

Stereo Vision IP Core

Data Sheet

(IP Version 6.0.0) April 7, 2021



Nerian Vision GmbH
Zettachring 2
70567 Stuttgart
Germany

Email: service@nerian.com
www.nerian.com

Contents

1	Introduction	4
2	Features	4
3	Background	5
3.1	Camera Alignment	5
3.2	Image Rectification	5
3.3	Camera Calibration	6
3.4	Disparity Maps	6
4	Stereo Vision Core Functionality	8
4.1	Image Input	8
4.1.1	8-Bit Monochrome	8
4.1.2	12-Bit Monochrome LSB Packed	10
4.1.3	12-Bit Monochrome GEV Packed	10
4.1.4	12-Bit Unpacked	10
4.1.5	8-Bit RGB	10
4.1.6	8-Bit Bayer Pattern	11
4.2	Image Output	11
4.3	Rectification	11
4.4	Image Pre-Processing	12
4.5	Stereo Matching	12
4.6	Cost Volume Post-Processing	13
4.6.1	Subpixel Optimization	13
4.6.2	Uniqueness Check	13
4.6.3	Consistency Check	14
4.7	Disparity Map Post-Proc.	14
4.7.1	Texture Filtering	14
4.7.2	Speckle Filtering	14
4.7.3	Gap Interpolation	14
4.7.4	Noise Reduction	15
5	DMA Core Functionality	15
5.1	Ports Connected to SVC	15
5.2	Interface Ports	15
5.3	Output Format Conversion	16
5.3.1	12-Bit Packed Output	17
5.3.2	12-Bit Split Output	17
5.3.3	16-Bit Output	18
6	Customization	18
6.1	SVC Customization Parameters	18
6.1.1	General Settings	18
6.1.2	Image Data	20
6.1.3	Processing	21
6.1.4	DMA Core Settings	22

6.2	DMA Core Customization Parameters	22
6.2.1	General Settings	22
6.2.2	DMA Settings	23
7	Supported Devices	24
8	Timing	24
9	Resource Usage	25
10	IO Signals	25
11	Registers	31
11.1	DMA Core Registers	33
11.1.1	0x00: Control	33
11.1.2	0x04: Status	36
11.1.3	0x08: Image Size	36
11.1.4	0x1C: Left Input Address Higher 32 Bits	36
11.1.5	0x20: Left Input Address Lower 32 Bits	37
11.1.6	0x24: Left Input Bytes Available	37
11.1.7	0x28: Right Input Address Higher 32 Bits	37
11.1.8	0x2C: Right Input Address Lower 32 Bits	37
11.1.9	0x30: Right Input Bytes Available	37
11.1.10	0x34: Input FIFO Info	37
11.1.11	0x38: Rectification Map Address Higher 32 Bits	38
11.1.12	0x3C: Rectification Map Address Lower 32 Bits	38
11.1.13	0x40: Rectification Map FIFO Info	38
11.1.14	0x44: Buffer Address Higher 32 Bits	38
11.1.15	0x48: Buffer Address Lower 32 Bits	38
11.1.16	0x4C: Buffer FIFO Info	39
11.1.17	0x50: DMA Errors	39
11.1.18	0x58: IP Version Number	39
11.1.19	0x5C: Left Output Address Higher 32 Bits	39
11.1.20	0x60: Left Output Address Lower 32 Bits	39
11.1.21	0x64: Left Output Bytes Available	39
11.1.22	0x68: Right Output Address Higher 32 Bits	40
11.1.23	0x6C: Right Output Address Lower 32 Bits	40
11.1.24	0x70: Right Output Bytes Available	40
11.1.25	0x74: Disparity Output Address Higher 32 Bits	40
11.1.26	0x78: Disparity Output Address Lower 32 Bits	40
11.1.27	0x7C: Disparity Output Bytes Available	40
11.1.28	0x80: Left / Right Output FIFO Info	41
11.1.29	0x84: Disparity output FIFO info	41
11.1.30	0x88 – 0xC4: Debugging Registers	41
11.2	SVC Registers	41
11.2.1	0x00: Control	41
11.2.2	0x04: Image Size	43
11.2.3	0x08: Algorithm Parameters 1	43

11.2.4	0x0C: Algorithm Parameters 2	43
11.2.5	0x10: License Key Higher 32 Bits	44
11.2.6	0x14: License Key Middle 32 Bits	44
11.2.7	0x18: License Key Lower 32 Bits	44
11.2.8	0x1C: Device DNA Higher 32 Bits	44
11.2.9	0x20: Device DNA Middle 32 Bits	44
11.2.10	0x24: Device DNA Lower 32 Bits	45
11.2.11	0x28: SVC Errors	45
11.2.12	0x2C: SGM Penalties	45
11.2.13	0x30: Subpixel Optimization ROI Offset	45
11.2.14	0x34: Subpixel Optimization ROI Size	46
11.2.15	0x38: IP Version Number	46
11.2.16	0x3C: Multi-Frame Subpixel Optimization	46
11.2.17	0x40: Statistics on Rectification Input Caches	47
11.2.18	0x44 – 0x98: Debugging Registers	47
12	Reference Design	47
13	Control Flow	48
13.1	One-Time Initializations	48
13.2	Per-Frame Control Flow	50
13.3	Result Retrieval	51
14	Migrating from Previous Versions	51
14.1	Migrating from Version 5.3 to 6.0	51
14.2	Migrating from Version 5.2 to 5.3	51
14.3	Migrating from Version 4.4 – 4.6 to 5.0 – 5.2	52
14.4	Migrating from Version 4.3 to 4.4	52
14.5	Migrating from Versions 4.0 – 4.2 to 4.3	52
14.6	Migrating from Version 3.x to 4.0	53

1 Introduction

The Stereo Vision Core (SVC) performs stereo matching on two grayscale or RGB input images. The images are first rectified to compensate for lens distortions and camera alignment errors. Stereo matching is then performed by applying a variation of the Semi Global Matching (SGM) algorithm as introduced by Hirschmüller (2005). Various post-processing methods are applied to improve the processing results. The output of the SVC is a subpixel accurate and dense disparity map, which is streamed over an AXI4-Stream interface.

To simplify the use of the SVC on devices with a shared system memory, such as the Xilinx Zynq SoC, an auxiliary core for direct memory access (DMA) is provided. This DMA core reads input data from memory through AXI3 or AXI4, and converts it into data streams that are suitable for the SVC. Likewise, the DMA core also collects the output data from the SVC and writes it back to memory.

Both IP cores are provided as encrypted RTL code. An IP block for each core is available for Xilinx Vivado IP Integrator.

2 Features

The SVC and DMA core comprise the following features:

- General processing architecture
 - Processing of grayscale images with a bit depth of 8 or 12 bits per pixel
 - Processing of color images with 8-bit RGB encoding
 - Stream-based processing of input images using either AXI4-Stream, AXI3 or AXI4
 - Configuration through AXI4-Lite interface
 - Output of disparity map starts before receiving the last pixel of both input images
 - Support for variable image sizes
 - Multi-clock design with faster clock for performance critical tasks
- Image rectification
 - Rectification using a pre-computed compressed rectification map
 - Bi-linear interpolation for subpixel accurate rectification
 - Pixel cache and DMA engine for efficient image data access
- Stereo matching
 - Stereo matching through a variation of the Semi-Global Matching (SGM) algorithm
 - Configurable disparity range up to 256 pixels
 - Configurable disparity offset
 - Configurable penalties P_1 and P_2 for small and large disparity variations

- Edge-dependent variation of penalties
- Pre-processing of input images for improved robustness against illumination variations and occlusions
- Post-processing
 - Subpixel optimization
 - Consistency check with configurable threshold
 - Uniqueness check with configurable threshold
 - Filling of small gaps through interpolation
 - Noise reduction
 - Speckle filtering
 - Filtering of untextured image areas

3 Background

3.1 Camera Alignment

For stereo vision, both cameras must be mounted on a plane with a displacement that is perpendicular to the cameras' optical axes. Furthermore, both cameras must be equipped with lenses that have an identical focal length. This arrangement is known as the *standard epipolar geometry*. An example for such a camera mounting is shown in Figure 1.

The distance between both cameras is referred to as *baseline distance*. Using a large baseline distance improves the depth resolution at high distances. A small baseline distances, on the other hand, allows for the observation of close objects. The baseline distance should be adjusted in conjunction with the lenses' focal length. An online tool for computing desirable combinations of baseline distance and focal length can be found on the Nerian Vision GmbH website¹.

3.2 Image Rectification

Even when carefully aligning both cameras, you are unlikely to receive images that match the expected result form an ideal standard epipolar geometry. The images are affected by various distortions that result from errors in the cameras' optics and

¹<https://nerian.com/support/resources/calculator/>



Figure 1: Example for standard epipolar geometry.

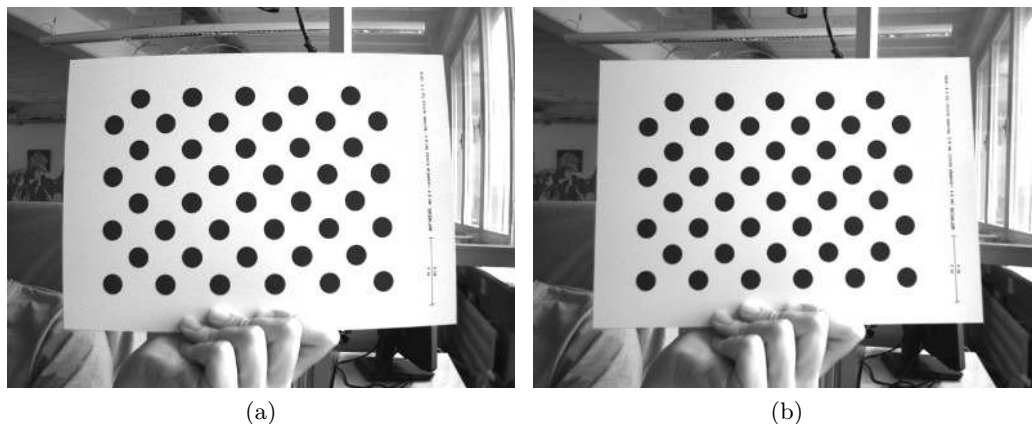


Figure 2: Example for (a) unrectified and (b) rectified camera image.

mounting. Therefore, the first processing step that needs to be performed is an image undistortion operation, which is known as *image rectification*.

Figure 2a shows an example camera image, where the camera was pointed towards a calibration board. The edges of the board appear slightly bent, due to radial distortions caused by the camera's optics. Figure 2b shows the same image after image rectification. This time, all edges of the calibration board are perfectly straight.

3.3 Camera Calibration

Image rectification requires precise knowledge of the cameras' projective parameters, which is obtained through *camera calibration*. This typically requires the recording of several sample images of a flat calibration board with a visible calibration pattern. From the observed projection of this pattern it is then possible to compute the calibration parameters. This process is not implemented by the SVC, but has to be performed in software. Source code for computing the calibration parameters from a set of camera images is available.

3.4 Disparity Maps

The stereo matching results are delivered by the SVC in the form of a *disparity map*, from the perspective of the reference camera. By default the left camera is the reference camera, but alternatively the right camera can be selected as reference camera during IP customization.

The disparity map associates each pixel in the reference camera image with a corresponding pixel in the other camera image (referred to as match camera). Because both images were previously rectified to match an ideal standard epipolar geometry, corresponding pixels should only differ in their horizontal coordinates. The disparity map thus only encodes a *horizontal coordinate difference*.

An example for a left camera image and the corresponding disparity map are shown in Figures 3a and 3b. Here the disparity map has been color coded, with blue hues reflecting small disparities, and red hues reflecting large disparities. As can be



Figure 3: Example for (a) left camera image and corresponding disparity map.

seen, the disparity is proportional to the inverse depth of the corresponding scene point.

The *disparity range* specifies the image region that is searched for finding pixel correspondences. In the example image, the color legend indicates that the disparity range reaches from 0 to 111 pixels. A large disparity range allows for very accurate measurements, but causes a high computational load and thus lowers the achievable frame rate. The SVC supports a configurable disparity range, which provides a choice between high precision or high speed.

It is possible to transform the disparity map into a set of 3D points. This can be done at a correct metric scale if the cameras have been calibrated properly. The transformation of a disparity map to a set of 3D points requires knowledge of the disparity-to-depth mapping matrix Q , which can be computed during camera calibration. The 3D location $\begin{pmatrix} x & y & z \end{pmatrix}^T$ of a point with image coordinates (u, v) and disparity d can be reconstructed as follows:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{1}{w} \cdot \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}, \text{ with } \begin{pmatrix} x' \\ y' \\ z' \\ w \end{pmatrix} = Q \cdot \begin{pmatrix} u \\ v \\ d \\ 1 \end{pmatrix}$$

An efficient implementation of this transformation is available in the API for the SceneScan stereo vision sensor.

The SVC computes disparity maps with a disparity resolution that is below one pixel. Disparity maps have a bit-depth of 12 bits, with the lower 4 bits of each value representing the fractional disparity component. It is thus necessary to divide each value in the disparity map by 16 in order to receive the correct disparity magnitude.

Several post-processing techniques are applied in order to improve the quality of the disparity maps. Some of these methods detect erroneous disparities and mark them as invalid. Invalid disparities are set to 0xFFFF, which corresponds to the decimal value 255.9375 and is the maximum value that can be stored in the 12-bit disparity map. In Figure 3b invalid disparities have been depicted as black.

4 Stereo Vision Core Functionality

The overall functionality of the SVC in its default configuration is depicted in Figure 4. Please note that the available ports differ when using the legacy rectification implementation. In this case, please refer to data sheets for IP core version 5.3 or earlier for further details.

The port `register_io`, which provides read and write access to all device registers, has been omitted in this figure for simplicity. This port complies to the AXI4-Lite standard (ARM, 2013) and acts as a communication slave.

The port `frame_complete` provides a binary signal which is asserted to 1 for one clock cycle, once processing of the current frame has been completed.

The remaining input and output ports all implement the AXI4-Stream protocol (ARM, 2010) and read/write image data, image rectification maps, image addresses or temporary buffer data. The purpose of each port and the involved processing is described in the subsequent sections.

Processing inside the SVC is divided into several sub-modules. Not all of these sub-modules are mandatory. Some can be deactivated through setting the appropriate device registers, or they can be removed from the IP core altogether if desired. A detailed description of each sub-module is provided below.

4.1 Image Input

During customization, a maximum image size must be specified that the SVC shall be able to handle. While the image height has only a marginal impact on the required FPGA resources, the image width will have a strong influence on the resource usage.

With the new image rectification introduced with version 6.0, the SVC will read the pixel data of an input image in the order that it is needed for image rectification. This means that image access is no longer linear. Therefore, the SVC provides a stream of *image addresses* through the `*_burst_to_dma` ports.

An image address is a memory address that is relative to the first byte of image pixel data. For each provided image address, the SVC expects a burst of data at the respective `*_burst_from_dma` port. The burst data must start from the given image address, with the burst length and size configured during IP customization (see Section 6.1). In a typical design the `*_burst_to_dma` and `*_burst_from_dma` ports are directly connected to the DMA core, which will handle all memory accesses. The SVC stores the image data in an internal cache memory, which significantly reduces the number of required burst transfers.

The SVC can be configured to use a pixel width of 12 or 8 bits for monochrome images, or 8 bits per channel for RGB images. When set to 12-bit monochrome, different encoding options can be chosen for the input image data. The encoding mode can be selected at runtime by writing to the device registers (see Section 11.2.1). All supported image encodings are described in the following.

4.1.1 8-Bit Monochrome

In this simple mode, an 8-bit monochrome encoding is assumed for the input data. No padding bytes are allowed between the pixel data.

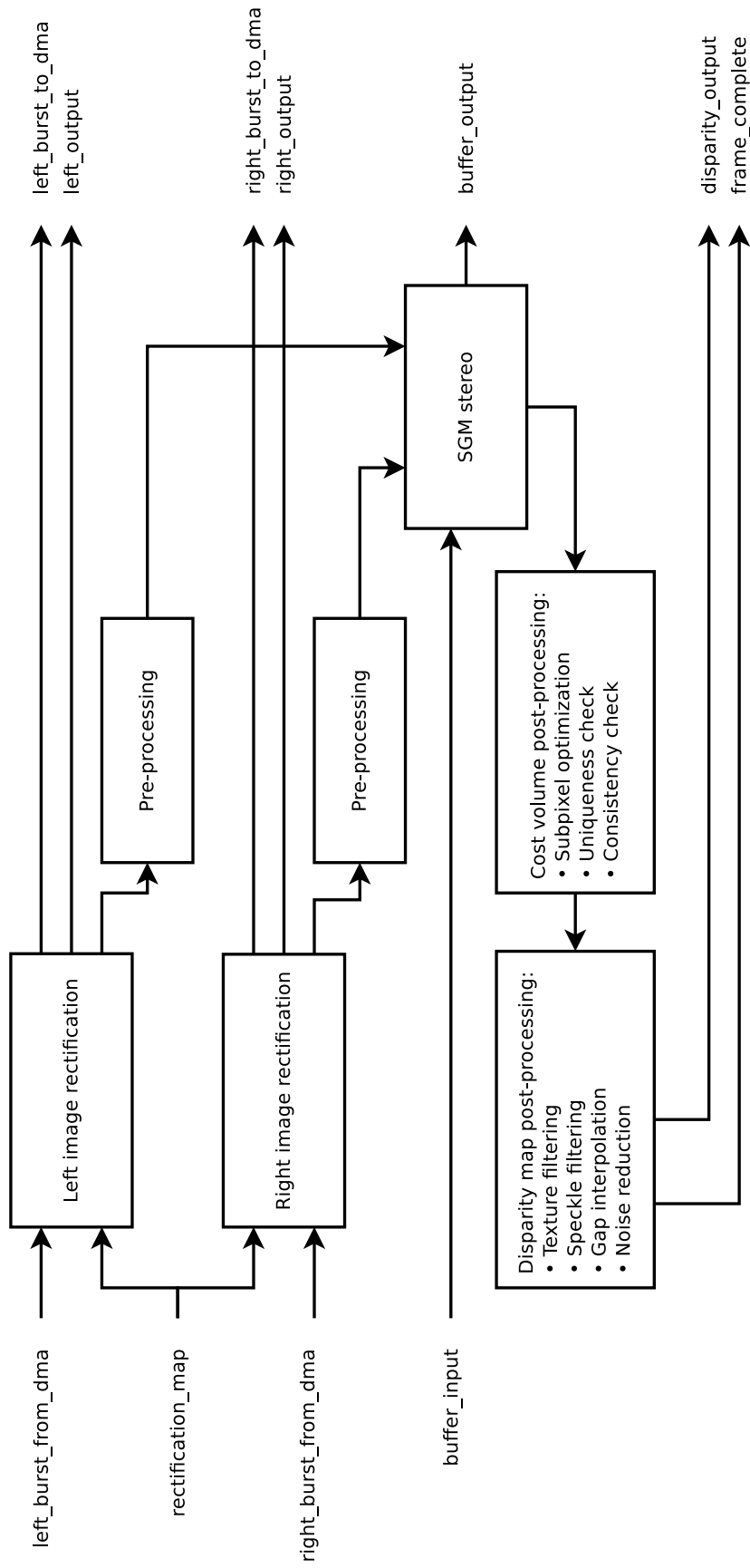
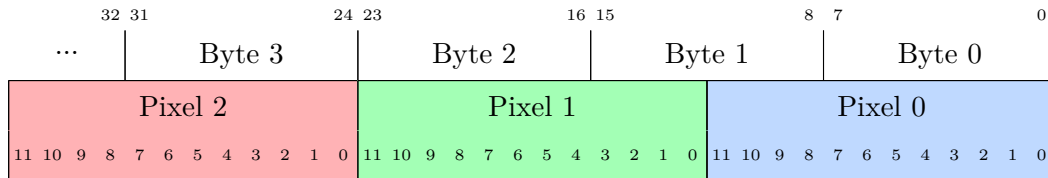


Figure 4: Block diagram of SVC functionality.

4.1.2 12-Bit Monochrome LSB Packed

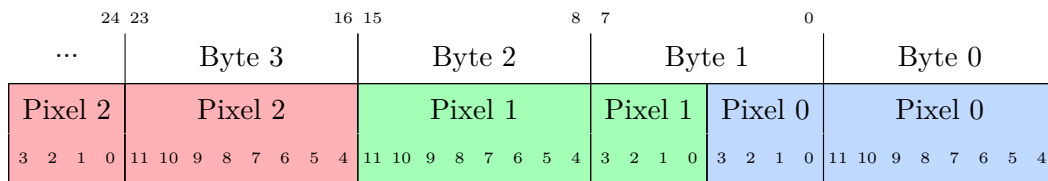
In this encoding mode, two 12-bit values are written to 3 bytes in memory. This happens in a least-significant-bit (LSB) alignment, without introducing additional padding bits. This means that data is filled LSB first in the lowest byte, and then continues to higher addresses, as depicted below:



This matches the Mono12p pixel format from the GenICam Pixel Format Naming Convention (European Machine Vision Association, 2016).

4.1.3 12-Bit Monochrome GEV Packed

Like the previous mode, the 12-bit GEV packing mode encodes two 12-bit values in 3 bytes. In this case, the upper 8 bits of the first pixel are written to the first byte, and the upper 8 bits of the second pixel are written to the third byte. A combination of the lower 4 bits of both pixels are written to the second byte in between. This encoding scheme is depicted below:



This scheme matches the Mono12Packed format from the GigE Vision standard (AIA, 2013). The advantage of this encoding is that it can be efficiently converted to 8-bits, by leaving out every third byte from the image data.

4.1.4 12-Bit Unpacked

In the unpacked 12-bit encoding mode, the image data is stored with 16 bits per pixel in memory. The SVC ignores the most significant 4 bits of each pixel when reading the image data.

4.1.5 8-Bit RGB

In the 8-bit RGB input mode the input image is a color image with 8 bits for each of the three color channels. Endianess does not matter for RGB images, which means that either the red or the blue color channel can be stored in the least or most significant bits. The green color channel, however, should be stored in the middle. The endianess of the output image will match the endianess of the input image. Either of the following encodings are hence possible:



Even though an RGB color image provides more information than a grayscale image with equal spatial resolution, a color image will not lead to better processing results. However, the color information will be preserved during image rectification, and the rectified color image can hence be used for onward processing. If color information is not required for onward processing, it is recommended to disable support for RGB images and directly provide monochrome image data, as this can save FPGA resources.

4.1.6 8-Bit Bayer Pattern

Prior to version 6.0, it was possible to directly process Bayer pattern images. Here the DMA core performed a conversion from Bayer pattern to RGB. With the new image rectification introduced in version 6.0, Bayer pattern conversion (‘demosai- ing’) is no longer possible on-the-fly, as image data is no longer read in a linear order. Hence, a conversion to RGB should be performed beforehand.

The older linear rectification is still available as an option for legacy projects. If you use the old rectification then Bayer pattern demosaicing is still supported. Please refer to the data sheet for version 5.3 for further details.

4.2 Image Output

Unlike the image inputs, all image outputs are provided as a linear stream of pixel data. If support for more than one pixel format is enabled, the data widths of all image output ports will be set to the pixel width of the largest pixel format. If a format with a shorter pixel width is used, then the unnecessary most significant bits will be set to zero.

4.3 Rectification

The SVC concurrently reads two input images from the `left_input` and `right_input` ports. The first processing step that is applied to the input data is image rectification. To perform image rectification, a pre-computed rectification map is required that is read from the dedicated input port `rectification_map`. The rectification map contains a subpixel accurate x- and y-offset for each pixel of the left and right input images. Bi-linear interpolation is applied to map the subpixel offsets to image intensities.

The offsets are interleaved such that reading from a single data stream is sufficient for finding the displacement vector for each pixel in both images. To save bandwidth, the rectification map is stored in a compressed form. On average one byte is required for encoding the displacement vector for a single pixel. Hence, the overall size of the

rectification map is equal to the size of two input images. Source code is provided for generating the rectification map from typical camera calibration parameters.

The left rectified image is always written to `left_output`, and the right rectified image to `right_output`, unless the IP's operation mode has been set to pass-through, in which case the input images are passed through without modification (see Section 11.2.1).

4.4 Image Pre-Processing

An image pre-processing method is applied to both input images. This causes the subsequent processing steps to be more robust towards illumination variations and occlusions.

4.5 Stereo Matching

Stereo matching is performed by applying a variation of the SGM algorithm by Hirschmüller (2005). SGM applies two penalties P_1 and P_2 for solutions with small and large disparity variations. In our case, we adapt both penalties according to image edges. For both, P_1 and P_2 , there exist two different penalty values, which are applied to edge and non-edge image pixels. These penalties can be configured at run-time through the SVC's registers (see Section 11.2.12).

The SVC can require several iterations for processing one pixel of the reference input image. In each iteration, the reference image pixel is compared to a group of pixels in the match image. The number of parallel pixel comparisons p can be configured through the SVC customization parameters (see Section 6.1). The number of iterations per reference image pixel n_i can be configured through the SVC control register (see Section 11.2.1).

A *disparity offset* o_d can also be configured through the SVC's registers, which indicates the smallest disparity value that will be considered during stereo matching. If $o_d \neq 0$ then the observable depth range will have an upper limit, as disparities smaller than o_d will not be allowed. The disparity offset o_d , iteration count n_i and the parallelization p determine the maximum disparity d_{max} :

$$d_{max} = o_d + n_i p - 1 \quad (1)$$

For storing intermediate processing results, the SGM sub-module requires write access to an external buffer through the port `buffer_output`. This buffer can be located in external memory, or if desired in the FPGA's block RAM. A lossy data compress is applied in order to reduce the required band-width and buffer size. The compression rate z can be configured during IP customization.

The total size s_b of the buffer can be computed as follows:

$$s_b = 3 \cdot (d_{max} - o_d + 1) \cdot w_{max} \cdot (1 - z) , \quad (2)$$

where d_{max} is the maximum disparity and w_{max} is the maximum supported image width.

Data is written linearly to the buffer, starting from byte offset 0 all the way through to the last byte in the buffer. Once the last byte has been written, the SVC sends out a rewind signal. Writing will then restart again at byte offset 0.

Similarly, the content of the same buffer is read back linearly through the port `buffer_input`, and reading restarts at byte offset 0 upon a corresponding rewind signal. It is ensured that reading and writing will never happen simultaneously on the same buffer data.

4.6 Cost Volume Post-Processing

The SGM stereo algorithm produces a *cost volume*, which encodes the matching costs for all valid combinations of left and right image pixels. Several of the applied post-processing techniques operate directly on this cost volume.

4.6.1 Subpixel Optimization

Subpixel optimization is the first applied post-processing technique. This step increases the accuracy of depth measurements by evaluating the matching costs to the left and right of the detected minimum for each pixel. A curve is fitted to the matching costs and its minimum is determined with subpixel accuracy.

The optimization algorithm is tuned automatically to provide the best possible subpixel measurements. If only a small region of interest (ROI) of the input image / disparity map is relevant, then this auto-tuning process can be constrained to only this ROI. In this case one should expect more accurate sub-pixel measurements inside the ROI. The ROI can be selected at run-time by writing to the SVC's registers (see Sections 11.2.13 and 11.2.14).

For the tuning process, the algorithm collects statistics over a period of time. In the default mode, statistics are collected over a single frame. The statistical information is then used to optimize the results of the next frame. Alternatively, a multi-frame mode can be selected through the SVC's registers (see Section 11.2.4), in which statistics are collected over more than one frame. In this mode, the SVC needs to be notified manually when the collection period should end.

The improved disparity estimates are encoded as fixed-point numbers. Currently the SVC supports 4 decimal bits for the subpixel optimized disparity. Hence, it is possible to measure disparities with a resolution of $1/16$ pixel. It is thus required to divide each disparity value by 16, when interpreting the final disparity map.

4.6.2 Uniqueness Check

Matches with a high matching uncertainty are discarded by imposing a uniqueness constraint. For a stereo match to be considered unique, the minimum matching cost c_{min} times a uniqueness factor $q \in [1, \infty)$ must be smaller than the cost for the next best match. This relation can be expressed in the following formula, where C is the set of matching costs for all valid pixel pairs and $c^* = c_{min}$ is the cost for the best match:

$$c^* \cdot q < \min \{C \setminus \{c_{min}\}\}. \quad (3)$$

Stereo matches that are discarded through the uniqueness check are assigned a disparity label of 0xFFFF.

4.6.3 Consistency Check

A consistency check is employed for removing further matches with high matching uncertainties. The common approach to this post-processing technique is to repeat stereo matching in the opposite matching direction, and then only retaining matches for which

$$|d_r - d_m| \leq t_c, \quad (4)$$

where d_r is the disparity from reference-to-match image matching, d_m the disparity from match-to-reference image matching, and t_c is the consistency check threshold.

In order to save FPGA resources, we refrain from re-running stereo matching a second time in the opposite matching direction. Rather, the match camera disparity map is inferred from the matching costs that have been gathered during the initial reference-to-match stereo matching. Pixels that do not pass the consistency check are again labeled with 0xFFFF.

4.7 Disparity Map Post-Processing

Following the cost volume post-processing, the cost volume is reduced to a disparity map (see Section 3.4). Additional post processing methods are then applied directly to the disparity values.

4.7.1 Texture Filtering

Matching image regions with little to no texture is particularly challenging. Especially if such regions occur close to image borders, this might lead to significant mismatches. In order to address this problem, a texture filter is applied. This filter computes a texture score s_t for each image pixel, which reflects the texture intensity within a local neighborhood. Pixels for which this score is below a configurable threshold t_t are again labeled with 0xFFFF in the computed disparity map.

4.7.2 Speckle Filtering

The aforementioned methods are not always able to identify and label all erroneous matches. Fortunately, the erroneous matches that remain tend to appear as small clusters of similar disparity. These *speckles* are then removed with a speckle filter. The speckle filter identifies connected components that are below a specified minimum size. The minimum speckle size is an IP core internal parameter that cannot be changed. However, it is possible to configure how many iterations of the speckle filter should be executed. A larger number of speckle filter iterations will result in larger speckles being removed. The pixels that belong to identified speckles are again labeled with 0xFFFF.

4.7.3 Gap Interpolation

The aforementioned post-processing techniques all remove pixels from the computed disparity map, which leaves gaps with no valid disparity data. If one such gap is small, it can be filled with valid disparities by interpolating the disparities from its

edges. Interpolation is only performed for gaps whose vertical and horizontal extent l_h and l_v fulfill the condition

$$\min \{l_h, l_v\} \leq l_{max}, \quad (5)$$

where l_{max} is the maximum gap width. Interpolation is also omitted if the disparities from the edge of the identified gap do not have a similar magnitude.

4.7.4 Noise Reduction

Finally, a noise reduction filter is applied to the generated disparity map. This filter performs a smoothing of the disparity map, while being aware of discontinuities and invalid disparities. If the operation mode is set to stereo matching (see Section 11.2.1), then the disparity map that results after this filter is directly written to the `disparity_output` port.

5 DMA Core Functionality

When using the SVC directly it is in the responsibility of the developer to provide all required data on the input ports and to collect the data from the output ports in time. In a typical setting, the input data is read from off-chip memory and the processing results are written back to memory. For systems with a shared system memory, such as the Zynq SoC, we provide a DMA core for fetching and writing data. The functionality of this core is depicted in the block diagram of Figure 5.

5.1 Ports Connected to SVC

Except for the clocks and `frame_complete` signals, the DMA core connects to all input and output ports of the SVC. In Figure 5, those ports are depicted on the right. The ports match the ones shown in Figure 4 on page 9, plus one further output and two inputs that were omitted previously for simplicity.

The new output is `system_resetn`, which is an active low reset signal. The reset signal is set to 0 if either the DMA core is reset itself, or if a soft reset is triggered through writing to the reset-bit in register 0x00. It is recommended that the SVC's reset input is connected to this output. Otherwise, the SVC will not be affected by a soft reset of the DMA core, which can lead to erroneous behavior.

The new SVC-specific input ports are `buffer_input_rewind` and `buffer_output_rewind`. These binary signals are asserted by the SVC when reading from or writing to the buffer memory shall restart from the beginning. It is important that reading does not start before this signal is asserted, as the relevant data might not yet have been written.

5.2 Interface Ports

All ports that are not connected to the SVC appear on the left-hand side of Figure 5. The port `register_io` provides read and write access to all device registers of the DMA core. This port complies to the AXI4-Lite standard (ARM, 2013) and acts as a communication slave.

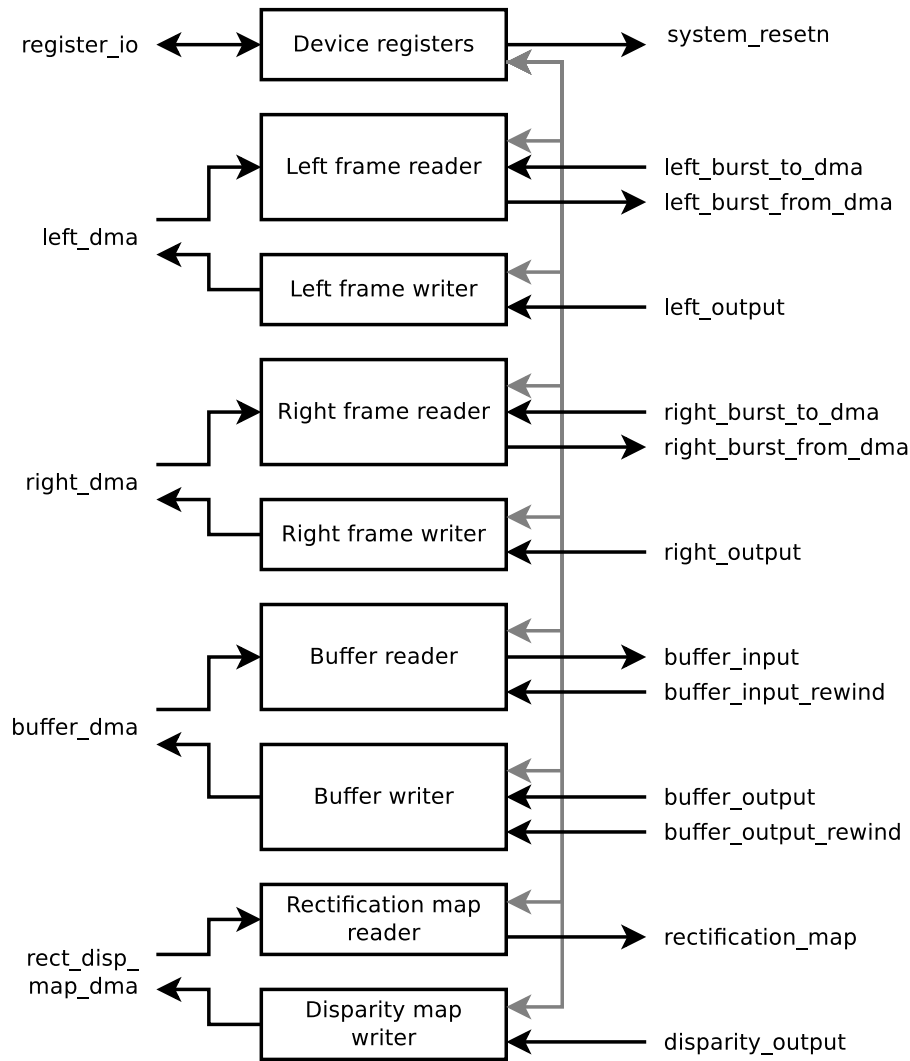


Figure 5: Block diagram of DMA core functionality.

The remaining ports follow the AXI3 or AXI4 standard (ARM, 2013) and act as communication masters. The `left_dma` port fetches the left input image and delivers the rectified left output image. Similarly, the `right_dma` port fetches the right input image and delivers the right rectified output image. The port `buffer_dma` serves for reading from and writing to the buffer memory. The `rect_disp_map_dma` port is used for fetching the rectification map and for delivering the output disparity map.

All fetch and write operations of the AXI3/AXI4 ports are controlled through the device registers. They contain the input and output memory addresses and can trigger read or write operations when set to a new value. More details on the device registers can be found in Section 11 on page 31.

5.3 Output Format Conversion

Before writing the output data to memory, the DMA core can apply different format conversions for 12-bit image data. As the disparity map typically has a 12-bit data

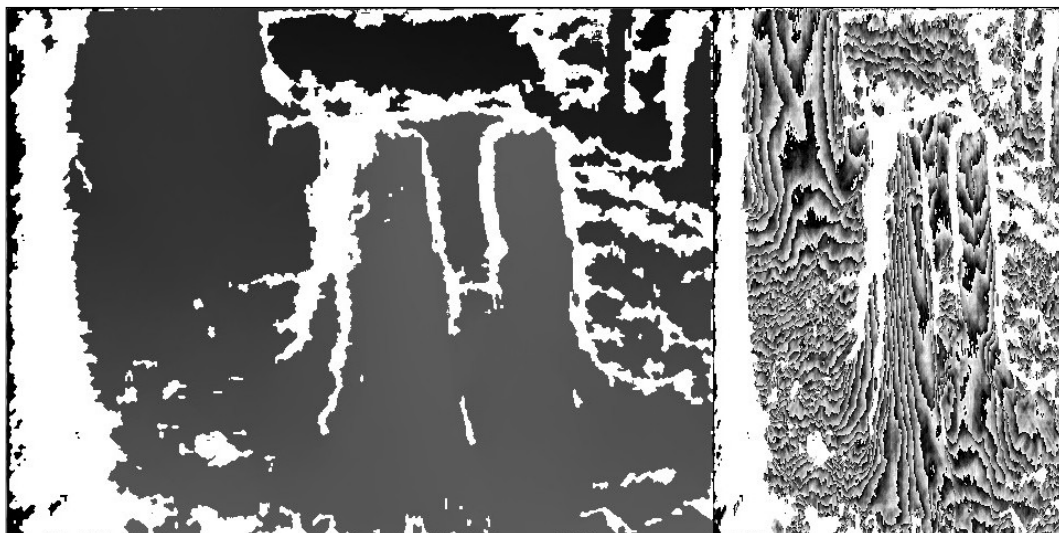


Figure 6: Example for disparity map with with 12-bit split encoding.

width, it is affected by the selected conversion option.

5.3.1 12-Bit Packed Output

The 12-bit packed mode is the simplest conversion mode. In this case two 12-bit values are written to 3 bytes in memory. This happens without introducing additional padding bits. This encoding matches the 12-bit LSB packed encoding from Section 4.1.2.

5.3.2 12-Bit Split Output

In this mode all 12-bit outputs are split into one 8-bit most-significant and one 4-bit least significant component. For the disparity map, the most-significant component matches the integer disparity, and the least-significant component matches the disparity decimal bits (see Section 4.6.1). These two components are combined into two new images, which are output row-wise interleaved, starting with the most significant component map.

It has to be considered that an element of the least-significant component map only has a size of 4-bits. Hence, two consecutive values are combined into a single byte. For this operation the first 4-bit element is written to the less-significant 4 bits, and the second element is written to the more-significant 4 bits of the 8-bit output value.

An example for the output of a 12-bit disparity map with the 12-bit split format conversion is shown in Figure 6. In this case, the merged output data can be interpreted as a row-wise sampled 8-bit image with dimensions $1.5w \times h$, where w and h are the width and height of the disparity map image.

5.3.3 16-Bit Output

If the 16-bit output is selected, all 12-bit values are inflated to 16 bits and written to two bytes in memory. This happens by introducing additional high-significant bits, which are set to 0. This matches the 12-bit unpacked encoding from Section 4.1.4.

6 Customization

Both, the SVC and the DMA core can be customized through several parameters. In Vivado's IP Integrator, these parameters can be set through the customization GUI. Screenshots of the customization windows for the SVC and DMA core are provided in Figures 7 and 8. Further parameters might be available for modification upon request. Please contact us if you have any special requirements.

6.1 SVC Customization Parameters

The SVC provides the customization parameters listed below. For some parameters a recommended value is provided, which we advise and use in our own products. All performance indicators that are provided in this document have been obtained with the recommended parameterization.

6.1.1 General Settings

FPGA Family: Needs to be set to UltraScale+ or 7-series, depending on for which FPGA the IP core should be synthesized.

Separate DNA clock: If enabled, the separate input clock `dna_clk` will be used for accessing the DNA port. This option is necessary if the base clock frequency does not match the clock frequency of the DNA port (typically 100 MHz). This clock must not be connected to the same source as `base_clk` or `fast_clk`, as a false path constraint between these clocks is automatically generated.

AXI Lite address width: Width in bits of an address for the `register_io` port. This should be 32 for Zynq-7000 devices and 40 for Zynq UltraScale+ devices.

Set registers to defaults on reset: If enabled, the internal configuration registers will be reset to their default values when the active-low reset signal is asserted to 0.

External processing buffer compression: A lossy data compression that is applied to the external buffer (see Section 4.5). Recommended values are 0% to 37.5%.

UltraRAM usage: The SVC is able to use UltraRAM instead of block RAM for some internal buffers. If your targeted device contains UltraRAM, then you can move some of the buffers to UltraRAM by setting this parameter to a value greater than 0.

Use new rectification implementation: Version 6.0 introduced a major re-design of the image rectification (see Section 4.1). For legacy systems it is still possible to use the older implementation by disabling this option. In this case

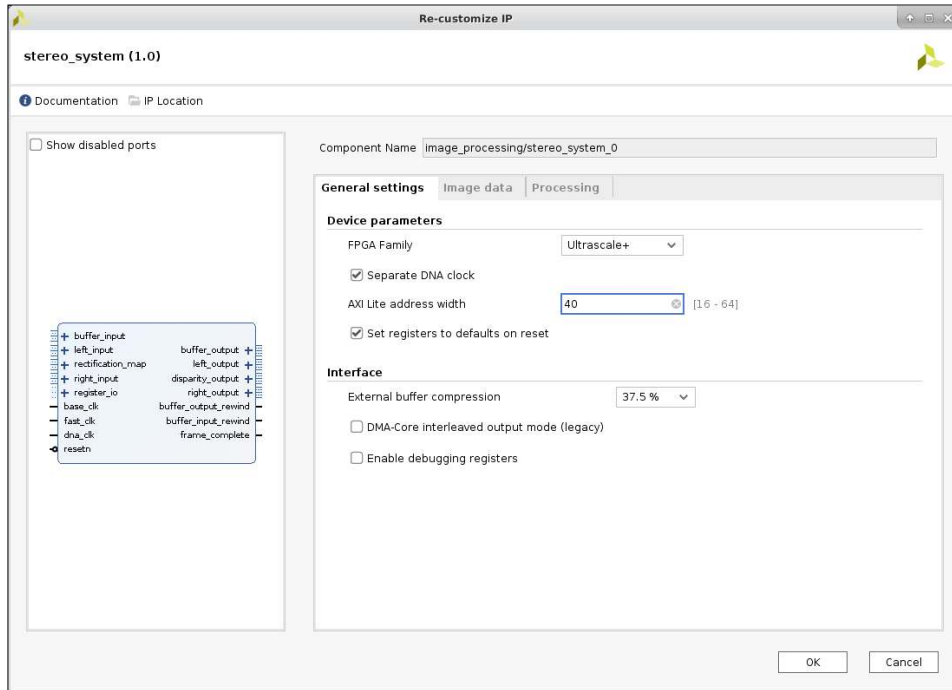


Figure 7: Customization parameters of SVC.

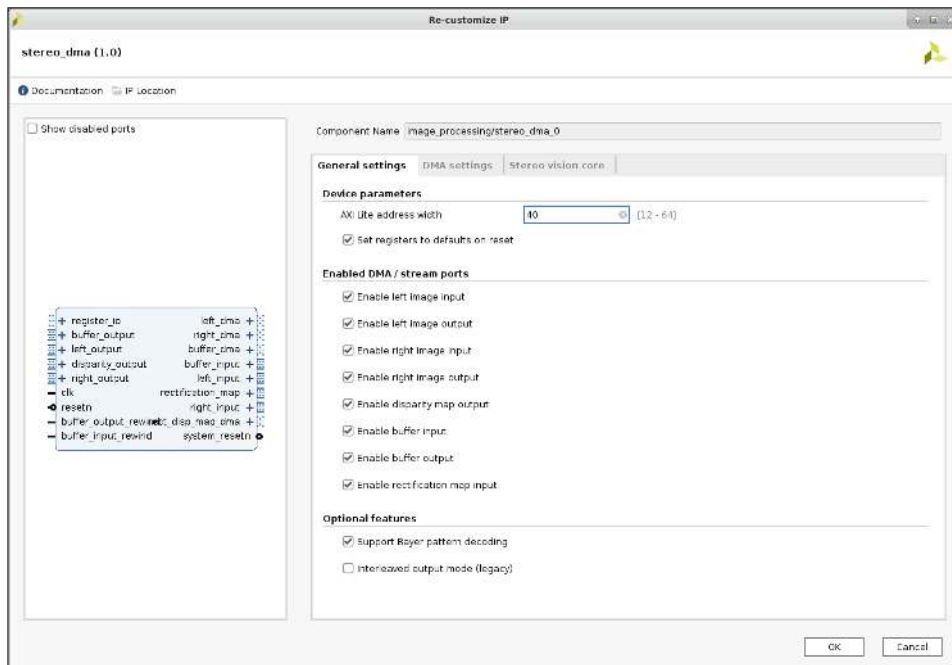


Figure 8: Customization parameters of DMA core.

please refer to the data sheet for version 5.3 or earlier for details about the rectification process.

Enable debugging registers: If enable, the SVC will hold additional registers with debugging information at the end of the register address space (see Section 11.2.18). These registers are intended for analysis by Nerian’s technical support team. Enabling this option will cause a slight increase in resource usage.

6.1.2 Image Data

Support 8-bit monochrome processing: If enabled, the IP is able to process input images in 8-bit monochrome encoding.

Support 12-bit monochrome processing: If enabled, the IP is able to process input images in 12-bit monochrome encoding

Support 8-bit RGB processing: If enabled, the IP is able to process input images in RGB encoding with 8 bits per color channel (24 bits in total).

Disparity width: The bit width of the output disparity map. This value should be large enough to allow for an output of the maximum disparity for the current parameterization. Otherwise, the least significant bits will be dropped. Please keep in mind that the disparities contain a 4-bit subpixel component (see Section 3.4). Recommended value: 12.

Pixel width: The bit width of one input pixel. This parameter inferred automatically from the enabled pixel formats.

Maximum image width: The maximum supported image width. This parameter has a significant impact on the block RAM usage. The actual image width is configured through the SVC registers and can be smaller (see Section 11.2.2). Recommended value for Zynq UltraScale+ ZU3: 1024.

Maximum image height: Maximum allowed height of an input image. The actual image height is configured through the SVC registers at runtime (see Section 11.2.2). This parameter has only a small impact on the resource usage. The value must be a multiple of the internal buffer size.

Determine maximum row stride automatically: If enabled, the maximum image row stride will be determined automatically from the configured maximum width and pixel format.

Maximum image row stride: If the option ‘*determine maximum row stride automatically*’ is not enabled, then the maximum row stride can be specified here. This option can be used to save resources if the automatically determined maximum stride would be larger than required.

6.1.3 Processing

Maximum rectification displacement: This option is only relevant if the legacy rectification method is used (see Section 6.1.1). The given value specifies the maximum offset that a pixel can be moved in vertical or horizontal direction during image rectification. This parameter has a significant impact on the block RAM usage of legacy systems. If a value of 0 is provided, then image rectification is disabled. Please note that the provided rectification map must be computed with respect to this parameter.

Number of pixels processed in parallel: The number of pixels p that are compared in parallel during SGM stereo matching. Together with the number of iterations, this parameter defines the disparity range (see Section 4.5). This parameter has a significant impact on the LUT usage. Recommended value for Zynq UltraScale+ ZU3: 32.

Maximum number of iterations per pixel: SGM stereo matching requires several iterations for processing a single pixel of the left input image (see Section 4.5). This parameter defines the maximum number of iterations that are allowed to be performed. Together the number of iterations and number of pixels processed in parallel define the disparity range. A lower number of iterations can be configured at runtime by writing to the SVC's registers (see Section 11.2.1). Recommended value for Zynq UltraScale+ ZU3: 8.

Optimize for low iteration count: If the IP core is configured for only few iterations per pixel, then the image processing will not be very efficient. In this case, additional buffers will be needed in order to guarantee a non-interrupted processing. These buffers are instantiated if this option is enabled. Please be aware that enabling this option will increase the block RAM usage. It is recommended to enable this option if the iteration count is less than 4, but even for higher iteration counts enabling this option can significantly improve the performance.

Maximum speckle filter iterations: In order to remove more speckles, the IP core can be configured to run multiple iterations of the speckle filter (see 4.7.2). The maximum number of iterations must be configured during customization. Increasing the maximum number of iterations will cause a moderate increase of the required FPGA resources.

Internal processing buffer size: Size of an internal buffer that is used by SGM stereo matching for caching computation results. This parameter has a high impact on the LUT and block RAM usage. Increasing this parameter significantly reduces the bandwidth that is required for reading from / writing to the external buffer. Recommended value for 32 parallel pixels processing: 16. Higher values are required for higher parallelization.

Matching direction: Specifies the direction in which stereo matching shall be performed. If 'left to right' is selected, then the left image will be the reference image and the right image will be the match image. If 'right to left' is selected, then the roles will be reversed. Default value: 'left to right'.

Generate repeatable results: If enabled, the IP core will attempt to generate repeatable processing results. This means that processing the same input image pair should produce the same disparity map. If this option is disabled, then results can vary with each processing iteration. Enabling this option will cause a minor increase in the resource usage. Even with this option enabled, minor difference in the output data can occur.

Rectification cache size: The image rectification requires an internal cache for buffering pixel data. The size of this cache can be specified through this parameter, and has a high impact on block RAM usage. A cache that is too small will lead to a reduced processing performance, which can be examined through the cache statistics register (see Section 11.2.17). Recommended value: 64.

6.1.4 DMA Core Settings

The parameters on the DMA core settings page must reflect the configuration of the DMA core to which the SVC shall be connected. If the settings don't match then interfacing won't be possible. The DMA core settings are described in the next section.

6.2 DMA Core Customization Parameters

Most of the SVC customization parameters are also available in the DMA core, on the *stereo vision core* settings page. It is important that these common parameters are configured identically in both cores. On top of the SVC parameters, the DMA core also provides its own customization parameters.

6.2.1 General Settings

AXI Lite address width: Width in bits of an address for the `register_io` port. This should be 32 for Zynq-7000 devices and 40 for Zynq UltraScale+ devices.

Set registers to defaults on reset: If enabled, the internal configuration registers will be reset to their default values when the active-low reset signal is asserted to 0.

Enable ... input/output: The AXI Stream input and output ports can be enabled or disabled individually. If a port is not required, then FPGA resources can be saved by disabling it.

Support Bayer pattern decoding: This option is only available if the legacy rectification implementation is used. If enabled, the DMA core can perform a Bayer pattern demosaicing through subsampling, while reading an input image.

Enable debugging registers: If enable, the DMA core will hold additional registers with debugging information at the end of the register address space (see Section 11.1.30). These registers are intended for analysis by Norian's technical support team. Enabling this option will cause a slight increase in resource usage.

6.2.2 DMA Settings

Buffer DMA AXI version: AXI protocol version for the `buffer_dma` port. This should be AXI3 for Zynq-7000 devices and AXI4 for Zynq UltraScale+ devices.

Buffer DMA address width: Width in bits of an address for the `buffer_dma` port. This should be 32 for Zynq-7000 devices and 49 for Zynq UltraScale+ devices.

Buffer DMA burst size: Width in bits of a data word for the `buffer_dma` port. This interface will be subject to high bandwidth data transfers. It should thus be configured to the highest data width that is supported by the external memory. This should be 64 for Zynq-7000 devices and 128 for Zynq UltraScale+ devices.

Buffer DMA burst length: Number of data transfers in one burst for the `buffer_dma` port. Recommended value for Zynq UltraScale+ ZU3: 32.

Buffer maximum concurrent write bursts: Maximum allowed number of outstanding write bursts for the `buffer_dma` port. Recommended value: 4.

Buffer maximum concurrent read bursts: Maximum allowed number of outstanding read bursts for the `buffer_dma` port. Recommended value: 4.

Buffer write FIFO size: Size of the FIFO buffer that is attached to the write channel of the `buffer_dma` port, measured in data words. Recommended value for Zynq UltraScale+ ZU3: 8.

Buffer read FIFO size: Size of the FIFO buffer that is attached to the read channel of the `buffer_dma` port, measured in data words. Recommended value for Zynq UltraScale+ ZU3: 8.

Other DMA AXI version: AXI protocol version for all ports other than `buffer_dma`. This should be AXI3 for Zynq-7000 devices and AXI4 for Zynq UltraScale+ devices.

Other DMA address width: Width in bits of an address for all ports other than `buffer_dma`. This should be 32 for Zynq-7000 devices and 49 for Zynq UltraScale+ devices.

Other DMA burst size: Width in bits of a data word for all ports other than `buffer_dma`. This should be 64 for Zynq-7000 devices and 128 for Zynq UltraScale+ devices.

Other DMA burst length: Number of data transfers in one burst for all ports other than `buffer_dma`. Recommended value for Zynq UltraScale+ ZU3: 32.

Other DMA maximum concurrent read bursts: Maximum allowed number of outstanding read bursts for all ports other than `buffer_dma`. Recommended value: 4.

12-bit encoding mode: The DMA core provides several ways for encoding 12-bit output data, such as the disparity map. Please refer to Section 5.3 for a description of the available encoding modes.

ARCACHE value / AWCACHE value: Cacheable bits that will be forwarded to AXI read / write transactions. Setting this parameter is important if cache coherent memory access should be used. Otherwise the default value "0011" is recommended.

ARPROT value / AWPROT value: Protection type bits that will be forwarded to AXI read/write transactions. Setting this parameter is important if cache coherent memory access should be used. Otherwise the default value "000" is recommended.

7 Supported Devices

The SVC has been field-tested on the Zynq-7000 and Zynq UltraScale+ SoCs. We thus recommend using these FPGA families. However, our IP core is also compatible to other Xilinx 7-Series and UltraScale+ devices. Please contact us to find out if your desired device is supported.

8 Timing

The SVC has been implemented as a multi-clock design. All input and output signals are associated with the *base clock*. When targeting the Zynq-7000 the recommended frequency for this clock is 100 MHz, and for a Zynq UltraScale+ SoC with speed grade 1 it is 150 MHz. In addition to the base clock, the SVC uses the so-called *fast clock* for clocking particularly performance critical tasks. The recommended frequency for this clock is 143 MHz for a Zynq 7000 SoC, and 300 MHz for a Zynq UltraScale+ SoC. Lower clock settings might be necessary when using a high degree of parallelization. The IP core will add a false path constraint between both clock domains.

If the DMA core is employed, its clock input has to be connected to the base clock. The recommended clock speed of the base clock matches the clock speed for the Zynq-7000 and UltraScale+ AXI memory interfaces. Hence, the DMA core can be directly connected to the Zynq's AXI slave ports.

The expected SVC processing delays when processing an input image of resolution 640×480 pixels with the recommended parameterization for the Zynq UltraScale+ ZU3 (32 times parallelization; see Section 6) and 4 iterations per pixel (i.e. 128 disparity levels) are listed in Table 1. The delays are given for a 150 MHz base clock and a 300 MHz fast clock, and are measured from the moment at which the first data item arrives at the SVC. As first output we consider the first byte of the computed disparity map. Consequently, the last output is the last byte of the disparity map, after which processing is complete.

The measurements were determined in simulation under the assumption that there is no significant memory delay. In a real-world system higher delays are expected, depending on the timing and throughput capabilities of the memory bus.

Table 1: SVC processing delays for default parameterization when processing input images of size 640×480 pixels and 128 disparity levels.

Delay	Fast clock cycles	Time
Delay until first output	136,000	0.45 ms
Delay until last output	1,890,000	6.30 ms

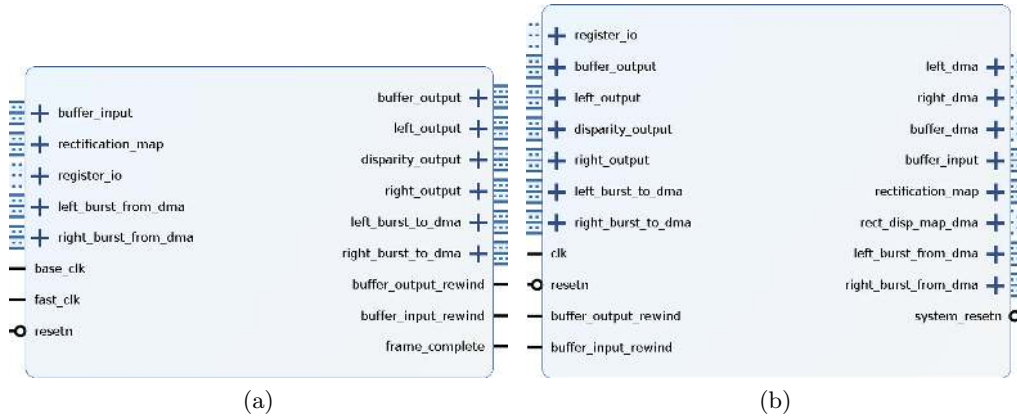


Figure 9: Interfaces of (a) SVC and (b) DMA core as shown by IP Integrator.

9 Resource Usage

Table 2 contains our recommended customization parameters when synthesizing the IP core for a Zynq UltraScale+ ZU3. The resource usage of the SVC and DMA Core when synthesizing with these parameters is shown in Table 3. The table further contains resource usage information for the individual sub-modules that have been identified in Section 4. These numbers provide an overview of the gains that can be achieved when removing one of the sub-modules from the SVC.

10 IO Signals

Figures 9a and 9b show a depiction of the SVC and DMA core as they appear in IP Integrator, which is part of Xilinx Vivado. Most of the shown ports have already been described in Sections 4 and 5. A detailed list of all input and output signals, including a breakdown of the AXI ports, is provided in Table 4 for the SVC. Likewise, Table 5 contains an equivalent list for the DMA core.

Table 2: SVC and DMA core customization parameters for resource usage evaluation.

Customization parameter	Value
External processing buffer compression	37.5%
Use new rectification implementation	yes
Enable debugging registers	no
Support 8-bit monochrome processing	yes
Support 12-bit monochrome processing	yes
Support 8-bit RGB processing	yes
Disparity width (bits)	12
Maximum image width	1024
Maximum image height	8192
Maximum image row stride	3072
Number of pixels processed in parallel	32
Maximum number of iterations per pixel	8
Optimize for low iteration count	no
Maximum speckle filter iterations	2
Internal processing buffer size	16
Matching direction	left to right
Generate repeatable results	no
Rectification cache size	no
Buffer maximum concurrent write bursts	4
Buffer maximum concurrent read bursts	4
Buffer write FIFO size	16
Buffer read FIFO size	16
Buffer DMA burst size	128
Buffer DMA burst length	16
Other DMA burst size	128
Other DMA burst length	16
Other DMA maximum concurrent read bursts	4
12-bit encoding mode	12-bit packed

Table 3: Resource usage of SVC and individual sub-modules.

Description	Slice LUTs	Registers	Memory	DSPs
SVC total usage	50,279	65,671	100.5	40
DMA core total usage	6,506	8,544	0.0	5
Image rectification	9,443	9,021	14.0	24
Image pre-processing	545	1,267	4.0	2
SGM stereo	23,666	39,148	60.5	0
Subpixel optimization	2,251	2,312	0.0	1
Uniqueness check	915	811	0.0	1
Consistency check	2,594	3,089	4.0	0
Texture filter	421	747	4.5	1
Speckle filter	7,630	5,310	7.0	2
Gap interpolation	1,029	1,326	4.0	6
Noise suppression	830	1,121	1.0	1
Others	955	1,519	1.5	2

Table 4: List of SVC input and output signals..

Signal name	i/o	Bits	Description
base_clk	i	1	Base clock source; this is the relevant clock for all input and output signals (see Section 8)
fast_clk	i	1	A faster clock for performance critical tasks (see Section 8)
dna_clk	i	1	Separate clock used for accessing the DNA port. This clock input is optional (see Section 6.1)
resetn	i	1	Global reset signal; active low
frame_complete	o	1	Signals that processing of the current frame has finished (see Section 4)

register_io signals

register_io_araddr	i	<i>variable</i>	Read address
register_io_arprot	i	3	Protection type; ignored!
register_io_arready	o	1	Read address ready; always 1!
register_io_arvalid	i	1	Read address valid
register_io_awaddr	i	<i>variable</i>	Write address
register_io_awprot	i	3	Protection type; ignored!
register_io_awready	o	1	Write address ready; always 1!
register_io_awvalid	i	1	Write address valid
register_io_bready	i	1	Response ready
register_io_bresp	o	2	Write response; always 00 ₂ (OK)!
register_io_bvalid	o	1	Write response valid
register_io_rdata	o	32	Read data

register_io_rready	i	1	Read ready
register_io_rresp	o	2	Read response; always 00 ₂ (OK)!
register_io_rvalid	o	1	Read valid
register_io_wdata	i	32	Write data
register_io_wready	o	1	Write ready
register_io_wstrb	i	4	Write strobes; ignored!
register_io_wvalid	i	1	Write valid

Signals for AXI4-Stream inputs

left_to_dma_ready	i	1	Ready to deliver a new DMA read address
left_to_dma_valid	o	1	DMA read address is valid
left_to_dma_address	o	<i>varied</i>	Left image address, from which a new burst shall be started.
left_from_dma_ready	o	1	Ready to receive read burst data
left_from_dma_valid	i	1	Read burst data is valid
left_from_dma_data	i	<i>varied</i>	One word of the current left read burst
right_to_dma_ready	i	1	Ready to deliver a new DMA read address
right_to_dma_valid	o	1	DMA read address is valid
right_to_dma_address	o	<i>varied</i>	Left image address, from which a new burst shall be started.
right_from_dma_ready	o	1	Ready to receive read burst data
right_from_dma_valid	i	1	Read burst data is valid
right_from_dma_data	i	<i>varied</i>	One word of the current left read burst
rectification_map_tready	o	1	Ready to receive rectification map
rectification_map_tvalid	i	1	Rectification map data is valid
rectification_map_tdata	i	32	Rectification map data
buffer_input_rewind	o	1	Reading from buffer shall restart from offset 0 (see Sections 4.5 and 5.1)
buffer_input_tready	o	1	Ready to receive buffer input
buffer_input_tvalid	i	1	Buffer input data is valid
buffer_input_tdata	i	<i>variable</i>	Buffer input data

Signals for AXI4-Stream outputs

left_output_tready	i	1	Ready to deliver left output
left_output_tvalid	o	1	Left output data is valid
left_output_tdata	o	<i>variable</i>	Left output data
right_output_tready	i	1	Ready to deliver left output
right_output_tvalid	o	1	Left output data is valid
right_output_tdata	o	<i>variable</i>	Left output data
disparity_output_tready	i	1	Ready to deliver disparity output
disparity_output_tvalid	o	1	Disparity output data is valid
disparity_output_tdata	o	<i>variable</i>	Disparity output data

buffer_output_rewind	o	1	Writing to buffer shall restart from offset 0 (see Sections 4.5 and 5.1)
buffer_output_tready	i	1	Ready to deliver buffer output
buffer_output_tvalid	o	1	Buffer output data is valid
buffer_output_tdata	o	<i>variable</i>	Buffer output data

Table 5: List of DMA core input and output signals.

Signal name	i/o	Bits	Description
clk	i	1	Main clock source; has to match the base clock from SVC
resetn	i	1	Global reset signal; active low
register_io signals			
register_io_araddr	i	<i>variable</i>	Read address
register_io_arprot	i	3	Protection type; ignored!
register_io_arready	o	1	Read address ready; always 1!
register_io_arvalid	i	1	Read address valid
register_io_awaddr	i	<i>variable</i>	Write address
register_io_awprot	i	3	Protection type; ignored!
register_io_awready	o	1	Write address ready; always 1!
register_io_awvalid	i	1	Write address valid
register_io_bready	i	1	Response ready
register_io_bresp	o	2	Write response; always 00 ₂ (OK)!
register_io_bvalid	o	1	Write response valid
register_io_rdata	o	32	Read data
register_io_rready	i	1	Read ready
register_io_rresp	o	2	Read response; always 00 ₂ (OK)!
register_io_rvalid	o	1	Read valid
register_io_wdata	i	32	Write data
register_io_wready	o	1	Write ready
register_io_wstrb	i	4	Write strobes; ignored!
register_io_wvalid	i	1	Write valid
buffer_dma / left_dma / right_dma / rect_disp_map_dma signals			
*_araddr	o	<i>variable</i>	Read address
*_arburst	o	2	Read burst type; always 01 ₂ (INCR)!
*_arcache	o	4	Read memory type; can be set in IP customization.
*_arlen	o	<i>variable</i>	Read burst length
*_arlock	o	2	Read lock type; always 00 ₂ (normal access)!
*_arprot	o	3	Read protection type; can be set in IP customization.
*_arqos	o	4	Read quality of service; always 0000 ₂ !
*_arready	i	1	Read ready
*_arsize	o	3	Read burst size; always 011 ₂ (8 bytes)!

*_arvalid	o	1	Read valid
*_awaddr	o	32	Write address
*_awburst	o	2	Write burst type; always 01 ₂ (INCR)!
*_awcache	o	4	Write memory type; can be set in IP customization.
*_awlen	o	<i>variable</i>	Write burst length
*_awlock	o	2	Write lock type; always 00 ₂ (normal access)!
*_awprot	o	3	Write protection type; can be set in IP customization.
*_awqos	o	4	Write quality of service; always 0000 ₂ !
*_awready	i	1	Write address ready
*_awsize	o	3	Write burst size; always 011 ₂ (8 bytes)!
*_awvalid	o	1	Write address valid
*_bready	o	1	Write response ready; always 1!
*_bresp	i	2	Write response
*_bvalid	i	1	Write response valid
*_rdata	i	<i>variable</i>	Read data
*_rlast	i	1	Read last
*_rready	o	1	Read ready
*_rresp	i	2	Read response
*_rvalid	i	1	Read last
*_wdata	o	<i>variable</i>	Write data
*_wlast	o	1	Write last
*_wready	i	1	Write ready
*_wstrb	o	8	Write strobes; always 1111111 ₂ !
*_wvalid	o	1	Write valid

rectification_map / buffer_input signals

*_ready	i	1	Ready to deliver input data
*_valid	o	1	Input data valid
*_data	o	<i>varied</i>	Data directed to SVC

left_output / right_output / disparity_output / buffer_output signals

*_ready	o	1	Ready to receive output data
*_valid	i	1	Output data is valid
*_data	i	<i>varied</i>	Data received from SVC

left_burst / right_burst signals

*_to_dma_ready	o	1	Ready to receive a new DMA read address
*_to_dma_valid	i	1	DMA read address is valid
*_to_dma_address	i	<i>varied</i>	Read image address. The DMA core will start a new read burst beginning at this address.
*_from_dma_ready	i	1	Ready to deliver read burst data
*_from_dma_valid	o	1	Read burst data is valid

*_from_dma_data	o	varied	One word of the current read burst
-----------------	---	--------	------------------------------------

Other signals directed to SVC

buffer_input_rewind	o	1	Reading from buffer shall restart from offset 0 (see Sections 4.5 and 5.1)
buffer_output_rewind	o	1	Writing to buffer shall restart from offset 0 (see Sections 4.5 and 5.1)
system_resetn	o	1	Active-low reset signal for SVC (see Section 5.1)

11 Registers

The SVC and DMA core each hold several registers that control the device behavior and provide information about the internal device state. Both IP cores have their own address spaces, starting at address 0x00. Please note that only the least significant address bits are evaluated and that reading from / writing to higher addresses will still affect the device registers.

A complete list of all available registers is shown in Table 6 for the DMA core, and in Table 7 for the SVC. All registers that have been marked with *r* are read-only. Writing to these registers will not produce an error but the new data is ignored.

Each register has a size of 32 bits. To simplify access from a CPU, the register addresses are always multiples of 4. Read and write operations must always be aligned to a 4-byte boundary. Reading from or writing to an address that is not a multiple of 4 is disallowed and has an undefined outcome. In the following, a description of all SVC and DMA core registers is provided, sorted by register address.

Table 6: Address space for DMA registers.

Address	Name	Read/ Write	Page
0x00	Control	r/w	33
0x04	Status	r	36
0x08	Image size	r/w	36
0x0C– 0x18	Obsolete	n/a	
0x1C	Left input address higher 32 bits	r/w	36
0x20	Left input address lower 32 bits	r/w	37
0x24	Left input bytes available	r/w	37
0x28	Right input address higher 32 bits	r/w	37
0x2C	Right input address lower 32 bits	r/w	37
0x30	Right input bytes available	r/w	37
0x34	Input FIFO info	r	37
0x38	Rectification map address higher 32 bits	r/w	38
0x3C	Rectification map address lower 32 bits	r/w	38
0x40	Rectification map FIFO info	r	38
0x44	Buffer address higher 32 bits	r/w	38
0x48	Buffer address lower 32 bits	r/w	38
0x4C	Buffer FIFO info	r	39
0x50	DMA errors	r	39
0x54	Reserved	n/a	-
0x58	IP version number	r	39
0x5C	Left output address higher 32 bits	r/w	39
0x60	Left output address lower 32 bits	r/w	39
0x64	Left output bytes available	r	39
0x68	Right output address higher 32 bits	r/w	40
0x6C	Right output address lower 32 bits	r/w	40
0x70	Right output bytes available	r	40
0x74	Disparity output address higher 32 bits	r/w	40
0x78	Disparity output address lower 32 bits	r/w	40
0x7C	Disparity output bytes available	r	40
0x80	Left / right output FIFO info	r	41
0x84	Disparity output FIFO info	r	41
0x88 – 0xC4	Debugging registers	r	41

Table 7: Address space for SVC registers.

Address	Name	Read/ Write	Page
0x00	Control	r/w	41
0x04	Image size	r/w	43
0x08	Algorithm parameters 1	r/w	43
0x0C	Algorithm parameters 2	r/w	43
0x10	License key higher 32 bits	r/w	44
0x14	License key middle 32 bits	r/w	44
0x18	License key lower 32 bits	r/w	44
0x1C	Device DNA higher 32 bits	r	44
0x20	Device DNA middle 32 bits	r	44
0x24	Device DNA lower 32 bits	r	45
0x28	SVC errors	r	45
0x2C	SGM penalties	r/w	45
0x30	Subpixel optimization ROI offset	r/w	45
0x34	Subpixel optimization ROI size	r/w	46
0x38	IP version number	r	46
0x3C	Multi-frame subpixel optimization	r	46
0x40	Statistics on rectification input caches	r	47
0x44 – 0x98	Debugging registers	r	47

11.1 DMA Core Registers

11.1.1 0x00: Control

General parameters that control the behavior of the SVC. Please note that whether a given input or output encoding mode is available depends on whether support for a particular pixel format has been enabled in the IP core customization (see Section 6.1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROE				LOE				DOE				<i>Reserved</i>				Iterations				RIE		LIE		R							

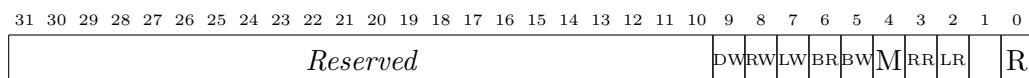
Name	Bits	Description
R	0	If set to 1 then the device performs a soft reset.
LIE	1 – 4	Specifies the encoding of the left input image. One of the following values must be selected, with 0000 being the default value:

		<p>0000 8-bit monochrome encoding (see Section 4.1.1).</p> <p>0001 12-bit monochrome LSB packed encoding (see Section 4.1.2).</p> <p>0010 12-bit monochrome GEV packed encoding (see Section 4.1.3).</p> <p>0011 12-bit monochrome unpacked encoding (see Section 4.1.4).</p> <p>0100 8-bit RGB encoding (see Section 4.1.5).</p> <p>1000 8-bit BG Bayer pattern that will be down-sampled (<i>only available with legacy rectification!</i>).</p> <p>1001 8-bit GB Bayer pattern that will be down-sampled (<i>only available with legacy rectification!</i>).</p> <p>1010 8-bit RG Bayer pattern that will be down-sampled (<i>only available with legacy rectification!</i>).</p> <p>1011 8-bit GR Bayer pattern that will be down-sampled (<i>only available with legacy rectification!</i>).</p>
RIE	5 – 7	<p>Encoding of the right input image. One of the following values must be selected, with 000 being the default value:</p> <p>000 Same as left input image.</p> <p>100 8-bit BG Bayer pattern that will be down-sampled (<i>only available with legacy rectification!</i>).</p> <p>101 8-bit GB Bayer pattern that will be down-sampled (<i>only available with legacy rectification!</i>).</p> <p>110 8-bit RG Bayer pattern that will be down-sampled (<i>only available with legacy rectification!</i>).</p> <p>111 8-bit GR Bayer pattern that will be down-sampled (see Section 4.1.6).</p>
Iterations	8 – 15	<p>Number of iterations per pixel (see Section 4.5). Must match the SVC configuration. Default value is the maximum number of iterations from the customization parameters (see Section 6.2). Minimum allowed value is 1.</p>
DOE	23 – 20	<p>Specifies the encoding of the output disparity map. The following values are possible, with 01 being the default value:</p>

		<p>00 The disparity map is encoded with 8 bits per pixel, of which 4 bits are fractional bits for the subpixel component. Hence the maximum disparity that can be represented is 15.</p> <p>01 The disparity map is encoded with 12 bits per pixel, of which 4 bits are fractional bits for the subpixel component. Hence the maximum disparity that can be represented is 255.</p>
LOE	24 – 27	<p>Specifies the encoding of the left output image. The following values are possible, with 00 being the default value:</p> <p>00 The output is encoded as an 8-bit monochrome image, even if the SVC/DMA core use a pixel width of 12 bits.</p> <p>01 The output is encoded as a 12-bit monochrome image. The encoding that is used depends on the 12-bit encoding mode that was selected during IP customization (see Section 6.2)</p> <p>10 The output is encoded as an 8-bit RGB image.</p>
ROE	28 – 31	<p>Specifies the encoding of the right output image. The following values are possible, with 01 being the default value:</p> <p>00 The output is encoded as an 8-bit monochrome image, even if the SVC/DMA core use a pixel width of 12 bits.</p> <p>01 The output is encoded as a 12-bit monochrome image. The encoding that is used depends on the 12-bit encoding mode that was selected during IP customization (see Section 6.2)</p> <p>10 The output is encoded as an 8-bit RGB image. This mode is only possible for pass-through or rectify mode of the SVC (see Section 11.2.1).</p>

11.1.2 0x04: Status

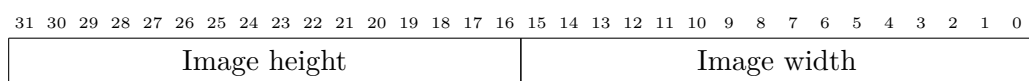
General device status information.



Name	Bits	Description
R	0	If 0 then the device is currently performing a soft or hard reset.
LR	2	If 1 then reading from <code>left_dma</code> has finished.
RR	3	If 1 then reading from <code>right_dma</code> has finished.
M	4	If 1 then reading from <code>rectification_map_dma</code> has finished.
BW	5	If 1 then writing to <code>buffer_dma</code> has finished.
BR	6	If 1 then reading from <code>buffer_dma</code> has finished.
LW	7	If 1 then writing to <code>left_dma</code> has finished.
RW	8	If 1 then writing to <code>right_dma</code> has finished.
DW	9	If 1 then writing to <code>disparity_dma</code> has finished.

11.1.3 0x08: Image Size

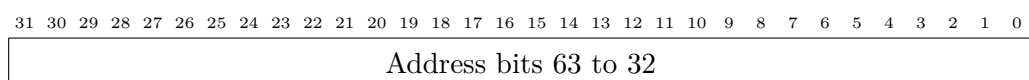
Dimensions of the left and right input images.



Name	Bits	Description
Image width	0 – 15	Width of an input image. This must match the parameterization of the SVC core. The image width must be a multiple of the number of pixels processed in parallel (see Section 6.1). If Bayer pattern images are used, this parameter must be set to the width of the downsampled RGB image. Default value: 0.
Image height	16 – 31	Height of an input image. This must match the parameterization of the SVC core. The image height must be a multiple of the internal processing buffer size (see Section 6.1). Default value: 0.

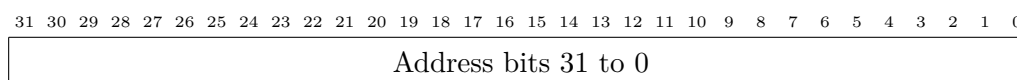
11.1.4 0x1C: Left Input Address Higher 32 Bits

Higher 32 bits of the left input image address. If the address width of the `left_dma` port is equal or less than 32 bits, then this register is ignored. This register must be written before the lower address register.

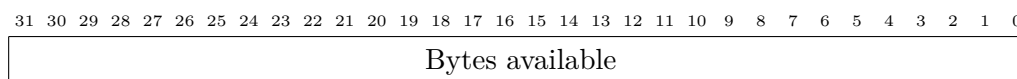


11.1.5 0x20: Left Input Address Lower 32 Bits

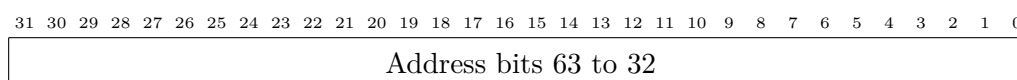
Lower 32 bits of the left input image address. This register must be written after the higher address register. The address must be aligned to the memory page size (typically 4 KB).

**11.1.6 0x24: Left Input Bytes Available**

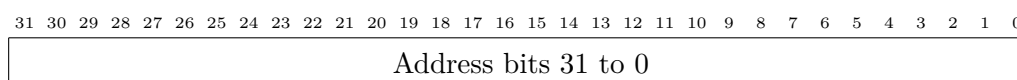
The number of bytes that can currently be read from `left_dma`, starting at the left input address. If this number is smaller than the frame size, then reading will pause once the specified number of bytes have been read. In this case reading will continue once a higher value is written to this register.

**11.1.7 0x28: Right Input Address Higher 32 Bits**

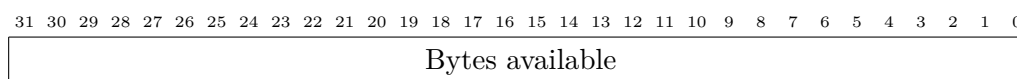
Higher 32 bits of the right input image address. If the address width of the `right_dma` port is equal or less than 32 bits, then this register is ignored. This register must be written before the lower address register.

**11.1.8 0x2C: Right Input Address Lower 32 Bits**

Lower 32 bits of the right input image address. This register must be written after the higher address register. The address must be aligned to the memory page size (typically 4 KB).

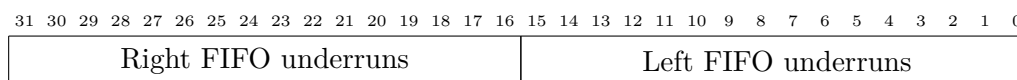
**11.1.9 0x30: Right Input Bytes Available**

The number of bytes that can currently be read from `right_dma`, starting at the right input address. If this number is smaller than the frame size, then reading will pause once the specified number of bytes have been read. In this case reading will continue once a higher value is written to this register.

**11.1.10 0x34: Input FIFO Info**

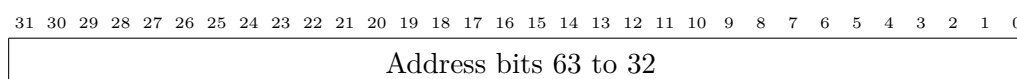
Statistics for the input FIFO buffers that are attached to `left_dma` and `right_dma`. Counters are reset with every new frame. This register is not available when using

the new rectification implementation.



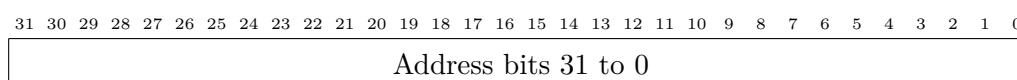
11.1.11 0x38: Rectification Map Address Higher 32 Bits

Higher 32 bits of the rectification map read address. If the address width of the `rectification_map_dma` port is equal or less than 32 bits, then this register is ignored. This register must be written before the lower address register.



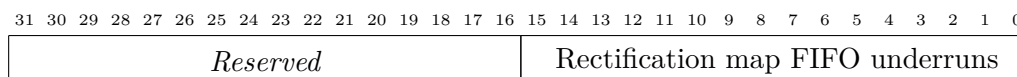
11.1.12 0x3C: Rectification Map Address Lower 32 Bits

Lower 32 bits of the rectification map read address. This register must be written after the higher address register. The address must be aligned to the memory page size (typically 4 KB). Reading from this address begins immediately after this register has been written. Reading continues until the full rectification map has been read from memory.



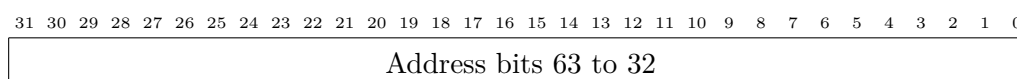
11.1.13 0x40: Rectification Map FIFO Info

Statistics for the rectification map FIFO buffer that is attached to `rectification_map_dma`. Counter is reset with every new frame.



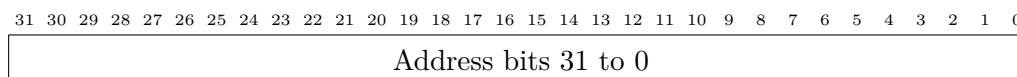
11.1.14 0x44: Buffer Address Higher 32 Bits

Higher 32 bits of the buffer address. If the address width of the `buffer_io` port is equal or less than 32 bits, then this register is ignored. This register must be written before the lower address register. The address is used for both, reading and writing data.



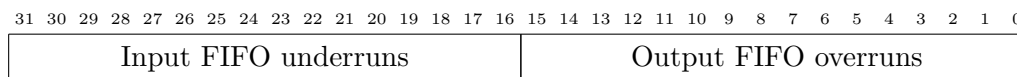
11.1.15 0x48: Buffer Address Lower 32 Bits

Lower 32 bits of the buffer address. This register must be written after the higher address register. The address is used for both, reading and writing data. The address must be aligned to the memory page size (typically 4 KB).

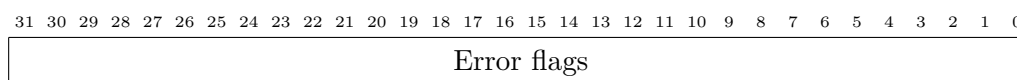


11.1.16 0x4C: Buffer FIFO Info

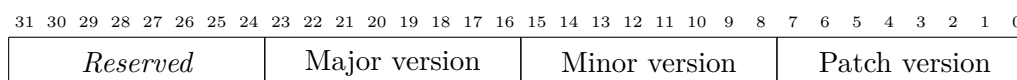
Statistics for the FIFO buffers that are attached to `buffer_dma`. Counters are reset with every new frame.

**11.1.17 0x50: DMA Errors**

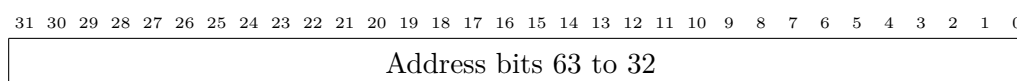
A collection of error flags that provide information on the internal error cause if the IP is malfunctioning. In normal operation, all bits in this register should have a value of 0. If one or more bits are set to 1 then this indicates an error and normal processing cannot continue.

**11.1.18 0x58: IP Version Number**

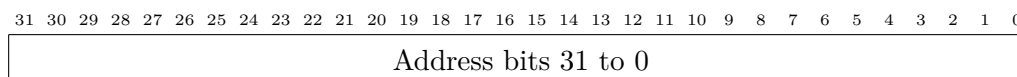
Version number of the IP, divided by major, minor and patch version.

**11.1.19 0x5C: Left Output Address Higher 32 Bits**

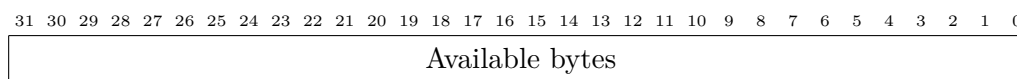
Higher 32 bits of the left output write address. If the address width of the `left_dma` port is equal or less than 32 bits, then this register is ignored. This register must be written before the lower address register.

**11.1.20 0x60: Left Output Address Lower 32 Bits**

Lower 32 bits of the left output write address. This register must be written after the higher address register. The address must be aligned to the memory page size (typically 4 KB). Writing to the destination address ends once one full frame has been written.

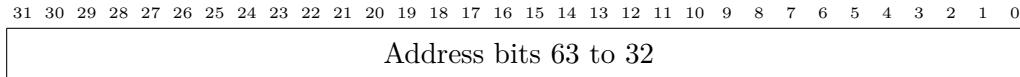
**11.1.21 0x64: Left Output Bytes Available**

The number of bytes that have successfully been written to the left output buffer since the start of the current frame.

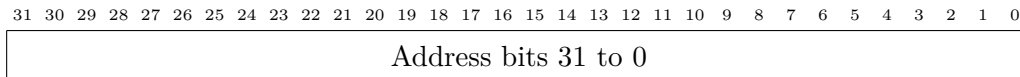


11.1.22 0x68: Right Output Address Higher 32 Bits

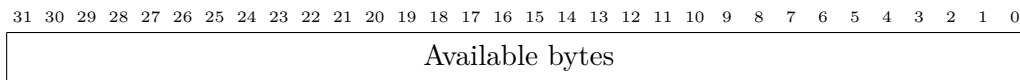
Higher 32 bits of the right output write address. If the address width of the `right_dma` port is equal or less than 32 bits, then this register is ignored. This register must be written before the lower address register.

**11.1.23 0x6C: Right Output Address Lower 32 Bits**

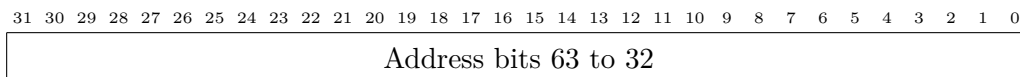
Lower 32 bits of the right output write address. This register must be written after the higher address register. The address must be aligned to the memory page size (typically 4 KB). Writing to the destination address ends once one full frame has been written.

**11.1.24 0x70: Right Output Bytes Available**

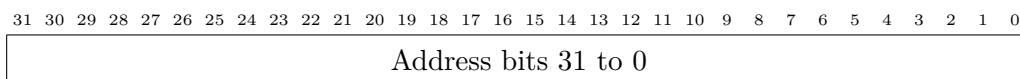
The number of bytes that have successfully been written to the right output buffer since the start of the current frame.

**11.1.25 0x74: Disparity Output Address Higher 32 Bits**

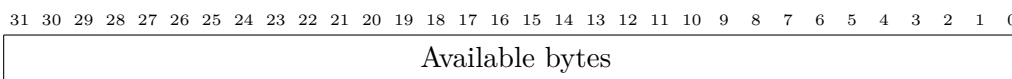
Higher 32 bits of the disparity output write address. If the address width of the `rect_disp_map_dma` port is equal or less than 32 bits, then this register is ignored. This register must be written before the lower address register.

**11.1.26 0x78: Disparity Output Address Lower 32 Bits**

Lower 32 bits of the disparity output write address. This register must be written after the higher address register. The address must be aligned to the memory page size (typically 4 KB). Writing to the destination address ends once one full frame has been written.

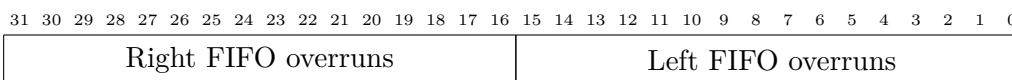
**11.1.27 0x7C: Disparity Output Bytes Available**

The number of bytes that have successfully been written to the disparity output buffer since the start of the current frame.



11.1.28 0x80: Left / Right Output FIFO Info

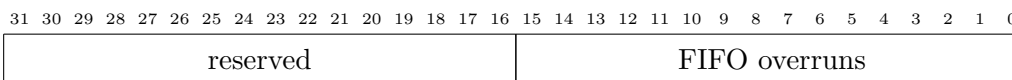
Statistics for the FIFO buffers that are used for writing the left and right output images. The counters are reset with every new frame.



Name	Bits	Description
Left FIFO overruns	0 – 15	Number of clock cycles for which image processing had to be paused because the left output image FIFO buffer was full.
Right FIFO overruns	16 – 31	Number of clock cycles for which image processing had to be paused because the right output image FIFO buffer was full.

11.1.29 0x84: Disparity output FIFO info

Statistics for the FIFO buffer that is used for writing the output disparity map. The counter is reset with every new frame.



Name	Bits	Description
FIFO overruns	0 – 15	Number of clock cycles for which image processing had to be paused because the disparity output FIFO buffer was full.

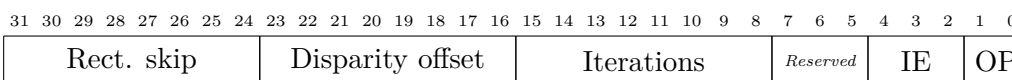
11.1.30 0x88 – 0xC4: Debugging Registers

If debugging registers have been enabled in the IP customization (see Section 6.2), then debugging information will be available in this register range. The contents of these registers is not documented and might differ between IP versions. These registers help Nerian's tech support in identifying the cause of potential problems in customer designs.

11.2 SVC Registers

11.2.1 0x00: Control

General parameters that control the behavior of the SVC.



Name	Bits	Description
OP	0 – 1	<p>Operation mode. Possible values are:</p> <p>00 Pass through. The SVC's left input is passed directly to the left output, and the right input is passed to the disparity output.</p> <p>01 Rectify. The rectification results are passed directly to the SVC's left and right output.</p> <p>10 Stereo matching (default). Stereo matching results are written to the SVC's disparity output, and the left and right rectified images are written to the left and right output.</p> <p>11 Reserved</p>
IE	2 – 4	<p>Specifies the encoding of the left and right input images. If the DMA core is used, then this parameter needs to be set together with the DMA core's IE parameter (see Section 11.1.1).</p> <p>000 8-bit monochrome encoding (see Section 4.1.1).</p> <p>001 12-bit monochrome LSB packed encoding (see Section 4.1.2).</p> <p>010 12-bit monochrome GEV packed encoding (see Section 4.1.3).</p> <p>011 12-bit monochrome unpacked encoding (see Section 4.1.4).</p> <p>100 8-bit RGB encoding (see Section 4.1.5).</p>
Iterations	8 – 15	<p>Number of iterations per pixel (see Section 4.5). Must match the DMA core configuration. Default value is the maximum number of iterations from the customization parameters (see Section 6.1). Minimum allowed value is 1.</p>
Disparity offset	16 – 23	<p>Offset for the disparity range in pixels (see Section 4.5). Default value: 0.</p>

Rect. skip	24 – 31	Number of pixels N that will be excluded from image rectification. If N is greater than 0, then the first N pixels will be passed through the image rectification without modification. This feature is useful if the image data contains embedded information at the beginning, such as timestamps or sequence numbers. Default value: 0.
------------	---------	--

11.2.2 0x04: Image Size

Dimensions of the left and right input images.

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
Image height	Image width

Name	Bits	Description
Image width	0 – 15	Width of an input image. This must match the parameterization of the DMA core. The image width must be a multiple of the number of pixels processed in parallel (see Section 6.1). Default value: 0.
Image height	16 – 31	Height of an input image. This must match the parameterization of the DMA core. The image height must be a multiple of the internal processing buffer size (see Section 6.1). Default value: 0.

11.2.3 0x08: Algorithm Parameters 1

Algorithmic parameters that can be changed at run-time.

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0			
Consist.	<i>Reserved</i>	Uniqueness factor	<i>Reserved</i>

Name	Bits	Description
Uniqueness Factor	16 – 24	Uniqueness factor q times 256. A value of 0 disables the uniqueness check (see Section 4.6.2). Default value: 320.
Consist.	28 – 31	Consistency check threshold t_c (see Section 4.6.3). Default value: 2.

11.2.4 0x0C: Algorithm Parameters 2

Further algorithmic parameters that can be changed at run-time.

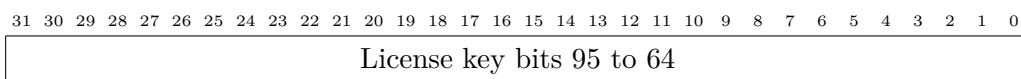
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0							
<i>Edge threshold</i>	<i>Reserved</i>	S. iter	Texture threshold	<i>Reserved</i>	N	G	C

Name	Bits	Description
C	0	If set to 1 then the consistency check is disabled (see Section 4.6.3). Default value: 0.

G	1	If set to 1 then the gap interpolation is disabled (see Section 4.7.3). Default value: 0.
N	2	If set to 1 then the noise reduction is disabled (see Section 4.7.4). Default value: 0.
Texture threshold	8 – 15	Threshold for the texture filter. A value of 0 disables the texture filter (see Section 4.7.1). Default value: 10.
S. iter	19 – 16	Number of iterations that the speckle filter should perform (see Section 4.7.2). Default value: maximum.
Edge threshold	31 – 24	Threshold that is used for edge detection, to make SGM adaptive to image edges (see Section 4.5). Small values will cause a high edge sensitivity and large values a low edge sensitivity. Default value: 23.

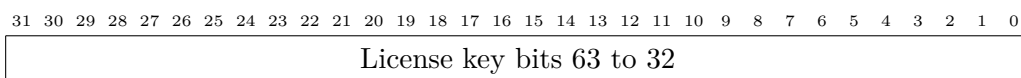
11.2.5 0x10: License Key Higher 32 Bits

The most-significant 32 bits of the device-specific license key.



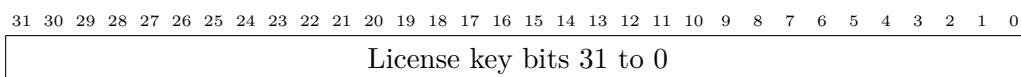
11.2.6 0x14: License Key Middle 32 Bits

The middle 32 bits of the device-specific license key.



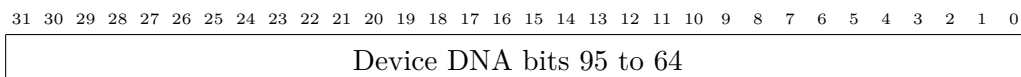
11.2.7 0x18: License Key Lower 32 Bits

The least-significant 32 bits of the device-specific license key.



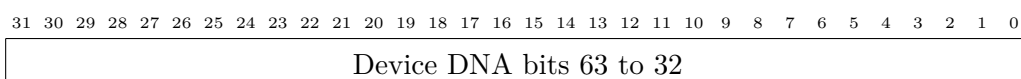
11.2.8 0x1C: Device DNA Higher 32 Bits

The most significant 32 bits of the 96-bit Xilinx device DNA. For 7-series devices, which have a 57-bit DNA, this register will always be 0.



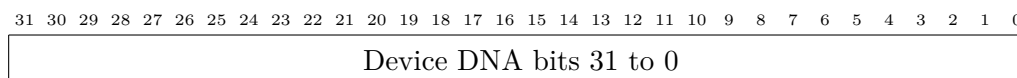
11.2.9 0x20: Device DNA Middle 32 Bits

The middle 32 bits of the 96-bit device DNA or most significant 25 bits of the 57-bit device DNA.

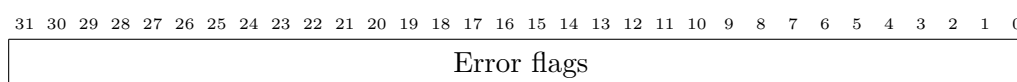


11.2.10 0x24: Device DNA Lower 32 Bits

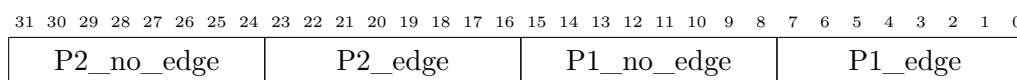
The least significant 32 bits of the 96-bit or 57-bit Xilinx device DNA.

**11.2.11 0x28: SVC Errors**

A collection of error flags that provide information on the internal error cause if the IP is malfunctioning. In normal operation, all bits in this register should have a value of 0. If one or more bits are set to 1 then this indicates an error and normal processing cannot continue.

**11.2.12 0x2C: SGM Penalties**

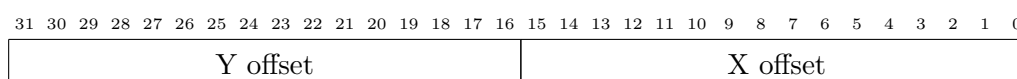
Penalty values for the SGM algorithm. As our SGM algorithm is edge adapting, we have two different values for P_1 and P_2 for edge and non-edge image pixels (see Section 4.5).



Name	Bits	Description
P1_edge	0 – 7	Penalty for small disparity changes at image edges. Default value: 3.
P1_no_edge	8 – 15	Penalty for small disparity changes outside image edges. Should be larger than P1_edge. Default value: 14.
P2_edge	16 – 23	Penalty for large disparity changes at image edges. Should be larger than P1_no_edge. Default value: 22.
P2_no_edge	24 – 31	Penalty for large disparity changes outside image edges. Should be larger than P2_no_edge. Default value: 65.

11.2.13 0x30: Subpixel Optimization ROI Offset

Offset of the region of interest (ROI) that is used for tuning the subpixel optimization (see Section 4.6.1). With the default parameters in this register and register 0x34, the subpixel optimization is tuned on the entire input image.



Name	Bits	Description
X offset	0 – 16	Horizontal offset of the ROI for subpixel optimization. Default value: 0.

Y offset	16 – 31	Vertical offset of the ROI for subpixel optimization. Default value: 0.
----------	---------	---

11.2.14 0x34: Subpixel Optimization ROI Size

Size of the region of interest (ROI) that is used for tuning the subpixel optimization (see Section 4.6.1). With the default parameters in this register and register 0x30, the subpixel optimization is tuned on the entire input image.

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	ROI height	ROI width
---	------------	-----------

Name	Bits	Description
ROI width	0 – 16	Width of the ROI for subpixel optimization. If this value is larger than the image width, then the full width of the input image is used. Default value: <i>0xFFFF</i> .
ROI height	16 – 31	Height of the ROI for subpixel optimization. If this value is larger than the image height, then the full height of the input image is be used. Default value: <i>0xFFFF</i> .

11.2.15 0x38: IP Version Number

Version number of the IP, divided by major, minor and patch version.

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	<i>Reserved</i>	Major version	Minor version	Patch version
---	-----------------	---------------	---------------	---------------

11.2.16 0x3C: Multi-Frame Subpixel Optimization

Parameters in this register allow the collection of subpixel statics over more than one frame (see Section 4.6.1).

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	Num Frames	<i>Reserved</i>	E	S
---	------------	-----------------	---	---

Name	Bits	Description
S	0	Subpixel optimization mode. This bit controls whether subpixel statistics should be collected over a single frame (= 0) or multiple frames (= 1). If set to 1, then either the end-of-statistics bit needs to be toggled when statistics collection should end, or the <i>Num Frames</i> parameter must be set to a value other than 0. Default value: 0.

E	1	End-of-statistics toggle bit. If the subpixel optimization mode is set to multiple frames and <i>Num Frames</i> is set to 0, then this bit controls the collection period. The collection period ends whenever this bit changes its value (0 to 1 or 1 to 0). Default value: 0.
Num Frames	16 – 31	If the subpixel optimization mode is set to multiple frames, then the number of frames for which subpixel statistics shall be collected can be specified in this parameter. When the given number of frames have been processed, the statistics is automatically updated. If this parameter is set to 0, no automatic update is performed and the end-of-statistics toggle bit must be used instead. Default value: 0.

11.2.17 0x40: Statistics on Rectification Input Caches

This register contains performance statistics for the cache memory that is used for image rectification. There is a separate cache for the left and right input images. For each cache, this register contains a counter that counts the *cache wait cycles*. As wait cycle we consider each clock cycle in which the cache was not able to deliver requested image data because the data was not yet in the cache.

Both counters need to be multiplied with 16 in order to get the true cycle count. A very high wait cycle count indicates a performance problem, which can result from highly warped input images or a too small cache.

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
Right wait cycles	Left wait cycles

11.2.18 0x44 – 0x98: Debugging Registers

If debugging registers have been enabled in the IP customization (see Section 6.1), then debugging information will be available in this register range. The contents of these registers is not documented and might differ between IP versions. These registers help Nerian’s tech support in identifying the cause of potential problems in customer designs.

12 Reference Design

When using the SVC in combination with the DMA core, it is important to connect the DMA core’s *clk* and the SVC’s *base_clk* clock inputs to the same clock source. All input and output signals of both IP cores are associated with this clock. The SVC’s *fast_clk* input can be connected to a faster clock, as described in Section 8 on page 24.

All inputs and outputs of the SVC except *frame_complete* shall be connected to the DMA core. The SVC’s *resetn* input shall be connected to the DMA core’s *system_resetn* output, such that it will also be reset when triggering a soft reset through DMA core’s register 0x00.

When using the provided IP cores on a Zynq SoC, it is usually desired that processing can be controlled by software, which is run on the Zynq's CPU cores. This requires that the device registers of both IP cores can be read and written from software. To facilitate this, the `register_io` ports of the SVC and DMA core need to be connected to one of the Zynq's general purpose AXI master interfaces. This requires an instance of the AXI interconnect IP. Both ports need to be mapped to different address ranges through the IP Integrator address editor.

For medium performance requirements, the DMA core's `*_dma` ports can be connected to the Zynq's high performance AXI slave interfaces. This allows reading input data from system memory, and writing the processing results back to memory. This configuration is depicted in Figure 10 for a complete system design.

If high-performance access to the processing results is required from the CPU, the cache coherent memory interfaces of the Zynq SOC should be used instead. When using the cache coherent memory interfaces, the `AxCACHE` and `AxPROT` parameters must be set appropriately (see Section 6.2). Also, the software (including the operating system) on the ARM CPU must be modified to work with cache coherent memory accesses. Please follow the instructions from the Xilinx Wiki² when using the cache coherent interfaces.

13 Control Flow

When using the DMA core, it is necessary to write to several device registers for processing an input stereo frame. As writing to some of these registers triggers certain actions, it is important to access them in a defined order. While many different access patterns lead to the desired result, we recommend using the reference control flow detailed in this section.

13.1 One-Time Initializations

After a hard or a soft reset the following registers should be written:

1. A value of 0 to DMA register 0x24 (*left input bytes available*).
2. A value of 0 to DMA register 0x30 (*right input bytes available*).
3. Format settings and iteration count to DMA register 0x00 (*control*).
4. Buffer memory address higher 32 bits to DMA register 0x44.
5. Buffer memory address lower 32 bits to DMA register 0x48.
6. Input image dimensions to DMA register 0x08.
7. License key higher 32 bits to SVC register 0x10.
8. License key mid 32 bits to SVC register 0x14.
9. License key lower 32 bits to SVC register 0x18.

²<https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842098/Zynq+UltraScale+MPSoC+Cache+Coherency>

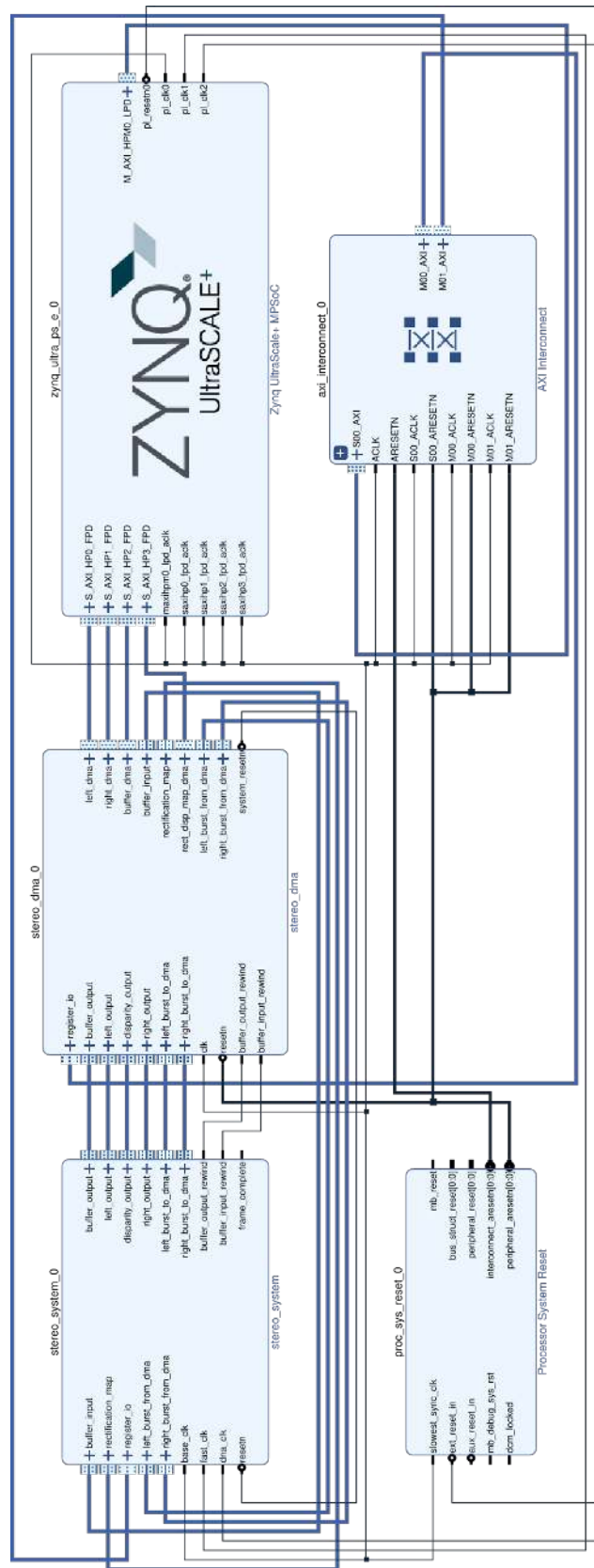


Figure 10: Reference design for Zynq SoC in IP Integrator.

10. Operation mode, iteration count and desired offset to SVC register 0x00 (*control*).
11. Input image dimensions to SVC register 0x04.
12. Algorithm parameters part 1 to SVC register 0x08.
13. Algorithm parameters part 2 to SVC register 0x0C.
14. SGM penalties to register 0x2C.

13.2 Per-Frame Control Flow

For each frame that should be processed, the following registers have to be written:

1. A value of 0 to DMA register 0x24 (*left input bytes available*).
2. A value of 0 to DMA register 0x30 (*right input bytes available*).
3. If the left rectified image shall be output:
 - (a) Left output address higher 32 bits to DMA register 0x5C.
 - (b) Left output address lower 32 bits to DMA register 0x60.
4. If the right rectified image shall be output:
 - (a) Right output address higher 32 bits to DMA register 0x68.
 - (b) Right output address lower 32 bits to DMA register 0x6C.
5. If the disparity map shall be output:
 - (a) Disparity output address higher 32 bits to DMA register 0x74.
 - (b) Disparity output address lower 32 bits to DMA register 0x78.
6. Left input address higher 32 bits to DMA register 0x1C.
7. Left input address lower 32 bits to DMA register 0x20.
8. Right input address higher 32 bits to DMA register 0x28.
9. Right input address lower 32 bits to DMA register 0x2C.
10. Rectification map input address higher 32 bits to DMA register 0x38.
11. Rectification map input address lower 32 bits to DMA register 0x3C.
12. Available left input bytes to DMA register 0x24.
13. Available right input bytes to DMA register 0x30.

13.3 Result Retrieval

When using the DMA core, the processing results can be retrieved directly from the selected memory locations that have been written to the DMA core's address registers. The number of valid output bytes at each memory location can be read from respective *bytes available* register. Processing is complete once this counter is equal to the expected output size (see Section 5.3). Alternatively, one can monitor the status bits in DMA register 0x04 (*status*) to determine when processing has finished.

14 Migrating from Previous Versions

The introduction of new features occasionally forces us to make changes to the register banks or the external interfaces. In this section you'll find a history of design-relevant changes for recent versions. When updating from older versions, please apply all changes chronologically, starting from the oldest change first.

14.1 Migrating from Version 5.3 to 6.0

1. Version 6.0 brings a new rectification implementation. The old implementation is still available if the new implementation is deselected in the customization parameters. If the new implementation is used, the image input ports of the SVC and DMA core will change. Please connect the ports with identical names together. If you prefer to keep the old rectification, no connection changes are needed.
2. The input encoding flags of the SVC's control register (see Section 11.2.1) have changed, and now match the flags for the DMA core.
3. Instead of specifying the maximum row stride in IP customization, we have moved back to specifying the maximum image width instead. It is still possible to manually enter a specific value for the maximum row stride, but with the new rectification the maximum stride has a much smaller impact on the resource usage.
4. With the new rectification it is no longer required to specify a maximum rectification displacement. The IP can now handle very larger displacements without problems. The old rectification maps are no longer compatible to the new rectification implementation.
5. Support for demosaicing of Bayer pattern images is no longer available for the new rectification implementation.
6. The previously deprecated interleaved output has been removed. If you are still using a design with interleaved output then you need to migrate to separate output images instead.

14.2 Migrating from Version 5.2 to 5.3

The two bit-flags *S* and *E* have been moved from SVC Register 0x0C to the new Register 0x3C.

14.3 Migrating from Version 4.4 – 4.6 to 5.0 – 5.2

Before version 5.0, the DMA core used to interleave all output data into a single memory buffer. Since version 5.0, the default behavior of the DMA core is to write each output image / map to separate memory buffers. The old interleaved output format is still available as an option for backwards compatibility. If you wish to keep the interleaved output then only the following migration steps are necessary.

1. Interleaved output mode must be activated in the IP customization of the SVC and DMA cores.
2. Make sure that the new DOE parameter in DMA register 0x00(*control*) is set to 01.
3. `rectification_map_dma` has been renamed to `rect_disp_map_dma`. The connection must be updated.

If you wish to migrate to the new output behavior with separate buffers, then the new registers 0x5C to 0x84 must be used correctly. Please refer to Section 13.

14.4 Migrating from Version 4.3 to 4.4

1. SVC customization

- (a) It is now possible to enable additional debugging registers during customization.

2. SVC registers

- (a) The SGM penalties configuration (P_1 and P_2) has been removed from register 0x08 and placed in its own dedicated register 0x2C (see Section 11.2.12). This change was necessary as the penalties are now edge adapting. For both penalties there is now a value for edge and for non-edge pixels.
- (b) A new edge detection threshold parameter has been added to register 0x08 (see Section 11.2.4). This parameter controls how sensitive the SGM algorithm will be to image edges.
- (c) A set of debugging registers has been added at the address range 0x44 – 0x98. These registers are only available if enabled in the IP customization.

14.5 Migrating from Versions 4.0 – 4.2 to 4.3

Version 4.3 introduced two register changes compared to previous versions. In the DMA core register, the control register (see Section 11.2.1) has changed as follows:

1. The *IE* parameter has been expanded from 3 to 4 bits in order to include the new Bayer pattern formats.
2. The *LOE* and *ROE* parameters have been moved to bit locations 24 – 27 and 28 – 31.

14.6 Migrating from Version 3.x to 4.0

Version 4.0 added support for RGB image processing. This extension required significant changes to the external interfaces as well as the register layout. The necessary design changes when migrating from version 3.x to a newer release are described below.

1. Interface changes

- (a) The SVC has a new output port `right_output`. This output needs to be connected to the equally named DMA core input port.

2. SVC customization

- (a) The input pixel formats that shall be supported need to be selected in the SVC and DMA core customization. Supported formats are 8-bit monochrome, 12-bit monochrome, 8-bit RGB.
- (b) The maximum row stride can now be configured independently to the maximum image width. If only one pixel format is supported, it is recommended to leave the option ‘determine maximum image width automatically’ enabled, and only configure the row stride. If more than one pixel format is supported, it might be useful to configure different values manually. Please refer to Section 4.2 for further details.
- (c) The new option ‘optimize for low iteration count’ is now available for configurations with only few iterations. Please refer to Section 6.1 for further details.

3. DMA core customization

- (a) The new SVC customization parameters also need to be configured in the DMA core.

4. SVC registers

- (a) The control register 0x00 has the new parameter *IE* that specifies the input encoding. This register must be set in accordance to the DMA core’s *IE* parameter. See Section 11.2.1
- (b) Register 0x28 has been added, which stores error flags. See Section 11.2.11.
- (c) The speckle filter disable flag in the algorithm parameters register 2 has been replaced by a configurable iteration count (see Section 11.2.4).

5. DMA core registers

- (a) The *IE* parameter in the control register 0x00 has been expanded from 2 to 4 bits, in order to allow the configuration of the RGB and bayer image formats. See Section 11.1.1.
- (b) The *OE* parameter, which specified the output encoding in the control register 0x00, has been replaced by two independent parameters *LOE* and *ROE* for the left and right image. See Section 11.1.1.

- (c) The new parameter O in the control register 0x00 specifies the operation mode. It needs to be set in accordance with the SVC's operation mode. See Section 11.1.1.
- (d) Register 0x50 has been added, which stores error flags. See Section 11.1.17.

In the SVC, the algorithm parameters register 2 (see Section 11.2.4) has changed. The speckle filter disable flag has been replaced by a configurable iteration count. To disable the speckle filter, an iteration count of 0 needs to be written.

Revision History

Revision	Date	Author(s)	Description
v6.0.0	April 7, 2021	KS	New rectification implementation and related changes.
v5.3.0	October 29, 2020	KS	Changes to multi-frame subpixel-optimization registers.
v5.2.0	August 7, 2020	KS	New multi-frame mode for subpixel-optimization.
v5.1.0	February 19, 2020	KS	New parameter: generate repeatable results.
v5.0.0	December 23, 2019	KS	New DMA core version with separate output buffers.
v4.6.0	November 7, 2019	KS	New improved subpixel optimization; version registers.
v4.5.0	September 4, 2019	KS	Added support for right-to-left matching and rectification skip.
v4.4.0	July 4, 2019	KS	Added edge-depending penalty adjustment and debug registers.
v4.3.0	March 8, 2019	KS	Added multiple speckle filter iterations and Bayer downsampling.
v4.2.0	January 22, 2019	KS	Added external buffer compression.
v4.1.0	January 4, 2019	KS	Allow iteration count of 1.
v4.0.0	December 5, 2018	KS	Updated for new IP core version 4.0. This version introduced RGB processing support.
v2.3	January 10, 2018	KS	New 12-bit input encoding modes. Updates to DMA control register.
v2.2	October 31, 2017	KS	Changed 12-bit packing format and clarified packing.
v2.1	September 7, 2017	KS	12-bit support, new packed 12-bit output, and bit depth conversion.
v2.0	July 29, 2017	KS	Major update for UltraScale+ devices.
v1.7	February 23, 2017	KS	Added description of concurrent of concurrent read / write bursts.
v1.6	January 27, 2017	KS	Fixes to control flow description.
v1.5	January 24, 2017	KS	Configurable disparity range and other updates for new IP core version 2.0.
v1.4	July 19, 2016	KS	Updated for new AXI interface. Added processing parameter recommendations.
v1.3	July 4, 2016	KS	Texture filter; updated timing and resource usage; changed AXI ID width.

v1.2	March 16, 2016	KS	Multi-clock design; speckle filter; variable image sizes; updated default parameterization, resource usage, timing and device registers.
v1.1	July 15, 2015	KS	Updated timing, resource usage and reference parameterization for optimized SVC.
v1.0	June 20, 2015	KS	Simplification of Section 2; minor rewording.
v0.2	June 4, 2015	KS	Split IP core into SVC and DMA core; added output merging; added subpixel optimization; updated resource usage, timing and registers to current version.
v0.1	April 10, 2015	KS	Initial revision

References

- AIA (2013). Video Streaming and Device Control Over Ethernet Standard. Version 2.0.
- ARM (2010). AMBA 4 AXI4-Stream Protocol. ARM IHI 0051A (ID030510).
- ARM (2013). AMBA AXI and ACE Protocol Specification. ARM IHI 0022E (ID022613).
- European Machine Vision Association (2016). Pixel Format Naming Convention. Version 2.1.
- Hirschmüller, H. (2005). Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 807–814.