SMOKE - Speeding up MPEG-4 Operational Kernels on Excalibur

Jonathan Hofman, Guido de Goede, Georgi Gaydadjiev and Stamatis Vassiliadis
Computer Engineering Laboratory,
Electrical Engineering, Mathematics and Computer Science Department,
Delft University of Technology,
Delft, The Netherlands
E-mail: {jonathanhofman, gdgoede, georgi}@Dutepp0.ET.TUDelft.NL
http://ce.et.tudelft.nl

Abstract—**This paper presents SMOKE hardware/software (HW/SW) co design exploration of an MPEG-4 decoder on Excalibur. SMOKE's main goal is to provide a full-featured hardware accelerated MPEG-4 decoder on the Altera Excalibur device providing adequate speed performance. The software part of SMOKE is based upon modified versions of an open source operating system and codec - Linux and XviD respectively. The SMOKE hardware accelerators are designed using VHDL. The complete HW/SW system is built using the Quartus design environment of Altera and a single makefile. It is shown that a speedup of 1.29 for the MPEG-4 decoder will be realizable.**

Key-words— **hardware-software co design; ARM processor; FPGA; reconfigurable hardware; MPEG-4**

I. Introduction

It is envisioned that the correct answer for future performance enhancement of embedded systems is through dedicated hardware for the computation intensive kernels of the targeted applications. Currently recognized applications are the multi-media encoders and decoders used in the new generation mobile devices. In order to preserve the flexibility of such systems, solutions involving general-purpose processors and reconfigurable hardware are emerging. The latter increases the complexity of the design process, hence new insights in this process are required. Therefore the SMOKE (Speeding up MPEG-4 Operational Kernels on Excalibur) project was initiated. SMOKE's main goal is to provide a full-featured hardware accelerated MPEG-4 decoder.

The target platform of SMOKE is DAMP (Delft Altera based Multimedia Platform) [5]. DAMP is based on the Altera Excalibur device [2] (EPXA1F672C3), which contains an ARM992T processor and a 100k FPGA fabric. The communication between the ARM processor and the FPGA fabric is via the high-speed AMBA bus. The integration of a general purpose processor and an FPGA into a single device, makes Excalibur well suited for hardware software co-design targeting computation intensive applications with a large number of data transfers. Besides Excalibur DAMP provides the necessary peripherals needed by stand-alone multimedia applications, like a VGA interface, a large SDRAM memory and a network interface.

To have basic support for networking, filesystem and debugging a version of the Linux kernel (version 2.4.19-rmk7) was ported to the DAMP platform. The MPEG-4 decoder used for SMOKE is based on the open source codec XviD [4]. The XviD codec was designed with portability in mind. Therefore porting the XviD codec to the DAMP platform did not require significant changes to the XviD codec sources.

The main contributions of this paper are:

- XviD evaluation based on profiling;
- implementation of the XviD hot spots in hardware;
- system integration and performance evaluation of SMOKE.

This paper is organized as follows: Section II discusses the software evaluation of the XviD codec and determines which parts can most likely be sped up. After the design of the hardware in the next sections III and IV, this expectation will be proven valid in the section V. Finally, section VI concludes the paper and gives some recommendations for future research.

II. XviD evaluation

In order to predict the impact of hardware acceleration of certain XviD kernels the following methodology was used. The XviD decoder was profiled using the profiling tool which is part of *gcc*. This has first been done on a pc running RedHat Linux 8.0 (*gcc* version *3.2*) to obtain initial indications as to which parts would benefit most of implementation in hardware.

On the pc, the functions requiring the most time were the colorspace conversion (from YUV to RGB) and the In-

verse Discrete Cosine Transform (IDCT). The colorspace conversion takes over 20% of the processing time and the IDCT uses almost 20%. These percentages are accurate up to 5%, since subsequent iterations of the profiling provided slightly different results. All resulted in these two functions consuming a large amount of time. Comparison of these measurements to profiling done on the DAMP platform, showed some noticeable differences. The initialization of the XviD decoder was responsible for most of the time on the DAMP platform. This is due to the limitations of this version of the DAMP platform, but its impact is negligible for longer movies. In that case the initialization is only a small part of the whole, since the initialization is needed only at the start of the decoding process. Besides the initialization function, the functions requiring the most time were again the colorspace conversion, the ICDT and some functions calling the IDCT.

The colorspace conversion draws attention, because this is an operation requiring a 3x3 matrix multiplication for each pixel, which takes significant time in software, but can be implemented easily in the VGA driver hardware. The internal colorspace of the XviD codec is YUV and the output of the MPEG-4 decoding needs to be RGB. This is one operation that should be optimized by hardware. The other operation to be optimized is the IDCT. It can be built separately from the functions calling it, requiring only the 64 values, and no global values. In the future the functions calling the IDCT could be designed into hardware as well, using an Excalibur device with a larger FPGA fabric.

Both the IDCT and the colorspace conversion are responsible for a large part of the execution time of the XviD decoder. It is expected that reducing the time needed for both functions by using hardware will noticeably reduce the decoding time of an MPEG-4 frame on the DAMP platform.

III. Inverse Discrete Cosine Transform

As stated in the profiling section, the (2-dimensional) IDCT is responsible for a considerable part of the decoding time of an MPEG-4 video. In this section the considerations for the implementation of the IDCT in hardware are elaborated.

The hardware part should enable the software to function as it did without hardware, without noticeable differences (except for speed). It should fit in the available reconfigurable hardware and has to be combined with the hardware to speed up the colorspace conversion. Furthermore, the time needed for communicating between the hardware and the software should be kept as small as possible.

To keep the communication overhead low, an efficient way to communicate is required. The IDCT requires 128 bytes to be sent to the hardware and sent back after finishing the IDCT. The transfer of the data and the IDCT operation in hardware need to finish within the time needed for the software to finish. In each cycle as much data as possible needs to be transfered.

When exploring the ways for the ARM CPU to communicate with the connected FPGA, several options are to be evaluated. There is one GPIO (General Purpose Input Output) register available for communication between the CPU and the FPGA. This register allows 4 bits to be transferred at a time, leaving it suitable only for signaling purposes. Another way of communicating with the hardware would be to use the AMBA bus and make the FPGA a slave on that bus. The AMBA bus is a high speed 32 bit bus available in the Excalibur for connecting different devices. A third option would be to use a DMA controller inside the FPGA to copy the data via the AMBA bus from memory accessible by the CPU, into a local memory on the FPGA. However both options using the AMBA bus would increase the communication overhead and the size of the hardware design, because extra hardware in the FPGA would be needed to communicate. Finally the preferred option would be to use the available dual-port RAM (DPRAM) , which is accessible by the CPU as well as by the FPGA. It does require little extra hardware inside the FPGA and it causes little communication overhead for the hardware as well. The CPU communicates with the DPRAM via the AMBA bus, as it does with all periphery. The hardware in the FPGA however communicates with the DPRAM directly. The DPRAM is large enough for the data that needs to be transfered. One IDCT will be performed each time, after which the results will be needed for the continuation of the MPEG-4 decoder.

The 2-dimensional (2-d) IDCT of a 64 value matrix can be calculated by first performing a 1-d IDCT operation on each row and then performing the 1-d IDCT on each column of the resulting values, as explained in [3]. This is the approach used by the XviD software. The hardware will be designed in the same manner, to prevent differences with the software caused by rounding.

The row operations must be performed before the column operations, but can be performed independently of other row operations. Column operations can also be performed independently of other column operations. When comparing row and column operations in software, only subtile differences appear between row and column operations. This enables the design of a piece of hardware capable of performing a row- or column operation when requested to do so. A device capable of only row operations would take up about 50% of the available gates in the

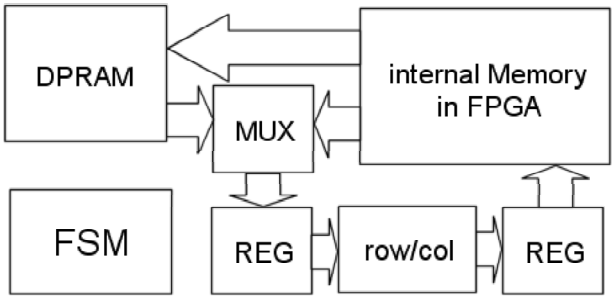

Fig. 1. Datapath of the implemented IDCT with row/col device.

FPGA fabric, when designed in hardware using Quartus II from Altera. When expanding it to perform the column operation as well, about 60% of the FPGA fabric was covered. This device will be called the row/col device from now on.

indent Using this row/col device some control logic is needed to load the appropriate row or column into the device, store the results for column operations and in the end writing back the results. The datapath in which this row/col device is embedded is depicted in figure 1. The DPRAM is capable of reading out 32 bits at a time, so four reads are needed for the 128 bits. Initial simulations of the row/col device showed that results of the operation are available after about 100 ns.

Tests show that the DPRAM can be read out by the hardware at 50 Mhz while still having the read value available after one clock cycle. A 40 Mhz clock is sufficient for the four values to be read out in 100 ns. The device uses extra registers to pipeline the design so that new values can be put into the row/col device at the same time as the previous results are read. The results from the row operations are stored into an internal RAM on the FPGA and fed back into the row/col device to perform the column operations. After each column operation, the results are stored back into the DPRAM. When the IDCT is finished the CPU will be notified by toggling a bit in the GPIO register.

In figure 1 the 'FSM' block contains a state machine, written in VHDL, to control the registers and the communication with the DPRAM. The next section will describe the design of the hardware for speeding up the colorspace (YUV to RGB) conversion and for controlling the VGA interface.

IV. Colorspace Conversion

As shown in section II XviD has another part besides the IDCT that is responsible for a large amount of the computation time. This is the colorspace conversion. The colorspace conversion is responsible for outputting the decoded image in the color format that is used by the calling application. XviD can convert into several colorspaces, for instance RGB (Red, Green and Blue) and YUV (Luminance, Chrominance blue, Chrominance red) [1]. These conversions are linear transformations, which can be implemented as matrix multiplications. A transformation can be reversed, but the limited precision of computer hardware results in rounding errors. These rounding errors can result in the loss of image quality.

Some colorspaces make use of the special properties of the human eye. The human eye is far more sensitive to brightness then to color [1]. This property makes it possible to lower the resolutions for the color information without resulting in noticeable differences for the human eye. The YUV colorspace separates the color information from the brightness and is therefore well suited to perform (lossy) compression. So one value can for example be stored for every pixel with information about the the luminance, while for the chrominances one value can be stored stored for four pixels. Resulting in compacter images.

The XviD decoder uses YUV as native colorspace. This means that all operations performed by the decoder are on images in the YUV colorspace. The YUV colorspace is stored in the yv12 format. The yv12 format stores one value for the luminance of every pixel, but for every square block of four pixels it stores one value for each of the chrominance parameters.

A. VGA Controller

The DAMP platform will output the decoded images via the VGA interface, which uses RGB as native colorspace. The VGA interface hardware needs to be implemented in the FPGA fabric, because no dedicated hardware, besides the digital to analog conversion, is available on DAMP. A simple VGA driver would only implement the possibility for displaying images in the RGB colorspace, while a more advanced VGA driver would adapt to the colorspace of the driving application, in this case YUV.

A.1 Simple VGA controller

The simple VGA controller consists of 3 main parts, a framebuffer, a DMA controller and a VGA driver. A block diagram of such a controller is depicted in figure 2. The framebuffer is located in the SRAM of the Excalibur device, while both the DMA controller and the VGA driver are placed in the FPGA fabric. Because of the limited size of the SRAM only frames with a maximum resolution of 160 by 120 pixels can be displayed. A framebuffer located in the SDRAM would eliminate this restriction. Unfortunately the current prototype of DAMP does not allow highspeed memory transfers from the SDRAM. Highspeed

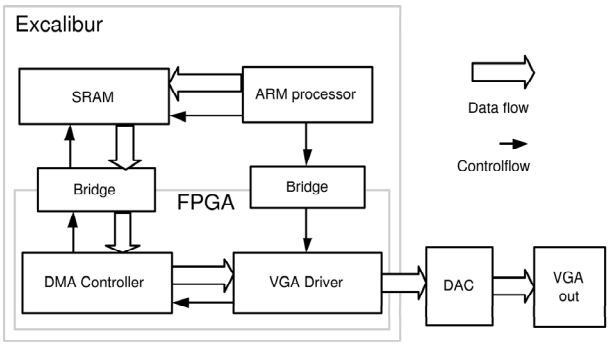

Fig. 2. Blockdiagram of the VGA controller.

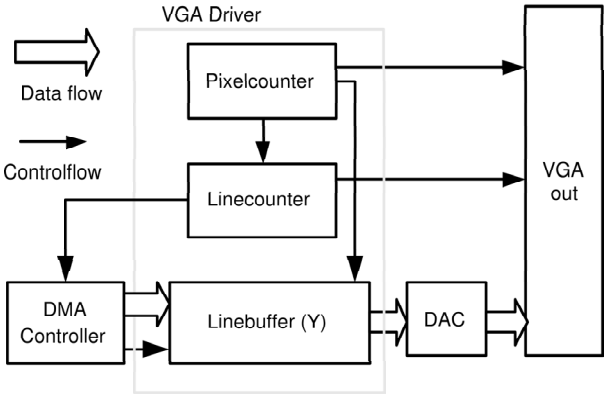

Fig. 3. The VGA driver that supports only RGB values.

memory transfers are required for the correct operation of the the VGA driver.

The VGA driver itself consist of several modules (depicted in figure 3). It contains a linebuffer, for holding the pixel information of the current line, a pixelcounter and linecounter, and some additional glue logic. The pixelcounter is directly connected to the linebuffer which feeds the DAC with the information of the current pixel. The pixelcounter also generates some control signals needed for the VGA interface and the linecounter. The linecounter is connected to the DMA controller, which needs to refresh the linebuffer after a line is written. With the information from the linecounter the address of the new line within the framebuffer can be calculated. The linebuffer is then filled with the data for the next line before the first pixel needs to be written. Furthermore the linecounter is also responsible for the generation of some control signals needed by the VGA interface.

A.2 Adapted VGA controller

The simple VGA driver worked completely in the RGB colorspace. Because the decoder operates in the YUV colorspace a conversion needs to be made. Performing this conversion in software consumes a significant amount of

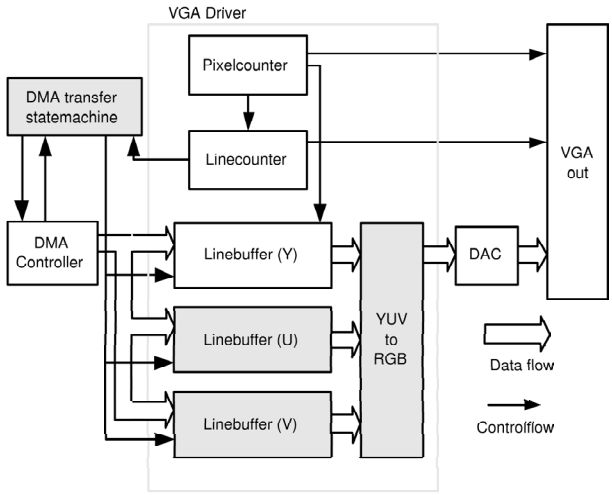

Fig. 4. Block diagram of the VGA driver supporting YUV. The grey blocks are added for the YUV support.

computation time, while the same can be achieved with a relatively small hardware implementation. Therefore only a few modules need to be added to the simple VGA driver. The conversion of the pixels will be made just before they are fed to the DAC. The datapath before the conversion needs to be extended to support the separate Y, U and V values. Therefore two linebuffers are added. This is depicted in figure 4. Also the DMA controller needs to perform three transfers on different memory locations, because of the separate locations in memory, where the Y, U and V planes are stored. For this purpose an additional control FSM at the top layer is added. The final modification is placing a colorspace conversion module between the output of the linebuffer and the DAC. It is possible to switch this module to RGB mode (pass-through) or YUV mode.

V. Experimental evaluation

As a result of the modifications mentioned in chapter III and IV a speedup of the XviD decoder can be expected. In this chapter the experiments for evaluating the speed-up and the results of these experiments are presented. For these experiments the same reference data was used as for the profiling process. All experiments were performed within the kernel of the Linux operating system.

To calculate the speedups from the execution times, Amdahl's law is used (1).

$$S_i = \frac{T_{sw}}{T_{sw} - T_{sw,i} + T_{hw,i}}, \tag{1}$$

where T_{sw} is the execution time of the decoder when none of the parts are accelerated by hardware. $T_{sw,i}$ is the execution time of a single kernel in software and $T_{hw,i}$ the

kernel	part of T_{SW}	kernel speedup	expected XviD speedup
IDCT	20%	1.25	1.04
colorspace	20%	12.28	1.23
both	40%	2.27	1.29

TABLE I
SPEEDUPS OF THE DIFFERENT KERNELS BY IMPLEMENTING THEM IN HARDWARE. ALSO THE EXPECTED SPEEDUPS FOR THE ENTIRE DECODER ARE GIVEN.

execution time of a single kernel in hardware.

When multiple kernels are accelerated using hardware the speedup is given by

$$S = \frac{T_{sw}}{T_{hw}} = \frac{T_{sw}}{T_{sw} - \sum T_{sw,i} + \sum T_{hw,i}}, \qquad (2)$$

where T_{hw} is the total execution time when all the kernels of interest are implemented in hardware. The maximum speedup can be calculated by letting $\sum T_{hw,i} \to 0$.

Due to the limitation of the clock resolution it is not possible within the decoder to measure the execution time of the separate kernels. Therefore the input data for the different kernels was extracted from the decoder during a test run. The extracted data was used to execute the specific kernels outside the decoder. In order to gain a higher clock resolution the accelerated kernels were run as a linux kernel module. A kernel module has better access to the lower level hardware, which made it possible to access the internal timers. One of these timers was available to measure the execution time of the accelerated kernels. The results of these measurements are stated in table I.

The obtained speedup from software to hardware for the IDCT when performing several thousand IDCTs on reference data is 1.25 and the obtained speedup from software to hardware for the colorspace conversion is 12.28.

From the speedups shown in table I, it is expected that when implemented into XviD, the speedup of the decoder due to the IDCT acceleration will be 1.04. For the colorspace conversion a speedup of 1.23 is expected. This is expected to yield a speedup of 1.29 on the entire XviD decoder, which is graphically depicted in figure 5.

VI. CONCLUSIONS

In this paper the implementation of SMOKE was presented. Two computation intensive kernels of the XviD MPEG-4 decoder were identified and implemented in hardware. These kernels, the IDCT and colorspace conversion, each are responsible for 20% of the decoders computation time. The implementation of the IDCT and colorspace conversion in hardware resulted in a speedup of *1.25* and *12.28* respectively. When extrapolated to the implementation into the entire XviD decoder, these results indicate a speedup of *1.29*. Such a speedup justifies hardware software co design for real life multimedia applications like video decoding.

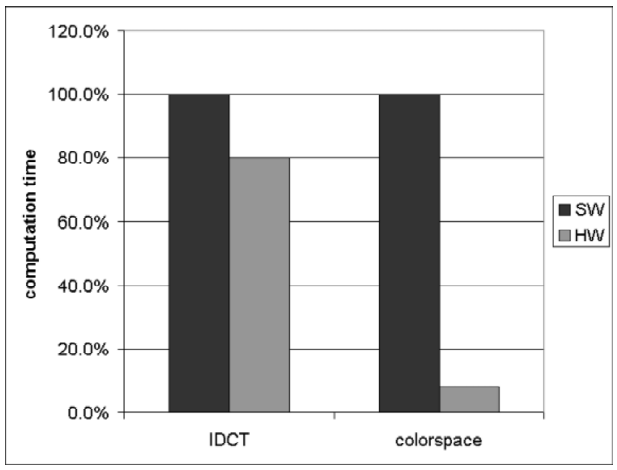

Fig. 5. Comparison of the computation time for the accelerated kernels.

It is recommended for the future to implement the accelerations into the decoder and to verify the extrapolated results. Furthermore it would be of interest to run the experiments on a revized version of the DAMP platform. This revized version of DAMP would have a higher memory bandwidth, making it better suited for multimedia applications.

REFERENCES

[1] P. Bourke. Ycc colour space and image compression, November 2000. http://astronomy.swin.edu.au/ pbourke/colour/ycc/.
[2] Altera Corporation. *Excalibur Hardware Reference Manual*, November 2002.
[3] A. B. Watson. Image compression using the discrete cosine transform. *Mathematica Journal*, 4:81–88, January 1994.
[4] XviD. http://www.xvid.org.
[5] W. Zwart, J. Eilers, G. N. Gaydadjiev, and S. D. Cotofana. Damp - delft altera-based multimedia platform. In *Proceedings ProRISC 2002*, pages 587–594, November 2002.