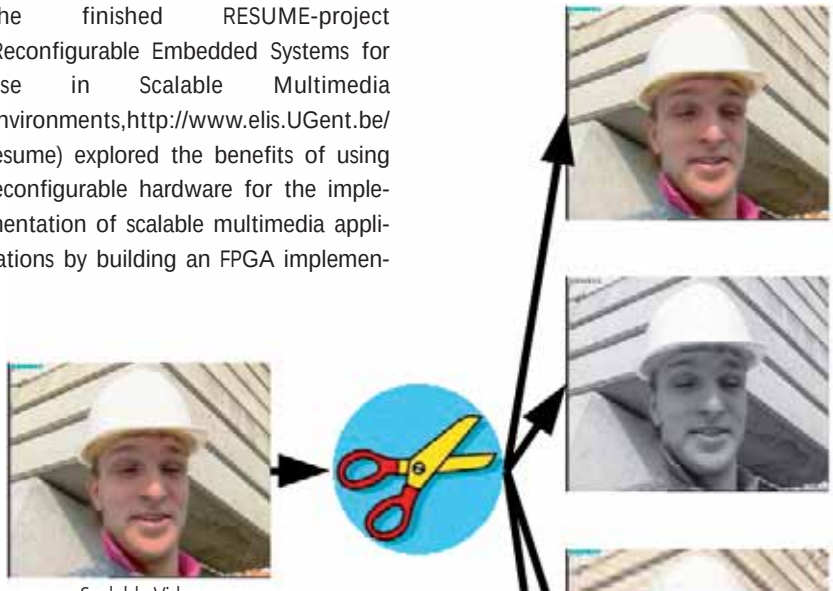
Client-side diversification has led the video-coding community to develop scalable video-codecs supporting efficient decoding at varying quality levels. In the RESUME project an FPGA-demonstrator was developed for a wavelet-based scalable video decoding scheme. For the implementation, state-of-the-art tools and rigorous testing methodologies were deployed to achieve real-time and lossless decoding.

UGent delivers first real-time wavelet-based scalable video decoder on FPGA

The finished RESUME-project (Reconfigurable Embedded Systems for Use in Scalable Multimedia Environments,http://www.elis.UGent.be/resume) explored the benefits of using reconfigurable hardware for the implementation of scalable multimedia applications by building an FPGA implementation of a real-time, scalable, wavelet-based video decoder. The term 'scalable video' refers to a coding scheme that can easily accommodate changes in quality of service (QoS) without the need for transcoding. A scalable video stream can be decoded at varying frame rates, resolutions and image qualities by simply skipping redundant parts in the video stream, only decoding those parts that will contribute to the displayed video.



Scalable Video

Such a scalable video codec has advantages for both the server (the provider of the content) and the clients. On the one hand the server scales well since it has to produce only one encoded video stream that can be broadcasted to all clients, irrespective of their QoS requirements. On the other hand the client (or the network) can easily adapt the decoding parameters to its needs. This way it is possible to optimize the use of the network, display, the required processing power, the required memory, ...

About RESUME:

The RESUME project was funded by the IWT and had four participating partners: IRIS (VUB), Paris (UGent), MML (UGent) and IMEC. The presented work was done by the Paris (Parallel Information Systems Group) research group of the Department of Electronics and Information Systems ELIS at Ghent University. The demonstrator was developed by Hendrik Eeckhaut, Mark Christiaens, Harald Devos and Philippe Faes with Dirk Stroobandt as promoter.

Scalability has a lot of advantages but comes at a cost. The decoding algorithm is computationally complex and really stresses the system bandwidth as it replaces the block-based DCT-approach with wavelets. This has a tremendous impact on the hardware architecture.

We implemented the complete decoding pipeline of our custom wavelet-based scalable video codec on a PCI development board equipped with a Stratix FPGA S60 and 256 MiB of DDR SDRAM

Demonstrator

continuation on page 6

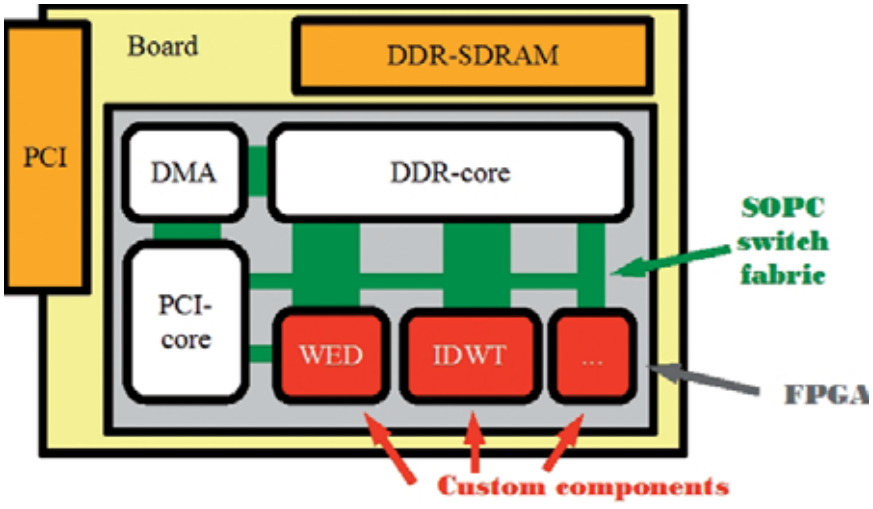
continuation from page 5

memory. The design goals were real-time, lossless decoding of CIF-sequences (352x288 pixels) at 25 frames per second. The FPGA board is plugged into a standard PC with two monitors, one dedicated to displaying the decoded video, the other to interact with the system.

Implementing a complete video codec is a complex undertaking that requires careful planning. We applied the following methodology. First the entire software code base was cleaned-up and we made sure that the algorithms used were properly understood. At this point, it was clear that the entire codec was too complex to be completely implemented in reconfigurable hardware. As a consequence, we chose to use a HW/SW-codesign approach leaving as much of the algorithm as possible in SW running on a CPU while implementing the time-critical parts in reconfigurable HW.

The hardware design was implemented using state-of-the-art tools such as Altera's SOPC (System-On-a-Programmable-Chip) Builder for component-based system integration. For each of the steps in the decoding pipeline we developed custom components to deliver the required hardware acceleration. Most components differ substantially from others with respect to their maximum clock frequency. To accommodate this, the design is subdivided into multiple clock domains. Thanks to the use of SOPC Builder this can be achieved fairly easily by assigning different clocks to each of the components.

Most components of the design were highly optimized to achieve real-time decoding. The wavelet entropy decoder, very similar to AVC's CABAC, was hammered until it produced one decoded symbol per clock cycle. We also developed an automatic IDWT generator based on polyhedral techniques (loop transformations). This resulted in a line-based IDWT especially tailored to the specific access pattern of the external on-board DDR-memory. For all development (hardware and software) we applied a write-tests-first strategy and used well founded engineering techniques as code reuse, refactoring, regression testing and continuous integration to continuously guard the quality of the entire design.



Hardware Overview

The major bottlenecks in the design were the limited bandwidth of the PCI-bus and the DDR-memory. Trading in the conventional block-based DCT transform for the wavelet-transform results in calculations on larger data blocks: wavelet sub bands instead of macro blocks. FPGAs no longer house sufficient internal memory to buffer the intermediate results of the decoding processes, forcing off-chip buffering. Of course, this results in large bandwidth requirements between the external memory and the FPGA. To alleviate this problem, much effort has been put in optimizing this communication through the use of DMA transfers in efficient burst mode.

Our approach was unique due to the large number of advanced tools and methods we combined. We built a hardware demonstrator that achieved the design goal of more than 25 lossless decoded CIF frames per second. Compared to our software implementation, run at 2 GHz on an AMD64 3500+, the FPGA-implementation at approximately 55 MHz is 3 times faster.

Now that we have a design capable of lossless decoding, we can investigate the options of reconfiguration with smaller or slower designs to adapt the hardware resource usage to lower quality requirements. This way we can effectively link the video scalability to real hardware scalability.