CT-JPEG04
JPEG Compression
Design Specification

December 2004

CT-JPEG04

A JPEG Compression Algorithm implementation

1.0 Introduction

This document describes Cadre Codesign (CCI) FPGA implementation for a JPEG compression algorithm based on the ISO/IEC 10918-1 standard. Dubbed CT-JPEG04, this development uses Xilinx ®'s Virtex-II FPGA as the reference platform hardware. Data is fed to the FPGA and is compressed by the CT-JPEG04 engine into a JPEG JFIF format. The core compresses data by leveraging configurable tables, i.e. quantization and Huffman tables.

The sensor interface specification was was replaced by a set of values, and other than a JFIF, no output interface were defined. CT-JPEG04 compresses incoming data of 10 pixels of 8 bits fed to the FPGA on every cycle of a 66 MHz clock and supports speeds of 500 frames per second at a resolution of 1280 x 1024. The compression ratio, depending on the quantization and Huffman tables, can vary from 0 to 100.

2.0 JPEG Architecture

2.1 Overview

This section describes in general terms, the different modules that are used in the CT-JPEG04 compression algorithm as shown in the diagram below.

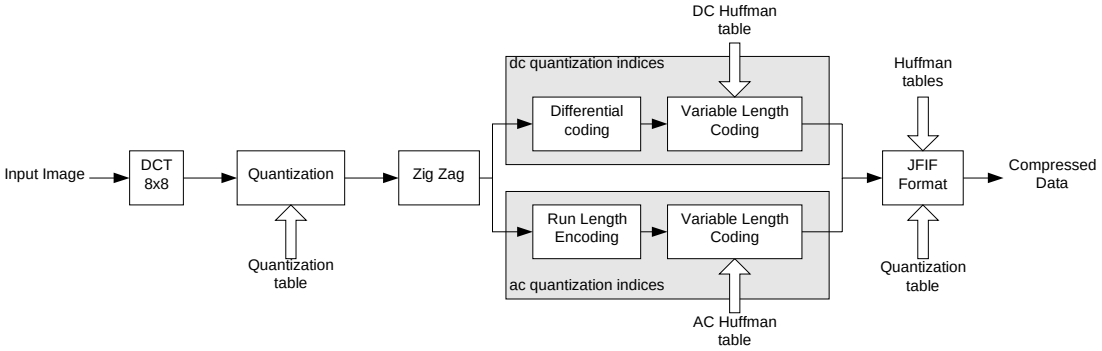

Figure 2.1.0 standard JPEG compression architecture

2.1.1 General description

The CT-JPEG04 primary function is to apply a DCT coefficients to input data. These coefficient apply one of the 64 cosine basic functions to various spatial frequencies (8 x 8 templates) to construct the original block.

Each DCT coefficient is uniformly quantized with a quantization step that is taken from a user-defined quantization table of 64, 1-byte elements. The quality and compression ratio of an encoded image can be changed by selecting q-table elements (usually by scaling up or down).

2.2 Design Implementation

The CT-JPEG04 core is composed of 9 modules (detailed below):

- Make Block
- DCT
- ZigZag
- Quantization
- DPCM RLE
- DC_EOB
- Huffman
- Reorder FIFO
- JFIF
- Local Controller

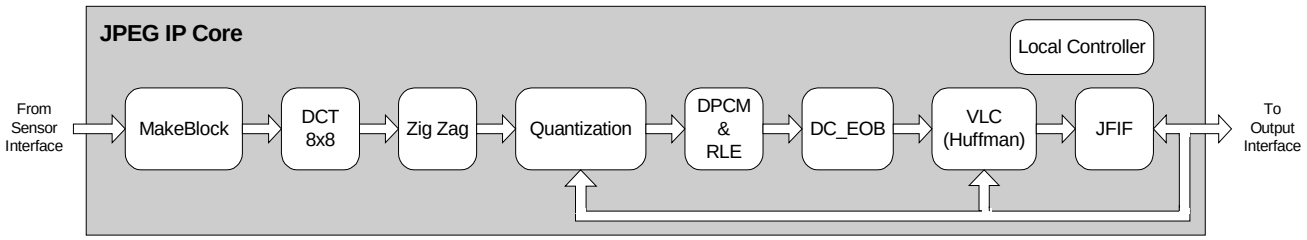

Figure 2.2.0 CT-JPEG04 core design implementation.

2.2.1 MakeBlock

The *MakeBlock* module memorizes the incoming pixels from the sensor interface, and builds 8x8 blocks required by the DCT transform. At each sensor interface clock cycle (66MHz), the *MakeBlock* takes from the incoming frame, of 1280 by 1024 pixel, line by line, 10 pixels of 8 bits. The *MakeBlock* module outputs blocks of 8x8 pixels at the rate of 8 pixels per clock period (85MHz).

2.2.2 DCT

The *DCT* module performs the 2 dimensions Discrete Cosine Transform (DCT). The *DCT* module (see: Figure 2.2.2 TDCT module architecture) is composed of 3 sub-modules: two *1-D DCT* and a *Transform* sub-modules. The *DCT* module outputs 8x8 DCT coefficients blocks at the rate eight DCT coefficients per cycle, where each DCT coefficient is 11 bits.

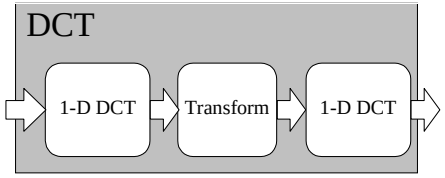

Figure 2.2.2 DCT module architecture

2.2.3 ZigZag

For each 8x8 quantized DCT coefficients block, the *ZigZag* module sorts the 64 coefficients into a single column of 64 lines according to their ascending frequency. The module, after a latency of 5 clock cycles, outputs a column at the throughput of 8 quantized DCT coefficients per clock cycle.

2.2.4 Quantization

The *Quantization* module divides each block of 8x8 DCT coefficients by a user defined quantization table of 8x8 integer. This step consists to force many DCT coefficients, especially the high frequencies ones, to a zero value. Each division result is truncated to the nearest integer value. At each clock cycle, the *Quantization* module can process 8 DCT coefficients, and produces 8 quantized DCT coefficients after one clock cycle of latency.

2.2.5 DPCM–RLE

Composed of two sub-modules, the Differential Pulse Code Modulation (*DPCM*) and the Run Length Encode (*RLE*), the *DPCM* sub-module treats only the DC coefficient while the *RLE* sub-module treats the AC coefficients.

2.2.6 DC_EOB

This module manages the eight pairs of Run Length count and AC coefficient produced at each clock cycle by the DPCM-RLE module. Depending on the free space in the FIFO memory, this module will force (or not) the highest frequency AC coefficients (by steps of eights coefficients) to zero in order to increase the free space in the FIFO, to preserve an overflow in the FIFO, and to preserve a lost of 8x8 block.

2.2.7 Huffman

Composed of three sub-modules (see: Figure 2.2.7 Huffman module): *Config*, *Code Generator, Huffman core,* the *Huffman core* sub-module produces one Huffman code and one amplitude code for each pair of Run Length count and quantized DCT coefficient. It outputs the number of valid bits for each Huffman or amplitude code, since they are defined on a variable number of bits. The Huffman are generated by the *Code Generator* sub-module, which uses the DC and AC tables are given by the user and stored in the Config sub-module.

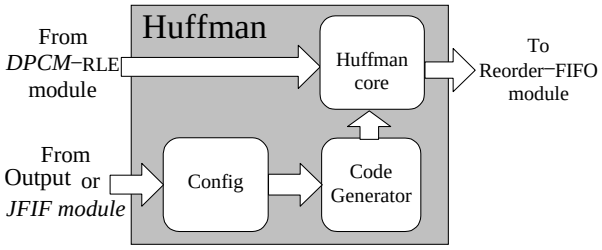

Figure 2.2.7 Huffman module

2.2.8 Reorder – FIFO

Sub divided in two sub-modules: the Reorder and the *FIFO*, this module concatenates, at each clock cycle (85MHz), 8-pairs of Huffman-amplitude codes into one long data string. Subsequently, the *Reorder* sub-module stores the long data string, in packet of 16 bits and in a chronological order, in the *FIFO* sub-module. At each clock cycle the *Reorder* sub-module can store up to 224 bits of information in the *FIFO*. The *FIFO* can memorize 262K of

information bits into 16 blocks of RAM, and can output if requested by the *JFIF* module 16 bits at each clock cycle (48MHz).

2.2.9 JFIF

The *JFIF* module sends output files, where each file corresponds to one frame, in the JFIF format through the Output interface. The file includes all the information to reconstruct the image, such as some characteristics of the image (size, color, type), the quantization table, the DC and AC tables of the *Huffman* module, and the information bits that correspond to the image.

3.0 Synthesis results

Table 1: Synthesis results[a]

Modules	Slices	BlkRam	Mult 18 x 18
MaleBlock	497	16	0
DCT	1940	0	26
ZigZag	627	0	0
Quantization	264	2	8
DC_EOB	83	0	0
DPCM&RLE	414	0	0
Huffman	668	9	0
Reorder	5560	16	0
JFIF	481	2	0
Controller	158	0	0
Total	10750	45	34

a. Synthesized with Symplify and placer and routed with Xilinx tools

