Stereo Vision IP Core

Data Sheet

(v1.4) July 19, 2016

Nerian Vision Technologies
Dr. Konstantin Schauwecker
Gotenstr. 9
70771 Leinfelden-Echterdingen
Germany

Email: service@nerian.com
http://nerian.com

Contents

1 Introduction **3**

2 Features **3**

3 Background **4**
3.1 Camera Alignment 4
3.2 Image Rectification 4
3.3 Camera Calibration 5
3.4 Disparity Maps 5

4 Stereo Vision Core Functionality **6**
4.1 Rectification 6
4.2 Image Pre-Processing 8
4.3 Stereo Matching 8
4.4 Cost Volume Post-Processing 9
4.4.1 Subpixel Optimization 9
4.4.2 Uniqueness Check 9
4.4.3 Consistency Check 9
4.5 Disparity Map Post-Processing 10
4.5.1 Texture Filtering 10
4.5.2 Speckle Filtering 10
4.5.3 Gap Interpolation 10
4.5.4 Noise Reduction 10

5 DMA Core Functionality **10**
5.1 Ports Connected to SVC 11
5.2 Interface Ports 12
5.3 Output Conversion 12
5.3.1 Split Disparity Map 12
5.3.2 Extended Disparities 13

6 Parameterization **13**

7 Supported Devices **13**

8 Timing **15**

9 Resource Usage **15**

10 IO Signals **15**

11 Registers **20**
11.1 DMA Core Registers 20
11.1.1 0x00: Control 20
11.1.2 0x04: Status 20
11.1.3 0x08: Image Size 22
11.1.4 0x0C: Output Address 22

11.1.5 0x10: Output Bytes Available 22
11.1.6 0x14: Output FIFO Info 22
11.1.7 0x18: Left Input Address 22
11.1.8 0x1C: Left Input Bytes Available 22
11.1.9 0x20: Right Input Address 23
11.1.10 0x24: Right Input Bytes Available 23
11.1.11 0x28: Input FIFO Info 23
11.1.12 0x2C: Rectification Map Address 23
11.1.13 0x30: Input FIFO Info 23
11.1.14 0x34: Buffer Address 23
11.1.15 0x38: Buffer FIFO Info 24
11.2 SVC Registers 24
11.2.1 0x00: Control 24
11.2.2 0x04: Image Size 24
11.2.3 0x08: Algorithm Parameters 1 24
11.2.4 0x0C: Algorithm Parameters 2 25
11.2.5 0x10: License Key Higher 32 Bits 25
11.2.6 0x14: License Key Lower 32 Bits 25
11.2.7 0x18: Device DNA Higher 25 Bits 25
11.2.8 0x1C: Device DNA Lower 32 Bits 25

12 Reference Design **26**

13 Control Flow **26**
13.1 One-Time Initializations 26
13.2 Per-Frame Control Flow 28
13.3 Result Retrieval 28

1 Introduction

The Stereo Vision Core (SVC) performs stereo matching on two grayscale 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 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.

Depending on the release, the IP cores are provided as encrypted and / or obfuscated netlist. An IP block for both cores 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 bits per pixel
  - Stream-based processing of input images using either AXI4-Stream or AXI3
  - 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
- Stereo matching
  - Stereo matching through a variation of the Semi-Global Matching (SGM) algorithm
  - Configurable penalties P_1 and P_2 for small and large disparity variations
  - 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 have to 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 Technologies website[1].

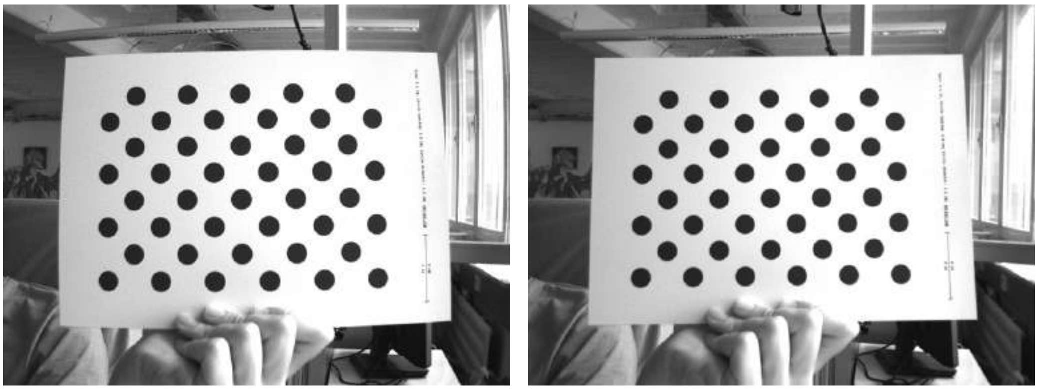
3.2 Image Rectification

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 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.

[1] http://nerian.com/support/resources/calculator/

Figure 1: Example for standard epipolar geometry.



(a) (b)

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

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 form a set of images is provided.

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 left camera. The disparity map assigns each pixel in the left camera image to a corresponding pixel in the right camera image. Because both images have to be 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. The disparity is proportional to the inverse depth of the corresponding scene point. It is thus 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:

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

(a) (b)

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

An efficient implementation of this transformation is available in the API for the SP1 stereo vision system.

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 `0xFFF`, 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 is depicted in Figure 4. The port `register_io` provides read and write access to the device registers, which keep all processing parameters. This port complies to the AXI4-Lite standard (ARM, 2013) and acts as a communication slave. The remaining input and output ports implement the AXI4-Stream protocol (ARM, 2010).

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 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

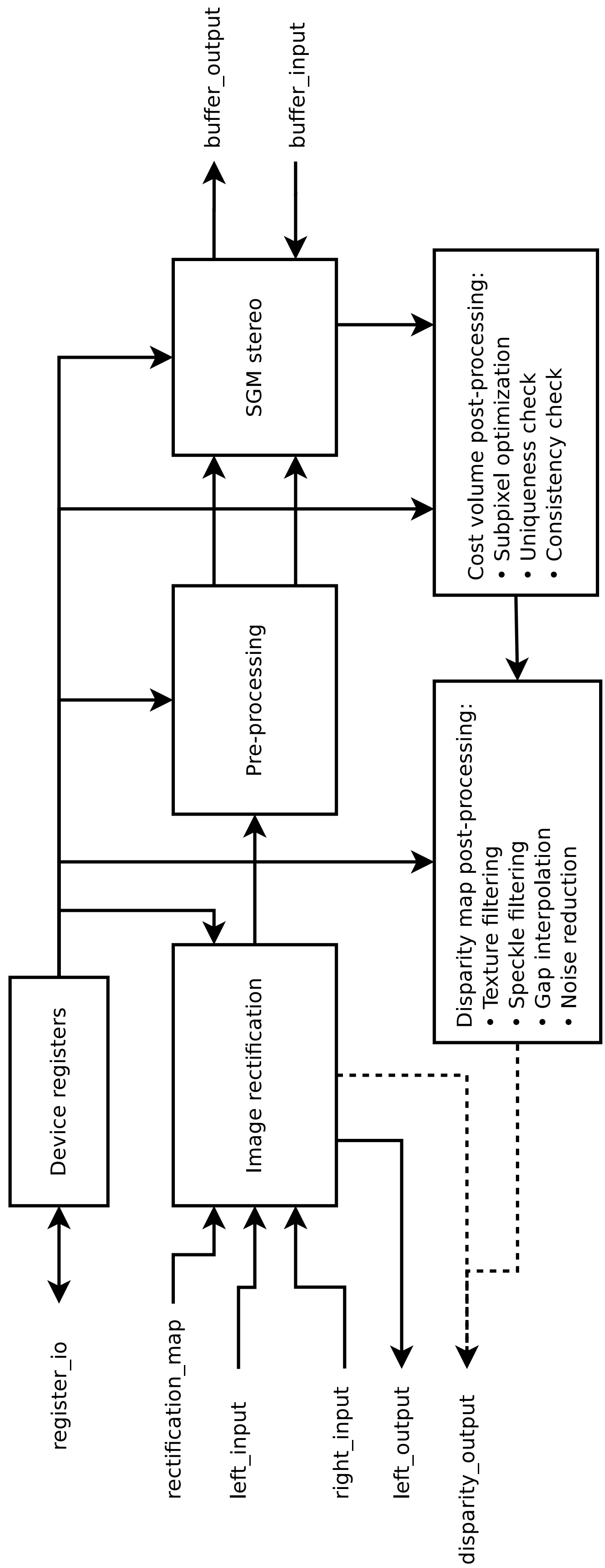


Figure 4: Block diagram of SVC functionality.

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.

Rectification is a window-based operation. Hence, the pixel offsets are limited by the employed window size. In our reference parameterization a window size of 79×79 pixels is used. This allows for offsets in the range of -39 to $+39$ pixels. If desired, the window size can be adjusted to allow for larger pixel displacements.

The left rectified image is always written to `left_output`. If desired, the right rectified image can be written to `disparity_output` by setting the appropriate operation mode in the SVC registers (see Section 11.2.1 on page 24). When outputting the right rectified image, stereo matching results are not available. Furthermore, as the `disparity_output` port is intended for delivering 12 bit wide subpixel accurate disparites, it has a larger than needed data width. When delivering the right rectified image over this output, the least significant four bits, which otherwise correspond to the subpixel component of a disparity value, are set to 0.

4.2 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.

4.3 Stereo Matching

Stereo matching is performed by applying a variation of the SGM algorithm by Hirschmüller (2005). The left input image is selected as reference image and matched against the right image data. The penalties P_1 and P_2 that are employed by SGM for small and large disparity variations can be configured at runtime though the SVC registers.

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

$$s_b = 3 \cdot (d_{max} + 1) \cdot w_{max} \ , \tag{1}$$

where d_{max} is the maximum disparity and w_{max} is the maximum supported image width. For the reference parameterization from Section 6, this buffer requires a total size of 262.5 KB.

Data is written linearly to the buffer, starting form 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.

4.4 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.

4.4.1 Subpixel Optimization

Subpixel optimization is the first applied post-processing technique. This step increase 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 improved disparity estimates are then 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.

4.4.2 Uniqueness Check

Matches with a high matching uncertainty are discard 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:

$$c^* \cdot q < \min \{C \setminus \{c_{\min}\}\} . \tag{2}$$

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

4.4.3 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 (in our case from the right image to the left image), and then only retaining matches for which

$$|d_l - d_r| \leq t_c, \tag{3}$$

where d_l is the disparity from left-to-right matching, d_r the disparity from right-to-left 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 right camera disparity map is inferred from the matching costs that have been gathered during the initial left-to-right stereo matching. Pixels that do not pass the consistency check are again labeled with 0xFFF.

4.5 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.

4.5.1 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 0xFFF in the computed disparity map.

4.5.2 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 controlled through the speckle filter window size w_s. The pixels that belong to identified speckles are again labeled with 0xFFF.

4.5.3 Gap Interpolation

The aforementioned post-processing techniques all remove pixels form 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

$$\min\{l_h, l_v\} \leq l_{max}, \tag{4}$$

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.

4.5.4 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

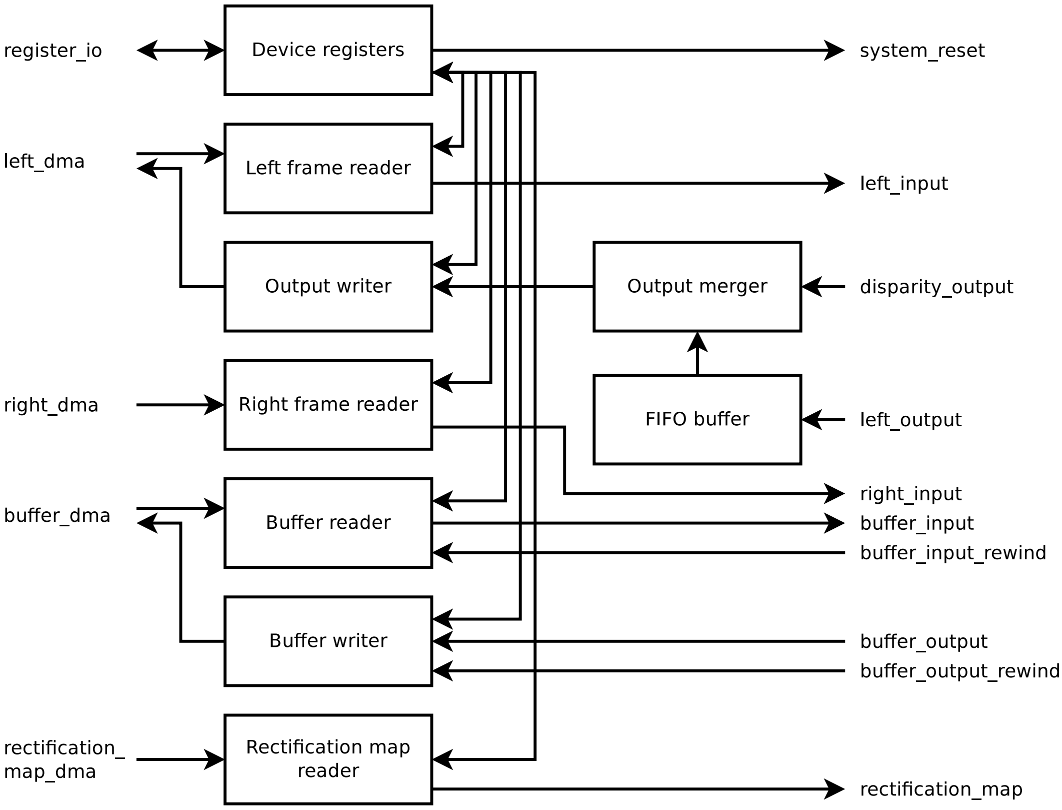


Figure 5: Block diagram of DMA core functionality.

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 clock 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 7, plus one further output and two inputs that were omitted previously for simplicity.

The new output is `system_reset`, 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.

The remaining ports follow the AXI3 standard (ARM, 2013) and act as communication masters. The `left_dma` port fetches the left input image and is also used for delivering the processing results. Similarly, the `right_dma` port is used for fetching the right input image. The port `buffer_dma` serves for reading from and writing to the buffer memory and the `rectification_map_dma` port is used for fetching the rectification map.

All fetch and write operations of the AXI3 ports are controlled through the device registers. They contain the input and output memory addresses and can trigger reading or writing operations when set to a new value. More details on the device registers can be found in Section 11 on page 20.

5.3 Output Conversion

The rectified left image and the disparity map that are output by the SVC are merged into a single data stream by the DMA core. This data stream is then output over the `left_dma` port. Because the disparity map is computed with a significantly higher delay than the rectified left image, the DMA core contains a sufficiently sized FIFO buffer for buffering the rectified left image data. For merging the two incoming data streams, the DMA core provides two possible options.

5.3.1 Split Disparity Map

In the first option the disparities are split into an 8-bit integer component and a 4-bit subpixel component, which consists of the disparity decimal bits (see Section 4.4.1). We hence receive two new maps, which are the *integer disparity map* and the *subpixel component map*.

An image row of the left rectified image, a row of the integer disparity map and a row of the subpixel component map are then output consecutively over the `left_dma` port. This interleaved output is repeated until there are no more remaining rows to be delivered.

It has to be considered that an element of the subpixel 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.

The merged output data can be interpreted as a row-wise sampled image with dimensions $2.5w \times h$, where w and h are the width and height of the input image. An example for the output of the DMA core when interpreted this way is shown in Figure 6. As can be seen, the output image is a horizontal arrangement of the left rectified image, the integer disparity map and the subpixel component map.

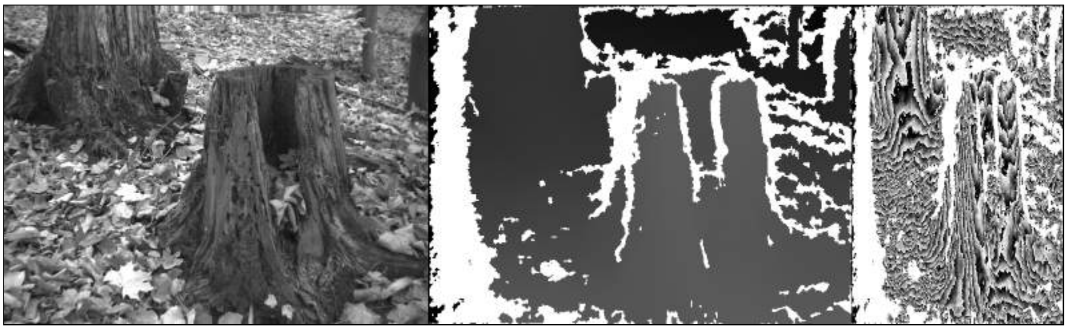


Figure 6: Example for interpreting the merged output as image when splitting the disparity map.

5.3.2 Extended Disparities

The alternative method to merging the SVC output is to not split the disparities into integer and subpixel components. Rather, the disparities are extended to 16 bits. This happens by introducing additional high-significant bits, which are set to 0.

The output is then again a row-wise interleaving of the left rectified image and the 16-bit extended disparity map. Compared to the first output option, this method produces more data due to the disparity extension. In total, the data that is delivered over the `left_output` port is equivalent to a $3w \times h$ sized image.

6 Parameterization

The SVC can be configured through several parameters, which are listed in Table 1. The table further includes our reference parameterization, which we recommend and use in our own products. All performance indicators that are provided in this document have been obtained with this reference parameterization.

Because the IP core is provided as a netlist, the parameterization cannot be changed by the user. Please contact us if you require a different parameter set, and we will provide you with one or more netlists for your evaluation. Please be aware that some parameters can have a great impact on the required FPGA resources and can influence the design timings. We will assist in finding an adequate parameterization for your application.

7 Supported Devices

The SVC has been field-tested on the Zynq 7000 SoC. We thus recommend using this FPGA family. However, our IP core is also compatible to other Xilinx 7-Series FPGAs. Please contact us to find out if your desired device is supported.

Table 1: Available parameters and reference parameterization.

Parameter	Description	Reference param.
Max. width w_{max}	Maximum width of an input image.	800
Max. height h_{max}	Maximum height of an input image.	800
Rectification window size	Window size that is used for image rectification (see Section 4.1).	79
Maximum disparity d_{max}	The maximum disparity label that is considered for stereo matching.	111
Pixel cycles	Number of clock cycles that SGM uses for processing a single pixel.	7
Maximum gap width l_{max}	The maximum extent in horizontal or vertical direction for a gap in the disparity map, such that it is still considered for gap interpolation (see Section 4.5.3).	5
Speckle filter window size w_s	Window size used by the speckle filter (see Section 4.5.2).	11
Split output disparity map	The disparities will be split into integer and subpixel components during output merging (DMA core only; see Section 5.3)	true
Input FIFO size	Number of bits that are FIFO-buffered for each AXI input. This value does not apply to the buffer input.	2048
Output FIFO size	Number of bits that are FIFO-buffered for each AXI output. This value does not apply to the buffer output.	1024
Buffer input FIFO size	Number of bits that are FIFO-buffered for the buffer AXI input.	43008
Buffer output FIFO size	Number of bits that are FIFO-buffered for the buffer AXI output.	43008

Table 2: SVC processing delays for reference parameterization when processing input images of size 640 × 480 pixels.

Delay	Base clock cycles	Time
Delay until first output	103,000	1.03 ms
Delay until last output	2,032,000	20.32 ms

8 Timing

The SVC has been implemented as a multi-clock design. All input and output signals are associated with the *base clock*. When synthesized for the Zynq 7000 SoC, this clock can have a frequency of up to 100 MHz. In addition to the base clock, the SVC uses the so-called *fast clock* for clocking particularly performance critical tasks. For the Zynq 7000 SoC, this clock can have a frequency of up to 125 MHz.

If the DMA core is employed, its clock input has to be connected to the base clock. The maximum clock speed of the base clock matches the clock speed for the Zynq's Static Memory Controller (SMC). Hence, the DMA core can be directly connected to the Zynq's memory interfaces.

The expected SVC processing delays when processing an input image of resolution 640×480 pixels with the discussed reference parameterization are listed in Table 2. The delays are given for a 100 MHz base clock and a 125 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 completed.

The measurements were determined under the assumption that new data is always available at the SVCs inputs. If the data to be processed is read from system memory, higher delays might occur due to the additional memory delays.

9 Resource Usage

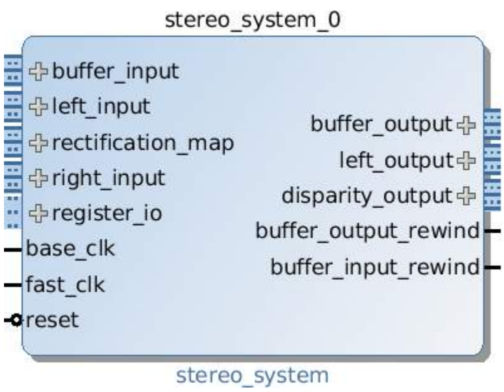
The total resource usage of the SVC and DMA core is listed in Table 3. The table further contains resource usage information for the individual sub-modules that have been identified in Section 4. These numbers provides an overview of the gains that can be achieved when removing one of the sub-modules from the SVC.

10 IO Signals

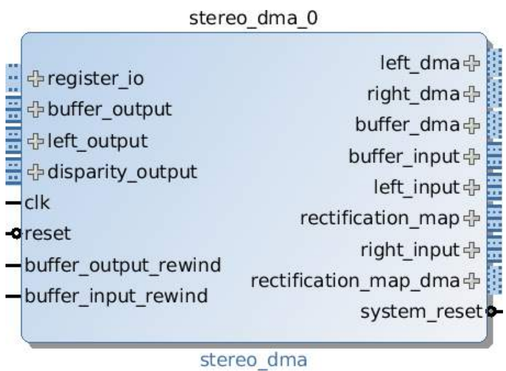
Figures 7a and 7b contain a depiction of the SVC and DMA core as they appear in IP Integrator, which is part of Xilinx Vivado. Most of the shown ports have already been described in Sections 4 and 5. A detailed list of all input and output signals, including a breakdown of the AXI ports, is provided in Table 4 for the SVC. Likewise, Table 5 contains an equivalent list for the DMA core.

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

Description	Slice LUTs	Registers	Memory	DSPs
SVC total usage	28,839	39,454	97.5	24
DMA core total usage	2,957	3,466	4.0	2
Image rectification	2,216	2,092	34.0	10
Image pre-processing	272	820	3.0	2
SGM stereo	15,195	23,377	42.0	2
Subpixel optimization	292	607	0.0	0
Uniqueness check	774	1,196	0.0	1
Consistency check	3,420	6,355	0.5	0
Texture filter	396	513	9.5	1
Speckle filter	4,159	2,544	3.5	1
Gap interpolation	644	675	4.0	6
Noise suppression	794	1,041	1.0	1
Others	677	234	0.0	0



(a)



(b)

Figure 7: Interfaces of (a) SVC and (b) DMA core as shown by IP Integrator.

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
fast_clk	i	1	A faster clock for performance critical tasks
reset	i	1	Gobal reset signal; active low
			register_io signals
register_io_araddr	i	32	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	32	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_input_tready	o	1	Ready to receive left image
left_input_tvalid	i	1	Left image data is valid
left_input_tdata	i	8	Left image data
right_input_tready	o	1	Ready to receive right image
right_input_tvalid	i	1	Right image data is valid
right_input_tdata	i	8	Right image data
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 4.3 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	384*	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	8	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	11*	Disparity output data
buffer_output_rewind	o	1	Writing to buffer shall restart from offset 0 (see Sections 4.3 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	384*	Buffer output data

* Size given for reference parameterization.

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
reset	i	1	Gobal reset signal; active low
			register_io signals
register_io_araddr	i	32	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	32	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_io / left_io / right_io / rect_map signals
∗_araddr	o	32	Read address
∗_arburst	o	2	Read burst type; always 01_2 (INCR)!

*_arcache	o	4	Read memory type; always 0011_2 (normal, non-cacheable, bufferable)!
*_arlen	o	4	Read burst length; always 1111_2 (16 transfers)!
*_arlock	o	2	Read lock type; always 00_2 (normal acces)!
*_arprot	o	3	Read protection type; always 000_2 (unprivileged secure data)!
*_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; always 0011_2 (normal, non-cacheable, bufferable)!
*_awlen	o	4	Write burst length; always 1111_2 (16 transfers)!
*_awlock	o	2	Write lock type; always 00_2 (normal acces)!
*_awprot	o	3	Write protection type; always 000_2 (unprivileged secure data)!
*_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; ignored!
*_bvalid	i	1	Write response valid
*_rdata	i	64	Read data
*_rlast	i	1	Read last
*_rready	o	1	Read ready
*_rresp	i	2	Read response; ignored!
*_rvalid	i	1	Read last
*_wdata	o	64	Write data
*_wlast	o	1	Write last
*_wready	i	1	Write ready
*_wstrb	o	8	Write strobes; always 11111111_2!
*_wvalid	o	1	Write valid

left_input / right_input / rectification_map / buffer_input signals

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

left_output / disparity_output / buffer_ output signals

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

			Other signals directed to SVC
system_reset	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 SVC, and in Table 7 for the DMA core. 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 four. 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 four 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.

11.1 DMA Core Registers

11.1.1 0x00: Control

General parameters that control the behavior of the SVC.

31 … 2	1	0
Reserved	R	

R If set to 1 then the device performs a soft reset.

11.1.2 0x04: Status

General device status information.

31 … 7	6	5	4	3	2	1	0
Reserved	BR	BW	M	RR	LR	LW	R

R If 0 then the device is currently performing a soft or hard reset.

LW If 1 then writing to `left_dma` has finished.

LR If 1 then reading from `left_dma` has finished.

RR If 1 then reading from `right_dma` has finished.

M If 1 then reading from `rectification_map_dma` has finished.

BW If 1 then writing to `buffer_dma` has finished.

BR If 1 then reading from `buffer_dma` has finished.

Table 6: Address space for SVC registers.

Address	Name	Read/Write
0x00	Control	r/w
0x04	Image size	r/w
0x08	Algorithm parameters 1	r/w
0x0C	Algorithm parameters 2	r/w
0x10	License key higher 32 bits	r/w
0x14	License key lower 32 bits	r/w
0x18	Device DNA higher 25 bits	r
0x1C	Device DNA lower 32 bits	r

Table 7: Address space for DMA registers.

Address	Name	Read/Write
0x00	Control	r/w
0x04	Status	r
0x08	Image size	r/w
0x0C	Output address	r/w
0x10	Output bytes available	r
0x14	Output FIFO info	r
0x18	Left input address	r/w
0x1C	Left input bytes available	r/w
0x20	Right input address	r/w
0x24	Right input bytes available	r/w
0x28	Input FIFO info	r
0x2C	Rectification map address	r/w
0x30	Rectification map FIFO info	r
0x34	Buffer address	r/w
0x38	Buffer FIFO info	r

11.1.3 0x08: Image Size

Dimensions of the left and right input images.

<table><tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td colspan="16">Image height</td><td colspan="16">Image width</td></tr></table>

11.1.4 0x0C: Output Address

Write address for `left_dma`. Writing ends once one full frame has been written to memory.

<table><tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td colspan="32">Address</td></tr></table>

11.1.5 0x10: Output Bytes Available

The number of bytes that have successfully been written to `left_dma` since the start of the current frame.

<table><tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td colspan="32">Available bytes</td></tr></table>

11.1.6 0x14: Output FIFO Info

Statistics for the output FIFO buffer that is attached to `left_dma`. The counter is reset with every new frame.

<table><tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td colspan="16">Reserved</td><td colspan="16">Output FIFO overruns</td></tr></table>

11.1.7 0x18: Left Input Address

Read address for `left_dma`. Reading from this address begins immediately after this register has been written. Reading continues until one full frame has been read from memory.

<table><tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td colspan="32">Address</td></tr></table>

11.1.8 0x1C: 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.

<table><tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td colspan="32">Bytes available</td></tr></table>

11.1.9 0x20: Right Input Address

Read address for `right_dma`. Reading from this address begins immediately after this register has been written. Reading continues until one full frame has been read from memory.

31–0
Address

11.1.10 0x24: 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.

31–0
Bytes available

11.1.11 0x28: 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.

31–16	15–0
Right FIFO underruns	Left FIFO underruns

11.1.12 0x2C: Rectification Map Address

Read address for `rectification_map_dma`. Reading from this address begins immediately after this register has been written. Reading continues until the full rectification map has been read from memory.

31–0
Address

11.1.13 0x30: Input FIFO Info

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

31–16	15–0
Reserved	Rectification map FIFO underruns

11.1.14 0x34: Buffer Address

Memory address for `buffer_io`. This address is used for both, reading and writing data.

31–0
Address

11.1.15 0x38: Buffer FIFO Info

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

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
Input FIFO underruns																Output FIFO overruns															

11.2 SVC Registers

11.2.1 0x00: Control

General parameters that control the behavior of the SVC.

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
Reserved																														OP	

OP Operation mode. Possible values are:

- **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.
- **01** Rectify. The rectification results are passed directly to the SVC's left and right output.
- **10** Stereo matching (default). Stereo matching results are written to the SVC's disparity output, and the left rectified image is written to the left output.
- **11** Reserved

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															

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.				Reserved			Uniqueness factor									P_2								P_1							

P_1 SGM penalty for small disparity variations (see Section 4.3). Default value: 6.

P_2 SGM penalty for large disparity variations (see Section 4.3). Default value: 22.

Uniqueness Factor Uniqueness factor q times 256. A value of 0 disables the uniqueness check (see Section 4.4.2). Default value: 320.

Consist. Consistency check threshold t_c (see Section 4.4.3). 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
Reserved																Texture threshold								Reserved				S	N	G	C

Texture threshold Threshold for the texture filter. A value of 0 disables the texture filter (see Section 4.5.1). Default value: 10.

C If set to 1 then the consistency check is disabled (see Section 4.4.3). Default value: 0.

G If set to 1 then the gap interpolation is disabled (see Section 4.5.3). Default value: 0.

N If set to 1 then the noise reduction is disabled (see Section 4.5.4). Default value: 0.

S If set to 1 then the speckle filtering is disabled (see Section 4.5.2). Default value: 0.

11.2.5 0x10: License Key Higher 32 Bits

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

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
License key bits 63 to 32																															

11.2.6 0x14: License Key Lower 32 Bits

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

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
License key bits 31 to 0																															

11.2.7 0x18: Device DNA Higher 25 Bits

The most significant 25 bits of the 57-bit Xilinx device DNA.

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
Reserved							Device DNA bits 56 to 32																								

11.2.8 0x1C: Device DNA Lower 32 Bits

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

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
Device DNA bits 31 to 0																															

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 15.

All inputs and outputs of the SVC shall be connected to the DMA core. The SVC's `reset` input shall be connected to the DMA core's `system_reset` output, such that it will also be reset when triggering a soft reset through DMA 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.

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. Please refer to Figure 8 for an illustration of the full reference design for a Zynq SoC.

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 0x1C *(left input bytes available)*.
2. A value of 0 to DMA register 0x24 *(right input bytes available)*.
3. Buffer memory address to DMA register 0x34 *(buffer address)*.
4. Input image dimensions to DMA register 0x08 *(image size)*.
5. License key most-significant 32 bits to SVC register 0x10.
6. License key least-significant 32 bits to SVC register 0x14.
7. Input image dimensions to SVC register 0x04 *(image size)*.
8. Algorithm parameters to SVC register 0x08 *(algorithm parameters 1)*.
9. Algorithm parameters to SVC register 0x0C *(algorithm parameters 2)*.

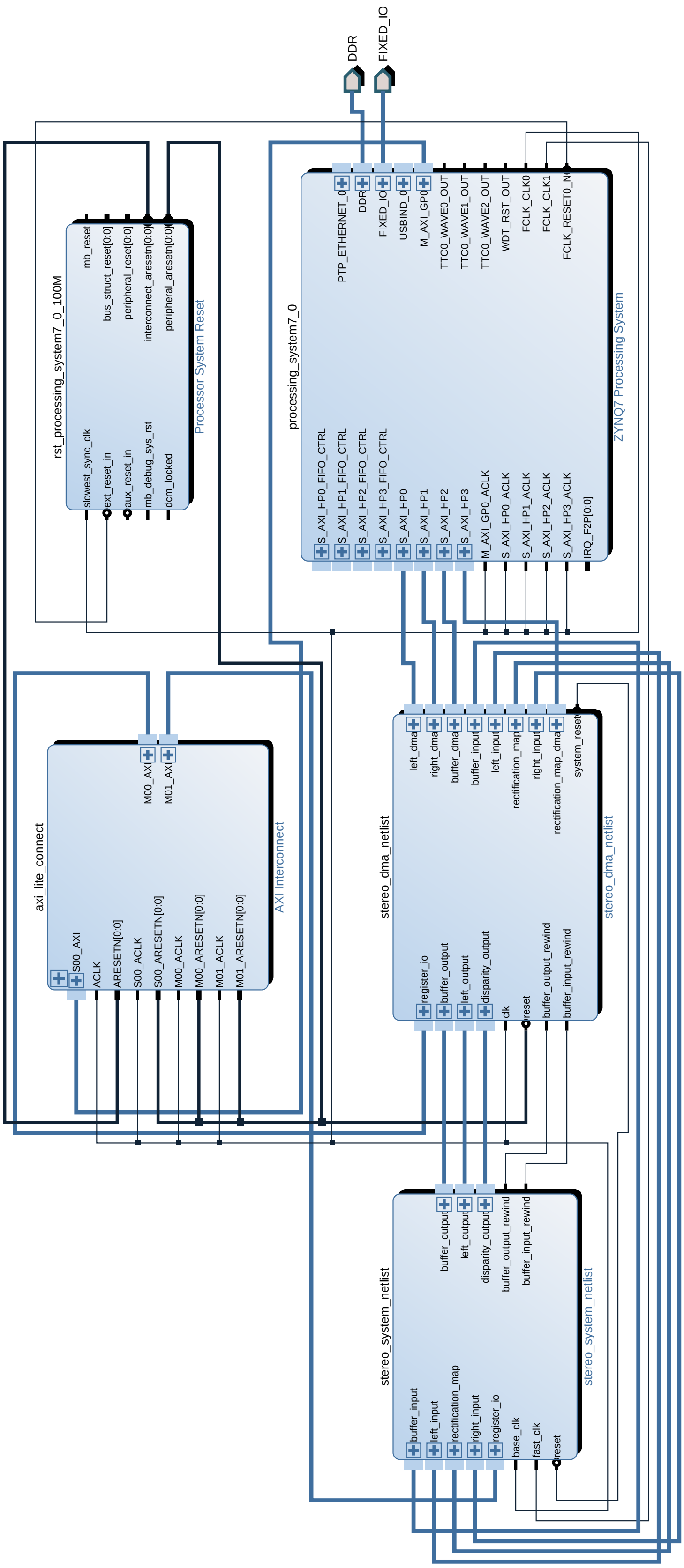


Figure 8: Reference design for Zynq SoC in IP Integrator.

10. Operation mode to write SVC 0x00 *(control)*.

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 0x1C *(left input bytes available)*.
2. A value of 0 to DMA register 0x24 *(right input bytes available)*.
3. Output address to DMA register 0x0C *(output address)*.
4. Left input address to DMA register 0x1C *(left input address)*.
5. Right input address to DMA register 0x24 *(right input address)*.
6. Rectification map input address to DMA register 0x2C *(rectification map address)*.
7. Available left input bytes to DMA register 0x1C *(left input bytes available)*.
8. Available right input bytes to DMA register 0x24 *(right input bytes available)*.

13.3 Result Retrieval

When using the DMA core, the processing results can be retrieved directly from the selected memory location that has been written to DMA register 0x0C *(output address)*. The number of valid output bytes can be read from DMA register 0x10 *(output bytes available)*. 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.

Revision History

Revision	**Date**	**Author(s)**	**Description**
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

ARM (2010). AMBA 4 AXI4-Stream Protocol. ARM IHI 0051A (ID030510).

ARM (2013). AMBA AXI and ACE Protocol Specification. ARM IHI 0022E (ID022613).

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.