JPEG codec library based on Microblaze

1. Introduction

This is an open source JPEG codec, including both encoder and decoder, for embedded systems. It can be fully synthesized and implemented on FPGA.

Different to a fully hardware implementation, this JPEG codec is designed based on Xilinx Microblaze processor with customized hardware accelerators. It is expected to achieve high flexibility, low complexity at little cost of size and performance. We aim to archive real time motion JPEG codec on a Xilinx Spartan X3S1000 equivalent FPGA (including I/O and memory controller). **

You can open the project with Xilinx EDK7.1 or higher and synthesize by Xilinx ISE7.1. * The verification hardware platform I use is Xilinx XUP board with a Xilinx XC2V30P on it. It provides necessary peripherals such as CF card for image storage and video output. The board can be obtained at the cost of 300 euro if you are in a university. Simulation is not tested yet.

The code here includes two parts, a JPEG codec library and a test bench. The library includes both hardware and software. The test bench is to read a BMP file from CF card, drive JPEG code library to compress it and write the JPG file back to CF card. You can also make your own design to play with camera and video output based on it.

The JPEG codec library can also be used as a library or IP core for image processing and video compression applications, for instance, MPEG codec. The IP cores can be integrated immediately. It is actually part of my master project and I try to write down in detail how I design and how to use it. Enjoy!

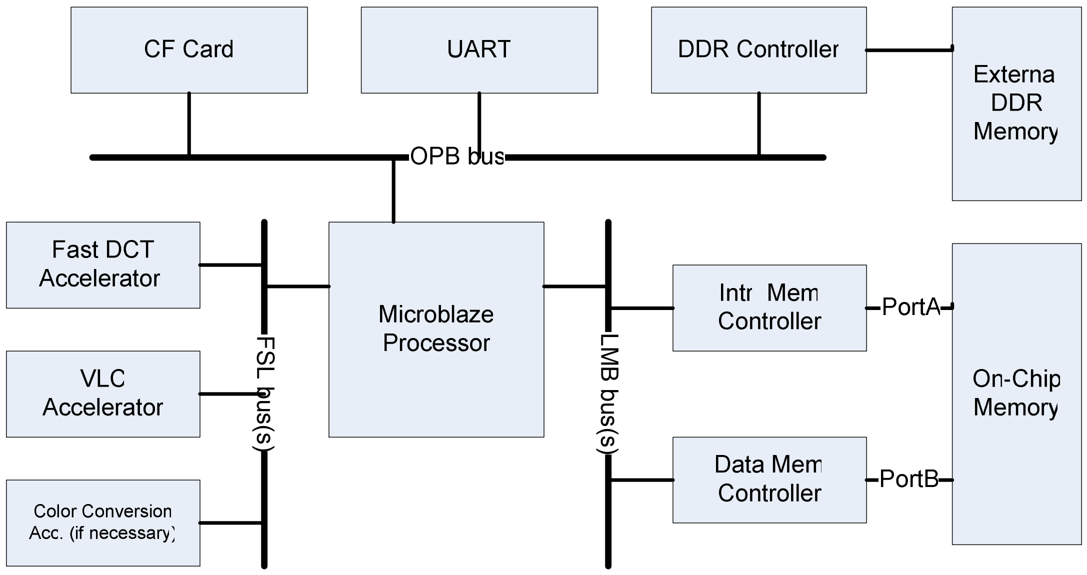
* Some intermediate version can only be open and synthesized by Xilinx EDK 8.1 and ISE 8.1, as indicated respectively.
** X3S1000: 1M Gates, 1920 CLBs, 432Kbits BRAM, Current implementation: 3460 CLBs, 589Kbits BRAM

2. Features

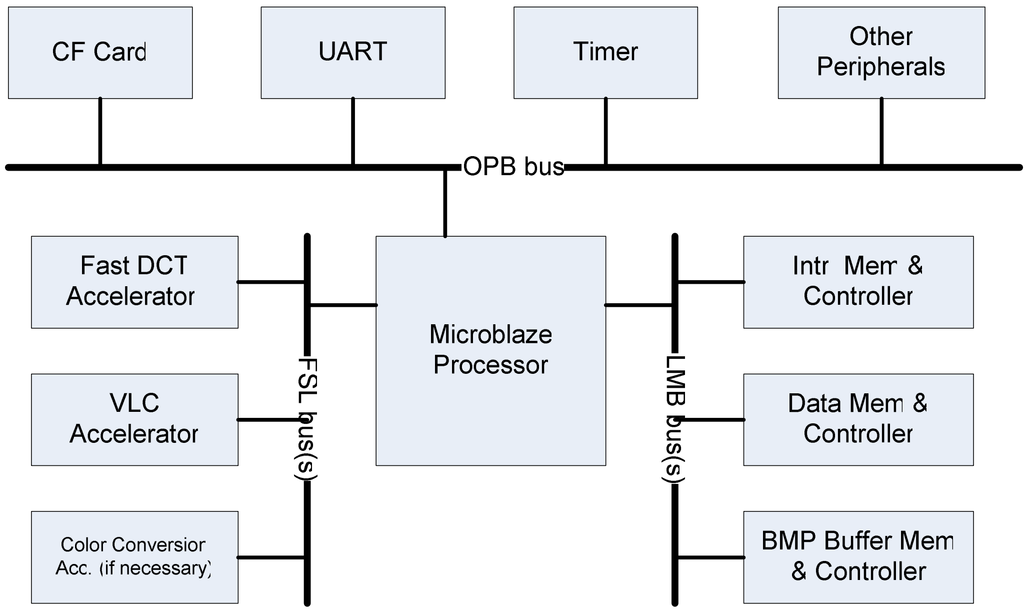
1) Baseline JPEG encoder and decoder, compatible to JPEG standard
2) RGB to YUV conversion, 4:2:0 subsampling

3. Architecture

3.1 Hardware Architecture



current architecture



architecture before v0.25

The main difference between current architecture and that before v0.25 is BMP buffer memory. Now external DDR memory is used while originally on-chip BRAM is used.

Code and data except for BMP buffer is still set to on-chip memory to achieve high performance. That's also the reason that cache is not used here.

In future implementation, all data and code can be moved to external memory to reduce on-chip resource utilization.

3.2 Data Flow

1) For whole BMP File from CF Card Controller -> OPB Bus -> Microblaze -> OPB Bus -> External DDR Memory.

The whole BMP file is read from CF card to BMP buffer memory. It is 246MB external DDR memory.

2) For every BMP macroblock from External Memory -> DLMB Bus ->Microblaze->DLMB Bus -> Data Memory

Every BMP macroblock is then read and convert to YUV color domain and stored in global arrays located in Data Memory.

3) For every YUV macroblock from Data Memory ->DLMB Bus ->Microblaze->DLMB Bus->Data Memory

It is DCT processing. Every YUV domain block is converted from spatial domain to frequency domain.

4) For every frequency domain macroblock from Data Memory->DLMB Bus -> Microblaze->OPB Bus->CF Card or ->DLMB Bus->Data Memory

Every macroblock is then quantized, zigzag scanned and Huffman encoded. The result is written into CF card. Due to file buffer in file system, not every block is written into CF card immediately.

4. Porting Guide

To port the design into other Xilinx FPGA board is not difficult. Almost all of these design files can be reused for other Xilinx FPGAs.

If you have other peripherals than CF card, for instance, camera, you need add your own peripheral and write your own code to deal with it. All of the code related to I/O is in *xupv2p.c* and all other files can be left untouched.

5. Further improvement

1) Design hardware accelerator to improve performance and reduce memory consumption.
2) Use external memory instead of on-chip memory to store instruction and data except BMP buffer.
3) Support more peripherals, for instance, camera and video output.
4) Try multiprocessor implementation and compare the result. Some of work can be found at http://www.opencores.org/projects.cgi/web/mpdma/overview.

6. For further information, please refer to

http://www.opencores.org/projects.cgi/web/mb-jpeg/overview.

Sunwei	sunwei388@gmail.com
Joris van Emden	joris.van.emden@gmail.com
Marcel Lauwerijssen	mlauwerijssen@morphcsore.com
Cristian Tena	tena.cristian@gmail.com

Appendix A Release Notes

You can checkout source code with CVS. It’s also possible to download bit file only from http://www.opencores.org/cvsweb.shtml/mb-jpeg/bitstreams/.

V0.25 2006/11/04

CVS Tag: STEP2_2c

Features

1. External Memory Support. BMP Buffer is moved to external memory and the BMP file to compress can be as large as the capacity of external memory

Code size

Text 25544 Data 5161 Bss 7892 total 38597 bytes

(CF card access, file system and I/O are all included)

Resource

Number of MULT18X18s	3 out of 136	2%
Number of RAMB16s	32 out of 136	23%
Number of SLICEs	1730 out of 13696	12%

Tool and Platform

EDK7.1, ISE7.1 and Xilinx XUP2PRO board.

V0.2 2006/09/15

CVS tag: STEP7_2

Features

1. 4:2:0 subsampling support. Compression ratio is doubled.

2. Reduce file system resource usage. For xilfatfs, CONFIG_BUFCACHE_SIZE 2560 (default 10240), CONFIG_MAXFILES 2 (default 5), CONFIG_WRITE true (default false)

Code size

Text 30920 Data 5156 Bss 13028 total 49104 bytes

(CF card access, file system and I/O are all included)

Tool and Platform

EDK8.1i2, ISE8.1i2 and Xilinx XUP2PRO board.

V0.11 2006/07/29

CVS tag: STEP2_2b

Features

The code is elaborate to reduce memory and resource usage. It is also platform independent.

Resource

Number 4 input LUTs:	2,049	out of	27,392	7%
Number of Block RAMs:	64	out of	136	47%
Number of MULT18X18s:	3	out of	136	2%

Block RAMs are used for instruction memory (16 blocks, 32KB), data memory (16 blocks, 32KB) and BMP buffer memory (32blocks, 64KB).

Code size

Text 30840 Data 5156 Bss 12276 total 48272 bytes

(CF card access, file system and I/O are all included)

Tool and Platform

EDK8.1i2, ISE8.1i2 and Xilinx XUP2PRO board.

V0.1 2006/07/19

CVS tag: STEP2_2b

Features

It includes a JPEG codec library and a testbench. It can read a BMP file (<64KB) named “image01.bmp” on CF card, compress it and write “image01.jpg” back to CF card.

Code size

Text 43284 Data 5680 Bss 32064 total 81028 bytes

(CF card access, file system and I/O are all included)

Tool and Platform

EDK8.1, ISE8.1 and Xilinx XUP2PRO board.

Appendix B Bug List

1. Non-8 multiplication image size is not supported.