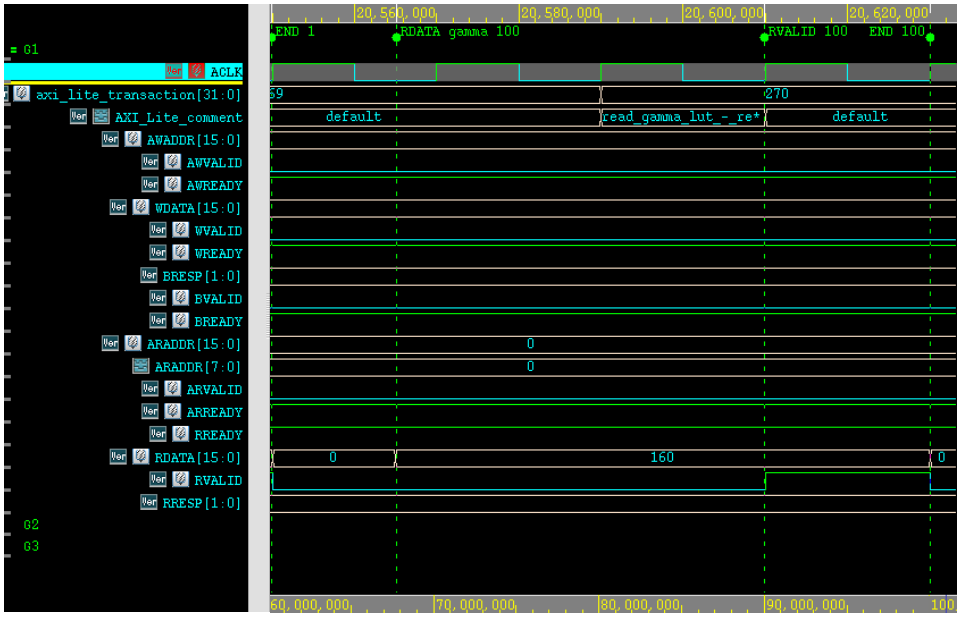

Figure 41 - AXI-Lite read "160" from addr "64" in gamma LUT (see Fig 31)

8.3. Verification – Test #4

Verifies contrast operation (with brightness and gamma LUT bypassed) in fully functional mode and with minimal delay from simulation start until parameter setting transactions pushed by user (AXI-Lite master emulated by testbench).

Flow input (1) – GIF file, as in [Figure 32].
Flow input (2.a) – Parameters for MATLAB reference model

Gamma	Contrast	Brightness
Off	On, factor 0.5	Off

Table 21 - Input (2.a) of test #4

Flow input (2.b) – Control and parameters file for testbench (AXI-Lite transactions)

Wait_ACLKS	ARESETn	AWADDR[15:12]-bin	AWADDR[7:0]-dec	AWVALID	WDATA	WVALID	BREADY	ARADDR[15:12]-bin	ARADDR[7:0]-dec	ARVALID	RREADY	comment
0	0	0	0	0	0	0	0	0	0	0	0	reset_params
0	1	**100**	0	1	**4**	**1**	1	0	0	0	0	**load_contrast_-_factor_0.5**
0	1	**1**	0	1	**2**	**1**	1	0	0	0	0	**load_enables_-_OFF,ON,OFF**
10	1	0	0	0	0	0	1	0	0	0	0	load_enables_-_write_response?

Table 22 - Input (2.b) of test #4

Flow output (3) – Comparison of selected frames reference model vs simulation with count of differing pixels.

Error count of 6 pixels in 1st frame only is the number of AXI-Stream clocks (up_ACLK) that it took for transactions in input (2.b) [Table 22] to be implemented by the design and affect actual output. This is an expected result we consider as good.

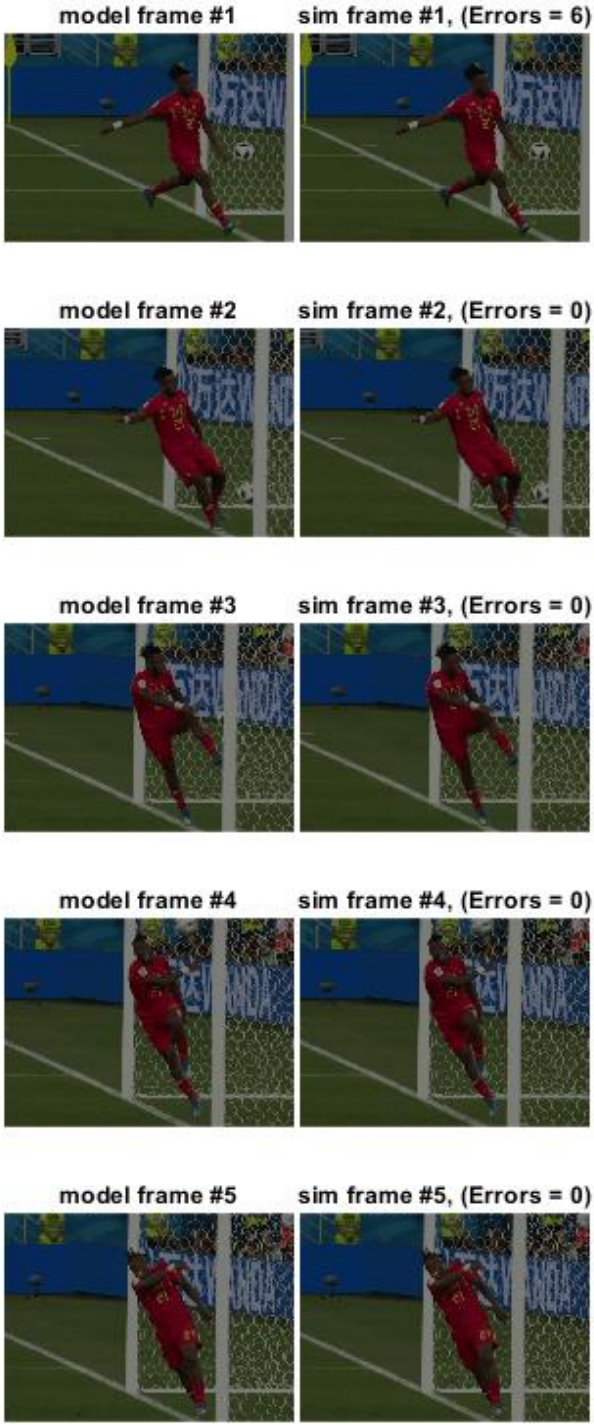

Figure 42 - Output (3) of test #4

8.4. Verification – Test #9

Verifies operation of all datapath modules enabled in fully functional mode.

Flow input (1) – GIF file, as in [Figure 32].
Flow input (2.a) – Parameters for MATLAB reference model

Gamma	Contrast	Brightness
On, factor 0.5	On, factor 0.5	On, factor -30

Table 23 - Input (2.a) of test #9

Flow input (2.b) – Control and parameters file for testbench (AXI-Lite transactions)
Part of full table with bulk of gamma LUT loading transactions omitted.

Wait_ACLKS	ARESETn	AWADDR[15:12]-bin	AWADDR[7:0]-dec	AWVALID	WDATA	WVALID	BREADY	ARADDR[15:12]-bin	ARADDR[7:0]-dec	ARVALID	RREADY	comment
0	**0**	0	0	0	0	0	0	0	0	0	0	**reset_params**
0	1	**1000**	**0**	**1**	**0**	1	1	0	0	0	0	**load_gamma_lut_-_start**
0	1	**1000**	**1**	**1**	**16**	1	1	0	0	0	0	**load_gamma_lut**
3	1	**1000**	**255**	**1**	**255**	1	1	0	0	0	0	**load_gamma_lut_-_end**
3	1	0	0	0	0	0	1	0	0	0	0	load_gamma_lut_-_last_write_response?
0	1	0	0	0	0	0	0	0	0	0	0	nothing
3	1	**100**	0	**1**	**4**	1	1	0	0	0	0	**load_contrast_-_factor_0.5**
3	1	0	0	0	0	0	1	0	0	0	0	load_contrast_-_write_response?
0	1	0	0	0	0	0	0	0	0	0	0	nothing
3	1	**10**	0	**1**	**-30**	1	1	0	0	0	0	**load_brigthness_-_factor_-30**
3	1	0	0	0	0	0	1	0	0	0	0	load_brigthness_-_write_response?
0	1	0	0	0	0	0	0	0	0	0	0	nothing
3	1	**1**	0	**1**	**7**	1	1	0	0	0	0	**load_enables_-_ON,ON,ON**
3	1	0	0	0	0	0	1	0	0	0	0	load_enables_-_write_response?

Table 24 - Input (2.b) of test #9

Flow output (3) – Comparison of selected frames reference model vs simulation with count of differing pixels.

Error count of 2067 pixels in 1st frame only is the number of AXI-Stream clocks (up_ACLK) that it took for transactions in input (2.b) [Table 24] to be implemented by the design and affect actual output. This is more than in test #4, as the full 256 values of the gamma LUT were loaded in this test. No mismatches in following frames prove the functionality of our design.

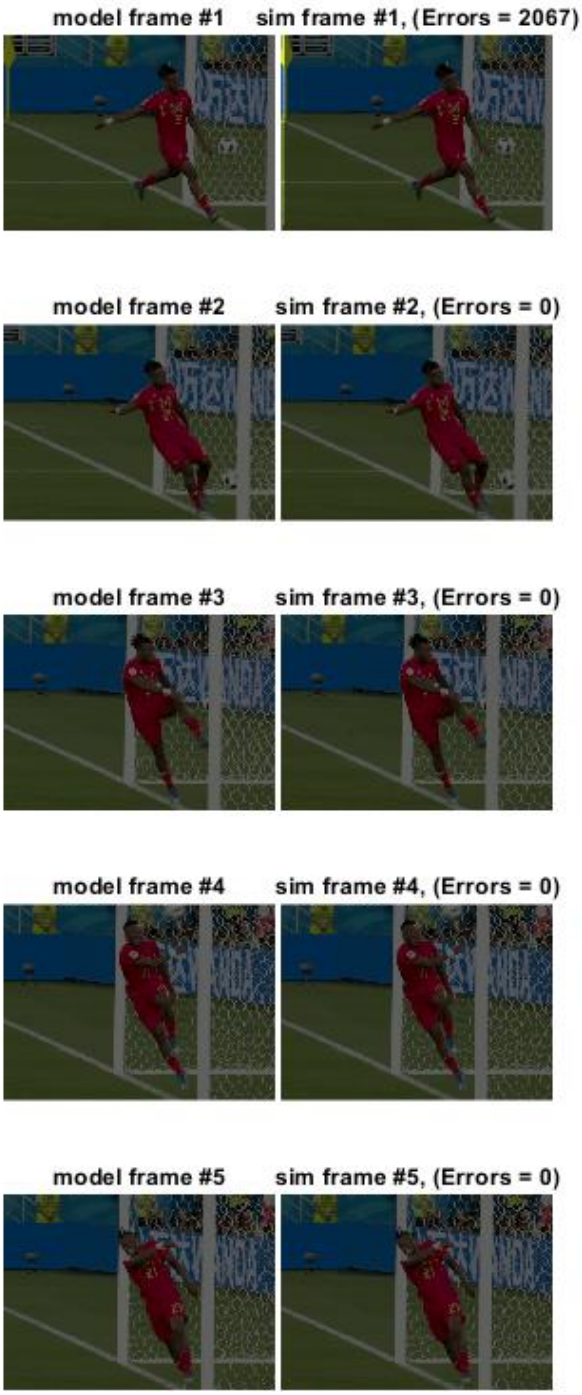

Figure 43 - Output (3) of test #9

9. Synthesis & Elaboration

Synthesis was implemented using Synopsys Design Vision for Tower TSL 018 technology according to VLSI lab manual [3].

9.1. Timing Analysis

2 clocks were defined for timing analysis at maximal operating frequencies of the design and the analysis was executed with following flags*: -path full -delay max -max_paths 1*.
Max delay paths per clock and cross-clocks resulted in a non-marginal positive slack.
The good results give us confidence in the CDC FIFO implementation.

Net name	**Usage**	**Period**	**Frequency**	**Worst slack**
up_ACLK	AXI-Stream clock. Defines data rate of datapath / Pixel IO freq. **Data rate of HD 30FPS video.**	$10\ ns$	$100\ MHz$	$3.53\ ns$ Met
ACLK	AXI-Lite clock. Defines data rate of parameter and control registers loading.	$20\ ns$	$50\ MHz$	$9.64\ ns$ Met

Table 25 - Synthesis Timing Analysis

9.2. Area Analysis

Area	**NAND2 eqv**
Combinational area	27506.50
Buffer/Inverter area	5997.00
Noncombinational area	61423.25
Macro/Black Box area	0.00
Net Interconnect area	35670.70
Total cell area	**88929.75**
Total area	**124600.45**

Table 26 - Synthesis Area Analysis

9.3. Power Analysis

Was executed with flag: *-analysis_effort low* and Global Operating Voltage = 1.8V.

The total operating power resulted in 59.45 mW.

Power Group	Internal Power mW	Switching Power mW	Leakage Power pW	Total Power mW	% of Total
io_pad	0.0000	0.0000	0.0000	0.0000	0.00%
memory	0.0000	0.0000	0.0000	0.0000	0.00%
black_box	0.0000	0.0000	0.0000	0.0000	0.00%
clock_network	0.0000	0.0000	0.0000	0.0000	0.00%
register	56.3816	0.0235	1.3441e+06	56.4064	94.87%
sequential	0.0000	0.0000	0.0000	0.0000	0.00%
combinational	0.4374	2.6121	3.9089e+05	3.0500	5.13%
Total	**56.8190 mW**	**2.6356 mW**	**1.7350e+06 pW**	**59.4564 mW**	**100%**

Table 27 - Synthesis Power Analysis

10. Layout

Layout was implemented using Cadende Innovus 20.11 for Tower TSL 018 technology according to the lab's manual [4] without external RAM.

The default die size was used, resulting in a total chip area of 10.273 mm^2.
Total area of standard cells with physical cells is 6.257 mm^2
Total area of standard cells without physical cells is 1.067 mm^2.

We consider this a good result for a 180 nm technology node – our design is production viable in terms of total area.

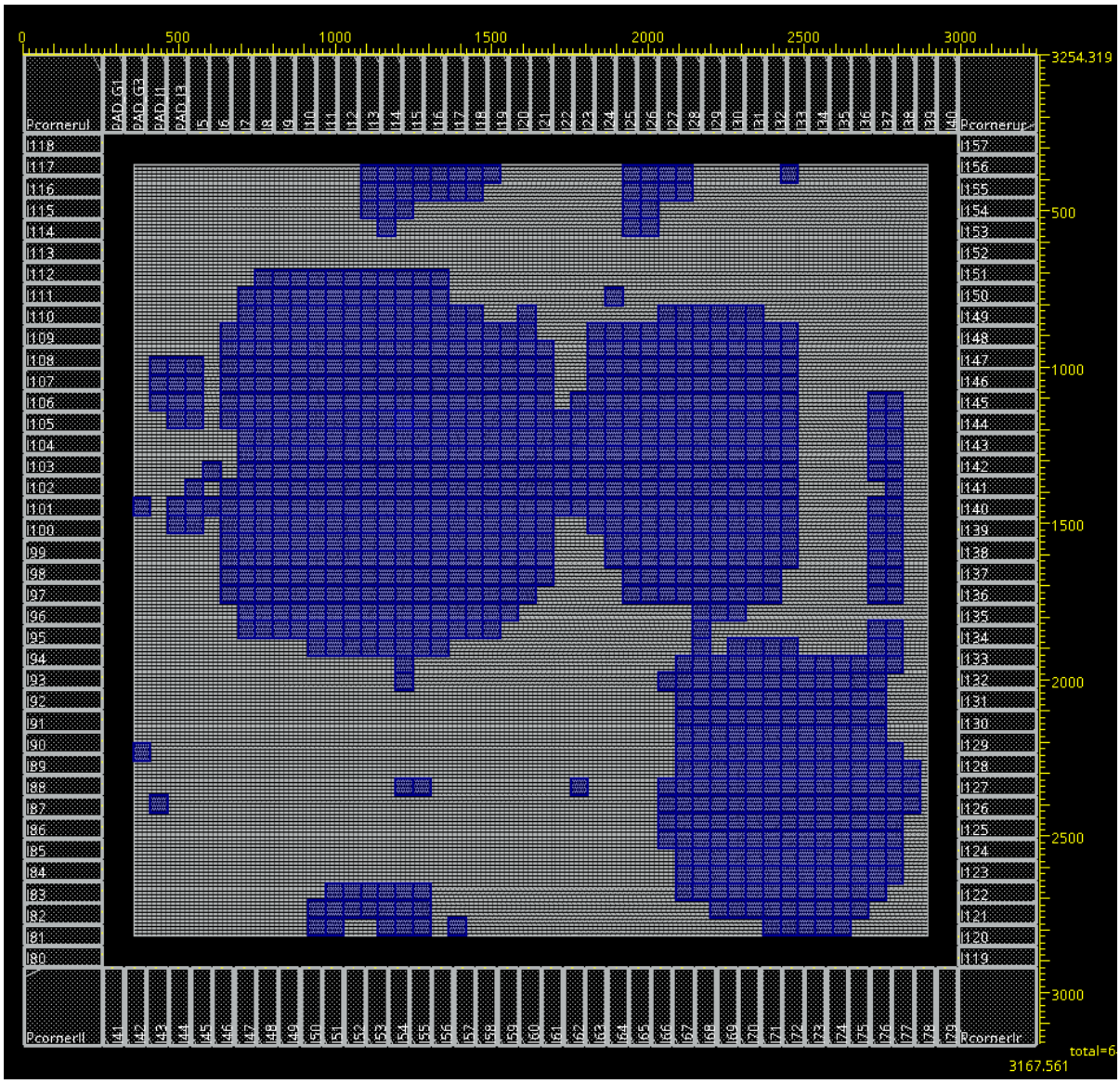

Figure 44 - Layout placement without metals. Sizes in µm.

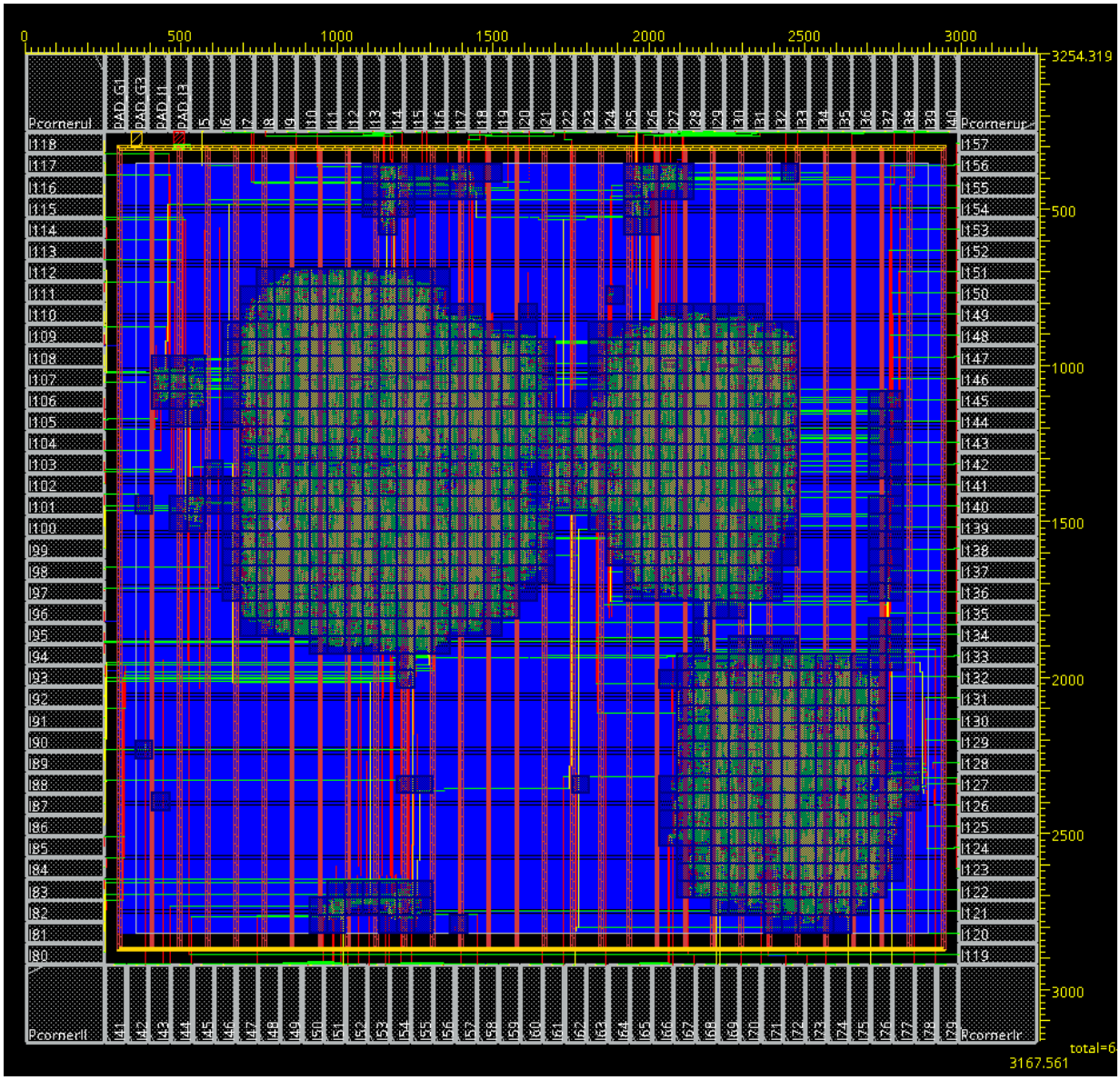

Figure 45 – Layout placement with metals. Sizes in µm.

11. Summary

We met and exceeded the maximal supported data rate expectation by using a pipelined, parallelized approach and building an optimized design.
The original requirement discussed with our supervisor was to support a video stream of 400x400 at 30FPS; it was reasoned that a design which will support contemporary video resolutions may be too heavy for a 180nm process to synthesize and layout for.
Through high optimization and proper planning of the data path architecture, our design supports HD 30FPS video delivered via AXI-Stream clocked at 100MHz [Table 25]. We surpassed the initial data rate expectation of ~28MBs by a factor of x6 achieving an analysis-based data rate ~186 MB/s within viable area and power constraints.

A critical factor in achieving this result was understanding the algorithm we are designing for. Once we gained the knowledge that the desired operations are performed on the single pixel's channel value level [Equation 1,Equation 2], and, critically, not on the frame level, we were able to design an architecture around that [Figure 3]. Thus, we could let go of an initial assumption the project was started with - that a full frame must be preloaded to memory to operate on [Page 49].

The major challenge we encountered was building to spec – correctly implementing an interface to industry-standard bus communication protocols with no reference model provided. AXI-Lite slave and AXI-Stream master and slave interface modules were designed as part of this project and verified by connecting master ↔ slave.
On the way, we learned about CDC and accounted for it in our design by integrating IP of FIFOs, as is widely practiced in the VLSI industry.

An early investment in building a functional verification/simulation environment and flow around files instead of hard-coded values allowed us to efficiently simulate a variety of scenarios by simply changing a table of values.

To conclude, our design has achieved the goals and requirements laid out in [page 13].

12. List of Abbreviations

4K	Resolution of 3840x2160 pixels
ALU	Arithmetic Logic Unit (digital design)
AXI	Advanced eXtensible Interface (protocol)
CDC	Clock Domain Crossing (digital design)
CSV	Comma-Separated Values (file type)
FIFO	First In – First Out (memory)
FPS	Frames per Second (video)
HD	High Definition video resolution of 1920x1080 pixels (video)
IP	Intellectual Property (digital design)
RGB	Red Green Blue (color space)
LUT	Look-Up Table (memory)

13. References

[1] ARM, “AMBA 4 AXI4-Stream Protocol,” 2010. https://developer.arm.com/documentation/ihi0051/a/ (accessed Feb. 24, 2023).

[2] ARM Limited, “AMBA AXI and ACE Protocol Specification,” *Arm*, 2010. https://developer.arm.com/documentation/ihi0022/e/AMBA-AXI4-Lite-Interface-Specification (accessed Feb. 24, 2023).

[3] G. Samuel and A. Stanislavsky, “חוברת הדרכה על כלי SYNOPSYS,” Oct. 2022. Accessed: Mar. 03, 2023. [Online]. Available: https://vlsi.eelabs.technion.ac.il/wp-content/uploads/sites/18/2022/10/synopsys2021_vcs_mx_dvV3.pdf.

[4] G. Samuel, “חוברת הדרכה על בניית Layout,” Aug. 2021. Accessed: Mar. 03, 2023. [Online]. Available: https://vlsi.eelabs.technion.ac.il/wp-content/uploads/sites/18/2021/08/innovus20_tsl018.pdf.

[5] Alasdair McAndrew; Hung-Hua Wang; Chun-Shun Tseng, *Introduction to Digital Image Processing with Matlab*. Victoria University of Technology, 2010.