Energy Scalability and the RESUME Scalable Video Codec

Harald Devos, Hendrik Eeckhaut,
Mark Christiaens and Dirk Stroobandt

Ghent University, ELIS Dept., Parallel Information Systems
Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium
Harald.Devos@elis.UGent.be

Abstract. In the context of the RESUME-project a scalable wavelet-based video decoder was built to demonstrate the benefits of reconfigurable hardware for scalable applications. *Scalable* video means that the quality of service (QoS), i.e., the frame rate, resolution, color depth, ... of the decoded video can easily be changed by only decoding those parts of the video stream that contribute to the desired QoS. With the emergence of high-performance FPGAs (Field Programmable Gate Array), both the required performance for real-time decoding and flexibility, by allowing reconfiguration, are offered.
Since the amount of calculations scales with the QoS, energy dissipation is expected to scale similarly. To investigate the relation between QoS and energy dissipation we actually measured the energy dissipation of a scalable video decoder implementation on a FPGA. The measurements show how dissipation effectively scales with the QoS, but also depends on the decoded data and the used design method. This is illustrated by comparing two different implementations of the inverse discrete wavelet transform (IDWT).

Keywords. Wavelet-based Scalable Video, Energy Measurement, Hardware Generation, FPGA

1 Introduction

Reconfigurable hardware, in particular FPGAs (Field Programmable Gate Array) [1], has found its way in a large class of systems and applications, thanks to the high computational power and low cost in comparison with ASICs (Application Specific Integrated Circuit). Until now their reconfigurability has mainly been used for upgrading a system and not for switching at run-time between several designs. One target of the RESUME project (Reconfigurable Embedded Systems for Use in scalable Multimedia Environments[1]) was to study how scalable video can benefit from an implementation on scalable (reconfigurable) hardware. *Scalable video* means that the quality of service (QoS), i.e., the image quality, frame rate, resolution and color depth of the decoded video, can be freely adapted without having to re-encode the video stream or having to decode the whole video stream if only a lower quality version is required. As a result, the video

[1] http://www.elis.ugent.be/resume

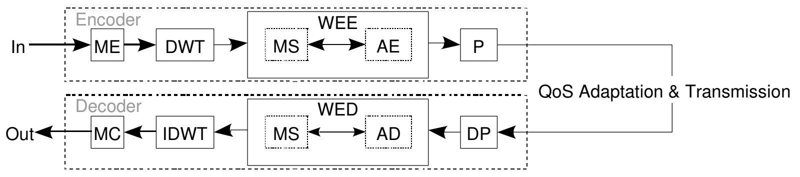


Fig. 1. High-level overview of the video encoder and decoder.

server only has to encode and store a single video stream for all different kinds of decoding platforms which can have varying computational power, network bandwidth or screen types.

The available QoS varies between different platforms, but can also be varied in time on a single platform, e.g., when the available bandwidth or battery power drops or different tasks on the platform have to share hardware resources. In the latter case reconfiguration of the hardware might become beneficial. The reconfiguration times of FPGAs, typically some tens of milliseconds, are often too large to use dynamic reconfiguration as a mean to enable time-multiplexing of hardware, e.g., when the device is not large enough to put all active hardware blocks next to each other in the available area. However, in the time scale of video this reconfiguration time is small, typically around the time one frame is displayed, and scaling hardware with scaling QoS becomes an option. Within the RESUME project a hardware (FPGA) implementation of a wavelet-based scalable video decoder has been built.

In this paper we study the relation between the energy dissipated by the decoder hardware and the delivered QoS. As manual hardware design is, in comparison with software development, very labor intensive and error prone, the design of some hardware blocks was used as an occasion to experiment with alternative design methods. As a result, our FPGA design was not optimized primarily for low power. Hence, when interpreting our results, we put more interest in the relative impact of scalability, rather than on the absolute power and energy figures. Moreover, since the implementation effort is so high, it is difficult to find any comparable results.

Unless mentioned otherwise, all measurements in this paper were performed on the well known reference video sequence 'Foreman'. Five GOPs (Group Of Pictures), i.e. 76 frames, of this sequence were encoded once at CIF (352×288 pixels) resolution and decoded at varying QoS.

2 Codec Algorithm and Scalability

The algorithmic structure of the RESUME scalable video coder and decoder (codec) is shown in Figure 1 and is described in [2]. The encoder consists of a motion estimation step (ME) [3] which exploits the temporal redundancy in the video stream by looking for similarities between adjacent frames. To obtain temporal scalability, motion is estimated in a hierarchical way. This hierarchical temporal decomposition enables decoding of the video stream at different frame rates because the decoder can choose up

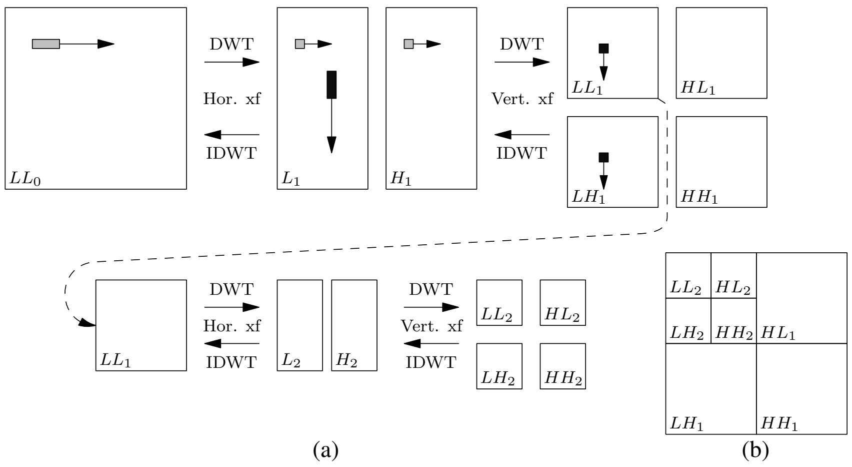


Fig. 2. Graphical representation of the 2-D (I)DWT over 2 levels. The 2-D DWT first filters the original image horizontally, line by line. The gray box in LL_0 represents the pixels that are read to calculate the elements indicated by the gray boxes in L_1 (low pass) and H_1 (high pass). The arrows indicate the scanning direction. Next, the results are filtered vertically, column by column as indicated by the black boxes. These two steps are repeated recursively on the LL subbands. The resulting subbands can be stored together within one data set with the same dimensions as the original frame (b). The 2-D IDWT consists of doing the inverse of each step.

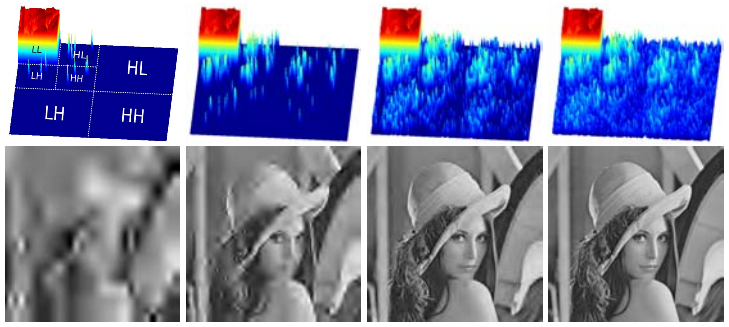


Fig. 3. Quality scalability: decoding more bit layers gives a more accurate wavelet-transformed frame. The distortions of the images on the bottom row are slightly exaggerated for visual clarity.

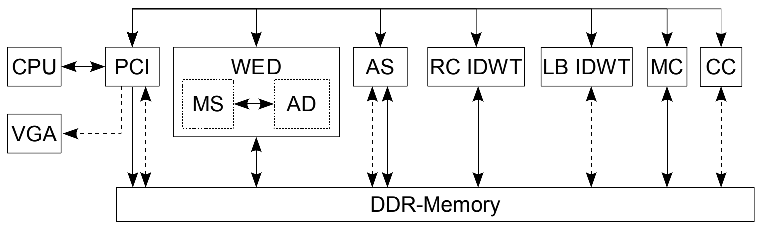


Fig. 4. Overview of hardware architecture of the RESUME decoder. Continuous lines indicate direct (master) write transfers, dashed lines indicate DMA (slave) transfers.

to which (temporal) level the stream is decoded. Each extra level doubles the frame rate. A discrete wavelet transform (DWT) [4] (Figure 2) separates the spatial low-pass and high-pass frequency components. Each LL-subband is a low resolution version of the original frame. The inverse discrete wavelet transform (IDWT) in the decoder can transform up to an arbitrary level, resulting in resolution scalability. The wavelet entropy encoder (WEE) [2] is responsible for entropy encoding the wavelet transformed frames. The frames are encoded bit layer by bit layer, yielding progressive accuracy of the wavelet coefficients (Figure 3) influencing the PSNR (Peak Signal to Noise Ratio) of the decoded frames. In the video community this is called quality scalability. The WEE consists of two parts: The model selector (MS) for statistical context modelling and the arithmetic encoder (AE) for the actual compression. Finally, the packetizer (P) packs all encoded parts of the video together in one bit stream. This video stream is adapted to the required QoS and transmitted to a decoder, that performs the inverse operations of the encoder.

3 Platform and Implementation

For demonstration purposes, the used hardware platform consists of an Altera PCI high-speed development board [5], equipped with a Stratix S60 FPGA and 256 MiB of DDR SDRAM, plugged into a standard PC with two monitors, one dedicated to displaying the decoded video, the other to interact with the system. The basic structure at the bottom of Figure 1 was refined to the architecture shown in Figure 4. The main control over the decoder resides on the host PC. All decoder blocks, but the depacketizer (DP), are implemented on the FPGA. Data is transferred from one block to another using the off-chip, but on-board, DDR SDRAM because the FPGA does not have enough internal memory to store the intermediate results.

As can be seen in Figure 4, the software architecture (Figure 1) was substantially modified. The entropy decoder was split into two separate components, the wavelet entropy decoder (WED) and the assembler (AS). The WED no longer produces ready-made wavelet frames but instead it constructs individual bit layers of the wavelet frames. The AS was introduced to reconstruct the wavelet frames from the individual bit layers. The use of the AS substantially improves the memory bandwidth. The frames produced

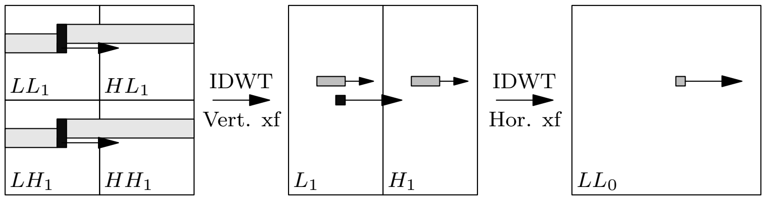


Fig. 5. Vertical filtering in a horizontal scan order leads to a line-based IDWT

by the motion compensator (MC) are in YUV format and are converted by the color convertor (CC) to the RGB color space. The resulting data is (DMA) transferred from the on-board DDR to a dedicated NVidia GeForce 5200 VGA card on the PCI bus.

Two variants of the IDWT were made. The first one (RC IDWT: Row-Column-wise, level-by-level, according to Figure 2) was used to investigate the suitability of SystemC to move from a high-level software implementation to a low level hardware description [6]. Using a single language at all levels of abstraction offered the advantage of using a single test bench throughout the entire design flow. However, with the introduction of SystemC-VHDL-Verilog cosimulators the use of a single language is no longer needed to have this advantage. The SystemC syntax at RTL-level is less decent compared with VHDL and most hardware development tools are not ready yet to work with SystemC. Therefore, it seems better to use each language at the appropriate level. This design could theoretically transform 79 gray-scale CIF frames/s, but in practice suffers from a large memory bottle neck. It needs too many accesses to the external DDR SDRAM, of which only half (the row-wise transforms) can be done in bursts. The resulting framerate drops to less than 15 frames/s (5 color frames/s for the entire codec).

To deal with this problem a second IDWT implementation was made. First, loop transformations were applied to improve the temporal and spatial locality of the data accesses, leading to a line-based IDWT (LB IDWT) (Figure 5). The transformations were partially automated by the WRaP-IT/URUK tool set [7]. We wrote CLooGVHDL, a back-end to this tool [8], to generate control hardware from the internal polyhedral representation used by this tool set. The data path was generated semi-automatically. After comparing several generated variants, one was selected to be extended with a memory hierarchy and integrated in the decoder. The semi-automatic generation of the hardware resulted in a huge reduction of the design time but comes with a large area cost, mainly due to excessive use of multiplexers.

The resulting hardware implementation achieves real-time, lossless decoding of CIF-sequences (352×288 pixels) at 25 frames per second. Synthesis results using Quartus 6.1 are shown in Table 1. The line-based IDWT clearly takes most of the resources, due to its immature design flow.

Measurements show that the execution times of the different components scale similarly with regard to temporal or spatial scalability. PSNR scalability (Figure 6) only influences the blocks that work on a bit plane level, i.e., the WED, AS, and DP. Note that the different blocks work in parallel and therefore, the sum of the their execution times is much larger than the execution time of the total decoder. Figure 6 suggests that

Table 1. Synthesis results of the video decoder. LE: number of logic elements, Regs: number of 1-bit registers (inside LEs), Mem: bits of on-chip RAM, 9×9: number of 9-bit multipliers, 18×18: number of 18-bit multipliers, Clk: clock frequency of the component in MHz. *Others* consists mostly of the Avalon Switch Fabric (Altera SOPC Builder) that interconnects the different blocks of the decoder and takes care of clock domain crossings. With the clock settings shown, the design decodes 26.5 lossless CIF-frames/s.

Component	LE	Regs	Mem (bits)	9×9	18×18	Clk (MHz)
AS	2988	1541	65024	0	2	65
CC	1635	668	36864	0	0	65
MC	2222	1116	25344	0	0	65
DDR	1334	978	4608	0	0	65
DMA	570	310	16384	0	0	65
RC IDWT	3439	1096	55296	0	5	58.6
LB IDWT	20473	2006	395752	0	9	58.6
PCI	4434	1840	23568	0	0	65
WED	4243	1709	107392	1	0	57
Others	10554	5855	0	0	0	
Total	51892	17119	730232	1	16	

our design is not real-time at the highest QoS settings. It takes 3.7 s to decode a 3.04 s sequence. This only seems so because the plots also include the latency, the time before the first frame is displayed (≈ 0.85 s).

4 Energy Measurement Setup

The FPGA board has no special provisioning for current measurement of the FPGA power supplies. To measure the current through the FPGA, which is controlled and powered over the PCI-interface, we used a PCI extender card (Sycard Technology, PCIextend 177) which allows to monitor all power supplies. The PCI-standard provides multiple power lines, of which only two are actually needed to run our design: the 3.3V line for the FPGA and the 5.0V line for the DDR. Because the current through the 5.0V line is very small, at most a few mA, it is ignored. The remaining 3.3V line is measured with a current probe (Tektronix TCP202, 3% accuracy) connected to an oscilloscope (Tektronix TDS7104).

As an illustration of the probe output, we plotted the oscilloscope trace of the inverse wavelet transform of one YUV-frame in Figure 7(a). It is easy to recognize the processing of the Y frame followed by the U and V frame, the latter two with a quarter of the frame size and execution time. Within a frame the three transformation levels of the wavelet are visible, also scaling with a factor four. After zooming in (Figure 7(b)), transforms on individual lines of the wavelet frame can be recognized.

The board draws 2 A when it is not decoding video and just waiting for input. This steady state current (I_{ss}) is used by the DDR memory chips, the clock oscillators, the

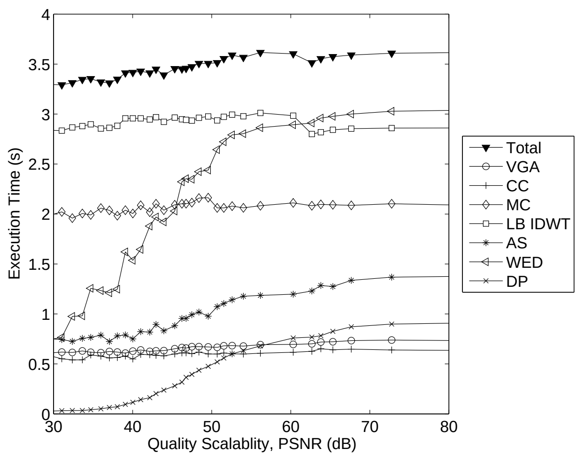
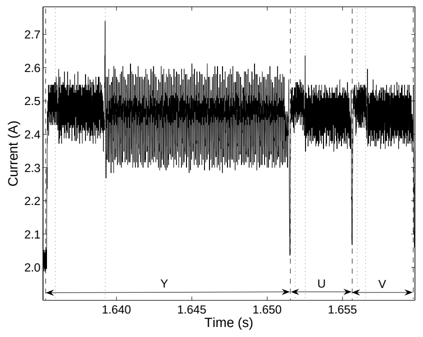
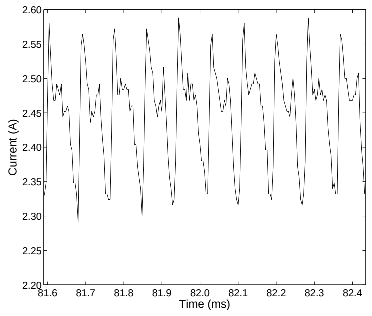


Fig. 6. Execution time for decoding 5 GOPS of the Foreman sequence for quality scalability.



(a)



(b)

Fig. 7. (a) Current drawn by the IDWT by transforming the three channels Y, U and V over 3 levels. (b) Zooming in on Fig. (a) unveils a periodic behavior. Each period corresponds with transforming two lines of the frame.

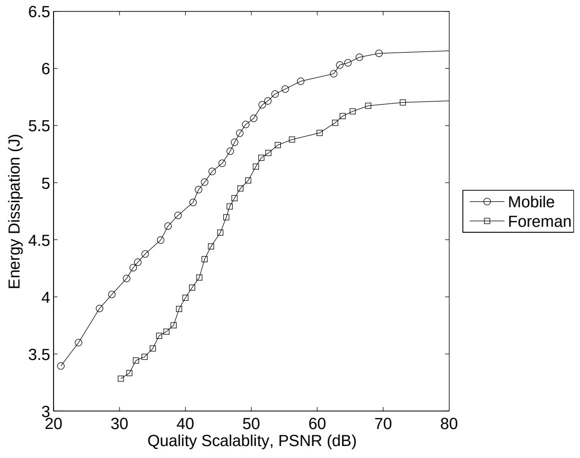


Fig. 8. Energy as a function of PSNR for two different video sequences, each 5 GOPs long.

voltage regulator circuits, glue logic, for distributing the clock signal inside the more than 75% filled FPGA, etc. Because we are only interested in the energy that is needed for the actual video decoding, the steady state current will be ignored in the remainder of this paper.

The temperature of the FPGA appeared to influence the measurements. E.g., the steady state current was higher after performing operations with a large power dissipation. By attaching a heat sink this effect was removed and the current measurements became reproducible.

Measuring the instantaneous current enables to determine the amount of power and energy that is needed to decode a video sequence. Energy (E) and Power (P) are calculated from the measured current ($i(t)$) by

$$\begin{aligned} i_n(t) &= i(t) - I_{ss} \\ P(t) &= V_{source} \times i_n(t) = 3.3\ \mathrm{V} \times i_n(t) \\ E &= \int P(t)dt \approx \sum_i P(t_i)\Delta t\ , \end{aligned}$$

where Δt is the sampling period of the oscilloscope.

5 Energy Scalability

The energy measurement process was automated to measure the energy dissipation for different quality (PSNR) settings. The results for decoding 5 GOPs of the Foreman and Mobile sequence are plotted in Figure 8. The energy dissipation clearly scales with the PSNR. Lossless decoding (∞ dB) dissipates almost twice the amount of energy

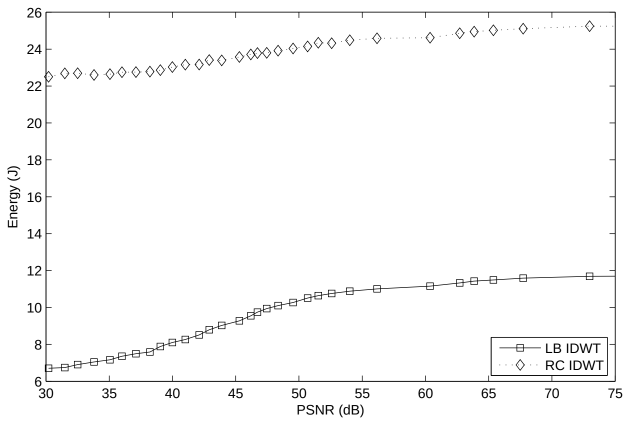


Fig. 9. Energy of the total video decoder as a function of PSNR when using the two different variants of the IDWT. Figures for decoding 10 GOPs (double of the other figures) of the Foreman sequence.

Table 2. Comparison of two decoders, each with one variant of the IDWT. Sequence of 10 GOPs (double of the other figures) of Foreman at full quality.

Variant	Time (s)	E (J)	P_{mean} (W)
RC IDWT	40	25.4	0.64
LB IDWT	10	11.6	1.16

needed for decoding at 30 dB. There is also a clear difference in dissipation between the different video sequences. The Mobile sequence needs at least 0.5 J more energy for decoding at the same quality.

A comparison of the two variants of the IDWT is shown in Figure 9. The hand-made row-column-wise IDWT is much smaller than the generated line-based IDWT, but uses a multiple of the energy, although the dissipated power is 2 times lower. This is because the decoding time is 4 times longer (Table 2). The higher energy is caused by the higher number of off-chip accesses to the external memory. The lower power is mainly a result of the fact that blocks are stalled when they wait for input data. In all other measurements the line-based IDWT is used.

Although only a single current through the entire design is measured, it is possible to get an idea of the distribution of energy dissipation among blocks. Therefore, the components were not only measured when working together, but also one by one. To that purpose the decoder control software was instrumented to log all hardware component instructions. This allows to replay them per component. To obtain relevant measurements the same input data is read by the components as if they were in the complete decoding pipeline. This was achieved by modifying the control software to never deallocate DDR-memory so that all intermediate results remain present and available for replay. The results of this approach for decoding the Foreman sequence at different quality scalability settings are plotted in Figure 10 (top).

As expected the WED significantly scales with increasing quality as it has to decode more bit planes; a factor ten between lossless and lowest quality. The IDWT scales with quality, despite the fact that the control flow and the number of calculations, are invariant. With increasing quality, a larger fraction of the wavelet coefficients, or a larger fraction of the bits within the coefficients, becomes non-zero, resulting in more signals that toggle and an increased energy dissipation. In the AS, MC and CC, the energy dissipation is almost invariant. The sum of the energy of these components differs from the total energy mostly because two components were not separately measured: the INPUT-step and the VGA-step. The INPUT-step, the part of the DP-step that copies the encoded data from the host PC to the FPGA-board, is expected to mildly scale with quality, since more data has to be transfered through the PCI and DDR core. The energy dissipation of the VGA-step is expected to be invariant, similar to CC. Another reason for the discrepancy between *Total* and *Sum* is the absence of interaction between the components when replaying them per component. When using all components concurrently in parallel, they compete to access the DDR. When quality increases, the data flow increases, resulting in more conflicting DDR requests, which leads to a slightly longer execution time and thus a slightly larger energy dissipation.

By combining the energy measurements (Figure 10 (top)) with the execution time (Figure 6) two kinds of power plots can be derived. If the energy E_c of a component c is divided by the execution time of the entire decoder T_t, one gets a measure of the *mean power*, $P_{m,c}$, of that component within the decoder (Figure 10 (middle)). This mean includes the time that the component is not active.

$$P_{m,c} = \frac{E_c}{T_t}$$

By using the execution time of the particular component T_c, instead of the total decoding time a measure of the *active power*, $P_{a,c}$, of a block is obtained, which is a mean over the execution time, excluding the non-active time (Figure 10 (bottom)).

$$P_{a,c} = \frac{E_c}{T_c}$$

The mean power plot is very similar to the energy dissipation plot, as the execution time varies little with the PSNR. The active power clearly differs. All components are roughly in the same region (0.2–0.4 W), except the IDWT and WED. For the IDWT this can be explained by the bit toggling mentioned above (cause of different energy, though the execution time is constant). In the WED a similar behavior can be found. In the most significant bit planes, which are needed for base quality, a high correlation between the bits is present. This augments the compression but also minimizes the bit toggling and statistical context model calculation. The lower the bit plane, the less correlation and more variance. Therefore, increasing the PSNR, by decoding more bit planes, makes the active power rise.

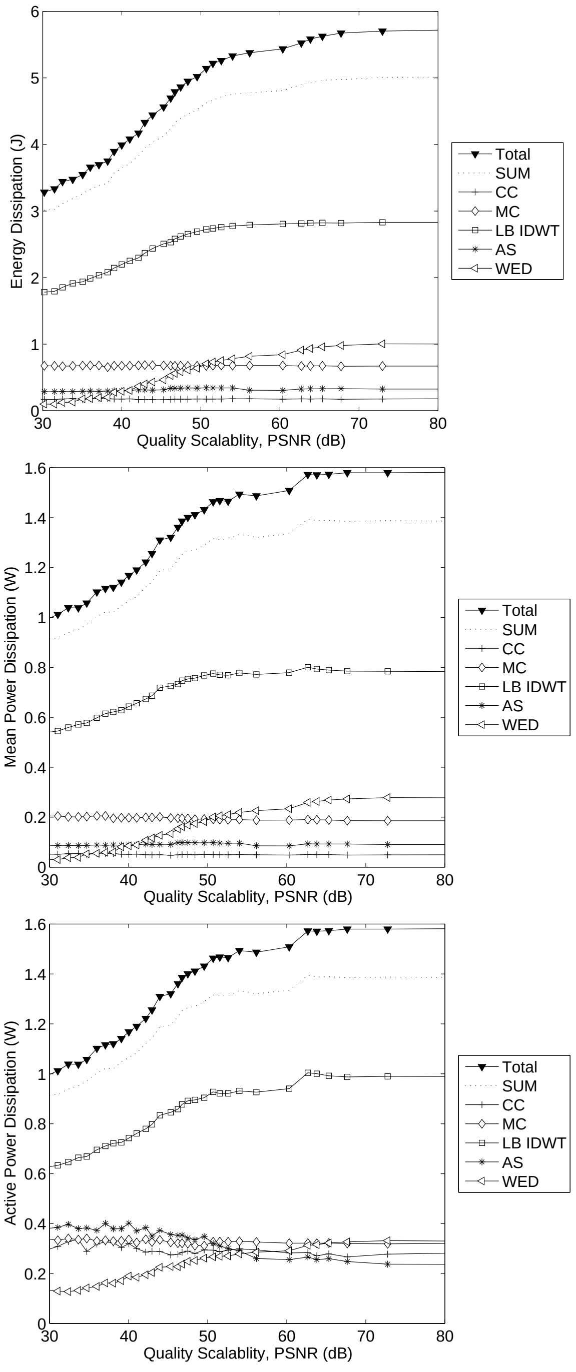


Fig. 10. Energy (top), mean power (middle) and estimation of active power (bottom) for decoding 5 GOPs (CIF resolution) of the Foreman sequence for quality scalability.

6 Conclusions

Wavelet-based, scalable video needs hardware acceleration to allow real-time decoding. This computational power can be offered by FPGAs, which also offer flexibility by allowing reconfiguration. In this paper we investigated the relation between QoS and energy and power dissipation. More energy is needed to decode at higher quality. Even though our video decoder was not designed for low power and only a single current could be measured, some interesting conclusions could be drawn. Rescaling QoS settings on real low power scalable video devices and reconfiguring, with one of several QoS-specific configurations, could lead to significant energy savings. Experiments with two variants of the IDWT showed that a semi-automatically generated, over-sized implementation, can outperform a manual design in speed and energy dissipation if bandwidth is a limiting factor.

Acknowledgment

This research is supported by the I.W.T., grant 020174, the F.W.O., grant G.0021.03, the GOA project 12.51B.02 of Ghent University and the Altera university program.

References

1. DeHon, A.: The density advantage of configurable computing. IEEE Computer **33** (2000) 41–49
2. Eeckhaut, H., Christiaens, M., Devos, H., Stroobandt, D.: Implementing a hardware-friendly wavelet entropy codec for scalable video. In: Proceedings of SPIE: Wavelet Applications in Industrial Processing III. Volume 6001., Boston, USA (2005) 169–179
3. Munteanu, A., Andreopoulos, Y., van der Schaar, M., Schelkens, P., Cornelis, J.: Control of the distortion variation in video coding systems based on motion compensated temporal filtering. In: International Conference on Image Processing (ICIP), IEEE (2003)
4. Mallat, S.G.: A theory for multiresolution signal decomposition: the wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence **11** (1989) 674–693
5. Altera: PCI High-Speed Development Kit, Stratix Pro Edition. 1.1.0 edn. (2005)
6. Devos, H., Eeckhaut, H., Schrauwen, B., Christiaens, M., Stroobandt, D.: Ever considered SystemC? In: Proceedings of the 15th ProRISC Workshop, Veldhoven (2004) 358–363
7. Cohen, A., Girbal, S., Parello, D., Sigler, M., Temam, O., Vasilache, N.: Facilitating the search for compositions of program transformations. In: ACM Int. Conf. on Supercomputing (ICS'05), Boston, Massachusetts. (2005)
8. Devos, H., Beyls, K., Christiaens, M., Van Campenhout, J., D'Hollander, E.H., Stroobandt, D.: Finding and applying loop transformations for generating optimized FPGA implementations. Transactions on High Performance Embedded Architectures and Compilers **1** (2007) 151–170