X_JPEG CODEC

February 28, 2000 Product Specification

300-2908 South Sheridan Way
Oakville, ON
Canada, L6J 7J8
Phone: +1 905 829 8889
Fax: +1 905 829 0888
E-mail: sales@xentec-inc.com
URL: www.xentec-inc.com

Features

- Supports Virtex and Virtex^{TM}-E devices
- 100% Baseline ISO/IEC 10918-1 JPEG compliant
- 8-bit/channel pixel depths
- Up to 4 quantization tables—programmable in RTL version
- Single clock cycle Huffman coding and decoding
- 4 Huffman tables (two AC and two DC)—programmable in RTL version
- Fully programmable MCU (Minimum Coded Unit)—programmable in both RTL and netlist versions
- Encode/Decode support (non simultaneous)
- Single clock per pixel encode and decode
- Support for up to 4-channel/component color
- Simple external interface
- Stallable design
- Hardware support for restart marker insertion
- Support for single, grey-scale component
- Four-channel Interface: Pixel In, Compressed Out, Compressed In, Pixel Out
- Internal registers interface
- Fully synchronous design
- Available as a fully functional and synthesizable VHDL or Verilog core including a test bench

AllianceCORE™ Facts

Core Specifics	
See Table 1	
Provided with Core	
Documentation	Datasheet JPEG Codec Programmer's Guide
Design File Formats	EDIF netlist, VHDL RTL source available extra
Constraints File	Jp_chip.ucf
Verification	Testbench, test vectors
Instantiation Templates	VHDL, Verilog
Reference Designs and Application Notes	JPEG Application Note
Additional Items	Demo Board
Simulation Tool Used	
ModelSim 5.3b	
Support	
Provided by Xentec, Inc.	

Applications

Typical applications for the JPEG CODEC include printers, desktop video editing, digital still cameras, surveillance systems, scanners, video conferencing, image archiving, and other consumer products.

Table 1: Core Implementation Data

Supported Family	Device Tested	CLB Slices	Clock IOBs[1]	IOBs[1]	Performance (MHz)	Xilinx Tools	Special Features
Virtex-E	V400E-8	3613	1	96	20	M2.1i	Block RAM, Select ROM
Virtex	V400-6	3618	1	96	18	M2.1i	Block RAM, Select ROM

Notes:
1. Assuming all core I/Os are routed off-chip

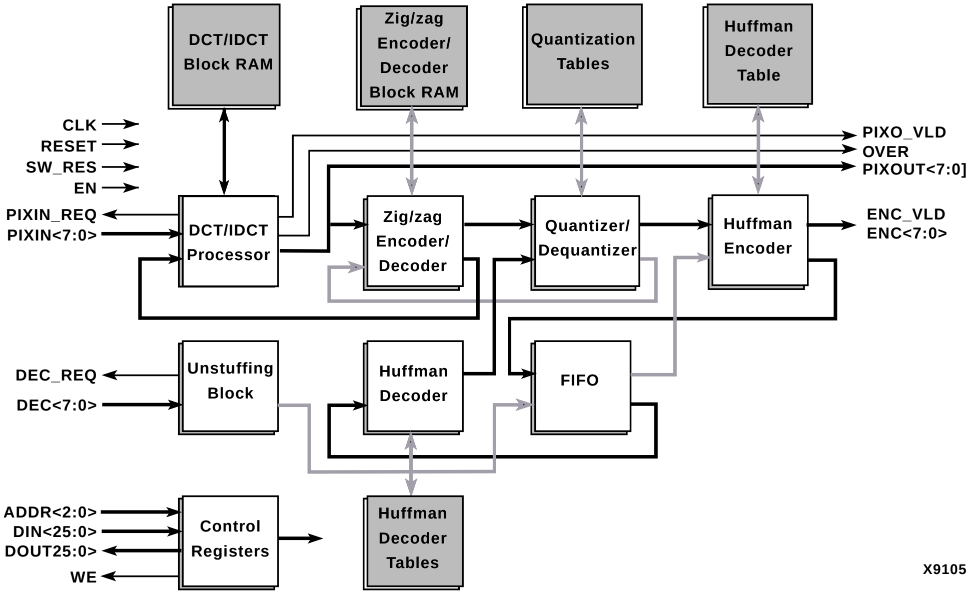

Figure 1: X-JPEG Block Diagram

General Description

This core is a fully ISO/IEC 10918-1 compliant implementation of the JPEG baseline algorithm. The simplicity of the design allows for high operational speed and makes it ideal for multimedia and color printing applications. It offers high performance and many features to meet your multimedia, digital video and digital printing applications.

Functional Description

The core implements all the steps necessary to encode image data and decode Entropy Coded Segments (ECS) data according to the JPEG baseline algorithm as specified in the ISO/IEC 10918-1 standard. The simplicity of the design allows for high operational speed and makes it ideal for multimedia and color printing applications.

The core is designed to accelerate ECS encoding and decoding, which form the most computing intensive part of the baseline JPEG algorithm.

During compression, the core accepts blocks of 8x8 pixel data and generates valid ECSs. If enabled, restart markers can also be inserted. Restart markers are the only markers that are generated by the core; parsing and generation of other JPEG markers can be performed with an additional device or in software, if a processor is part of the designed system. Please contact Xentec for more information. During decoding, the core accepts ECSs and outputs pixel data. If enabled, restart markers are recognized and acted upon.

The core supports up to four color components (with associated quantization tables) and up to two DC and two AC Huffman tables.

As shown in Figure 1, some blocks are shared by both the encoding and decoding processes; however, because processing never happens at the same time, there is no conflict.

Control Registers

The core is fully controlled and programmed through the Register Programming. Data specified on the DIN bus is written synchronously in the register indicated by the address ADDR, when the write enable signal WE is high. The content of the registers indicated by the bus ADDR is read asynchronously from port DOUT.

Minimum Coding Unit (MCU)

The MCU data instructs the codec on which quantization and Huffman tables to use during the data processing for each color component. As part of the MCU description, the number of 8x8 blocks (data units) per color component can also be specified. The composition of the Minimum Coding Unit (MCU) is fully programmable and the details are available in the JPEG Codec Programmer’s Guide.

In the case of a single color component, the MCU coincides with an 8x8-pixel block. In the case of multiple color components, the MCU is composed by at least one block per component. However, in some cases, such as for color spaces like YUV in video applications, the MCU can be formed by 4

Y pixel blocks, 2 U blocks and 2 V blocks. Full MCU programmability allows to encode and decode color components with different sampling factors, such as 4Y+2U+2V, 4Y+U+V, R+G+B, C+M+Y+K, and many others.

This allows great flexibility in the interleaving the color components with different sub-sampling rates as required by different applications, such as color printing and video editing.

Stalling

The core can be stalled at any time by lowering the EN signal. This will cause the core to suspend the encoding or decoding process. Operations will resume as soon as the EN signal is set to high. Control Registers also allow the start and stop of the codec through set/reset of the least significant bit in the Register ‘0’.

The codec output can be stalled only by stalling the entire core using the EN signal. In this case, input data pixels or ECS data must be stopped as well.

Encoding Process

The encoding process compresses 8x8-pixel blocks (data units) into valid ECSs. See Figure 2.

Before starting the encoding process, a description of the MCU must be programmed in the core’s internal registers. This information will be used by the core to correctly encode the incoming pixel blocks. Please refer to the JPEG Codec Programmer’s Guide for detailed information on register and table programming.

The encoding process is started by writing any value with the least significant bit set in the register 0. The encoding process can be stopped at any time by writing any value with the least significant bit reset in the register 0. The encoding process starts on the cycle immediately after writing a valid starting value into register 0. However, the core can be started when the value of the EN signal is low. This allows the core to start and wait for the first valid pixel (indicated by EN going high) to actually begin the encoding process. As previously noted, the core is stallable from the input side, so if EN is invalid, encoding is suspended until the next valid pixel.

As mentioned before, the core can also be stopped, regardless of the value of the EN signal, by resetting the register 0. This allows aborting the encoding process at any time. Pixels belonging to a data unit (8x8 block) are expected in rows. This means that input samples must be provided in the order X_{00}, X_{01}, ..., X_{07}, X_{10}, ..., X_{70}, ..., X_{77}.

No pause is necessary between two different blocks and encoding can proceed at the rate of one pixel per clock.

The encoder expects data units in the order specified by the MCU composition. For example, in the case of JFIF images, with MCU=4Y+U+V, the core expects 4 luminance data units, followed by the U and V data units.

Each incoming pixel is level shifted before being transformed with the Discrete Cosine Transform (DCT). The latter is performed by the DCT/IDCT processor, (see Fig. 1) which uses the dual port BlockRAM (64x14 bits) to store intermediate results.

The coefficients of the transformed 8x8 block are then rearranged in zig-zag order. This is performed by the Zig-Zag Encoder/Decoder (see Fig. 1) with the help of the Zig-zag Encoder/Decoder block RAMs (Two, single ported 64x11 bits).

The rearranged coefficients are then quantized by the Quantizer/Dequantizer (see Fig. 1) according to the appropriate quantization table. Each data unit is quantized according to the color component to which it belongs . The quantization coefficients are stored in the Quantization Tables. The Huffman Encoder (see Fig. 1) then performs the last two steps of the encoding algorithm: generation of the run length/amplitude pairs and their encoding using Huffman codes.

The Huffman codes are stored in the Huffman Encoder Table. This is a single ported, 384x12 bits RAM.

The Huffman Encoder also uses a small FIFO shown in Fig.1.

The output of the Huffman Encoder is ECS data. A valid ECS byte from the port ENC is indicated by the signal ENC_VLD. The end of the encoding process is indicated by the OVER signal going high.

Decoding Process

The decoding process decompresses valid ECSs into 8X8 pixel blocks (data units). See Figure 3.

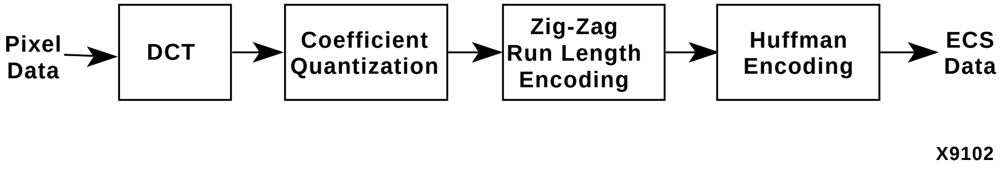

Figure 2:   JPEG Codec Encoding Process

Before starting the decoding process, a description of the MCU must be programmed in the core's internal registers, as mentioned above. This information will be used by the core to decode incoming ECS data correctly.

Please refer to the JPEG Code Programmer's Guide for detailed information on register and table programming.

The decoding process is started by writing any value with the least significant bit set in the register 0. The encoding process can be stopped at any time by writing any value with the least significant bit reset in the register 0. The decoding process starts on the cycle immediately after writing a valid starting value into register 0.

The DEC_REQ output will go high, indicating that it requires 8 bits of ECS data at the DEC input. The requested data is captured by the core at the next clock edge. A request for ECS data from the core cannot be ignored, unless the entire core is stalled by lowering the EN input.

In fact, it is possible to start the core when the value of the EN signal is low. This allows the core to start and wait for the external process to be ready (indicated by EN going high) to actually begin the decoding process.

As previously noted, the core is stallable from the input side, so if EN is low, decoding is suspended until EN is raised again.

As the core can turn the DEC_REQ signal high at any time, it is recommended that the process that feeds the core produce the next ECS data as soon as the requested one is clocked into the core. By having the next data ready, unnecessary stalling of the core will be avoided.

Each ECS data at the input of the core is passed to the Unstuffing Block (see Fig. 1). This block strips the incoming data of restart markers (in order for them to be processed by the next stage). It also detects any non-restart marker that indicates the end of ECS data and the end of decoding. The Unstuffing Block uses the same small FIFO used by the encoding process (see Fig. 1).

The Huffman Decoder (see Fig. 1) extracts run length/ amplitudes pairs from the incoming ECS data. This is achieved with the help of the Huffman Decoder Tables (a group of 3 single ported memories, 4x100 bits, 64x9 bits and 336x8 bits). The Huffman Decoder will use the appropriate AC or DC table according to the MCU description contained in the core register. This module will also expand run length/amplitude pairs into a stream of coefficients in zig-zag order.

These coefficients are then passed to the Quantizer/ Dequantizer (see Fig. 1). Each coefficient will be dequantized according to the appropriate quantization table.

The coefficients are then passed to the Zig-Zag Encoder/ Decoder (see Fig. 1), where the zig-zag order is reversed.

Finally, the Inverse Discrete Cosine Transform (IDCT) is applied on the resulting block of data by the DCT/IDCT Processor (see Fig. 1).

The IDCT output is then level shifted before it arrives at the PIXOUT output. Each valid pixel is indicated by a high PIX_VLD signal. Pixels appear at the output row by row, in the same order as expected at the input for encoding (as described above).

Provided that the code is not stalled by the EN signal, the core will produce a valid pixel per clock without any interruptions or gaps between data units.

When the core has finished decoding the PIX_VLD signal will go low and the OVER signal will pulse for one cycle.

Core Modifications

All encoding/decoding tables in this version are fixed, but can be modified upon the user's request. Please contact Xentec for further information.

Pinout

The pinout of the JPEG CODEC core has not been fixed to specific FPGA I/O, thereby allowing flexibility with a user's application. Signal names are shown in Figure 1 and described in Table 2.

Table 2: Core Signal Pinout

Signal	Signal Direction	Description
CLK	Input	Clock
RESET	Input	Reset, active high
SW_RES	Input	Soft reset, active high
EN	Input	Core enable
PIXIN_REQ	Output	Pixel input data request

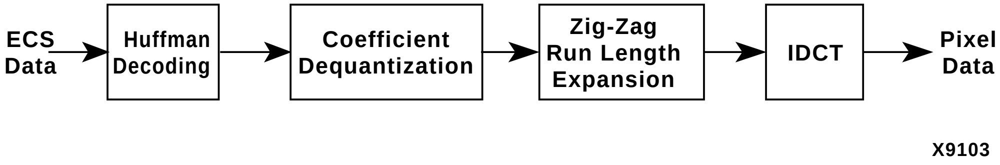

Figure 3: JPEG Codec Decoding Process

Table 2: Core Signal Pinout

Signal	Signal Direction	Description
PIXIN<7:0>	Input	Pixel input data
DEC_REQ	Output	Decoder input data request
DEC<7:0>	Output	Decoder input data
ADDR<2:0>	Input	Internal register address
DIN<25:0>	Input	Internal register input data
DOUT<25:0>	Input	Internal register output data
WE	Input	Pixel input data request
PIXO_VLD	Output	Pixel output data valid
OVER	Output	End of decoding signal
PIXOUT<7:0>	Output	Pixel output data
ENC_VLD	Output	Encoder output data valid
ENC<7:0>	Output	Encoder output data

Verification Methods

Extensive functional (pre-synthesis) and timing (post-synthesis) simulation has been performed, using the Model Technology ModelSim simulator. Simulation scenarios (including data files) and the test bench used for design verification are provided with the core.

Recommended Design Experience

Good knowledge of the ISO/IEC 10918-1 specification for the baseline algorithm, as well as its terminology, is recommended. Users should be familiar with Verilog or VHDL synthesis and simulation and Xilinx design flows as well.

Ordering Information

The X_JPEG CODEC core is provided under license by Xentec for use in Xilinx programmable logic devices. RTL synthesizable source code is also available. Please contact Xentec for information about pricing, terms and conditions of sale.

Xentec reserves the right to change any specification detailed in this document at any time without notice, and assumes no responsibility for any error in this document.

Related Information

Xilinx Programmable Logic

For information on Xilinx programmable logic or development system software, contact your local Xilinx sales office, or:

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Phone:+1 408-559-7778
Fax: +1 408-559-7114
URL: www.xilinx.com

For general Xilinx literature, contact:

Phone: +1 800-231-3386 (inside the US)
+1 408-879-5017 (outside the US)
E-mail: literature@xilinx.com

For AllianceCORE™ specific information, contact:

Phone: +1 408-879-5381
E-mail: alliancecore@xilinx.com
URL: www.xilinx.com/products/logicore/alliance/tblpart.htm