TASK CONCURRENCY ANALYSIS AND EXPLORATION OF VISUAL TEXTURE DECODER ON A HETEROGENEOUS PLATFORM

*Z. Ma†, C. Wong†, E. Delfose, J. Vounckx, F. Catthoor**

IMEC,
DESICS division
Kapeldreef 75, Leuven, Belgium

S. Himpe, G. Deconinck

K. U. Leuven,
Electrical Engineering Department
Leuven, Belgium

ABSTRACT

The emerging of mobile multimedia terminals has given rise to growing demands for the power-efficient and scalable image transmission. Visual Texture Coding (VTC) has attracted increasing attention due to its scalability when transmitting still images. Nowadays, the implementation of such VTC decoders has not yet considered the need of energy performance trade-offs at the system-level. We have applied systematic system-level design techniques to analyze the VTC decoder and explore its timing-energy trade-off space by using our concurrent task scheduling exploration techniques. The approach presented in this paper allows a system designer to select the optimal heterogeneous platform configuration for a given speed of the VTC decoder while minimizing the global energy consumption.

1. INTRODUCTION

As multimedia applications are being implemented in portable devices and networked scenarios, optimizing resource usage is becoming a key concept. In this context, scalable and power-efficient image coding algorithms have received significant attention recently. The Visual Texture Coding (VTC) [2] scheme of the MPEG-4 standard provides multimedia application designers much more scalability and user interactions to transmit still images, and therefore is employed in many low-cost terminals in low-bandwidth networks [3, 4, 5]. The current design policy of 'divide-and-conquer'[6, 7, 11, 12] leads to the analysis and optimization of the VTC decoder-like multimedia applications for energy consumption without considering the fact that such an application will be embedded into a whole system as a module. To reduce the overall system energy consumption, a designer needs to trade off the energy and performance of the module while still fulfilling the QoS requirements for the whole system. To arrive at really good solutions, this trade-off should even be performed while dynamicall following the real system characteristics, e.g. the wireless channels and data stream contents[10]. Such trade-off space, especially for multi-processor heterogeneous platforms, is too large to explore manually. To address this challenge, we have constructed a system-level model to represent the VTC decoder and then applied our design-time static scheduling approach on this model. The trade-off space generated by the static sub-task scheduling provides a system designer the possibility to select a suitable working point in order to reduce the energy consumption for the total system. It also allows to select an appropriate working point at run-time[9]. An overview [1] of our analysis and exploration flow followed in this paper is illustrated in Fig.1 while the run-time phase is left out.

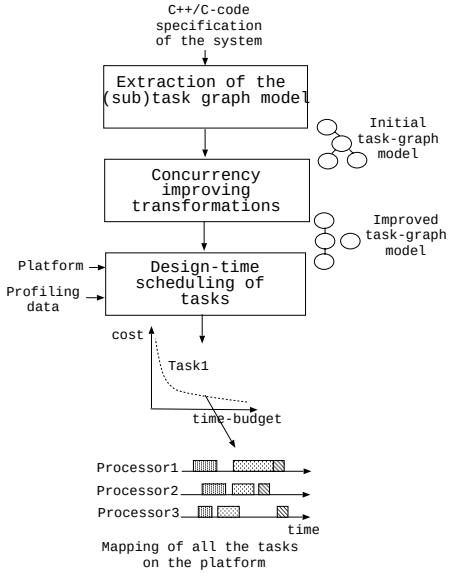

Fig. 1. Task concurrency analysis and exploration flow

This paper is organized as follows: Section 2 briefly introduces the motivations of our analysis and exploration. Section 3 presents the systematic analysis of the VTC decoder at the system-level. Section 4 introduces our static

*also professor of K. U. Leuven. † also Ph.D. students of K. U. Leuven

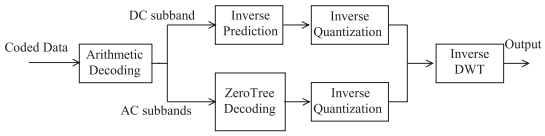

Fig. 2. Block diagram of the VTC decoder

scheduling tool. Section 5 presents the profiling of the VTC decoder. Section 6 reports and discusses the scheduling results. Finally, Section 7 summarizes the paper.

2. MOTIVATIONS

A VTC decoder is basically a pipeline of the following modules: Arithmetic decoding of the DC sub-band using a predictive scheme; Arithmetic decoding of the bit-stream into quantized wavelet coefficients and the significance map for AC sub-bands; Zero-tree decoding of the higher sub-band wavelet coefficients; Inverse quantization of the wavelet coefficients; Composition of the texture using inverse discrete wavelet transform (IDWT). The block diagram of the VTC decoder is shown in the Fig.2, more details can be found in [2, 11, 12].

Traditionally, VTC is implemented into ASIC chips [4, 6, 7]. Besides its expensive design cycles, an ASIC chip does not have enough programmability to keep up with the fast evolving requirements for the multimedia terminals even if it is partly parameterized. Furthermore, the VTC decoder is only one component of the whole multimedia system; in order to optimize the global energy consumption at the system-level, a designer needs to trade off the energy distribution for components. In other words, a component should provide the designer trade-off space to decide at which speed this component needs to run to fulfill the QoS requirements as well as to minimize the energy consumption of the whole system. To this end, a systematic system design approach is needed that can map these applications cost- and power-efficiently to the target 'platform' realization while meeting all real-time and other constraints. Voltage/frequency scaling combined with appropriate task scheduling and dynamic power management techniques are key in this context. However, in existing approaches that overall system trade-off is not exploited and only a single point in the trade-off space is generated.

In this paper we will use an approach derived from [1, 8] to analyze and optimize the VTC decoder. By doing so, we will show that the trade-off space is large and the Pareto curves generated by the task concurrency management techniques for the different tasks in the application can be used to achieve overall power gains for a given system-level performance constraint. Moreover, we will show that heterogeneous platforms can provide even larger trade-off space than conventional homogeneous platforms.

3. TASK GRAPH EXTRACTION AND OPTIMIZATION FOR THE VTC DECODER

First of all, we build a task graph of the application to explore the system-level trade-offs. This especially involves the identification of suitable sub-tasks for which Vdd/frequency scaling can be applied and different processors can be assigned. In this section, the basic task graph extracted from the VTC decoder code will firstly be presented. Then transformations applied to improve the basic graph for scheduling will be shown.

3.1. Building the basic task graph

The basic graph of the VTC code is mainly generated by tracing the program execution. Since the VTC code is specified in an object-oriented manner with C++, every invoked method is traced and made visible in the basic graph. The resulting task graph is shown in Fig.3. The bubbles in the task graph represent the boundaries of invoked methods. A bubble contained by a larger one corresponds to a method called by the method represented by the larger bubble. In this paper, we call each bubble a sub-task.

As shown in the basic task graph, three bottlenecks marked by shadowed bubbles are presented in the VTC code. They account for 57%, 30% and 10% of the execution time, respectively. Only the first one is further analyzed in the model, because its execution time is the largest. If a speed-up is required, the last two bottlenecks are likely to be implemented with a dedicated accelerator data-path and hence kept as a black box in our task graph.

The first bottleneck is a large loop to deal with the Texture Unit (TU) decoding. One TU is decoded during each iteration of this loop. The top half of Fig.4 illustrates the task graph of one iteration. A real-life image processed by the VTC decoder will have up to 256 rectangle blocks. To generate one block, the decoder will process 9 TU's, of which there are 3 Y-element TU's, 3 U-element TU's and 3 V-element TU's. The pixels processed when decoding a Y-element TU are four times as many as those of an U- or V-element TU.

3.2. Basic task graph transformations

After extracting the basic task graph for the VTC decoder, we transform the basic graph in order to increase its concurrency and to handle the varying workload. The transformations will introduce system-level trade-offs and thus allow the designer to map the application more efficiently onto a given platform. The transformations are guided by the profiling data and understanding the nature of the VTC code.

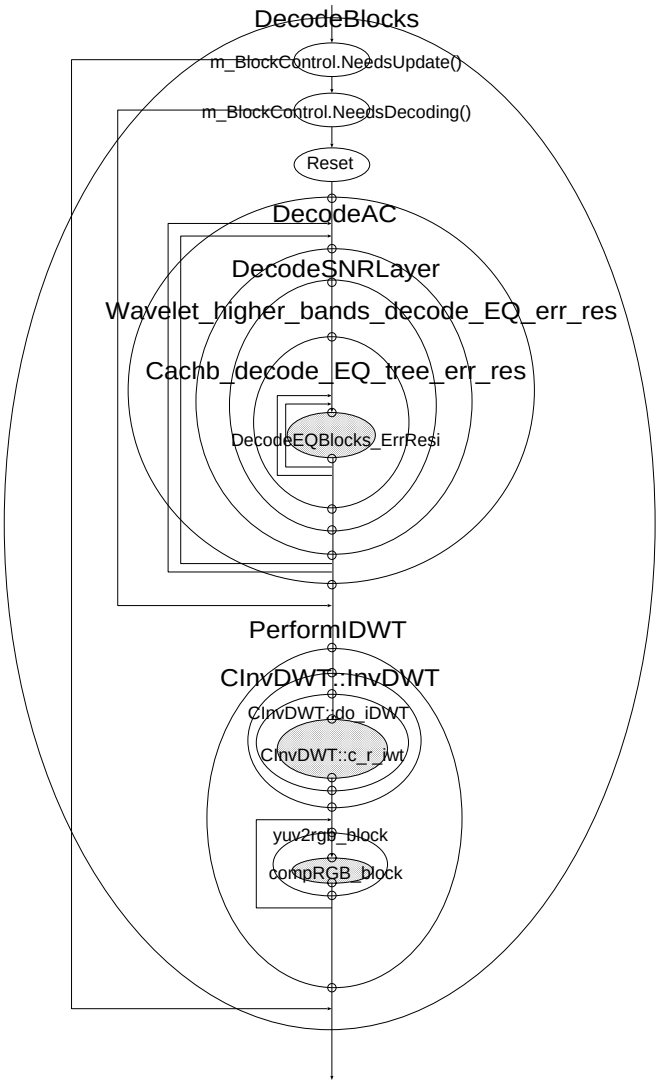

Fig. 3. Basic task graph

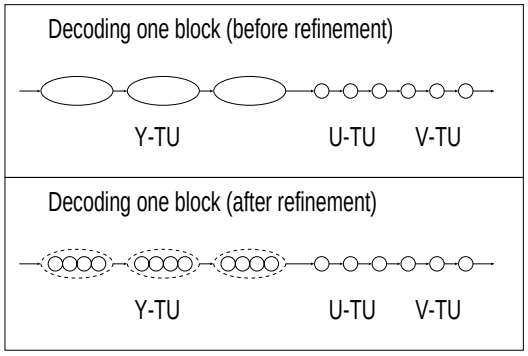

Fig. 4. Refining the sub-tasks

3.2.1. Refining the task graph

As mentioned in the subsection 3.1, one TU is decoded within each iteration of the identified bottleneck loop. The decoding procedure was a recursion function which is difficult to be further analyzed. To refine the analysis granularity in the bottleneck, we have simplified the task graph by transforming the recursion function into an iterative function. This transformation not only provides more details of the TU decoding in the task graph, it also gives us the freedom to balance the workload of decoding different types of TU's: because decoding a Y-element TU requires to process four times as many pixels as those for U- or V-element TU, we can now easily divide the Y-element decoding into four sub-tasks and still keep the U- or Y-element decoding as one sub-task. After such refinement, the decoding of one image block consists 18 balanced sub-tasks. This is illustrated in Fig.4.

3.2.2. Clustering the sub-tasks

The VTC decoding procedure has varying workload because when decoding one TU, certain numbers of its pixels may be skipped according to the content of the encoded image. Conventional ASIC chips have difficulties to handle this sort of dynamism efficiently in terms of energy consumption, because they are designed to work under the fixed worst scenario, i.e. the largest workload scenario.

The amount of workload of the VTC decoder depends on the content of encoded images and hence cannot be predicted at design-time, it is however possible to predict the approximate workload at run-time. According to the profiling data, we have observed that the workload is correlated for the TU decoding sub-tasks within the neighboring blocks. In other words, the workload of the TU decoding sub-tasks located in two neighboring image blocks does not

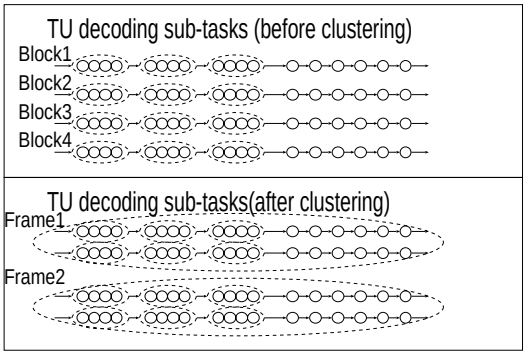

Fig. 5. Clustering the sub-tasks

have significant change. Therefore, if clustering the sub-tasks of two consecutive image block into one frame, we can predict the workload of the remaining sub-tasks after executing the first one. Then the Vdd/frequency scaling and the different processors assignment can be applied on each of these frames by using our design-time static scheduling tool. Such static scheduling should be applied for each workload scenario of a sub-task frame, then the operating system will choose an appropriate schedule for each running sub-task frame to reduce the overall system-wide energy consumption[9]. Fig. 5 illustrates these sub-task frames.

4. STATIC SUB-TASK SCHEDULING

After the task graph transformation, we now have identified a set of critical sub-tasks contained in a frame. Sub-tasks in such a sub-task frames have fixed workloads associated with a certain run-time scenario. For a sub-task frame under a given scenario, we will map and schedule its sub-tasks to different processors of a heterogeneous platform by using our static scheduling tool. The *time budget vs. energy consumption* trade-off space given by this tool will allow a system designer to choose a scheduling result which is appropriate to reduce the system-wide energy consumption. Moreover, this scheduling result always has the lowest energy consumption at the given time budget. We have developed and implemented the scheduling tool based on a set of scheduling heuristics [8]

5. VTC PLATFORM PROFILING

Many modern signal processing systems have varying workload by nature. Voltage scaling has been recognized as an efficient way to minimize energy consumption for such systems [17]. However, the effect of voltage scaling on one type of processors is still limited, given the wide range of varying workload for the VTC decoder. For example, it is observed that the peak workload is more than 20 times

	RISC	VLIW
Time ($\times 10^{-3}$Sec.)	0.75/4.00	1.13/3.98
Energy ($\times 10^{-3}$J.)	0.67/4.43	0.60/1.99

Table 1. Simulation results of two extreme workloads

higher than the typical workload in our simulations. An appropriate platform for such signal processing systems would consist heterogeneous processors, i.e. the high throughput VLIW processors and the cost-efficient RISC processors.

We assume a heterogeneous platform of four processors as the target platform for the static scheduling: two StrongARM-like [16] RISC processors and two TriMedia-like [15] VLIW processors combined with different Vdd and frequencies. To provide the energy consumption and execution time figures for the sub-tasks running on different processors, we simulate the VTC decoder code on two simulators: the Armulator for the profiling data on the RISC processors and the TriMedia simulator for the profiling data on the VLIW processors

The reference profiling data for RISC processors come from the simulation on a revised version of Armulator, which simulates a StrongARM processor running at 133MHz with a Vdd of 1.55V [13, 16] and calculates the energy consumption by using an instruction-level cycle-accurate energy model [18]. For processors running at different Vdd and frequencies, we derive the execution time and energy consumption figures by the following formulas: $P \propto V^2$ and $T \propto 1/V$, where P represents power, T represents execution time and V represents Vdd.

The reference profiling data for VLIW processors are achieved by simulating VTC decoder on the TriMedia simulator, which simulates a TM-1300 TriMedia processor running at 100MHz. We use an energy consumption model given in [15] to calculate the overall energy consumption for the VTC decoder code and then distribute the consumed energy to each sub-task according to its workload.

Since the execution time as well as the energy consumption of a sub-task will change under different scenarios, we have run the simulator for the most interesting scenarios. For simplicity, this paper only presents the profiling data under two scenarios: the lowest workload scenario and the highest one.

The profiling results of the sub-tasks in the TU decoding loop are shown in the Table 1.

6. SCHEDULING RESULTS AND DISCUSSION

The static scheduling is applied on a sub-task frame (Fig.5) whose sub-tasks are annotated with reference execution time and energy consumption figures. The target platform for the scheduling experiments has two StrongARM-like RISC processors running at 266MHz/3.1V and 171MHz/2.0V and

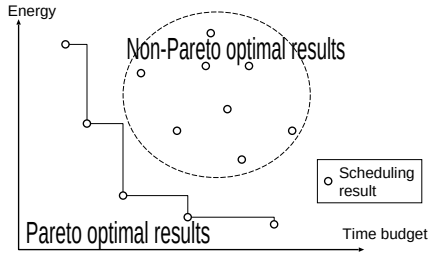

Fig. 6. Pareto optimal *vs* Non-Pareto optimal points

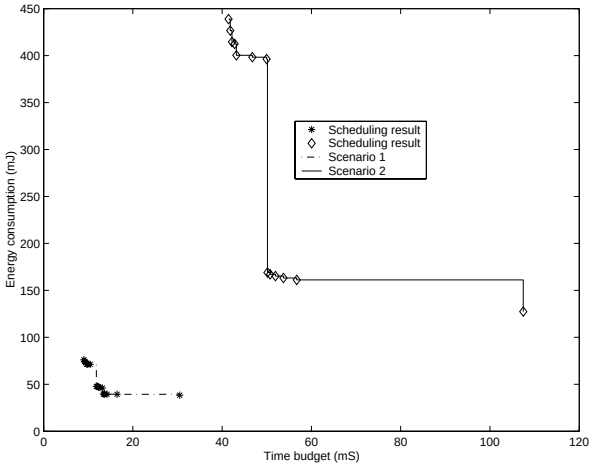

Fig. 7. Static scheduling results represented by the Pareo curves for 2 extreme types of image contents

two TriMedia-like VLIW processors running at 166MHz/2.5V and 133MHz/2.0V.

We call a scheduling result as a Pareo optimal result only if this result has the lowest energy consumption among all scheduling results that the scheduler has explored for a given time budget. This is illustrated in Fig.6. The scheduling results of two scenarios representing a very heavy and a very light weight in terms of the image block content are illustrated in the Fig.7. Each Pareto point in Fig.7 represents an optimal scheduling result. For example, Fig.8 illustrates a scheduling result represented by one Pareto point.

It is shown in Fig.7 that a system designer can trade off the energy consumption by a factor of 3 in both scenarios.

To evaluate the effect of the heterogeneous platform, we also made scheduling experiments for a platform with four RISC processors running at 3.1V, 2.4V, 2.2V and 2.0V, respectively. The scheduling results on this 4-RISC-platform are shown together with the results from the 2-RISC-2-VLIW-platform in Fig.9. Although the Pareto points achieved for these two different platforms in the lowest workload scenarios are very close, the heterogeneous platform provides the

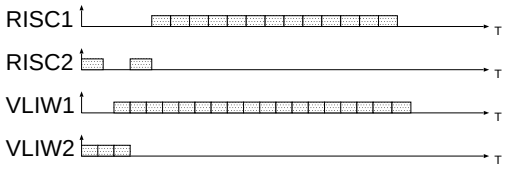

Fig. 8. One Pareto optimal scheduling result (RISC1 and VLIW1 have higher Vdd)

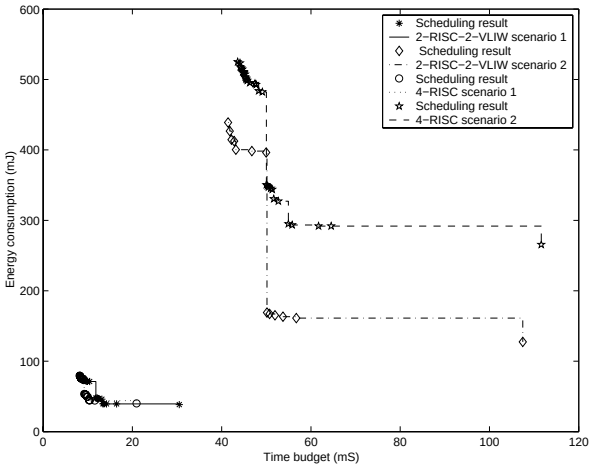

Fig. 9. Comparison of scheduling results from two different platforms

scheduling result with on average 55% lower energy consumption for a given time budget in the worst workload scenario.

7. CONCLUSIONS

Exploring the system-level timing-energy trade-offs is crucial for an optimal implementation of the VTC decoder. Due to the dynamic nature of the VTC decoding, identifying suitable sub-tasks is especially critical for the design-time static scheduler. We have identified the appropriate sub-tasks and clustered them into a scheduling frame by using a systematic methodology. Our static scheduling tool has successfully explored the trade-off space which would be infeasible to explore manually.

The task concurrency management analysis and optimization provides a system designer the capability to trade off timing and energy in order to find out the global energy optimal implementation. This task concurrency management approach combined with Vdd/frequency scaling is a valuable step toward applying the VTC decoding schemes on low-power mobile multimedia terminals utilizing the combination of heterogeneous programmable processors.

8. REFERENCES

[1] F.Thoen and F.Catthoor,“Modeling, Verification and Exploration of Task-Level Concurrency in Real-Time Embedded Systems”, Kluwer Academic, Boston,1999

[2] “MPEG4 Audio-visual Compression Standard”, Text of ISO/IEC 14496-5/FPDAM1,ISO/IEC JTC1/SC29/WG11/MPEG99/N3309, Noordwijkerhout, Mar. 2000

[3] G. Lafruit, et al.,“MPEG-4 Visual Texture Coding: Variform, yet Temperately Complex”, IWSSIP, Jun. 2001

[4] L. Nachtergaele, et al.,“Implementation of a scalable MPEG-4 wavelet-based visual texture compression system”, Design Automation Conference, LA, Jun. 1999

[5] G. Lafruit, et al.,“View-dependent, scalable Texture Streaming in 3D QoS with MPEG-4 Visual Texture Coding” , IEEE Tr. Circuits and Systems for Video technology, Special issue on AFX (submitted), 2002

[6] M. Boó, and T. Lang,“A VLSI architecture for arithmetic coding of multilevel images”, IEEE Tr. on Circuits and systemsII:Analog and Digital signal Processing, 1997

[7] B. Vanhoof, et al.,“A scalable architecture for MPEG-4 wavelet quantization”, Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology. Vol. 23, 1, Oct. 1999

[8] C. Wong, et al.,“Task Concurrency Management Methodology to Schedule The MPEG-4 IM1 Player on A Highly Parallel Processor Platform”, Proc. of the Ninth International Symp. on Hardware/Software Codesign, 2001

[9] P. Yang, et al.,“Energy-aware Run-time Scheduling for Embedded Multiprocessor SoCs”, IEEE Design and Test of Computers, Special ITC Issue, 2001

[10] P. Yang, et al.,“Managing Dynamic Concurrent Tasks in Embedded Real-time Multimedia Systems”, ISSS, Japan, 2002

[11] I. Sodagar, et al.,“Scalable Wavelet Coding for Synthetic/Natural Hybrid Images”, IEEE Tr. on Circuits and Systems for Video technology Vol 9 No.2, Mar 1999

[12] J. Shapiro,“Embedded Image Coding Using Zerotrees of Wavelet Coefficients”, Tr. on Signal Processing, 41(12), Dec. 1993

[13] Advanced RISC Machine Ltd.,“Developer Suite Version 1.2”, 2001

[14] Philips semiconductors,“TM-1300 High-speed Low-cost Enhanced PCI, VLIW Media Processor”, Hot Chip Symposium, 1999

[15] Philips semiconductors,“TM-1300 Power Consumption Study”, Feb. 2001

[16] Intel,“Intel StrongARM SA-1110 Microprocessor Brief Datasheet”, 2000

[17] A. Chandrakasan, et al.,“Data Driven Signal Processing: An Approach for Energy Efficient Computing”, ISLPED, USA, 1996

[18] T. Simunic, L. Benini, G.De Micheli,“Cycle-Accurate Simulation of Energy Consumption in Embedded Systems”, DAC, 1999