| Prepared | No | |
|---|---|---|
| Krister Karlsson | 2009:001 | |
| *Doc Response/Approved* | *Date* | Rev |
| Marcus Erlandsson | 2009-08-15 | 0.02 |

# Definition of the
# H.264 Baseline Encoder Project at OpenCores

## Abstract
The purpose of this document is to define the H.264 Baseline encoder project, and to suggest how it should be carried out.

# The Proposed SoC

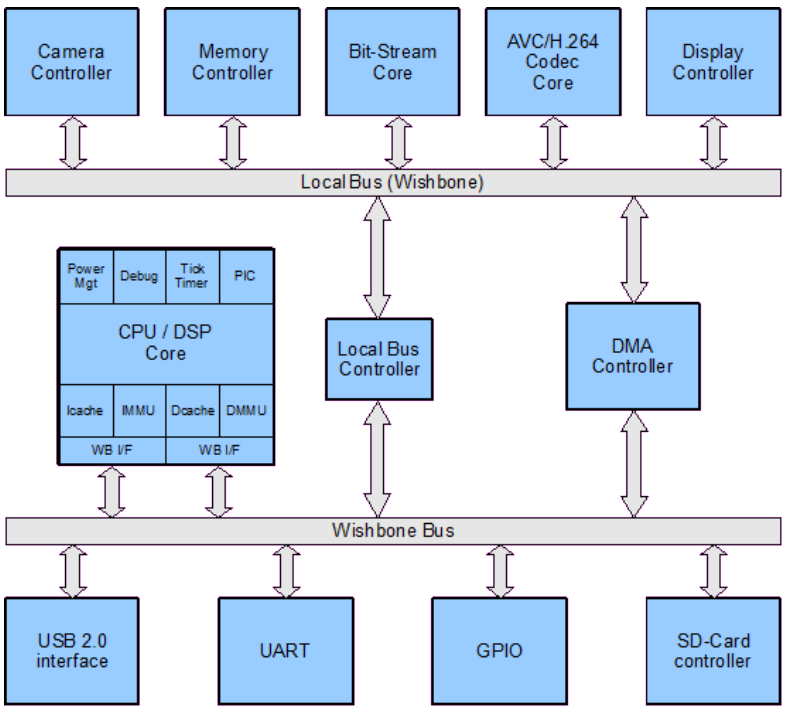
This project was described for the first time in the OpenCores Newsletter June 2009. The full article is available at http://opencores.org/?do=newsletter&2009=06#n3. Part of the message in this article is repeated here in order to state what we want to accomplish with this project.

The OC-team wants to define and start a new multimedia project pursued within the OpenCores community. The long term goal is to be able to support various multimedia standards and products by expanding our collection of media IP cores. However, we need to have a first reasonable goal.

**The first goal for the multimedia project is a 'complete' H.264 Baseline (progressive) encoder that supports both intra- and inter-prediction (I- and P-slices)**. At present time there is one project at OpenCores (the "Video compression systems" project maintained by Richard Herveille and Andy Henson) that supports H.264 Baseline encoding with intra-prediction (but not yet inter-prediction). This seems like a good starting point for the implementation of HW accelerators.

Video coding algorithms require both (complex) decision-making and intense mathematical calculations. This is especially true for the encoder. Therefore we propose a heterogeneous SoC architecture, containing a programmable RISC DSP core and HW accelerator blocks specific for video compression. The RISC core features conditional execution and branching and is therefore efficient for the decision-making. Typically the RISC core takes care of the overall system control, the user interaction, and perhaps some of the less intense video coding at the higher layers. The HW accelerator(s) take care of the video processing at the macroblock level (prediction and interpolation, transformation, quantisation, entropy coding, filtering, colour conversion etc). The heterogeneous SoC architecture attempts to combine the (re-) programming flexibility of the RISC DSP core with the ASIC-like performance and efficiency of the HW accelerators. It allows for a more precise fine-tuned matching of intense calculation and decision-making.

So, what we have in mind right now is a SoC architecture similar to the one below.




- The Camera Controller is an interface to a CCD or CMOS image sensor of a camera.
- The Memory Controller is connected to the external system memory (off-core or off-chip), typically SDRAM. The reference frame buffer is located in this memory.
- The Bit-Stream Core is very useful for a decoder. The codewords of a compressed bit-stream usually have any bit-width within a certain range. They follow in a sequence after each other and are therefore not nicely aligned to certain memory address boundaries. Finding such codewords in the bit-stream can be a painful job for a microprocessor, meaning a lot of shifting, masking, and other bit-manipulation operations. The purpose of the Bit-Stream Core is to manage the bit-stream reading and help the decoder to quickly find codewords.
  The Bit-Stream Core may also serve an encoder when writing codewords into the bit-stream.
- The first goal is that the AVC/H.264 Codec Core should be a H.264 Baseline encoder.
- The Display Controller supports the chosen display device.
- The DMA Controller manages the data traffic between off-chip and on-chip memories.
- The main CPU in the system is an OpenRISC processor (e.g. OR1200).
- The interface blocks shown should be understood as examples. Others may apply as well.

# The Proposed Way

Let us pair OpenCores with Open Source!

We think it is a good idea to start this project with a 'simple' solution. When we have this solution working and feel fairly satisfied, we move on to more optimized and refined solutions.
We propose that this project is carried out in the following three basic steps:

1. Porting of an open source H.264 Baseline encoder SW.
2. Profiling of the open source H.264 Baseline encoder SW.
3. Optimization by means of HW acceleration.

**Porting**
The first solution of the H.264 encoder is a pure SW implementation. We do this by porting an existing and available open source H.264 Baseline encoder SW to the OpenRISC processor platform. The goal at this point is to have a first working solution on our platform that performs 'reasonably' well.

There are obvious reasons for using available open source SW. First, we should no re-invent the wheel. It is unnecessary to write encoder SW of our own when it already exists open source SW that has been used by several other groups and is verified to a large extent. Second, even if it probably would be a very learning and interesting experience, writing new encoder SW from scratch is a very time consuming task. It also requires detailed knowledge of the complete encoder system. And third, using open source SW goes very well with our own OpenCores design philosophy.

For this project we suggest the SW offered by VideoLAN (http://www.videolan.org).

VideoLAN is a software project, run by volunteers, backed-up by a non-profit organisation, which produces free and open source software for multimedia, released under the GNU General Public License. Their main software is the cross-platform VLC media player.
VideoLAN is also the home of the x264 encoder project, which is used by several other software projects (amongst them MEncoder and ffmpeg).

**Profiling**
Profiling serves many purposes. First, it is a good way to get acquainted with the SW; its overall structure and what the various underlying function are doing. Second, profiling supports performance evaluation; the performance needed by various parts of the software and the performance given by the processor platform. We can measure the number of function calls and execution times and thereby identify critical algorithms (at the macroblock level). This will give us important information about which parts may need HW acceleration. Third, profiling supports the building of statistics for representative video material.

We think that the porting and the profiling of the SW can start in parallel. Profiling on a PC platform will give us relative performance numbers. When the SW has been ported to the OpenRISC platform the profiling continues there. Now, profiling on the target platform will give us absolute performance numbers.

**Optimization**
When we have a working SW implementation on the OpenRISC platform as well as the profiling results, it is time for optimization. Certain critical algorithms will be supported by HW accelerators, and the original SW for such an algorithm will be replaced by a device driver for the accelerator.
There are of course several possible architectures for a collection of accelerators, for instance one central accelerator that does 'everything', or several smaller ones. But we do not have to make any decision about this right now. We will probably discuss this allot within the OpenCores team when the time is right.