JPEG Encoder IP Core Setup Guide on Altera Quartus Qsys

VISENGI

Hardware & Software Engineering

www.visengi.com

Revision History

Rev.	Date	Description
1.0	2014-11-20	Initial documentation
1.1	2015-08-06	Describe AXI interfaces' endianness. Update optional AXI4-ST output width.

Table of Contents

VISENGI

Hardware & Software Engineering

VISENGI S.L.
Address: c. Juan de Herrera, 24 - 3 D
Santander – 39002 – SPAIN
EU VAT#: ESB39736509

Web: www.visengi.com
Phone: (+34) 942 50 80 15
Email: support@visengi.com

www.visengi.com

Setting Up

The following guide describes how to set up the Altera Quartus Qsys drag'n'drop instance deliverables for **implementing VISENGI's JPEG Encoder IP core on Altera projects**.

Requirements

- Altera Quartus 14.0 or newer (Web or Subscription Edition).

List of deliverables

Two versions of the same JPEG Encoder IP core are delivered, the only difference being the bus used on the Data I/O Interface (for Pixel Input and Coded JPG Output):

- **jpeg_encoder_hw.tcl:** JPEG Encoder instance description, with one AXI4-Lite interface for configuration, and one AXI3 interface for Pixel Input and JPG Output, to include the IP core into Altera Qsys' IP library and be able to instantiate it into designs.

- **jpeg_encoder_st_hw.tcl:** JPEG Encoder instance description, with one AXI4-Lite interface for configuration, one AXI4-Stream interface for Pixel Input and another AXI4-Stream interface for JPG Output, to include the IP core into Altera Qsys' IP library and be able to instantiate it into designs.

- **jpeg_encoder_stin_hw.tcl:** JPEG Encoder instance description, with one AXI4-Lite interface for configuration, one AXI4-Stream interface for Pixel Input and one AXI3 interface for JPG Output, to include the IP core into Altera Qsys' IP library and be able to instantiate it into designs.

The AXI3 interfaces contain DMA logic to enable connecting them directly to high speed buses to external memory (or directly to a memory controller port) for data I/O.

On the other hand, AXI-4 Stream interfaces are meant usually to be connected as part of a pixel processing pipeline, for example.

More in-depth information on these interfaces follows in the next pages.

Evaluation limits

The main purpose of these files is to show how simple it is to embed a JPEG encoder into any design.

As such, please note that the .tcl files are missing the JPEG Encoder netlist itself. Hence, the design can be fully connected/implemented but not simulated, nor synthesized into an FPGA bitstream.

The licensed version of the IP core consists of the same files, but with an added netlist, allowing bitstream generation without changes to the Qsys system implemented here.

Setting up the deliverables

In order to design JPEG systems with Altera Quartus' Qsys GUI tool, the accompanying files "jpeg_encoder_hw.tcl" (AXI data interfaces), "jpeg_encoder_st_hw.tcl" (AXI4-Stream data interfaces), and "jpeg_encoder_stin_hw.tcl" (AXI and AXI4-Stream data interfaces) must be added to Altera's IP library by following these steps:

1. Locate the Altera Quartus SW installation folder (for example "C:\Program Files\Altera\14.0") .

2. Create a new sub-folder called "visengi" within Altera SW's "ip" subfolder (for example: "C:\Program Files\Altera\14.0\ip\visengi").

3. Copy the three ".tcl" files to the new "visengi" folder.

Instantiation

In order to instantiate the JPEGE IP core into a Qsys design, the following steps should be followed:

1. Open Altera Quartus SW, and then the project where the JPEGE is to be integrated.
2. Open the Qsys tool (Tools menu) and its .qsys project of choice (or create a new one).
3. In the IP Library, go to "DSP" and "Video and Image Processing"
4. The IP core will appear as "JPEG Encoder (AXI IO)" for the AXI only I/O interface, "JPEG Encoder (AXI4-ST IO)" for AXI4-Stream only I/O interfaces, and "JPEG Encoder (AXI4-ST In, AXI Out)" for AXI4-Stream pixel input and AXI data output interfaces
5. Select the preferred one and then click on the "Add" button.
6. The new instance will now appear in the design without any connections.

AXI vs AXI4-Stream Interfaces

The three JPEG Encoder blocks delivered feature the three most common interfacing possibilities of the IP core's Data I/O interface.

Firstly, let's review the interfaces that are the same in all versions:

- **Configuration Interface (S_AXI):** AXI4-Lite slave with a 32 bits interface to control all the necessary parameters of encoding through a small Configuration register set. This is a low speed interface.

- **Interrupt Interface (jpege_rdy_int_o):** A rising-edge interrupt is available, signaling when one image encoding has finished. It can also be checked through the Configuration register set.

Actually, the only interface that varies between the provided blocks is the Pixel Input and Coded JPG Output, or **Data I/O, interface**:

1) For "**JPEG Encoder (AXI IO)**" IP core name ("jpeg_encoder_hw.tcl"):

The "M_AXI" bus (used to read **input pixels** and write the coded **JPG output**) is one **AXI3** Master, with a 64 bits data width, embedded DMA engine for direct connection to a memory controller, and user-set addressing parameters through the Configuration register set. Data is in **little-endian** format.

2) For "**JPEG Encoder (AXI4-ST IO)**" IP core name ("jpeg_encoder_st_hw.tcl"):

The **Pixel Input AXI4-Stream** Slave Interface is labeled "S_AXIS_PIX_IN" and can be connected to a Video IP AXI4-Stream Interface that feeds pixels in a row-wise manner (such as from an image sensor or video input). Data width is 24 bits.

The Coded **JPG Output AXI4-Stream** Master Interface is labeled "M_AXIS_JPG_OUT". It outputs bytes in a sequential manner in **big-endian** format. If they are written one after the other incrementally into a single file, the result is a .jpg file. Data width is 64 bits.
Check VISENGI's documentation for encapsulation of .jpg files as one MJPEG .avi video file.

3) For "**JPEG Encoder (AXI4-ST In, AXI Out)**" IP core name ("jpeg_encoder_stin_hw.tcl"):

This third version features a mix of the above:

The **Pixel Input AXI4-Stream** Slave Interface is labeled "S_AXIS_PIX_IN" and can be connected to a Video IP AXI4-Stream Interface that feeds pixels in a row-wise manner (such as from an image sensor or video input). Data width is 24 bits.

The "M_AXI" bus (write only for the Coded **JPG Output**) is **AXI3** Master, with a 64 bits data width, embedded DMA engine for direct connection to a memory controller, and user-set addressing parameters through the Configuration register set. Data is in **little-endian** format.

Interfacing

The following interfacing example will feature the AXI I/O variant of the JPEG Encoder, since a complete example is possible without other IP cores like a pixel input pipeline (from a camera).

In order to allow the JPEGE IP core to maintain its constant number of pixels encoded per clock cycle, the Data I/O interface must be connected to a high speed AXI slave with direct access to memory, capable of the required pixel bandwidth.

CPU/DDR Configuration:

In this **connection example** we will suppose a **Cyclone V SX** (or Arria V SX) FPGA, that is: an FPGA with embedded ARM CPUs.

> NOTE: the JPEGE IP core is fully autonomous, the use of a CPU in this example is just for convenience. It is only employed to configure the compression parameters (frame width/height, quality, DMA addresses, etc) through the AXI4-Lite interface. However, a soft-core CPU (such as Nios II) or even a simple FSM, can be used for the same purpose (on non-ARM enabled FPGAs).

The CPU is supposed to be the "Arria V/Cyclone V Hard Processor System" already at the top of the Qsys "System Contents" tab.

Double click on it to open the Parameters window and, on the **"FPGA Interfaces"** tab:

- Under the **"AXI Bridges"** section, make sure that the "Lightweight HPS-to-FPGA interface width" is set to "32-bit". This is where the AXI4-Lite Configuration interface (S_AXI) will go.

- Under the **"FPGA-to-HPS SDRAM Interface"** section, click the "+" button to add a new interface, select type "AXI-3" and width "64". This fast-speed AXI Slave interface will be used for the "AXI Data I/O Interface (M_AXI)", since it connects directly to DDR memory.

- Under the **"Interrupts"** section, make sure that there is a mark on the checkbox labeled "Enable FPGA-to-HPS Interrupts". This is where the JPEGE Interrupt output (jpege_rdy_int_o) will connect to the CPU.

Close the Parameters window for the changes to take effect.

NOTE: On a non-ARM enabled FPGA, the S_AXI AXI4-Lite low speed bus should be connected to a soft-core CPU, or a suitable FSM, for configuration; whereas the AXI3 Data interface should be connected to a DDR Memory Controller port.

Interconnection:

Now all the necessary endpoints should be available in the Qsys "System Contents" tab. To properly connect the IP core these instructions should be followed.

To create the connections between ports/buses: click on the grayed out empty circles that make up the connections described, at the first column of the "System Contents" tab.

- The "reset_sink" Reset Input port must be connected to an active-low reset source.

  On Qsys, this could be the "clk_reset" Reset Output of a "Clock Source" block.

- The "clock_sink" Clock Input port must be connected to a rising-edge active clock source.

  On Qsys, this could be the "clk" Clock Output of a "Clock Source" block.

- The "jpege_rdy_int_o" Interrupt Sender port must be connected to a rising-edge active interrupt receiver.

  On Qsys, this could be the "f2h_irq0" Interrupt Receiver of the "Arria V/Cyclone V Hard Processor System" block, the interrupt connections are done at the far-right side column of the "System Contents" tab.

- The "S_AXI" bus (used to access the JPEGE's Configuration Registers) is a standard AXI4-Lite interface of data width 32 bits.

  On Qsys, connect it to the low speed "h2f_lw_axi_master" bus of the "Arria V/Cyclone V Hard Processor System" block. Multiple components may be sharing this bus, which is fine.

- The "M_AXI" bus (used to read input pixels and write the coded JPG file) is a standard AXI3 interface of data width 64 bits, which can be directly connected to a memory controller, since it already embeds the necessary DMA engine.

  On Qsys, for maximum performance, it is recommended to connect it to the "f2h_sdram0_data" AXI slave, which is directly connected to the CPU's shared DDR Memory Controller.

NOTE: "f2h_sdram*_data" buses all connect the FPGA to the CPU's shared DDR memory controller. However, if there are other memory controllers on the FPGA side, they may be used instead, in order to ensure the maximum throughput for both the ARM CPU and JPEGE IP core.

Address Map:

Finally, click on the "Address Map" Qsys tab and find the following intersections in the table:

- **Row "jpeg_encoder_*.S_AXI" and Column "hps_*.h2f_lw_axi_master"**: set here the assigned memory range to access the Configuration register set of the JPEGE IP core (remember that there is an added offset at run-time, which in the ARM on the SX FPGAs is 0xFF20_0000).

For example: typing "0x0001_1000" will make the address range become "0x0001_1000 – 0x0001_107f" (hence, accessible on SW at 0xFF21_1000 to 0xFF21_107F).

- **Row "hps_*.f2h_sdram0_data" and Column "jpeg_encoder_*.M_AXI"**: set here the total memory range, by typing "0x0000_0000", which will become "0x0000_0000 – 0xFFFF_FFFF". The actual address range used by the JPEGE will be set in the Configuration register set.

To finalize, go to the File menu and Save, and then click on the button at the bottom-right labeled "Generate HDL....", accept the default settings and wait for the system to be validated and generated, if there are missing connections they will be reported.

When this is done, close the window and then click on the "Finish" button.

The JPEG encoding system is ready!

For any question on how to best interface this IP core for your application,
please do not hesitate to send an email to info@visengi.com

