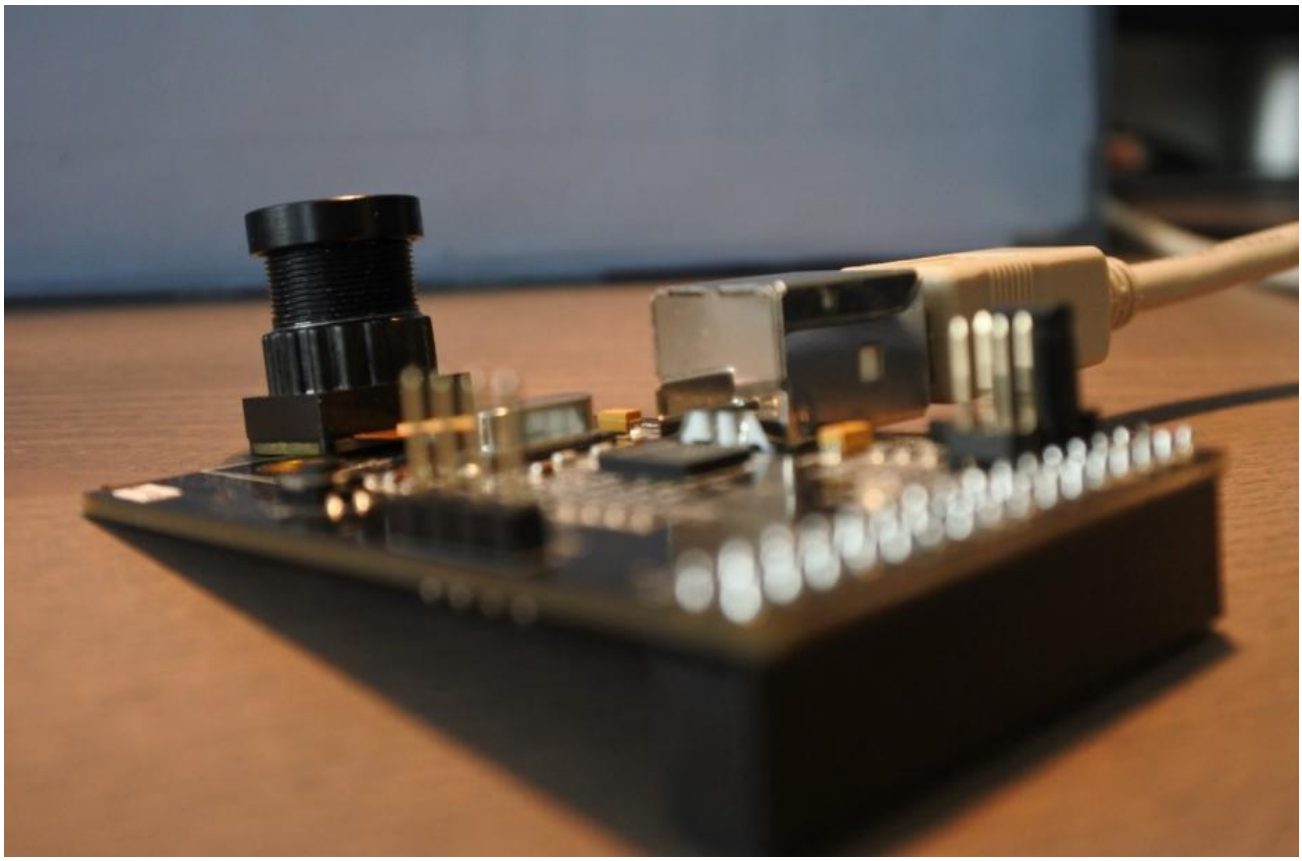
Data Sheet

Bit-MIPI CSI-2 Rx IP

A very compact, from 500 LUTs/FFs, camera sensor receiver IP, interface converting from CSI-2 to AXI4-Stream Video standard



Introduction

The **BitCsi2Rx** IP is a very compact receiver for camera sensor signals, to be used in an FPGA or ASIC.

It receives camera signals in accordance with the MIPI CSI-2 and D-PHY specifications. BitCsi2Rx converts these signals to parallel video signals for AXI4-Stream Video standard. To make it easier to interface BitCsi2Rx with various other design blocks, it also outputs fval (frame valid) and lval (line valid) signals for synchronization.

Deliveries

- Licenses for protected or readable VHDL source code
- VHDL testbench with stimuli and checkers
- User Guide
- Expert technical support and maintenance

Licensing

- For more information or license purchase: info@bitsim.com

A MIPI CSI-2 transmitter IP is also available from BitSim, **BitCsi2Tx**.

1. System Description

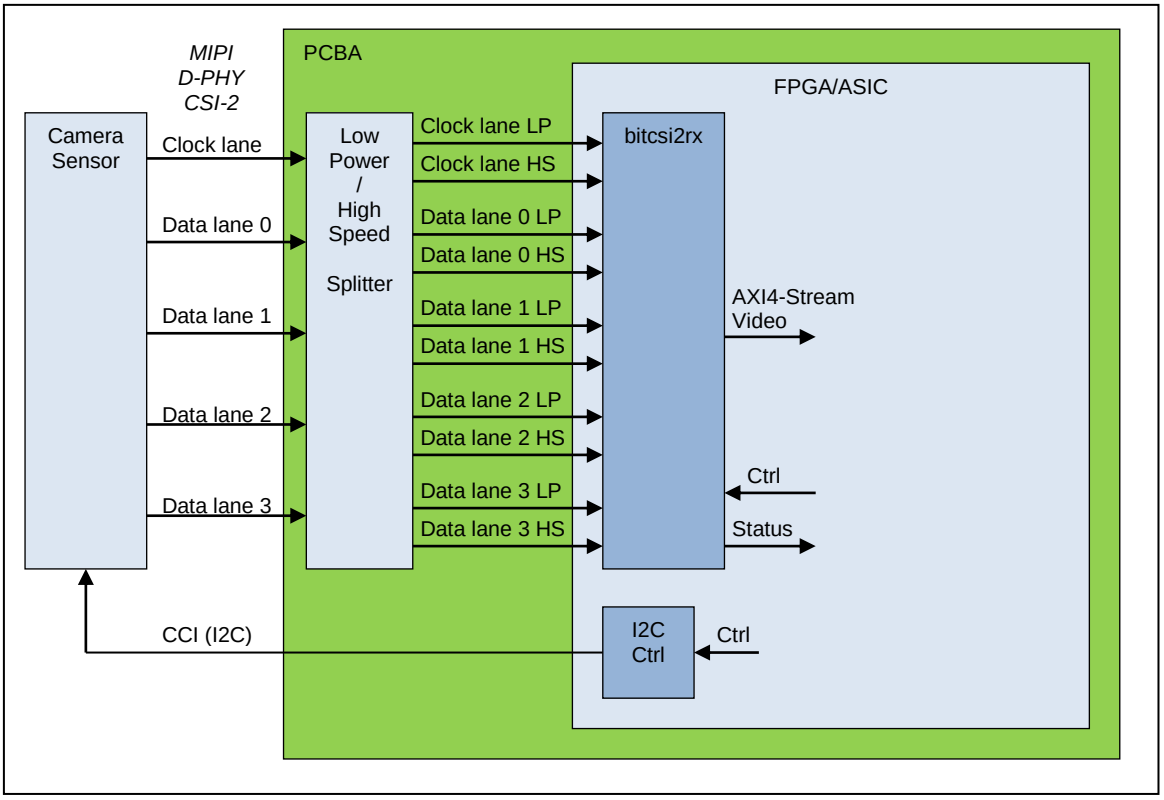


Figure 1 BitCsi2Rx in a system

The MIPI D-PHY standard uses a signal-pair for each lane. Each pair consists of a positive and a negative signal. The pairs switch between high-speed mode and low power mode. The high-speed mode is used when there is video data to send during active video, and the low power mode is used when there is no data, which is during horizontal and vertical blanking.

In high-speed mode, the pair is used as low-voltage differential signals. In low-power mode, the two signals of the pair are used as independent single-ended signals with higher signal levels. Most FPGA families do not support switching between low-power differential mode and higher-voltage single ended mode during run-time.

Therefore, external circuitry is required to split the incoming camera signal into separate low-voltage differential FPGA-pins/balls and higher-voltage single ended pins/ball. This can be done with a simple resistor net, or with specialized chips.

The bitcsi2rx does not include a Camera Control Interface (CCI). CCI can easily be implemented with an I2C controller IP in the FPGA/ASIC, or an I2C controller peripheral in a SoC.

2. Features

- 1 to 4 data lanes
- Up to 2.5 Gb/lane
- Also support Xilinx UltraScale+ family, with embedded D-PHY IOs
- Number of data lanes can be configured statically at synthesis-time and dynamically during run-time
- Supported CSI-2 Data Types:
  RAW8/10, YUV422_8/10, RGB888, RAW12/14/16, Generic, User Defined
- AXI4-Stream Video output with additional synchronization signals
- Virtual Channel Support
- ECC checking and correction for packet headers and short packets
- Checksum checking for packet payload data
- D-PHY protocol decoding included
- Clock-lane/data-lanes deskew
- Full High Speed/Low Power mode support
- Test/debug features
- Written in VHDL, prepared for instantiation in Verilog or SystemVerilog design

3. Block Description

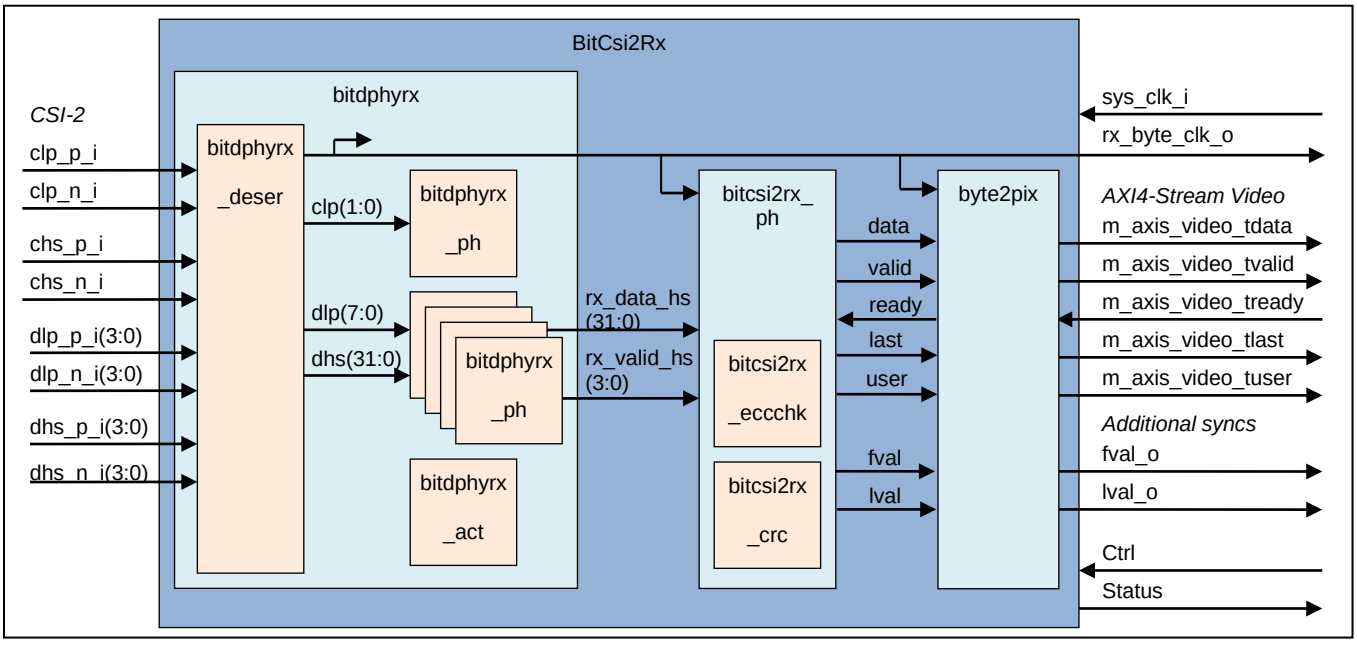


Figure 2 BitCsi2Rx block diagram, configured with four data lanes

The CSI-2 input consists of a clock lane and one to four data lanes. Each lane is a signal pair with a positive and negative half. The lanes must be split in a Low Power part and a High-Speed part outside the FPGA/ASIC.

bitdphyrx extracts a byte clock from the incoming clock lane, and deserializes the data in the data lanes. It also interprets the D-PHY protocol and outputs the D-PHY packets. This is the only block in bitcsi2rx that is tailored to a specific FPGA family or ASIC technology. All other blocks are written in generic, technology independent code. The data lanes can be individually adjusted for shorter or longer delay in run-time, to compensate for differing lane lengths on the PCB.

bitcsi2rx_ph handles the CSI-2 protocol and extracts the video- and sync information. It outputs video as AXI4-Stream Video with additional sync signals to make it easier to interface to various other design blocks.

byte2pix converts the 8-bit data blocks into the correct data format as specified by the data_type indicator. This results in 8, 10, 12, 14 or 16-bits data per lane. The total width of the m_axis_video_tdata signal becomes the data_type width multiplied with the number of lanes. If fewer data bits than the largest possible are used, the effective part is placed at the MSB end of the data word.

4. Ports

This is not a complete list, it is only intended as an overview.

Port	Range	Dir	Description
Clock and reset			
sys_clk_i	-	In	System clock
sys_rst_i	-	In	Synchronous reset in sys_clk_i domain
rx_byte_clk_o	-	Out	Byte clock generated from clock lane input
rx_byte_rst_o	-	Out	Synchronous reset in rx_byte_clk_o domain
Control, asynchronous			
enable_i	-	In	Enables BitCsi2Rx
Control, sys_clk_i domain			
t_clk_term_en_i	-	In	D-PHY timing setting
t_clk_settle_i	-	In	D-PHY timing setting
t_clk_miss_i	-	In	D-PHY timing setting
t_clk_lp11_timeout	-	In	D-PHY timing setting
clk_stop2prpr_ena_i	-	In	Accepts clock lane stop state protocol violations
Control, rx_byte_clk_o domain			
num_datalanes_i	-	In	Number of MIPI CSI-2 data lanes to use (run-time selection). 00 = 1 data lane 01 = 2 data lanes 10 = 3 data lanes 11 = 4 data lanes
checksum_ena_i	-	In	Enables or disables checksum checking of long packets. Checksum is an optional feature in the MIPI CSI-2 standard, so all CSI-2 transmitters don’t support checksum

Port	Range	Dir	Description
dhs_delay_i	-	In	Pad input delay for dhs_p_i/dhs_n_i, for compensating for clock-to-data skew. Each data lane has an individual delay value
dhs_delay_stb_i	-	In	Pulse 1 to apply the dhs_delay_i value
t_d_term_en_i	-	In	D-PHY timing setting.
t_hs_settle_i	-	In	D-PHY timing setting.
t_d_lp11_timeout_i	-	In	D-PHY timing setting.
dat_stop2prpr_ena_i	-	In	Accepts data lane stop state protocol violations
CSI-2 settings, rx_byte_clk_o domain			
vc_o	-	Out	Virtual channel number of current or last received packet
dt_o	-	Out	CSI-2 Data type of current or last received packet
wc_o	-	Out	Word count of current or last received packet
dt_long_o	-	Out	CSI-2 Data type of current or last received long packet.
frame_num_o	-	Out	Frame number included in the last START FRAME packet.
AXI4-Stream Video Master output interface, rx_byte_clk_o domain			
m_axis_video_tdata	-	Out	Video data output
m_axis_video_tvalid	-	Out	Valid pixel output
m_axis_video_tready	-	In	AXI4-Stream Video slave ready to receive data
m_axis_video_tlast	-	Out	End of Line
m_axis_video_tuser	-	Out	Start of Frame

Port	Range	Dir	Description
Additional video synchronization signals			
fval_o	-	Out	Frame valid
lval_o	-	Out	Line valid
sof_packet_o	-	Out	Pulses when FRAME START packet received
eof_packet_o	-	Out	Pulses when FRAME END packet received
D-PHY RX error flags, rx_byte_clk_o domain			
err_sot_hs_o	-	Out	Pulses when detected correctable D-PHY Sync word error
err_sot_sync_hs_o	-	Out	Pulses when detected non-correctable D-PHY Sync word error
Serial clock lane diff inputs from pads, Low Power and High Speed			
clp_p_i	-	In	Clock lane Low Power, Positive
clp_n_i	-	In	Clock lane Low Power, Negative
chs_p_i	-	In	Clock lane High Speed, Positive
chs_n_i	-	In	Clock lane High Speed, Negative
Serial data lane diff inputs from pads, Low Power and High Speed			
dlp_p_i	-	In	Data lanes Low Power, Positive
dlp_n_i	-	In	Data lanes Low Power, Negative
dhs_p_i	-	In	Data lanes High Speed, Positive
dhs_n_i	-	In	Data lanes High Speed, Negative

Port	Range	Dir	Description
Status, rx_byte_clk_o domain			
dt_err_o	-	Out	CSI-2 data type is not supported by the receiver
ecc_ok_o	-	Out	Successful ECC check of received packet header
ecc_ecc_err_o	-	Out	Unsuccessful ECC check of received packet header. The error was in the ECC byte itself, not in the data bytes
ecc_corr_err_o	-	Out	Unsuccessful ECC check of received packet header. The error was corrected
ecc_err_o	-	Out	Unsuccessful ECC check of received packet header. The error could not be corrected
cs_ok_o	-	Out	Successful checksum check of received payload data
cs_err_o	-	Out	Pulses if unsuccessful checksum check of received payload data
Test/debug, rx_byte_clk_o domain			
clp_o	-	Out	Current logical level on clp_p_i resp clp_n_i.
clp_active_o	-	Out	Activity (toggling) on clp_p_i resp_clp_n_i
chs_active_o	-	Out	Activity (toggling) on chs_p_i/ chs_n_i
dlp_active_o	-	Out	Activity (toggling) on dlp_p_i resp_dlp_n_i for the individual data lanes
dhs_active_o	-	Out	Activity (toggling) on dhs_p_i resp_dhs_n_i for the individual data lanes
clk_err_protviol_stop_o	-	Out	Data lane stop state protocol violation is detected
Test/debug, rx_byte_clk_o domain			
cs_insert_err_i	-	In	Inserts packet payload checksum error for testing
ignore_ecc_err_i	-	In	Ignores packet header ECC errors and receives packets even if ECC check was unsuccessful. Useful when debugging prototypes with poor signal integrity
wc_force_i	-	In	Ignores the word count information in the received packet headers and uses the value from wc_force_i instead. Useful when ignore_ecc_err_i is 1, and if expected word count is known
wc_force_en_i	-	In	See wc_force_i
dlp_o	-	Out	Current logical level on dlp_p_i resp dlp_n_i for the individual data lanes
err_protviol_stop_o	-	Out	Data lane stop state protocol violation is detected

5. Supported CSI-2 Data Types

The following tables lists supported combinations of number of data lanes and CSI-2 data types. The Lanes column shows the number of data lanes selected with the `num_datalanes_i` input port of BitCSI2Rx. The Bits column refers to bit numbers in the `m_axis_video_tdata` output port of BitCsi2Rx.

Code	Data Type	Lanes	Bits	Purpose
0x10	Generic 8-bit Null	1	7 0	Data
		2	15:0	Data
		3	23:0	Data
		4	31:0	Data
0x11	Generic 8-bit Blanking Data	1	7:0	Data
		2	15:0	Data
		3	23 0	Data
		4	31:0	Data
0x1E	YUV422 8-bit	2	7 0	Y0/Y1
			15:8	U0/V0
0x1F	YUV422 10-bit	2	9:0	Y0/Y1
			19:10	U0/V0
0x24	RGB888	3	7:0	G
			15:8	B
			23:16	R
0x2A	RAW8 (usually used for Bayer data)	1	7:0	Data
		2	15:0	Data
		3	23:0	Data
		4	31:0	Data
0x2B	RAW10	1	9:0	Data
		2	19:0	Data
		4	39:0	Data
0x2C	RAW12	1	11:0	Data
		2	23:0	Data
		4	47:0	Data
0x2D	RAW14	1	13:0	Data
		2	27:0	Data
		4	55:0	Data (2 values)
0x2E	RAW16	1	15:0	Data
		2	31:0	Data (2 values)
		4	63:0	Data (4 values)
0x30	User Defined 8-bit Data Type 1	1	7:0	Data
0x31	User Defined 8-bit Data Type 2	2	15:0	Data
0x32	User Defined 8-bit Data Type 3	3	23:0	Data
0x33	User Defined 8-bit Data Type 4	4	31:0	Data
0x34	User Defined 8-bit Data Type 5			
0x35	User Defined 8-bit Data Type 6			
0x36	User Defined 8-bit Data Type 7			
0x37	User Defined 8-bit Data Type 8			

Example:
When the desired data type is 0x2A RAW8, then one, two, three or four data lanes are supported. If one data lane is used, the data is carried on bits 7:0 on the output from BitCSI2Rx. If two lanes are used, the data is carried on bits 15:0. The least significant byte (7:0) carries the first image byte, and 15:8 carries the second.

When the desired data type is 0x1E YUV422 8-bit, two data lanes can be used, but not one, three or four. On the output from CSI2Rx; bits 7:0 carries Y0 (luminance sample of first pixel), and bits 15:8 carries U0 (blue chrominance sample of first pixel). In the next clock cycle; bits 7:0 carries Y1 (luminance sample of second pixel), and bits 15:8 carries V0 (chrominance sample of first pixel). This sequence is repeated for the following pixels.

The IP supports dynamic change of data type meaning that the active part of the data width may be smaller than the maximum allowed data width. The active part is placed at the MSB end of the component and unused bits are padded with zero. See User's guide for details.

6. Performance

Performance of the IP is dependent of the technology used for implementation and the user configuration of the IP.

Figure 3 shows typical resource utilization values collected from implementations for an Artix 7 FPGA from Xilinx.

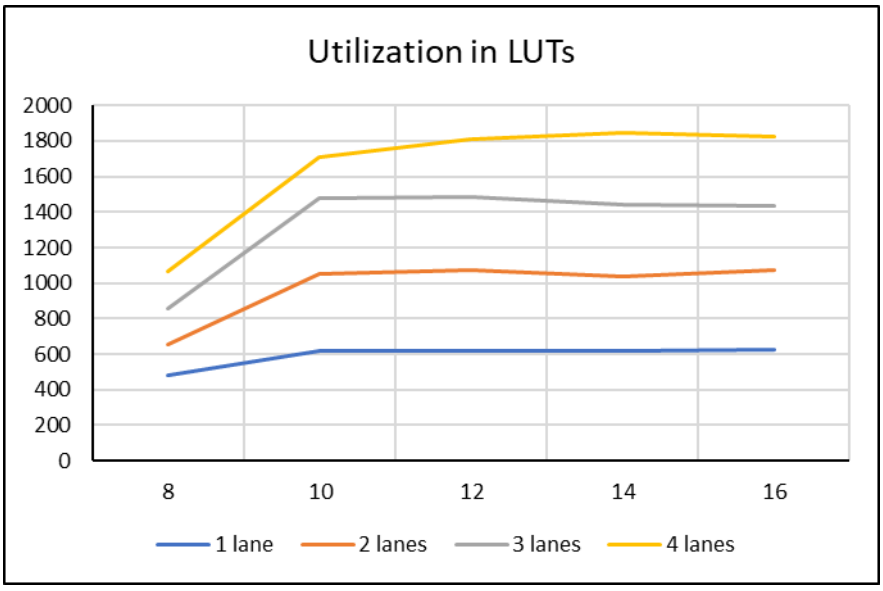


Figure 3 Resource utilization as a function of data width and number of lanes

The IP will be able to handle at least 800 Mbit/s per lane in most technologies. Please contact BitSim for more details.

Use of Information

Information in this document is provided solely to enable system and software implementers to use BitSim products. There are no expressed or implied copyright licenses granted hereunder to design or program devices or design or fabricate any integrated circuits based on the information in this document.
BitSim reserves the right to make changes without further notice to any products herein.
BitSim makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does BitSim assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages.
All operating parameters, including “Typical”, must be validated for each customer application by customer’s technical experts.
BitSim does not convey any license under its patent rights nor the rights of others.
BitSim products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the BitSim product could create a situation where personal injury or death may occur. Should Buyer purchase or use BitSim products for any such unintended or unauthorized application, Buyer shall indemnify and hold BitSim and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that BitSim was negligent regarding the design or manufacture of the part.

BitSim™ and the BitSim logo are trademarks of BitSim AB
© BitSim AB 2020. All rights reserved