GigE Vision Reference Design

Document Revision A-0.2.3, 2009-05-26

Sensor to Image GmbH

Lechtorstr. 20
D-86956 Schongau
Germany

http://www.sensor-to-image.de

Contents

Terms and Abbreviations....................3

Overview....................4

Structure of the Design....................4

Basic Operation....................5

Delivered Archive....................5

Hardware....................6

Main Board....................6

Expansion Board....................7

Connectors....................7

FPGA....................8

Firmware....................9

Makefile....................9

Cygwin Environment and GNU Make....................9

Makefile Targets....................10

Bootloader....................10

Commands....................11

Boot Flash Memory....................12

Parameters EEPROM....................14

Standalone Operation....................15

User Application and GigE Library....................16

GigE Vision Registers....................17

Remote Access to the SPI Flash Memory....................17

Remote Access to the Configuration EEPROM....................18

Video Processor....................20

CPU Interface....................21

GCSR Register Bits....................22

Padding Register Bits....................22

Revision History....................24

Terms and Abbreviations

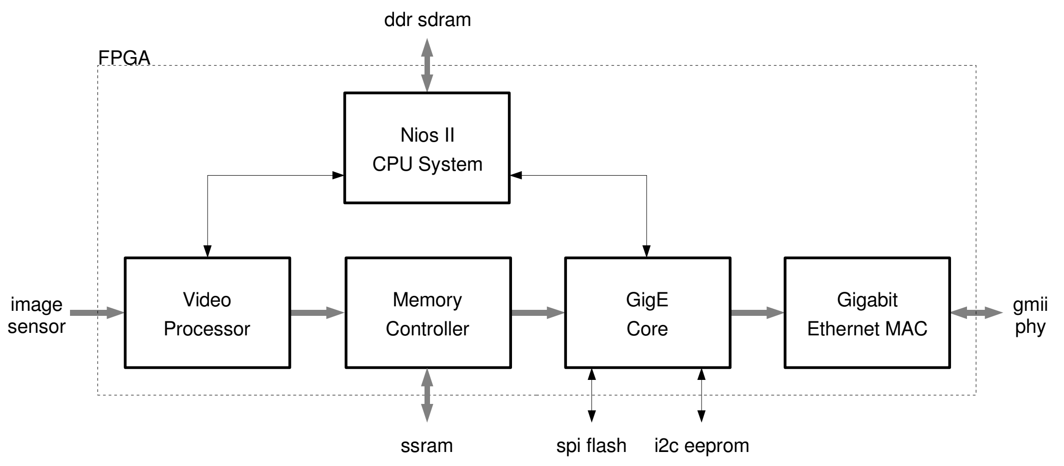
AOI	Area of Interest.
DDR	Double Data Rate interface. Signals of this interface might change its state on both rising and falling edges of a clock.
GVCP	GigE Vision Control Protocol. See the GigE Vision Specification for details.
GVSP	GigE Vision Streaming Protocol. See the GigE Vision Specification for details.
HSMC	High Speed Mezzanine Card.
PoE	Power over Ethernet.
SDR	Single Data Rate interface. Signals of this interface may change its state either on rising or on falling edge of a clock but not on both.
SDRAM	In this document it always means SDR Synchronous Dynamic RAM unless there is explicitly stated DDR or DDR2.
SSRAM	Synchronous Static RAM.

Overview

The GigE Vision Reference Design is the official system for evaluation of the whole set of GigE IP cores. The reference design is based on the Altera Cyclone III FPGA Starter Kit and the CANCam-GigE/HSMC expansion board. The design is done in Quartus II version 9.0 and corresponding SOPC Builder.

Structure of the Design

Figure 1. Block diagram of the reference design



The top level VHDL source file gvrd.vhd instantiates all the modules required to form fully functional GigE Vision camera. The design consists of following modules:

- Processor system cpu.sopc. This is an SOPC Builder generated processor system based on Nios II CPU and Avalon bus peripherals for interfacing the CPU to an external code memory and the rest of the reference design.
- Video processor module video.vhd. This modules provides set of basic registers required for a GigE Vision camera. Additionally it converts video data stream from the image sensor to the format required by the memory controller.
- GigE memory controller memory.qxp. This module operates as a special video frame-buffer which forms data packets for the GigE core. It handles half of the packet resend feature of the GigE Vision streaming protocol.
- GigE core gige.qxp. This module handles all the low-level networking features to the rest of the system. It forms the GigE Vision stream channel and provides networking

interface for the CPU system. Additionally it handles half of the packet resend feature.

- Gigabit Ethernet MAC mac.vhd. This is just VHDL wrapper for the QXP netlist of the Gigabit MAC core.

Basic Operation

The reference design forms a basic GigE Vision camera. The streaming channel is handled completely in hardware and therefore it is capable to reach maximum data throughput of the Gigabit Ethernet network. The control protocol together with supporting features are handled by the Nios II CPU in software. Basic video processing features are handled partially in hardware and partially directly by the image sensor.

Delivered Archive

The reference design is delivered as an archive containing the Quartus II project and corresponding firmware. Structure of the archive is following:

./gvrd-c3-q90

Directory containing the Quartus II 9.0 project of the reference design including the SOPC Builder CPU system.

./gvrd-c3-q90/software

This directory contains source code of the firmware, static GigE library, binary file of the bootloader, and support files like Makefile, linker scripts and so on.

⚠ **In following text, the root of the extracted archive will be always indicated as . (dot).**

Hardware

The reference design is based on the Altera Cyclone III FPGA Starter Kit and the CANCam-GigE/HSMC expansion board. Basic characteristics of the HSMC expansion board are summarized in Table 1.

Table 1. Basic parameters of the CANCam-GigE/HSMC expansion board

Parameter	Value	Units
Board dimensions	83.8 x 78.7	mm
DC supply voltages	12, 3.3	V
Total power consumption	tbd	W
Ethernet interface	1000BASE-T	
Power over Ethernet	IEEE 802.3af	
Resolution of the image sensor	752 x 480	pixels
Color mode of the image sensor	Grayscale or Bayer pattern	

Main Board

The reference design is based on the Altera Cyclone III FPGA Starter Kit. The board contains following main components:

- Cyclone III EP3C25F324 FPGA
- Embedded USB-Blaster circuitry allowing download of FPGA configuration files
- Power supply
- 256 Mbit of DDR SDRAM
- 1 Mbyte of synchronous SRAM
- 16 Mbytes of Intel P30/P33 parallel flash
- 50 MHz on-board oscillator
- Six push buttons total, four user controlled
- Seven LEDs total, four user controlled
- HSMC connector
- USB Type B connector

For detailed description of the board see the documentation on the Altera website http://www.altera.com/products/devkits/altera/kit-cyc3-starter.html.

Expansion Board

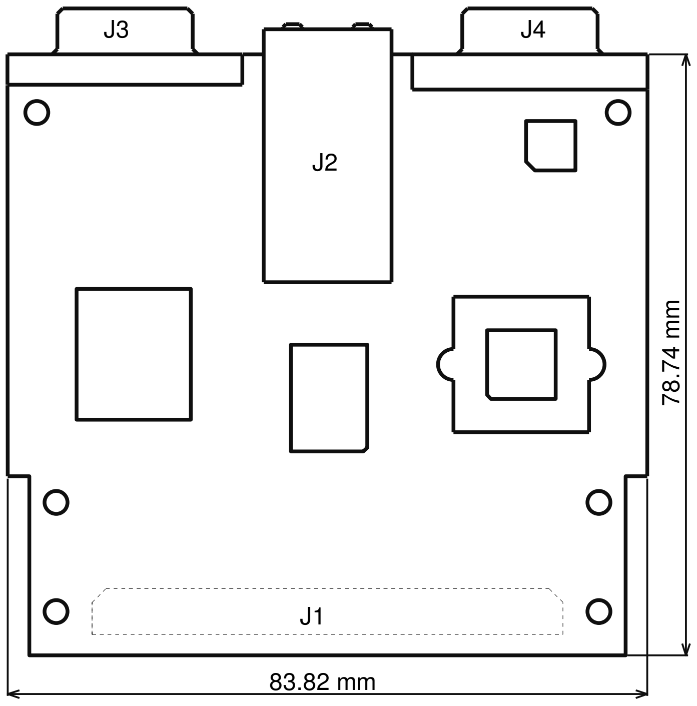
The expansion board was designed to present features of the GigE Vision IP cores and to allow their simple evaluation. The expansion board contains following components:

- PoE capable Cat.6 RJ45 Ethernet jack
- 9 pin d-sub connector for RS-232 interface
- 15 pin d-sub connector for VGA output
- HSMC connector to connect to the main board
- 8 kB EEPROM for non-volatile storage of user data
- 8 MB Intel SPI flash for code storage and IP licensing
- Gigabit Ethernet PHY device
- IEEE 802.3af compliant PoE circuitry
- VGA digital to analog converter
- Aptina MT9V022 grayscale or color CMOS image sensor 752×480 pixels

Connectors

Top-view layout of the expansion board showing position and orientation of the connectors is shown in Figure 2.

Figure 2. Layout of the expansion board



The expansion board contains four connectors. The J1 HSMC header on bottom side of the PCB is dedicated for connection to the main board. J2 is Cat.6 Ethernet jack with power over Ethernet feature. Connector J3 is 9 pin d-sub for serial console and J4 is 15 pin d-sub for VGA output.

FPGA

The reference design itself is implemented in the Cyclone III FPGA. Block diagram is shown in Figure 1. Detailed description of the IP cores delivered as QXP netlists can be found in their respective documentation. The rest of the system is delivered as VHDL source codes to show example how to use the cores to create working GigE Vision system.

The reference design consumes around 60% of the EP3C25 logic resources. Table 1 shows precise numbers of device utilization.

Table 2. Reference design FPGA resource utilization

Module	Logic Cells	Registers	M9Ks	DSPs	PLLs
GigE core	5603	3049	21	0	0
Memory controller	2197	1415	4	2	0
GMAC core	969	544	0	0	0
CPU	6167	3726	23	4	2
Video processor	397	312	2	0	0
Total including top level	15465	9123	50	6	2

Firmware

The firmware is formed of C source code files and precompiled static library implementing all the functionality required by the GigE Vision specification. Structure of the firmware directory ./gvrd-c3-q90/software/ is following:

Makefile	GNU make control file for auxiliary project related tasks.
common/	This directory contains the GigE library libgige.a, backup of the bootloader file boot.hex, and device description XML file gvrd_v10.xml.
gvrd/	The control application for the Nios II IDE.
gvrd_syslib/	System description library for the gvrd application.
scripts/	Auxiliary Perl scripts are placed into this directory.

Makefile

Additional task that are not handled by the Quartus II or Nios II IDE can be controlled using the GNU make tool. Following section is for Windows users not familiar with UNIX shell and make utility. Users running Altera development tools under GNU/Linux operating system are usually used to use these tools.

Cygwin Environment and GNU Make

All the UNIX-like tools required for successful generation of the executable and bitstream are part of the Altera development environment under Windows operating systems. To enter the command line run the Nios II Command Shell from the Altera/Nios II EDS start menu. Now at the command line change current working directory to the firmware directory. The cygwin environment maps Windows disks to UNIX filesystem so that to change current working directory to <ref-design-path>\software on disk D: you must type following command:

```
cd /cygdrive/d/<ref-design-path>/software
```

When you are in the firmware directory, you can control build process of the auxiliary files issuing following command:

```
make <target>
```

where the <target> is one of the Makefile targets described in following section.

Makefile Targets

The Makefile offers following targets to control tasks of the build process:

bin	Creates raw binary file gvrd.bin from the ELF file and displays its length and checksum in hexadecimal.
eeprom	Generates binary image of the configuration EEPROM and displays its length and checksum in hex. The image can is used to set default parameters of the GigE device.
clean	Deletes all generated files.

Bootloader

The bootloader is delivered as a HEX file together with the firmware. It is located in the Quartus II project directory as boot.hex file and its copy is kept under the software/common directory. Its contents is used to fill the on-chip boot RAM of the Nios II based CPU system during project compilation in Quartus II. Basic function of the bootloader is to load the firmware from SPI flash into program memory and start it. Additionally it allows to upload binary files from a PC using serial line and program these files into the flash memory or parameters EEPROM.

The bootloader communicates with the PC using serial line set to following parameters:

Speed:	115200 Bd
Data bits:	8
Parity:	none
Stop bits:	1
Flow control:	none

After successful configuration of the FPGA using JTAG or from SPI flash the bootloader displays some version information and waits for 3 seconds printing out dots. When no key is pressed during this period, the bootloader tries to load a firmware from flash and if it is successful, it starts it.

When a key is pressed during the initial timeout period or when there is an error while loading the firmware from flash, the bootloader enters its command line. User has a full access to all the features of the bootloader from the command line.

The bootloader signalizes its current status using two general-purpose outputs. These can be used by a custom hardware. States of the bootloader are summarized in Table 3.

Table 3. Bootloader status

sys_gpo[1]	sys_gpo[0]	State
0	0	The bootloader is working
0	1	Boot failure, bootloader is entering command line
1	0	Bootloader is passing control over to the application
1	1	Undefined state

Commands

Command line mode of the bootloader allows user full control over the bootloader features. The commands are entered pressing one key only. The command interpreter is case-insensitive. List of available bootloader commands is shown in Table 4.

Table 4. Bootloader commands

Key	Command	Description
U	Upload	Upload binary file using Xmodem protocol over a serial line to the program memory
D	Download[1]	Download content of the program memory using serial line
L	Load	Load data from flash into program memory
S	Save	Save data from program memory into flash
R	Run	Execute program from the program memory
N	Number	Read serial number of the flash memory
E	Evaluation	Read contents of the SPI GCSR register to check status of the evaluation control (serial number check and evaluation period)

The Load and Save commands expect additional argument at the command line. These additional arguments are listed in Table 5. The EEPROM command needs confirmation – pressing the Y key as "yes" – before rewriting contents of the parameters EEPROM.

Table 5. Additional arguments of the Load and Save bootloader commands

Key	Command	Description
B	Bitstream	The command operates with the FPGA bitstream
S	Secondary Bitstream	Load or save secondary (backup) FPGA bitstream
X	XML File	Load or save an XML camera description file
A	Application	The command operates with the firmware application
E	EEPROM	Load or save contents of the parameters EEPROM

The most important commands for system operation are Upload, Load, and Save.

Upload After issuing the upload command bootloader periodically tries to start the

1 The Download command is disabled in current version of the bootloader.

Xmodem file transfer over a serial line. Whenever the sending side (a PC) starts sending data, the bootloader stores it into the program memory. After end of the transfer the bootloader prints out length of the received file and its checksum. These values might be checked on a PC using the software/scripts/cksum.pl Perl script.

Load This command expects one parameter and then it loads corresponding section of the flash memory or EEPROM into the program memory and prints out its length and checksum.

Save This command expects one parameter and according to it then stores the current contents of the program memory into the flash memory or EEPROM.

Boot Flash Memory

The bootloader uses an SPI flash memory for storage of the FPGA bitstreams, firmware, and device description XML file. Memory map of the flash memory is shown in Table 6.

⚠ **The reference design based on the Cyclone III Starter Kit does not use the FPGA bitstream stored in the boot SPI flash memory. Full description is kept for compatibility with other reference designs. The Cyclone III Starter Kit based reference design configures the FPGA from the parallel flash assembled on the main board.**

Table 6. Flash memory map

0x7F FFFF – 0x40 0000	Application	4 MB
0x3F FFFF – 0x30 0000	Reserved	1 MB
0x2F FFFF – 0x21 0000	XML File	960 kB
0x20 FFFF – 0x20 0000	Allocation Table	64 kB
0x1F FFFF – 0x10 0000	Secondary Bitstream	1 MB
0x0F FFFF – 0x00 0000	Bitstream	1 MB

There is no special filesystem implemented in the flash memory. Its structure is rather fixed to keep the design simple. After power up the FPGA loads its configuration starting from address zero which should be a bitstream. The secondary bitstream is a backup when there would be a problem configuring the FPGA using the first bitstream.

The allocation table contains information about data stored in each section. It holds the length of the section in bytes and its checksum which is used during boot process to check data integrity. Table 7 shows occupation of the allocation table.

Table 7. Layout of the flash memory allocation table

0x20 FFFF : 0x20 0030	Reserved
0x20 002C	Application checksum
0x20 0028	Application length
0x20 0024	Reserved
0x20 0020	Reserved
0x20 001C	XML file checksum
0x20 0018	XML file length
0x20 0014	Reserved
0x20 0010	Reserved
0x20 000C	Secondary bitstream checksum
0x20 0008	Secondary bitstream length
0x20 0004	Bitstream checksum
0x20 0000	Bitstream length

All values in the allocation table are 32 bit unsigned integer numbers. The length items mean number of used bytes within a corresponding section. The checksums are calculated as 32 bit sum of all used bytes of a section.

Parameters EEPROM

The I^2C EEPROM is used for non-volatile storage of constants and parameters required by the GigE Vision device. Detailed description of the EEPROM operation including its memory map can be found in the GigE Vision IP Specification.

There is a Perl script eeprom.pl in the scripts directory provided to allow user to prepare his own default settings of the parameters stored in the EEPROM. This script expects several parameters on its command line. These parameters are then used to assemble a binary image of the EPROM contents. The script can be executed from command line calling perl scripts/eeprom.pl [optional_parameters] -x <xml-file> -o <output-file>. Table 8 shows command line options of the eeprom.pl script.

Table 8. Command line options of the eeprom.pl script

Option	Value	Description
-m (--mac)	hh:hh:hh:hh:hh:hh	Ethernet MAC address of the device
-c (--ipcfg)	d8	Enabled IP configuration methods, allowed values are 1 for static, 2 for DHCP, 4 for LLA, and all their "or" combinations
-i (--ip)	d8.d8.d8.d8	Static IP address of the device
-n (--net)	d8.d8.d8.d8	Static IP network mask
-g (--gw)	d8.d8.d8.d8	Static IP default gateway
-p (--port)	d16	GVCP port number
-u (--uname)	text	User defined name, maximum length is 15 characters
-d (--dest)	d8.d8.d8.d8	Destination IP address for bidirectional version
-s (--stream)	d16	GVSP port number for bidirectional version
-e (--serial)	text	Serial number/ID of the device
-t (--text)		Force the script to generate C source file containing initialized array instead of EEPROM binary image
-x (--xml)	text	Path to existing device description XML file
-o (--output)	text	Name of the output C source or binary image file

The command line arguments are expected in form "option value". The -x and -o options are mandatory and must be always present. The other options are optional and predefined default values will be used when options will be omitted.

The values in Table 8 use simple symbols to represent actual parameters. The h represents a single hexadecimal digit, d8 means 8-bit unsigned decimal number, d16 stands for 16-bit unsigned decimal number, and text means a text string. When it is necessary to use a whitespace character within a text, enclose the whole text string into quotation marks.

Standalone Operation

To make the whole system run standalone, it is necessary to follow these steps:

- After any change in hardware generate up-to-date FPGA programming file in the Quartus II.
- Generate the firmware application binary. Build the application inside the Nios II IDE and create required binary file calling make bin from the software directory. Generated file gvrd.bin will be placed into the software/common directory.
- Connect JTAG cable and power supply to the reference board and switch the supply on.
- Start your favorite serial terminal program and setup its serial line parameters according to the Bootloader section of this chapter.
- Configure the FPGA from the Quartus II (program device). In the serial terminal

window press any key after bootloader reports its version and starts printing dots.

- In bootloader press the U key to upload the bitstream binary file to the program memory. The bootloader tries to start Xmodem file transfer now. Use a send file using Xmodem protocol feature of you serial terminal program to send the gvrd.rbf file from the Quartus II project directory to the device. After the end of file is detected, the bootloader reports size and checksum of the received file. You can check it comparing with a result of the perl scripts/cksum.pl -x ../bitstream.bin command.
- Now you can save the bitstream to the flash memory pressing the S key followed by the B key what means “save bitstream”.
- Upload the application binary the same way as the bitstream. Press the U key to call the upload command, send using Xmodem protocol the commnon/gvrd.bin and check its length and checksum according to result of the cksum.pl script.
- Save the application into the flash pressing the S key followed by the A key what means “save application”.
- Generate your default configuration EEPROM image calling perl scripts/eeprom.pl [options] -x common/gvision_v10.xml common/eeprom.bin or simply make eeprom. The make command uses default parameters entered into the makefile.
- Upload the EEPROM image pressing the U key and sending the common/eeprom.bin file using Xmodem protocol.
- Save the EEPROM contents pressing the S key followed by the E key what means “save eeprom”. Be patient, programming the whole EEPROM takes approximately 20 seconds.
- Upload the XML file using the U key and save it into flash memory using the S key followed by the X key what stands for “save XML”.
- If everything passed smoothly, you can switch the power off and back on. You should see the bootloader information in the serial terminal, then several dots while waiting for boot, and finally it should report boot information and start the application.

User Application and GigE Library

The sample application is delivered as C source codes with the GigE static library. The sources are placed in the software/gvrd directory. The user code is separated from the GigE Vision related code. The user application does not need to deal with the GigE Vision protocol. Instead it uses services of the GigE library. The library can be found in software/common/libgige.a.

The GigE library handles all the networking functionality of the reference design except the GigE Vision Streaming Protocol. The GigE static library has been compiled using Nios 2 GCC compiler that is part of the Nios 2 Embedded Design Suite 9.0.

Detailed description of the library can be found in the GigE Vision IP Specification.

GigE Vision Registers

From the perspective of the GigE Vision application, the control of the GigE Vision device is done using set of registers. Table 9 shows the register address map as it is implemented in the reference design.

Table 9. GigE Vision control register address space

0xFFFF FFFF 0xFF80 0000	Access to Boot SPI Flash Memory	8 MB
0xFF7F FFFF 0xFF7F E000	Configuration EEPROM	8 kB
0xFF7F DFFF 0x0000 A000	Reference Design Control Registers	4087.953 MB
0x0000 9FFF 0x0000 0000	GigE Vision Bootstrap Registers	40 kB

The GigE Vision register address space is separated into four regions. The GigE Vision Bootstrap Registers region is specified in the GigE Vision Specification. See the specification for detailed information. The address space starting at address 0x0000A000 is dedicated to manufacturer-specific register space. This region is divided into three sub-regions in the reference design. One part is related to the reference design control registers while the top of the address space is dedicated to accessing the configuration EEPROM and boot SPI flash memories. Address map of the reference design control registers is described in following chapter.

Remote Access to the SPI Flash Memory

The reference design implements a way how to access the SPI flash memory remotely. This is useful when it is necessary to update firmware of a device which is connected to a network but is not directly reachable and therefore it is not possible to use bootloader as described above. Remote access to the SPI flash memory is provided using last 8 MB of the GigE Vision manufacturer-specific register address space.

Reading contents of the SPI flash memory is straightforward. The GigE Vision register address is directly translated into SPI memory address and four bytes starting at this translated address are read. The SPI address is

$$addr = reg - 0\text{xFF800000}$$

where the reg is address of the GigE Vision register accessed by the GigE Vision application running at a PC. This allows the GigE Vision application to access whole content of the SPI configuration/boot flash memory.

Write access to the SPI flash memory is slightly more complicated than reading. There is no straight way to implement direct random write access. Writing data into the flash is interfered by 64 kB write buffer and an address register. See the address mapping in Table 10.

Table 10. SPI flash memory write access registers

0xFFFF FFFF 0xFF80 0000	Reserved (write-protected)	8127.996 kB
0xFF81 0000	Write Address	4 B
0xFF80 FFFF 0xFF80 0000	Write Buffer	64 kB

Write accesses to the SPI flash memory are organized in 64 kB blocks. To write a data block into the flash memory, the application must fill the write buffer with 64 kB of data. If smaller number of bytes than 64 kB is required to be written, the remaining space of the buffer should be filled with 0xFF bytes. When the buffer is ready to be written, the application must write starting SPI flash memory address into the write address register. The firmware then erases 64 kB block of the flash memory and programs it using data from the write buffer.

Closer description of the boot SPI flash memory can be found above in Bootloader section of this chapter.

Remote Access to the Configuration EEPROM

The reference design allows the GigE Vision application to remotely update contents of the configuration EEPROM. It is implemented using access to a dedicated address space region from 0xFF7FE000 to 0xFF7FFFFF.

The EEPROM is accessed simply using read and write commands. The GigE Vision register address is directly translated into EEPROM address and four bytes starting at this translated address are read or written. The EEPROM address is

$$addr = reg - 0\text{xFF7FE000}$$

where the reg is address of the GigE Vision register accessed by the GigE Vision application running at a PC.

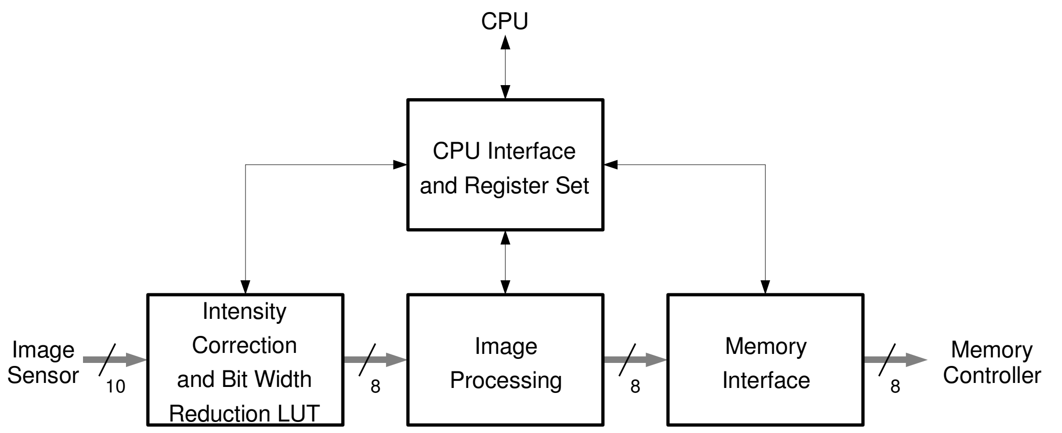
Access to the configuration EEPROM is not described in a device description XML file and therefore it is hidden for any standard GigE Vision application. This feature is supported by special software from Sensor to Image only.

The EEPROM image useful for the update is generated by the eeprom.pl script described above in this chapter.

Video Processor

The reference design comes with very simple video processor. It is the simplest image sensor video input block which might be used as a base for customer video processor. Figure 3 shows block diagram of the video processor unit.

Figure 3. Block diagram of the video processor


CPU
CPU Interface and Register Set
Image Sensor
10
Intensity Correction and Bit Width Reduction LUT
8
Image Processing
8
Memory Interface
8
Memory Controller

The basic video processor unit corrects video signal gain using a look-up table. The look-up table also converts input 10 bit pixel data width to 8 bits. The 8 bit pixel data with corrected intensity can be used for video processing and then it is sent out to the memory controller. Interfacing to the CPU and basic video register set is provided using its dedicated block.

The image processing block is empty in the reference design. When additional image processing is required, this block must be replaced by the processing algorithm. This module must use input and output signals listed in Table 11. Direction of the signals is shown from perspective of the image processing module. The width is stated in bits.

Table 11. LUT output signals useful for custom image processing

Signal	Width	Dir.	Description
lut_frame	1	in	Frame valid from the image sensor
lut_line	1	in	Line valid from the image sensor
lut_strobe	1	in	LED strobe output of the sensor
lut_data	8	in	Pixel data from the look-up table
proc_frame	1	out	Processor frame valid
proc_line	1	out	Processor line valid

Signal	Width	Dir.	Description
proc_data	8	out	Processed pixel data

CPU Interface

The CPU interface contains set of registers required for basic GigE Vision camera. Additionally it allows access to the input look-up table from the CPU software. The example firmware provides a way how to access these registers from a GigE Vision application running at a PC. It maps the video registers to the manufacturer-specific register space of the GigE Vision application allowing the application accessing them. Table 12 shows list of video registers provided by the reference design.

Table 12. Video processor CPU registers

GigE Application		Firmware		Name	Bits	Description
Address	Acc.	Offset	Acc.			
0x0000A000	r/w	0x0000	r/w	gcsr	31..0	Global video control and status register; see below for details
0x0000A004	r/w	0x0004	r/w	width	11..0	Image frame width in pixels; allowed values are from 1 to max_width
0x0000A008	r/w	0x0008	r/w	height	11..0	Image frame height in lines; allowed values are from 1 to max_height
0x0000A00C	r/w	0x000C	r/w	offs_x	11..0	Horizontal offset of the AOI; it must be in range from 0 to (max_width - 1)
0x0000A010	r/w	0x0010	r/w	offs_y	11..0	Vertical offset of the AOI; it must be in range from 0 to (max_height - 1)
0x0000A014	r	0x0014	r/w	padding	31..0	Line and frame padding in bytes; see below for details
0x0000A018	r	0x0018	r/w	pixfmt	31..0	Pixel format according to the GigE Vision Specification
0x0000A01C	r	–	–	max_width	31..0	Maximum image width of the image sensor
0x0000A020	r	–	–	max_height	31..0	Maximum image height of the image sensor
0x0000A024	r	–	–	total_bpf	31..0	Total number of data bytes transferred for each image frame
0x0000A028	r	–	–	acq_mode	31..0	Current acquisition mode; the reference design supports continuous mode (1) only

GigE Application		Firmware		Name	Bits	Description
Address	Acc.	Offset	Acc.			
0x0000A02C : 0x000197FF		–	–			Not used
0x00019800 : 0x00019FFF	r/w	–	–	lut	512 × 31..0	Video input look-up table; only first 256 dwords are used for the LUT

There are two address and access type columns in Table 12. The GigE Application columns are related to the GigE Vision application running at a PC. The Firmware columns are valid for the firmware running in the embedded CPU of the reference design. The firmware offset means relative offset of the register address according to the EPC interface base address. Actual address of a video register is then

$$address = base + offset$$

where the base is typically EPC_VIDEO_BASE defined in the system.h header file.

GCSR Register Bits

This is the global video control and status register. Description of its particular bits is shown in Table 13.

Table 13. Video gcsr register bits

Bit(s)	Name	Access	Default	Description
31	acquisition	r/w	0	Video acquisition control; setting this bit to 1 starts continuous video acquisition, resetting it to 0 stops the acquisition
30..1				Not used
0	endianness	r/w	0	LUT access data endianness; 0 means big-endian, 1 stands for little-endian

Padding Register Bits

This register sets the padding information of the GVSP leader packet. It is read-only for the GigE Vision application. Description of its particular bits is shown in Table 14.

Table 14. Video padding register bits

Bit(s)	Name	Access	Default	Description
31..16	line_pad	r/w	0x0000	Number of insignificant bytes placed at the end of each video line as padding

Bit(s)	Name	Access	Default	Description
15..0	frame_pad	r/w	0x0000	Number of insignificant bytes placed at the end of each video frame as padding

The reference design does not need line padding and therefore the line_pad is kept zero. The frame_pad value depends on number of pixels per frame. If the number of frame pixels is even, the frame padding is zero. When the number of frame pixels is odd, the frame padding is 1 byte. This behavior comes from 16 bit width of the memory controller input interface where the number of input bytes is always even.

Revision History

Date	Version	Revision
2009-04-09	A-0.1.0	Initial draft of the Altera version
2009-04-22	A-0.2.0	First public draft – updated firmware related sections
2009-04-29	A-0.2.1	Updated resource utilization table
2009-05-18	A-0.2.2	Updated boot flash memory allocation table
2009-05-26	A-0.2.3	Added description of the bootloader status outputs