Resource Prediction for Media Stream Decoding

Juan Hamers Lieven Eeckhout
ELIS Department, Ghent University, Belgium
Email: {jmhamers,leeckhou}@elis.UGent.be

Abstract

Resource prediction refers to predicting required compute power and energy resources for consuming a service on a device. Resource prediction is extremely useful in a client-server setup where the client requests a media service from the server or content provider. The content provider (in cooperation with the client) can then determine what service quality to deliver given the client's available resources.

This paper proposes a practical approach to predicting resources for decoding media streams. The idea is to group frames with similar decode complexity from various media streams in the content provider's database into so called scenarios. Client profiling using scenario representatives characterizes the client's computational power. This enables the content provider for predicting decode time, decode energy and quality of service for a media stream of interest once deployed on the client.

1 Introduction

Resource prediction, *i.e.*, predicting client side compute power and energy required for consuming a service, has an important application in today's media systems. The content provider could inform a client about the required resources for consuming the requested service. This is especially relevant for battery-operated and resource-constrained clients. For example, if a complex high-resolution video stream is demanded, the content provider could warn the client that its resources will be insufficient for guaranteeing a given quality of service. Or, the client could be suggested that the available battery lifetime will be insufficient to view the entire video stream the client asks for; the client could then acquire the same video stream in a smaller resolution, or the content provider may do that automatically.

This paper proposes a resource prediction method that operates at the content provider side. The key idea of our proposal is to exploit the notion of similarities both within and across media streams. An offline analysis determines frames across the various media streams in the content provider's database that exhibit similar decode complexity, *i.e.*, require similar compute power and energy consumption at decode time. These similarly behaving frames are determined based on their encoding and are called scenarios. Scenario information is then maintained in the content provider's database for the various media streams. This scenario information is platform-independent, *i.e.*, is independent of the system configuration and/or decoder version of a particular client. Resource prediction is then performed by profiling scenario representatives on the client and by subsequently relating the profiling info to the frames in the media stream of interest.

The media stream decoding application that we specifically focus on in this paper is video decoding, however, our scheme is also applicable to other media stream processing applications. We apply scenario-aware resource prediction for predicting decode time, decode energy and quality of service. Our experimental results using the H.264 video decoder show good accuracy for all three purposes. For example, our method predicts decode time and energy with a worst case error less than 4% and an average 1.4% error.

This paper is organized as follows. We first discuss how scenarios are identified by the content provider. We subsequently discuss how this scenario information can be used for predicting required resources. After having detailed our experimental setup, we then evaluate the proposed scenario-aware resource prediction method.

2 Scenario identification

Figure 1 (on the left) illustrates how video streams are annotated with scenario information by the content provider. The content provider collects a macroblock profile for all video streams in his database. A *macroblock profile* counts the number of macroblocks of a given type for all frames in a video stream — note that a frame is built up from macroblocks and that each macroblock can be of a specific type or encoding style. The purpose of a macroblock profile is to characterize the decode complexity in a platform-independent way, *i.e.*, a macroblock profile is independent of the decoder as well as the system on which the video stream is to be decoded. A macroblock profile thus is a matrix in which the rows are the consecutive frames in the video stream and in which the columns are the macroblock counts for each of the macroblock types.

Once a macroblock profile is collected for all video streams in the database, all frames correspond to points in a multidimensional space, which we call the *frame space*,

978-3-9810801-2-4/DATE07 © 2007 EDAA

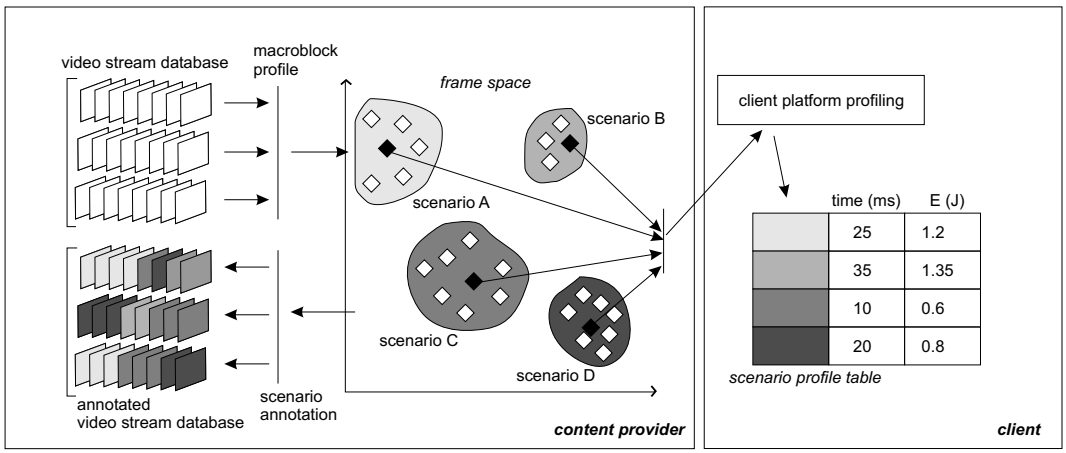

Figure 1. Annotating video streams with scenario information by the content provider (on the left), finding a scenario representative (in the middle) and filling the scenario profile table at the client side (on the right).

see Figure 1. The various dimensions in the frame space are the number of macroblocks of a given type; this is a 22-dimensional space in our experiments. We then apply cluster analysis in the frame space. Cluster analysis finds groups of frames, which we call *scenarios*, based on their macroblock characteristics. The idea is that frames belonging to a given scenario show similar macroblock characteristics, whereas frames from different scenarios show dissimilar macroblock characteristics. Note that cluster analysis is done on the collection of frames from all video streams in the database. As such, scenarios may be formed consisting of frames from different video streams.

The scenario identifiers are then to be maintained by the content provider in a scenario database.

Note that identifying scenarios is a one-time cost. Whenever a new video stream is added to the database though, a macroblock profile needs to be computed for the new video stream, and scenario identifiers need to be determined for all frames in the video stream.

We now discuss in more detail the following issues related to scenario identification by the content provider: (i) macroblock profiling, (ii) identifying the video stream scenarios using cluster analysis, and (iii) maintaining the scenario information for all video streams.

2.1 Macroblock profiling

Macroblock profiling captures a platform-independent image of the coding complexity at the frame level. Although the discussion below on macroblock profiling is geared towards the H.264/AVC decoder [18] that we target in this work, similar profiles can be computed for other types of media streams. A macroblock consists of 16×16 picture elements. We identify the following macroblock types — there are 21 macroblock types in total.

- An intra prediction macroblock only uses already transmitted macroblocks of the same image for predicting samples of the given macroblock. There are two flavors of intra prediction macroblocks, namely 16×16 (type 1) and 4×4 (type 2). The 4×4 macroblock consists of 16 4×4 subblocks which are separately encoded.

- An inter prediction or a motion compensated macroblock uses previously transmitted images for predicting samples of the given macroblock. An inter prediction macroblock can be divided into 4 partitions, namely 16×16 (type 3), 16×8 (type 4), 8×16 (type 5) and 8×8. The 8×8 submacroblock can be further subdivided into 8×4, 4×8 and 4×4 partitions. As such, we consider 15 macroblock types for 8×8 macroblocks depending on the partitioning within the submacroblocks.

- The final macroblock type is the skip macroblock type (type 21) which means that the current macroblock is the same as the corresponding macroblock of the previous frame.

For each of these 21 macroblocks, macroblock profiling computes the number of each of these macroblock types per frame. Note that not all of the above macroblock types appear for all frame types, *e.g.*, I frames do not contain inter prediction macroblocks. In such a case, the macroblock type count then equals zero.

Next to these macroblock type counts, macroblock profiling also measures the residual byte count per frame. (The residual bytes allows the decoder to correct the predictions based on the macroblock encoding.) We normalize the residual byte count in order to bring it in the range of the macroblock type counts — note that we will use the macroblock profile to build the frame space.

2.2 Video stream scenario identification

We identify video stream scenarios through cluster analysis in the frame space. Cluster analysis [15] is a data analysis technique that is aimed at clustering n cases, in our case video frames, based on the measurements of p variables,

in our case macroblock characteristics. The final goal is to obtain a number of groups, containing frames that exhibit 'similar' behavior, which we call scenarios. There exist two commonly used types of clustering techniques, namely linkage clustering and K-means clustering. We advocate K-means clustering because it is known to scale better with an increased number of cases which may be the case for large video stream databases. K-means clustering produces exactly *K* clusters with the greatest possible distinction. The algorithm works as follows. In each iteration, the distance is calculated for each case to the center of each cluster. A case is then assigned to the closest cluster. As such, new cluster centers can be computed. This algorithm is iterated until no more changes are observed.

2.3 Maintaining scenario identifiers

Maintaining scenario identifiers can be done using a scenario database or using a separate scenario stream associated with each respective video stream [2]. Since scenario information needs to be stored for all video streams, it is of primary importance to limit the storage cost at the content provider side.

3 Scenario-driven resource prediction

The scenario IDs annotated to the video streams can be exploited for predicting required resources. Before actually predicting the required resources on the client side, we first need to profile the client by building the scenario profile table, see Figure 1 (on the right). The *scenario profile table (SPT)* contains the decode time and/or decode energy (depending on what resource needs to be predicted) for decoding a frame belonging to each of the scenarios.

3.1 Client profiling

In order to enable efficient client profiling, the content provider needs to determine a *scenario representative* for each scenario, see Figure 1 (in the middle). The scenario representatives can be determined as part of the cluster analysis that the content provider needs to do for identifying the video stream scenarios. We select the frame that is closest to the cluster's centroid as the scenario representative.

The content provider sends these scenario representatives to the client for client profiling. The client then decodes these scenario representatives and monitors (i) how long it takes to decode each scenario representative and (ii) how much energy is consumed for decoding each scenario representative. Monitoring the decode time and consumed energy can be done using user accessible hardware performance counters that are available on modern microprocessors [14]. The end result of the client profiling process is a filled up SPT that summarizes the decode time and energy per scenario.

3.2 Resource prediction

For enabling scenario-driven resource prediction, the client has to communicate the SPT to the content provider, or alternatively, for popular clients, the SPT can be cached by the content provider. Predicting the required compute and energy resources for decoding a given video stream on the given client is then trivial. The content provider only needs to combine the scenario ID information for the video stream of interest with the information in the SPT. For example, if the goal is to estimate the amount of energy required for decoding a video stream on the given client, the content provider needs to make a weighted sum over the energy column in the SPT with the weights being the number of frames in a given video stream belonging to a given scenario. A similar method applies to estimating decode time; the total decode time is estimated by a weighted sum over the decode time column in the SPT. Yet another potential application is to estimate quality-of-service, *i.e.*, the number of missed frame deadlines when decoding the video stream at a given clock frequency. This is done by counting the number of frames in the video stream for which the decode time exceeds the frame decode deadline.

video stream	abbr	JVT class
akiyo	ak	A
coast guard	cg	B
container	con	A
foreman	for	B
hall monitor	hall	A
head w/ glasses	hd	A
mobile	mob	C
mother daughter	md	A
news	news	B
silent	sil	B
stefan	ste	C
table	tab	C

Table 1. The video streams and abbreviations used throughout the paper along with their JVT [3] classification.

Window ROB/LSQ	32/16
Cache hierarchy	64KB L1 I/D-caches, 1MB unified L2
Latencies (L1/L2/MEM)	L1: 2 cycles; L2: 20 cycles; MEM: 80ns
Branch predictor	Hybrid 4K tables, 3 cycle front-end pipeline
Processor width	4-wide
Functional units	4 integer ALUs, 2 memory ports

Table 2. Baseline processor model considered in this study.

4 Experimental setup

Our experiments are done using the H.264 Advanced Video Coding (AVC) codec [18]. AVC is the new generation compression algorithm for consumer digital video. In our measurements we use version JM6.1 of the reference software of the JVT/AVC codec [1].

In our evaluations we use twelve video sequences, see Table 1. Each video sequence contains 297 frames; this corresponds to approximately 10 seconds of video at a decode

rate of 30 frames per second. The results presented in this paper are obtained for these video streams in CIF resolution (352×288 pixels per frame). Further, we consider content-adaptive variable-length coding (CAVLC) and a IPPPP... GOP structure, *i.e.*, there is one I frame followed by 15 P frames.

The performance results presented in this paper were obtained using detailed cycle-level processor simulations using the SimpleScalar/Alpha v3.0 tool set [5]. Microprocessor energy consumption is estimated using Wattch v1.02 [4]. These simulation tools were extended to model frequency scaling as well as voltage scaling. When applying both frequency and voltage scaling we vary voltage with frequency based on $f \sim \frac{(V-V_{th})^{\alpha}}{V}$ [13] using 100MHz frequency steps. We also model the time cost for changing the processor operating frequency. The baseline processor model used in this paper is a contemporary 4-wide superscalar microarchitecture, *i.e.*, up to four instructions can be issued per cycle, see Table 2. We also evaluated our scenario-based resource prediction technique on other processor architectures, both wider and narrower microarchitectures, and obtained very similar results. These results are not reported here because of space constraints.

Note that in all of our experiments, we assume a leave-one-out methodology. This means that when evaluating the prediction accuracy for a given video stream we leave that video stream out of the content provider's database for building the scenarios. This reflects what is to be expected in practice whenever a new video stream is added to the content provider's database.

5 Evaluation

We now evaluate the proposed scenario-based resource prediction method. We consider three flavors of resource prediction: predicting decode time, predicting quality of service and predicting energy consumption. Finally, we also evaluate the storage cost at the content provider side for maintaining the scenario IDs. In this paper, we assume 32 scenarios which we found to achieve good prediction accuracy. We do not include a detailed exploration of the impact of the number of scenarios here because of space constraints.

5.1 Predicting decode time

We first evaluate the prediction accuracy for predicting the decode time for a given video stream on a given client. When assuming a 2GHz clock frequency at the client side[1], the decode time for decoding the entire video stream can be predicted with an error of at most 4% and an average error of 1.4%, see Figure 2. Figure 3 gives a more detailed view by showing the absolute prediction error for predicting decode times for individual frames. The graph shows the cumulative distribution function of per-frame prediction errors. For example, 90% of all frames have a decode time

[1]We have to set the client clock frequency to 2GHz in order to meet the 30ms frame deadlines using the reference decoder.

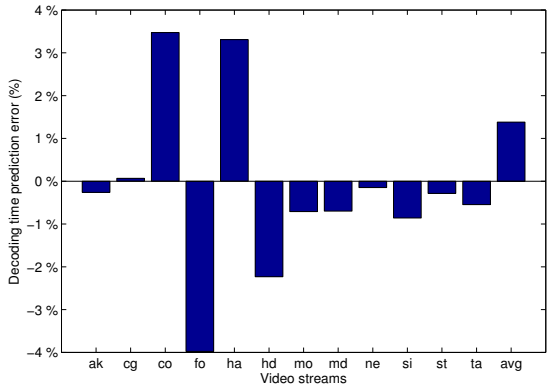

Figure 2. Predicting video stream decode times.

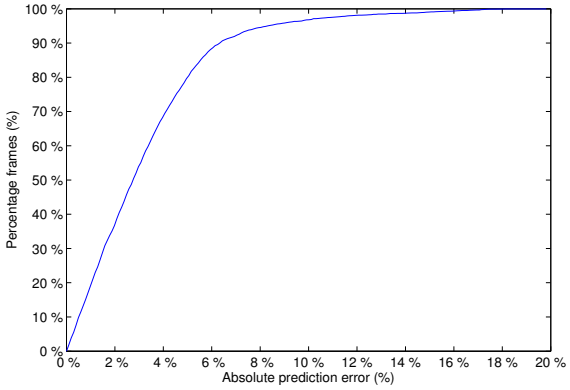

Figure 3. Predicting individual frame decode times. The percentage frames (on the vertical axis) for which the decode time prediction error is less than a given percentage (on the horizontal axis).

prediction error of less than 6.3%. Or, 96.8% of all frames have a decode time prediction error of less than 10%.

5.2 Predicting quality of service

Resource prediction can also be used to predict quality of service. More in particular, the content provider can predict whether the client platform is computationally powerful enough for decoding a given video stream. In other words, the content provider can make an estimate of the number of frame deadlines that will be missed by the client. Figure 4 quantifies the accuracy for predicting the quality of service. The two curves show the real percentage missed frame deadlines and the predicted percentage missed frame deadlines as a function of the client's clock frequency. When running the client's clock frequency at 1.2GHz, approximately 40% of all frame deadlines will be missed; however, at 2.5GHz, no deadlines will be missed. This graph clearly shows that the estimated number of missed frame deadlines matches the real number of missed frame deadlines very closely over the entire client's clock frequency range. The

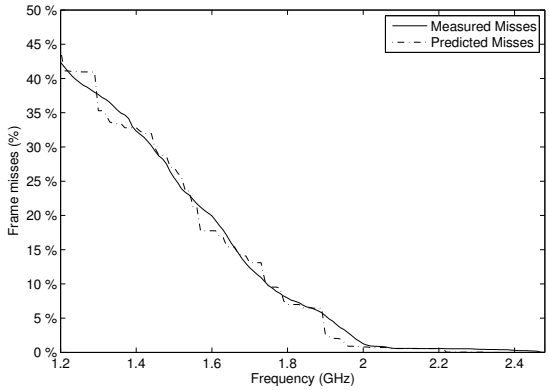

Figure 4. Predicting the number of missed frame deadlines (in 10MHz steps on the horizontal axis).

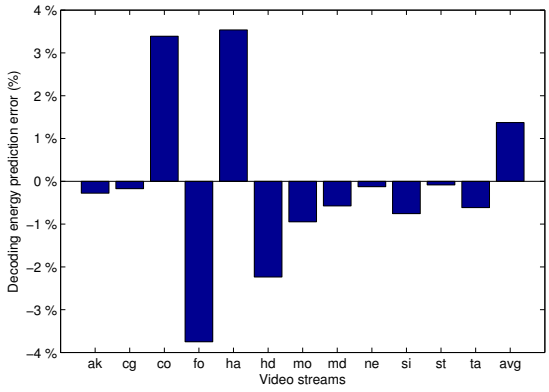

Figure 5. Energy prediction error for a 2 GHz client clock frequency.

prediction error is never larger than 4%.

5.3 Predicting energy consumption

We now evaluate the accuracy of scenario-based video stream analysis for predicting the overall energy consumption. The results are shown in Figure 5 for decoding at a 2GHz clock frequency — we obtained similar results over the entire 1.2GHz to 2.5GHz range. The average (absolute) prediction error is 1.4%; the maximum prediction error is observed for the container, foreman and hall monitor video streams (3.5% to 3.8%).

The above results were obtained when decoding a video stream at a fixed clock frequency. However, recent research has considered applying dynamic voltage and frequency scaling (DVFS) to reduce the energy consumed in video stream decoding. The idea is to lower the clock frequency and supply voltage so that energy consumption is minimized while still meeting the frame deadlines. Scenario-aware resource prediction can also be used to predict the energy consumed by such energy-efficient video decoding solutions. Figure 6 shows the error when predicting the amount of energy consumed for decoding the given video

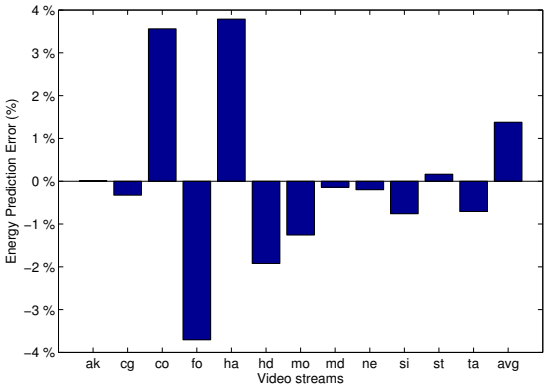

Figure 6. Energy prediction error for a DVFS-aware client.

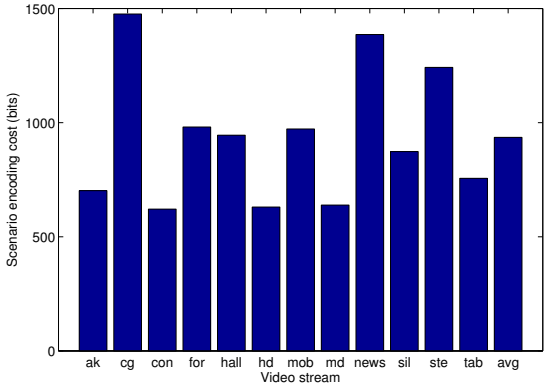

Figure 7. Total storage cost per video stream for maintaining scenario IDs.

stream using the DVFS-driven video decoding method presented in [10]. The average prediction error is 1.4%; the maximum prediction error is never larger than 4%.

5.4 Cost

Scenario-aware resource prediction incurs a cost at the content provider side, namely scenario IDs need to be maintained for all video streams in the database. The average cost per video stream assuming run-length encoding is less than 1K bits, see Figure 7. This corresponds to less than 100 bits per second on average. Or, a one-hour video only requires approximately 45KB of storage. The cost for uncompressed scenario information is approximately 66KB per hour of video. This is an overhead of less than 0.02%.

6 Related work

Video decoder complexity analysis. Horowitz *et al.* [11] present a detailed platform-independent video decoder complexity analysis of the H.264/AVC video decoder. The characterization is done in terms of macroblock types and the number of fundamental operations required to perform the key decoder subfunctions.

Mattavelli and Brunetton [17] also characterize video streams in a platform-independent manner by counting the number of inter prediction macroblocks, the number of intra prediction macroblocks, the number of skip macroblocks, *etc*. For predicting the decoder performance on a particular platform, they then employ linear regression to learn how to weight the platform-independent metrics for making decode time predictions. There are two important limitations with this approach. First, Mattavelli and Burnetton require "a relatively large set of video sequences with associated measurements". This is impractical on resource-constrained embedded devices as profiling the client may take considerable time. In case of scenario-based resource prediction on the other hand, client profiling is done very quickly using only a few frames, namely the scenario representatives. Second, the storage requirements on the content provider side for Mattavelli and Brunetton's technique is very large (at least two orders of magnitude larger) compared to scenario-based resource prediction. Mattavelli and Brunetton need to store regression weights for all platform-independent metrics per frame; this would be hundreds of bits per frame in our setup. We only need a few bits for encoding the scenario ID per frame.

Energy-efficient video decoding. Several proposals to energy-efficient video decoding have been made in recent literature [6, 7, 8, 12, 13, 16]. Several of these proposals do an online prediction of how fast the client's clock frequency needs to be set in order to minimize energy and still meet the frame deadlines. However, these approaches do not make a priori resource predictions at the content provider's side.

WCET estimation. Gheorghita *et al.* [9] use scenarios for worst case execution times (WCET) estimation. WCET estimation is a different problem than resource prediction. WCET estimation needs formally proved execution time bounds; resource prediction on the other hand strives at making realistic performance, QoS and energy estimates.

7 Conclusion

Resource prediction is useful in media stream applications where content providers deliver services to clients. This paper proposed scenario-aware resource prediction for predicting decode time, decode energy and quality of service. The key idea of this approach is to identify platform-independent scenarios across the media streams in the content provider's database. A scenario collects frames with similar decode complexity. Our experimental results using the H.264 video decoder showed accurate resource predictions. Per-stream decode time and energy can be predicted with an average error of 1.4% and a maximum error of 4%. Predicting the number of missed frame deadlines can be done with an error of at most 4%.

Acknowledgements

Juan Hamers is supported by a BOF grant from Ghent University. Lieven Eeckhout is a Postdoctoral Fellow with the Fund for Scientific Research–Flanders (Belgium) (FWO–Vlaanderen). This research is also partly supported by the IWT, Ghent University and the HiPEAC Network of Excellence.

References

[1] H.264/AVC reference software. `http://iphome.hhi.de/suehring/tml/download/`.

[2] Information technology – coding of audio-visual objects – part 14: Mp4 file format. ISO/IEC 14496-14:2003.

[3] MPEG-4 video verification model version 18.0. ISO/IEC JTC1/SC29/WG11 N3908, Jan. 2001.

[4] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *ISCA*, pages 83–94, June 2000.

[5] D. C. Burger and T. M. Austin. The SimpleScalar Tool Set. Computer Architecture News, 1997. See also `http://www.simplescalar.com` for more information.

[6] K. Choi, K. Dantu, W.-C. Cheng, and M. Pedram. Frame-based dynamic voltage and frequency scaling for a MPEG decoder. In *ICCAD*, pages 732–737, Nov. 2002.

[7] E.-Y. Chung, L. Benini, and G. De Micheli. Contents provider-assisted dynamic voltage scaling for low energy multimedia applications. In *ISLPED*, pages 42–47, Aug. 2002.

[8] S. V. Gheorghita, T. Basten, and H. Corporaal. Intra-task scenario-aware voltage scheduling. In *CASES*, pages 177–184, Sept. 2005.

[9] S. V. Gheorghita, S. Stuijk, T. Basten, and H. Corporaal. Automatic scenario detection for improved WCET estimation. In *DAC*, pages 101–104, June 2005.

[10] J. Hamers, L. Eeckhout, and K. De Bosschere. Exploiting video stream similarity for energy-efficient decoding. In *MMM*, Jan. 2007.

[11] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro. H.264/AVC baseline profile decoder complexity analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):704–716, July 2003.

[12] Y. Huang, S. Chakraborty, and Y. Wang. Using offline bitstream analysis for power-aware video decoding in portable devices. In *MM*, pages 299–302, Nov. 2005.

[13] C. J. Hughes, J. Srinivasan, and S. V. Adve. Saving energy with architectural and frequency adaptations for multimedia applications. In *MICRO*, pages 250–261, Dec. 2001.

[14] C. Isci and M. Martonosi. Runtime power monitoring in high-end processors: Methodology and empirical data. In *MICRO*, pages 93–104, Dec. 2003.

[15] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, fifth edition, 2002.

[16] Z. Lu, J. Hein, M. Humphrey, M. Stan, J. Lach, and K. Skadron. Control-theoretic dynamic frequency and voltage scaling for multimedia workloads. In *CASES*, pages 156–163, Oct. 2002.

[17] M. Mattavelli and S. Brunetton. Implementing real-time video decoding on multimedia processors by complexity prediction techniques. *IEEE Transactions on Consumer Electronics*, 44(3):760–767, Aug. 1998.

[18] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi. Video coding with H.264/AVC: Tools, performance and complexity. *IEEE Circuits and Systems Magazine*, 4(1):7–28, Jan. 2004.