Stereo Vision IP Core

Data Sheet

(v1.3) July 4, 2016

nerian
VISION TECHNOLOGIES

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

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

Contents

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 a data stream that is suitable for the SVC. Likewise, the DMA core also collects the output data from the SVC and writes it back to memory.

The SVC is provided as an encrypted netlist. For the DMA core, source code can be provided if required. 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
  - 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 Stereo Vision Core Functionality

The overall functionality of the SVC is depicted in Figure 1. The displayed input and output ports implement a simplified subset of 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.

3.1 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. During this step, image pixels are displaced in order to compensate for lens distortions and camera alignment errors. Bi-linear interpolation is applied to allow for pixel displacements with subpixel accurate offsets.

To perform the 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. 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 pixel offsets in the range of -39 to $+39$. 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 device register as explained in Section 10 on page 16. When outputting the right rectified image, stereo matching results are not available. Further, as the `disparity_output` port is intended for delivering subpixel accurate disparity maps, it has a larger than needed data width. When delivering the right rectified image over this output, the least significant bits, which otherwise correspond to the subpixel component of a disparity value, are set to 0.

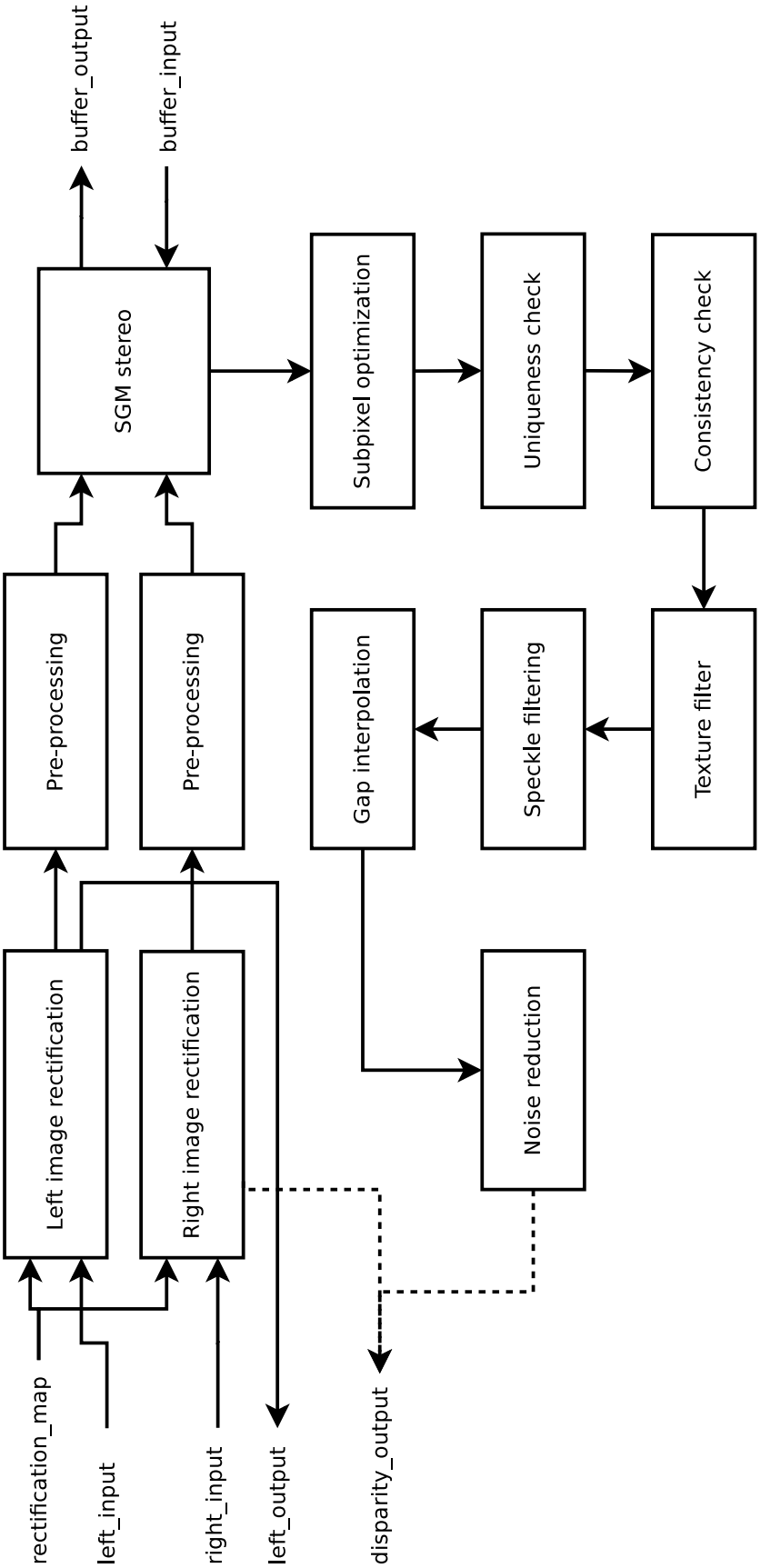


Figure 1: Block diagram of SVC functionality.

3.2 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.

3.3 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 device 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 5, 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`. It is ensured that reading and writing will never happen simultaneously on the same buffer data.

3.4 Subpixel Optimization

Subpixel optimization is applied to the results from stereo matching in order to increase the accuracy of depth measurements. In this step, the costs to the left and to the right of the minimum matching cost are evaluated. 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.

3.5 Uniqueness Check

A range of post-processing methods is applied to the data from stereo matching. First we discard matches with a high matching uncertainty 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 feature 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, which corresponds to the decimal value 255.9375 and is the maximum value that can be transmitted through the `disparity_output` port.

3.6 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 retain 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.

3.7 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.

3.8 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.

3.9 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.

3.10 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. The final disparity map, which results after this filter, is directly written to the `disparity_output` port.

4 DMA Core Functionality

When using the SVC 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 2.

4.1 Ports Connected to SVC

Except for the clock signals, the DMA core connects to all input and output ports of the SVC. In Figure 2, those ports are depicted on the right. The ports match the ones shown in Figure 1 on page 5, plus two further outputs and two inputs that were omitted previously for simplicity.

The first new SVC-specific output port is `write_registers`. This port delivers the contents of all writable device registers, which are held by the DMA core. The SVC only requires a subset of the device registers, as a large fraction is used by the DMA core for controlling memory access. The registers that are needed by the SVC occupy the address range from 0x00 to 0x14. Thus, only the lower 192 bits of the `write_registers` port need to be connected to the SVC.

Another 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 of 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.

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.

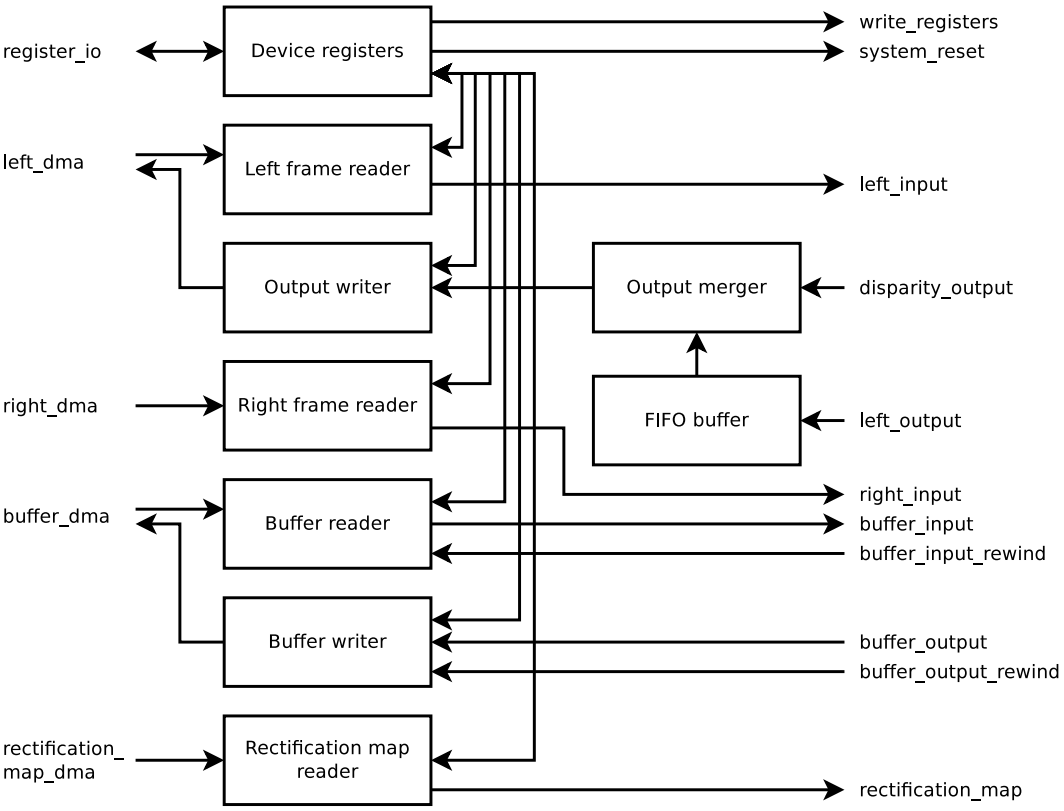


Figure 2: Block diagram of DMA core functionality.

4.2 Interface Ports

All ports that are not connected to the SVC appear on the left hand side of Figure 2. The port `register_io` provides read and write access to all device registers, which are held by 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 when set to a new value. More details on the device registers can be found in Section 10 on page 16.

4.3 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 written out over the `left_dma` port. Because the disparity map is output with a significantly higher delay than the rectified left image, the SVC core contains a sufficiently sized

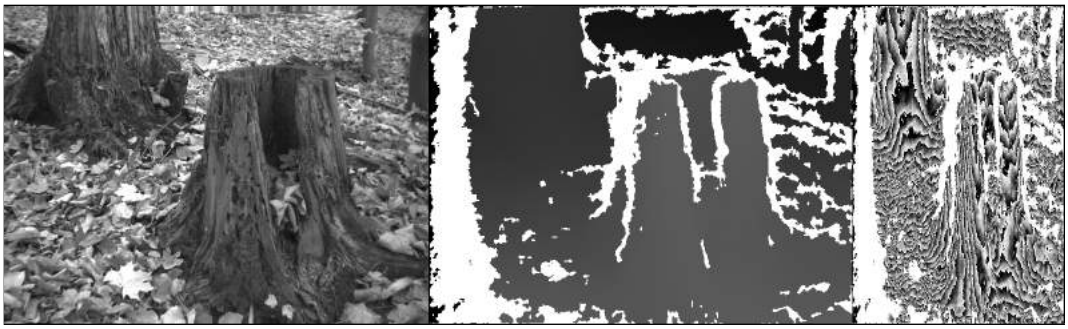


Figure 3: Example for interpreting the merged output as image when splitting the disparity map.

FIFO buffer for buffering the rectified left image data. For merging the two incoming data streams, the DMA core provides two possible options.

4.3.1 Split Disparity Map

In the first option the disparities are split into an integer component (extended to 8 bit) and a 4-bit subpixel component, which consists of the disparity decimal bits (see Section 3.4). 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 we always write the less-significant 4-bits first, and the more-significant 4-bits last.

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 3. 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.

4.3.2 Extended Disparities

The alternative method for merging the SVC output does 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.

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 3.1).	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 3.9).	5
Speckle filter window size w_s	Window size used by the speckle filter (see Section 3.8).	11
Split output disparity map	The disparities will be split into integer and subpixel components during output merging (DMA core only; see Section 4.3)	true
Input FIFO size	Number of data items that are FIFO-buffered for each AXI input. This value does not apply to the buffer input.	8
Output FIFO size	Number of data items that are FIFO-buffered for each AXI output. This value does not apply to the buffer output.	4
Buffer input FIFO size	Number of data items that are FIFO-buffered for the buffer AXI input.	8
Buffer output FIFO size	Number of data items that are FIFO-buffered for the buffer AXI output.	8

5 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. Be aware that some parameters can have a great impact on the required FPGA resources and can influence the design timings. We will assist you in finding an adequate parameterization for your application.

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

6 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.

7 Timing

The SVC has been implemented as a multi-clock design. All input and output signals are associated to 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 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 complete.

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.

8 Resource Usage

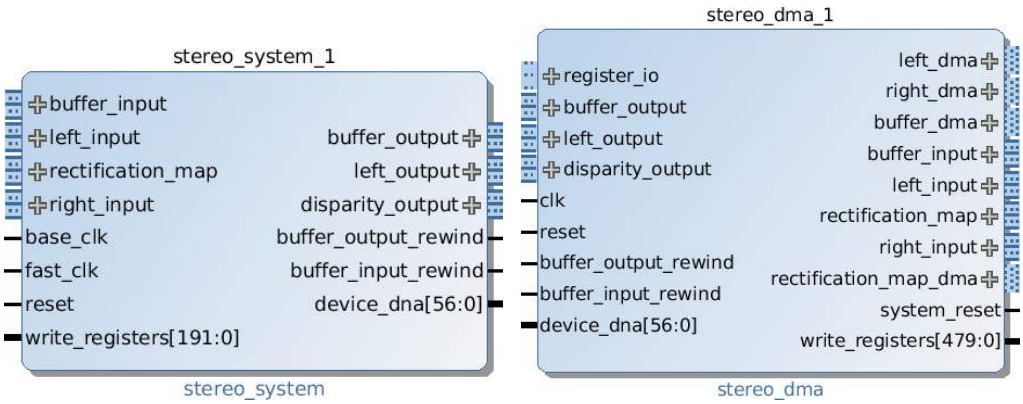
The total resource usage of the SVC and DMA core is listed in Table 3. The table further contains resource usage information for the individual sub-modules that have been identified in Figure 1 on page 5. These numbers provides an overview of the gains that can be achieved when removing one of the sub-modules from the core.

9 IO Signals

Figures 4a and 4b 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 3 and 4. A detailed list of all input and output signals,

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

Description	Slice LUTs	Registers	Memory	DSPs
SVC total usage	28,655	39,220	97.5	24
DMA core total usage	2,954	3,544	4.0	2
Image rectification	2,249	2,092	34.0	10
Image pre-processing	264	820	3.0	2
SGM stereo	15,211	23,377	42.0	2
Subpixel optimization	291	607	0.0	0
Uniqueness check	774	1,196	0.0	1
Consistency check	3,521	6,355	0.5	0
Texture filter	406	513	9.5	1
Speckle filter	4,160	2,544	3.5	1
Gap interpolation	646	675	4.0	6
Noise suppression	812	1,041	1.0	1
Others	321	0	0.0	0



(a) (b)

Figure 4: Interfaces of (a) SVC and (b) DMA core as shown by IP Integrator.

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 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
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
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 3.3 and 4.1)
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
buffer_input_rewind	o	1	Reading from buffer shall restart from offset 0 (see Sections 3.3 and 4.1)
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
write_registers	i	192	Writable device registers, forwarded by DMA core (see Section 4.1)

* 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)!
*_arid	o	0	Read address ID; Not used!
*_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)!

*_awid	o	0	Write address ID; Not used!
*_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
*_bid	i	0	Write response ID; Not used!
*_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
*_rid	i	0	Read ID; Not used!
*_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
*_wid	o	0	Write ID; Not used!
*_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 4.1)
write_registers	o	416	Writable device registers forwarded to SVC (see Section 4.1)

10 Registers

The DMA core holds several registers that control the device behavior and provide information about the internal device state. Please note that read- and write-registers

each have a separate address space. Hence, writing to a given address has no effect on the data that is received when reading from the same address. A list of all read and write registers can be found in Tables 6 and 7.

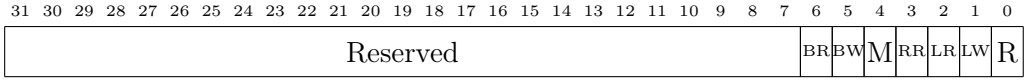
The contents of all write registers is concatenated and output through the DMA core's `write_registers` port. As shown in Table 7, the lower 6 write registers are relevant for the SVC. Hence, the lower 192 bits of the DMA core's `write_registers` output should be wired to the SVC's `write_registers` input (see Section 11 for an example). If the SVC is used without the DMA core, the relevant register values have to be provided directly to the SVC.

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 read and write registers is provided, sorted by register address and related IP core.

10.1 DMA Core Read Registers

10.1.1 0x00: Status

General device status information.



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.

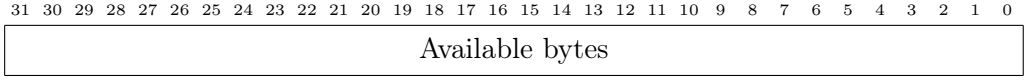
M If 1 then reading from `rectification_map_dma` has finished.

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

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

10.1.2 0x04: Output Bytes Available

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



10.1.3 0x08: Input FIFO Info

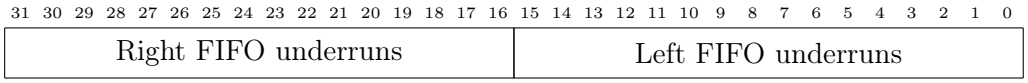
Statistics for the input FIFO buffers that are attached to `left_dma` and `right_dma`. Counters are reset with every new frame.

Table 6: Address space for read registers of DMA core.

Address	**Name**	**Related IP core**
0x00	Status	DMA
0x04	Output bytes available	DMA
0x08	Input FIFO info	DMA
0x0C	Output FIFO info	DMA
0x10	Buffer FIFO info	DMA
0x14	Device DNA higher 25 bits	DMA
0x18	Device DNA lower 32 bits	DMA

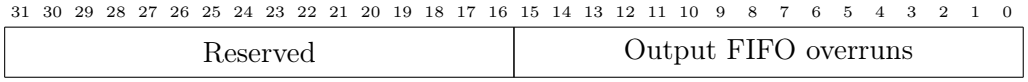
Table 7: Address space for write registers of SVC and DMA core.

Address	**Name**	**Related IP core**
0x00	Control	SVC
0x04	Image size	SVC
0x08	Algorithm parameters 1	SVC
0x0C	Algorithm parameters 2	SVC
0x10	License key higher 32 bits	SVC
0x14	License key lower 32 bits	SVC
0x18	Output address	DMA
0x1C	Left input address	DMA
0x20	Left input bytes available	DMA
0x24	Right input address	DMA
0x28	Right input bytes available	DMA
0x2C	Rectification map address	DMA
0x30	Buffer address	DMA



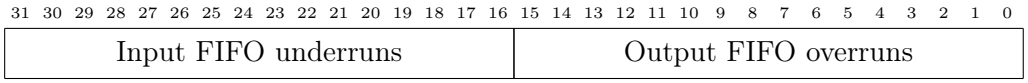
10.1.4 0x0C: Output FIFO Info

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



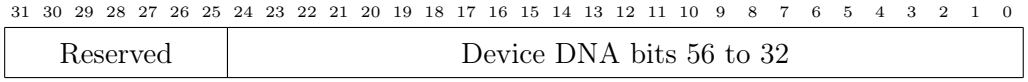
10.1.5 0x10: Buffer FIFO Info

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



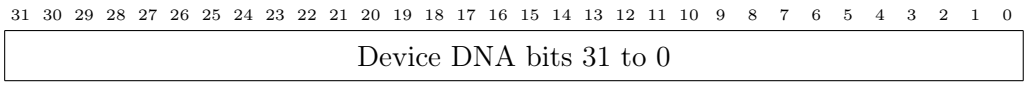
10.1.6 0x14: Device DNA Higher 25 Bits

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



10.1.7 0x18: Device DNA Lower 32 Bits

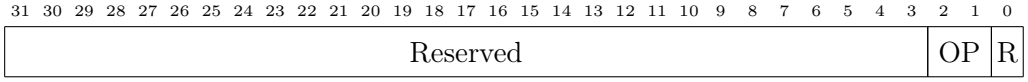
The least significant 32 bits of the 57-bit Xilinx device DNA.



10.2 SVC Write Registers

10.2.1 0x00: Control

General parameters that control the behavior of the SVC.



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

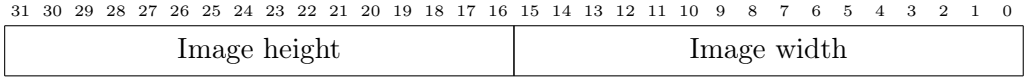
OP Operation mode. Possible values are:

- **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. Stereo matching results are written to the SVC's disparity output, and the left rectified image is written to the left output.

11 Reserved

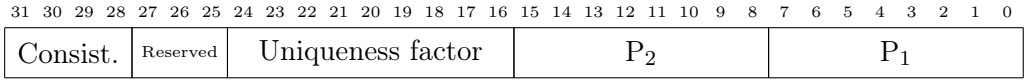
10.2.2 0x04: Image Size

Dimensions of the left and right input images.



10.2.3 0x08: Algorithm Parameters 1

Algorithmic parameters that can be changed at run-time.



P_1 SGM penalty for small disparity variations (see Section 3.3).

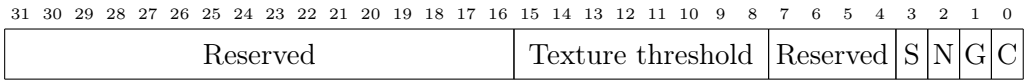
P_2 SGM penalty for large disparity variations (see Section 3.3).

Uniqueness Factor Uniqueness factor q times 256. A value of 0 disables the uniqueness check (see Section 3.5).

Consist. Consistency check threshold t_c (see Section 3.6).

10.2.4 0x0C: Algorithm Parameters 2

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



Texture threshold Threshold for the texture filter. A value of 0 disables the texture filter (see Section 3.7).

C If set to 1 then the consistency check is disabled (see Section 3.6).

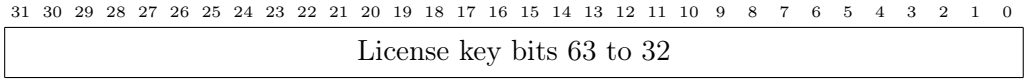
G If set to 1 then the gap interpolation is disabled (see Section 3.9).

N If set to 1 then the noise reduction is disabled (see Section 3.10).

S If set to 1 then the speckle filtering is disabled (see Section 3.8).

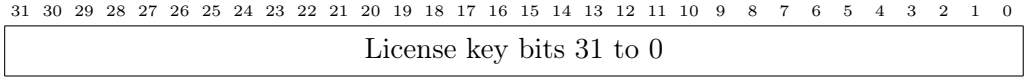
10.2.5 0x10: License Key Higher 32 Bits

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



10.2.6 0x14: License Key Lower 32 Bits

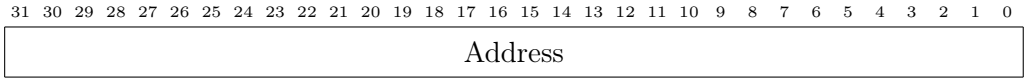
The least-significant 32 bits of the device-specific license key.



10.3 DMA Core Write Registers

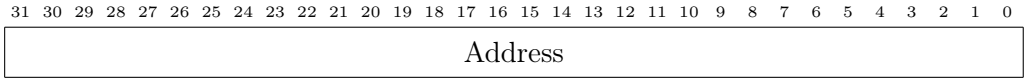
10.3.1 0x18: Output Address

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



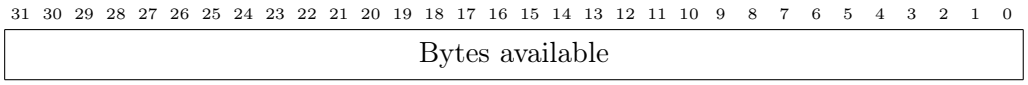
10.3.2 0x1C: Left Input Address

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.



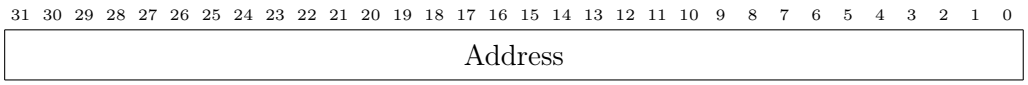
10.3.3 0x20: Left Input Bytes Available

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.



10.3.4 0x24: Right Input Address

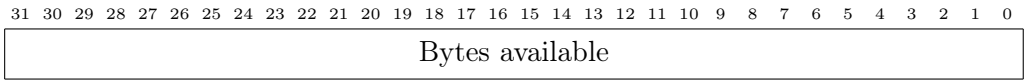
Read address for `right_dma`. Reading from this address begins immediately after this register has been written. Reading continues until one full frame has been read from memory.



10.3.5 0x28: Right Input Bytes Available

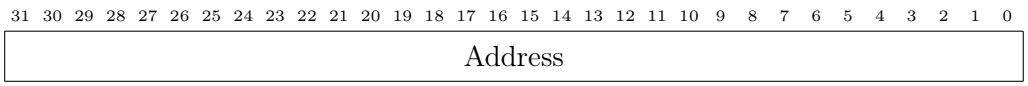
The number of bytes that can currently be read from `right_dma`, starting at the right input address. If this number is smaller than the frame size then reading will pause once the specified number of bytes have been read. In this case reading will

continue once a higher value is written to this register.



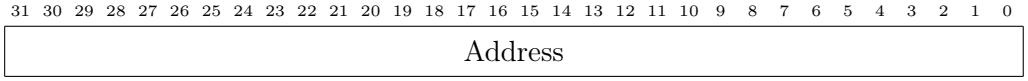
10.3.6 0x2C: Rectification Map Address

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.



10.3.7 0x30: Buffer Address

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



11 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 have to be associated with this clock. The SVC's `fast_clk` input can be connected to a faster clock, as described in Section 7 on page 12.

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 the device registers. As the SVC only requires a subset of the available device registers, the width of the `write_registers` input of the SVC is smaller than the corresponding output of the DMA core. In order to make the necessary connection, an instance of the *slice* IP block is required, as visible in Figure 5. The slice block needs to be configured to pass through the lower 192 bits of the DMA core's `write_registers` output.

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 can be read and written from software. To facilitate this, the `register_io` port of the DMA core needs to be connected to one of the Zynq's general purpose AXI master interfaces. Due to the fact that `register_io` complies to the AXI4-Lite standard, an AXI interconnect block is necessary to make this connection.

All the remaining `*_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 5 for the full reference design.

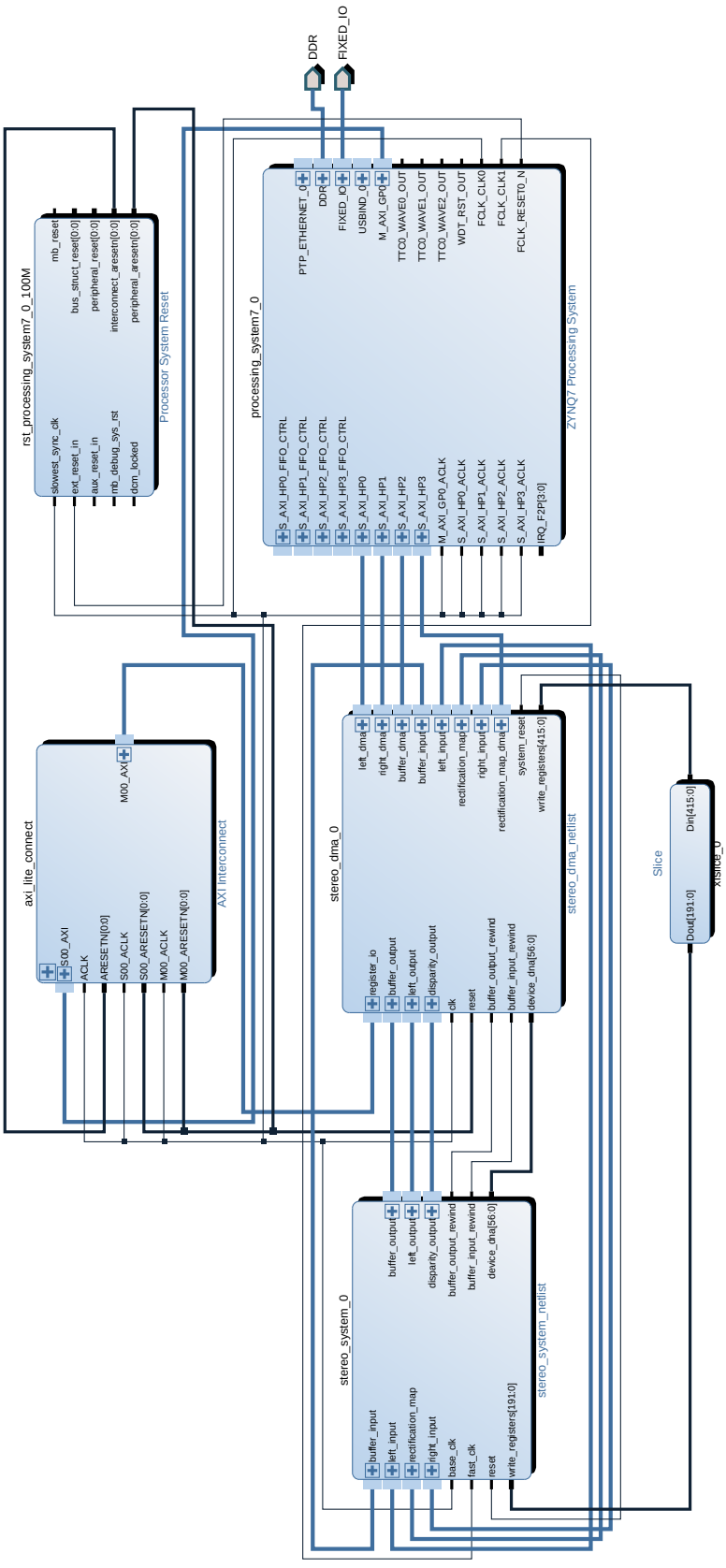


Figure 5: Reference design for Zynq SOC in IP Integrator.

12 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.

12.1 One-Time Initializations

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

1. License key most-significant 32 bits to write register 0x10.
2. License key least-significant 32 bits to write register 0x14.
3. A value of 0 to write register 0x20 *(left input bytes available)*.
4. A value of 0 to write register 0x28 *(right input bytes available)*.
5. Buffer memory address to write register 0x30 *(buffer address)*.
6. Input image dimensions to write register 0x04 *(image size)*.
7. Algorithm parameters to write register 0x08 *(algorithm parameters 1)*.
8. Algorithm parameters to write register 0x0C *(algorithm parameters 2)*.
9. Operation mode to write register 0x00 *(control)*.

12.2 Per-Frame Control Flow

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

1. A value of 0 to write register 0x20 *(left input bytes available)*.
2. A value of 0 to write register 0x28 *(right input bytes available)*.
3. Output address to write register 0x18 *(output address)*.
4. Left input address to write register 0x20 *(left input address)*.
5. Right input address to write register 0x28 *(right input address)*.
6. Rectification map input address to write register 0x2C *(rectification map address)*.
7. Available left input bytes to write register 0x20 *(left input bytes available)*.
8. Available right input bytes to write register 0x28 *(right input bytes available)*.

12.3 Result Retrieval

When using the DMA core, the processing results can be retrieved directly from the selected memory location that has been written to write register 0x18 *(output address)*. The number of valid output bytes can be read from read register 0x04 *(output bytes available)*. Processing is complete once these counters are equal to the expected output size (see Section 4.3). Alternatively, one can monitor the status bits in read register 0x00 *(status)* to determine when processing has finished.

Revision History

Revision	Date	Author(s)	Description
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.