FACHHOCHSCHULE HOCHSCHULE FÜR
STUTTGART TECHNIK

UNIVERSITY OF APPLIED SCIENCES

Department of Mathematics and
Computer Science

Winter Term 2001/2002

Implementation of a JPEG Decoder for a 16 bit Microcontroller

by

Stefan Kuhr

Abstract of a Thesis Submitted

in Partial Fulfillment of the

Requirements for the Degree

MASTER OF SCIENCE

in

Software Technology

Thesis Committee:

Prof. Dr. Uwe Müßigmann, Chair
Prof. Dr. Peter Hauber
Rolf Kofink, Sony Corp.

Abstract

This thesis is a feasibility study that should examine, whether a typical 16-bit microcontroller with a graphics accelerator unit, as used for On-Screen-Displays in High-End TV sets, is capable of decoding JPEG files of commonly used file sizes and image resolutions. The prior expectations were that, for the outcome of this thesis to become part of a competitive commercial product, the time for decoding a typical image from a digital still camera should be in the range of 5 $\sim$ 10s. Yet, it was unclear whether such a performance could be achieved with the microprocessor at hand.

One part of the problem was considered market research in order to find a suitable software package that could be ported to the microcontroller. Another part was getting intimate knowledge of the well-known algorithms for computing the DCT (Discrete Cosine Transform) which is the transformation that is at the heart of JPEG encoding and decoding, in order to be able to find improvements and potential bottlenecks in the software packages to be evaluated. The third part of the problem was considered the implementation, the port to the microcontroller and performance tuning of the solution.

Very soon after the start of this thesis it turned out that there is not much of a choice for software packages that could be used for JPEG decoding. More or less all successful commercial and non-commercial products at the time of writing use a library that was written by a group of volunteers, the Independent JPEG Group (IJG). The IJG distributes its library, called JPEGLib, under very modest licensing requirements which, besides from the high portability, excellent performance, quality and stability of the code, has been key to the success of this library. Since the library is written entirely in C, portability across a wide variety of operating system environments is possible with the additional benefit of high performance by using C as a language that is very close to the actual hardware. Though the microcontroller in use has a very unusual segmented architecture, porting the JPEGLib library to the microcontroller used throughout this thesis was only a matter of 2 $\sim$ 3 weeks, since a lot of the problematic issues for segmented architectures had previously been solved in the JPEGLib for MS-DOS.

As to the DCT algorithms known today, several fast algorithms, including the fastest scaled and unscaled one-dimensional DCT algorithms were examined, as well as some historical algorithms and one fast two-dimensional algorithm which is based on the fastest scaled one-dimensional algorithm. Unfortunately, time was too short for an actual implementation of the two-dimensional algorithm but for an experienced coder it should be no problem to implement the algorithm after reading and understanding the associated chapters of this thesis.

Another problem that became apparent throughout this thesis was the necessity to scale a decoded image to the limited spatial resolution of a TV set or a monitor. The algorithms to scale down images already during the decoding process by factors of 2, 4 and 8 that are in use in the JPEGLib could be verified and suggestions could be made, where these algorithms can be improved.

In order to scale to an arbitrary resolution, the scaling process has to be performed in the spatial domain. A very simple and inaccurate implementation for this which only maps to each destination pixel its associated source pixel could be made. Unfortunately, time was too short to implement more sophisticated methods for this purpose which care about aliasing and calculate destination pixel values as weighted sums of source pixel values.

Especially for the usage of the JPEGLib in embedded systems, suggestions and actual source code implementations could be made on how to improve decoding performance by using fast on-chip memory locations.

Results

The first result that could be made was that the controller is fully capable of doing JPEG decoding for all files that typically come from digital still cameras, at least at the time of writing. Initial worries, that the limited memory space the controller can access (8 MBytes of DRAM) could not be sufficient, at least not for the common 3.3 megapixel format (2084×1536 pixels) turned out to be a non-issue, because the algorithms that scale down during decoding can be employed and require far less memory before scaling in the spatial domain has to be performed.

A very interesting result is the overall computational complexity of JPEG decoding. JPEG files that are found on the World Wide Web typically reflect more or less a file size versus quality tradeoff decision of the web designer and very often file size is the crucial criterion. Typical JPEG images found on the World Wide Web also have resolutions that are in the range of only a few ten or hundred pixels in each direction. The microcontroller used during this thesis can decode such images withing acceptable time. For digital still camera pictures however, users usually want to use the maximum resolution and the best quality that is possible. JPEG however can only achieve its extraordinary compression rates if some quality loss is accepted. Since the part of inverting the DCT during the decoding process takes a fixed amount of time ($O(1)$) for a given image resolution, the time needed to reverse the entropy encoding scheme in JPEG far outweighs the time for the inverse DCT and the color dedoding stage for such high-quality images. This leads to the assumption that entropy decoding, which is normally Huffman decoding, has a computational complexity of roughly $O(N)$ for constant image resolution and chroma subsampling, with N being the file size and thus image quality. At least this is what measuring the performance suggests. Figure 1 shows the result for one and the same image with a resolution of 2048×1536 pixels and 4:2:0 subsampling mode at different file sizes and thus different levels of quality . The DCT step that is required for this image seems to have a computational complexity of $O(N)$ as well, but this is misleading. The JPEGLib which was used to decode the images uses some sort of adaptive DCT that checks whether certain values of the DCT coefficients are zero and in that case uses a far less complex scheme to calculate the coefficients. The higher the image quality becomes, the less probable these zero values occur and the higher the need to calculate the full DCT becomes. Because there is an upper bound for the number of

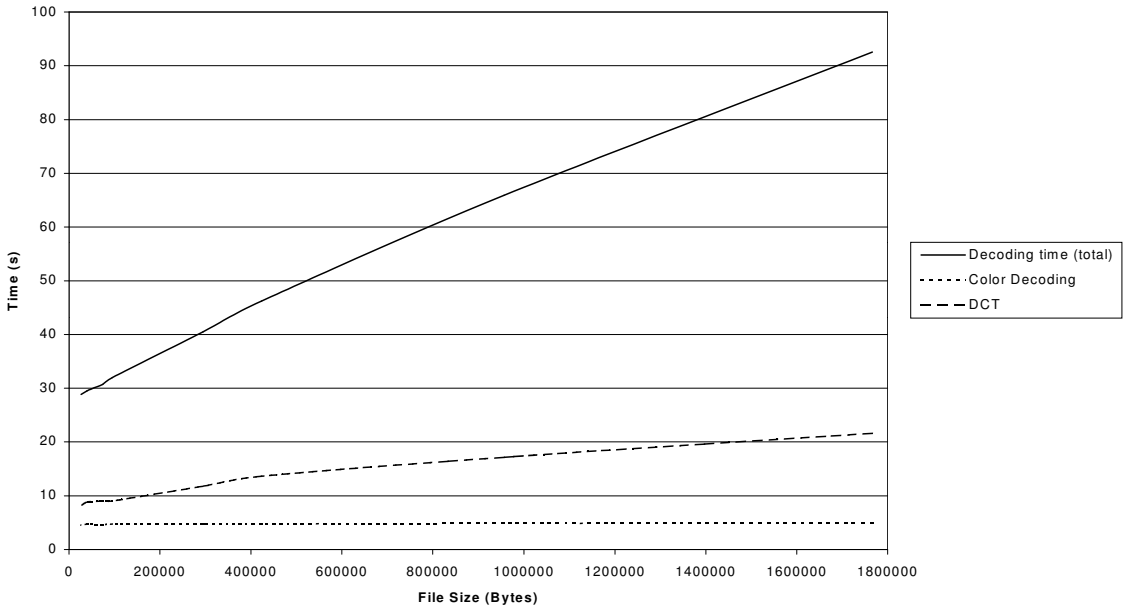

Figure 1: Decoding times for a series of images with 2048×1536 pixels resolution (4:2:0 subsampling mode, downsampled to a fourth)

blocks in the image that need to be transformed with a full DCT if image resolution and subsampling is constant, it can be safely assumed that the DCT step during decoding, much like the color decoding step, which transforms pixels from the YCbCr color space to RGB, has a computational complexity of $O(1)$.

To give an impression of the performance of the microcontroller which can be roughly categorized to have 16.5 MIPS, a JPEG image of size 1.35 MBytes with 4:2:0 chroma subsampling and a size of 2048×1536 pixels can be decoded in about 49 s. This means that for a successful commercial product at least tenfold processing power or speed is required. It can be expected that future successors of the microcontroller will meet this criterion.