Resource Estimation Methodology for Multimedia Applications

Hari Kalva, Ravi Shankar, Tuhina Patel and Camilo Cruz
Dept. of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431

ABSTRACT

Reducing the product development cycle time is one of the most important and challenging problems faced by the industry today. As the functionality and complexity of devices increases, so does the time required to design, test, and develop the devices. Developing products rapidly in the face of this increasing complexity requires new methodologies and tools. This paper presents a methodology for estimating the resources consumed by a video decoder. The proposed methodology enables resource estimation based on high level user requirements. Component architecture for a H.264 video decoder is developed to enable design space exploration. The resources required to decode H.264 video are estimated based on a measure of the complexity of the H.264 bitstreams and the target architecture. The proposed approach is based on the hypothesis that the complexity of a H.264 video bitstream significantly influences resource consumption and the complexity of a bitstream can thus be used to determine resource estimation. The bitstream complexity is characterized to capture the data dependencies using a process called Bitstream Abstraction. The decoder is componentized and component level resource requirements determined in a process called Decoder Abstraction. The proposed methodology uses Bitstream Abstraction together with Decoder Abstraction to estimate resource requirements. A component model for the H.264 video decoder is developed. Resources consumed by each component are determined using the VTune performance analyzer. These resource estimates and video bitstream complexity are used in developing a parametric model for resource estimation based on bitstream complexity. The proposed methodology enables high level resource estimation for multimedia applications without a need for extensive and time consuming simulations.

Keywords: Resource estimation, H.264, components, multimedia, architecture

1. INTRODUCTION

The continuing advances in computing and communication technologies are expected to make digital video communication an integral part of next generation information systems. Most of the mobile devices today have built-in cameras and an increasing number are expected to have some basic video capability. User expectations for mobile devices are also continually increasing. What was a feature in a high-end mobile phone a year ago is likely to become a standard feature on next generation phones. With changing user expectations and given the diversity of user preferences and requirements, new devices that meet specific user needs have to be continually developed; one-size-fits-all policy will not work. Device manufacturers have to quickly develop and ship devices to satisfy user demands. The current product development cycle of 24 months has to be drastically reduced to be able to conceive and ship products in a matter of few weeks or less. Such a drastic reduction in product development cycle time requires new approach to product planning and design.

One of the keys to reducing the design cycle time is the ability to determine the resources required to run an application on the device. The resource requirements have to be determined *before* a device is designed and without the extensive and time consuming simulations. With most of the design practices today, devices are first built based on rough specifications and applications and features are added or dropped depending on the platform capability. This approach leads to over design if designer is conservative or under design in which case features have to be reduced. Such a design approach results in inefficient use of resources and leads to longer cycle times. Figure 1 depicts the larger goal of this project. Given user requirements for video, our methodology and tools provide quick estimates of resource

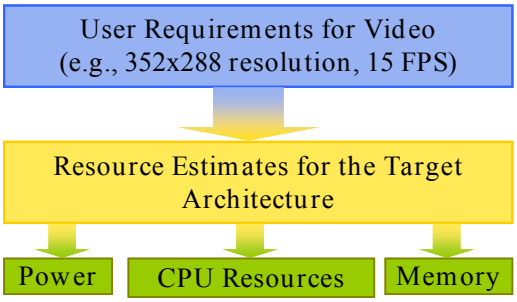

Fig 1. Resource estimation methodology

This work was supported by Motorola under the *One Pass to Production* project

Multimedia Computing and Networking 2007, edited by
Roger Zimmermann, Carsten Griwodz, Proc. of SPIE-IS&T Electronic Imaging,
SPIE Vol. 6504, 65040L, © 2007 SPIE-IS&T · 0277-786X/07/$15

requirements. By considering the application and systems at a very high level, we are likely sacrificing accuracy for speed. However, the reduced accuracy at the architects level is a non-factor in the over all product design.

2. BACKGROUND

Power estimation has been widely studied at various levels of abstraction; from gate level to the system level [1]. A resource estimation model for DFT cores is presented in [2]. This model relies on the RTL level Verilog descriptions of the DFTcore. This is a too low a level from an architect's point of view. Our goal is to enable an architect to select components to build the system based on the resource estimates. A simulation model for power estimation for video encoders is reported in [3]. This approach uses a model exclusively developed for designing video coprocessors and is not applicable to resource estimation on general processors or DSPs. A methodology for estimating power consumption in FPGAs is presented in [4]. This approach does not address the application specific power consumption estimation. Estimating power consumption due to software execution has been well studied [5,6,7,8]. There is limited amount work in the area of power aware video playback and the existing work primarily deals with system level adaptation such as frequency scaling [9,10]. Power consumption due to the display activity was studied in [11]. These approaches do not consider the data dependencies inherent to video decoders and cannot be used directly for power estimation. These approaches also require a completely defined model of the target platform to estimate the power. At the architects' level, most of the system details are abstracted and new approaches are necessary. Furthermore, while power is a key concern, computing requirements (instructions, processor speed, memory, latency) are more important at the architects' level.

Our resource estimation methodology takes data dependencies in to consideration and enables resource estimation including power and computing resources.

3. METHODOLOGY AND TOOLS

The proposed methodology is developed with the H.264 video decoder as an example. The key factors that influence the proposed approach are: 1) data dependencies and 2) target architecture. The proposed approach abstracts the data dependencies using a process called Bitstream Abstraction. The decoder is componentized and component level resource requirements determined in a process called Decoder Abstraction. The proposed methodology uses Bitstream Abstraction together with Decoder Abstraction to develop a resource estimation model. In this section we first introduce the elements of the methodology and then describe the integrated flow in Section 4.

3.1. H.264 Video Decoder

The H.264 video coding standard has been developed recently through the joint work of the ITU's video coding experts group (VCEG) and ISO motion pictures experts group (MPEG) [12,13]. The H.264 video coding standard is flexible and offers a number of tools to support a range of applications with very low as well as very high bitrate requirements. The H.264 video uses the same hybrid coding approach that is used in the other MPEG video standards: motion compensated transform coding. Figure 2 depicts the main components of the H.264 video decoder. A component model of the H.264 video decoder was developed based on the functional partitioning of the decoder. The key components of the decoder modeled for resource estimation purpose are: 1) entropy decoder (VLD), 2) Inverse transform and quantization, 3) Intra prediction, 4) Inter prediction, and 5) Deblocking.

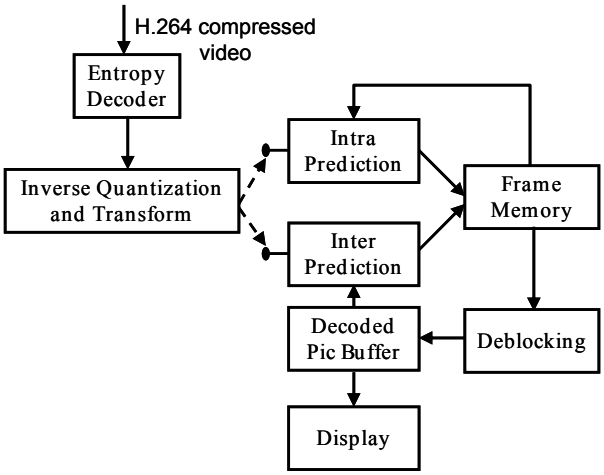

Fig. 2. H.264 Video Decoder

3.2. VTune Performance Analyzer

The Intel VTune Performance Analyzer is a software application that helps locate performance bottlenecks in code, i.e. it locates and displays portions of code with a large amount of activity over a given time [14]. In our methodology,

VTune is used to model high level resource requirements (instructions, cycles, reads, writes, and cache). The H.264 video decoder is built and run in VTune environment to estimate the computing resources required. A power estimation methodology is developed that relies on transitions on the data and address buses. Since VTune does not provide a means of tracing the bus transitions, a second tool, Armulator, was used to model power dissipation.

3.3. Armulator

Armulator is an ARM instruction set simulator that can give a detailed trace of the address and data bus transactions. Armulator is used to run the H.264 video decoder and gather bus transactions. These bus transactions are then used to determine the number of transitions (0 to 1 or 1 to 0 transition) on the data and address bus. A larger portion of power dissipated can be attributed to the bus activity [15], modeling and measuring the transitions gives an estimate of power dissipation. A similar approach was used in [16] to measure bus transitions and then use alternative coding such as Grey codes to reduce the power dissipation.

3.4. Bitstream Abstraction

The Bitstream Abstraction as used in this paper refers to the characterization of a compressed bitstream with a few parameters that significantly influence the amount of computing resources required to decode the bitstream. Content dependencies are inherent in video coding and resources required to encode/decode a video also depend on the content as well as the quality of the video. Complexity description should take these dependencies into consideration and use metrics that are content independent. Furthermore, complexity estimation and description should use metrics that will allow the estimation of the actual computing resources required for specific device architecture. The computing resources required to execute a program can be described in terms of the processor usage, execution time, memory usage, and power consumption. The amount of such resources required varies depending on the underlying hardware architecture.

A bitstream complexity can be characterized at different levels of abstraction: frame level to pixel level. A detailed analysis of complexity estimation is presented in our prior work [17]. The complexity analysis of the H.264 video decoding reported by Horowitz et. al. discusses the implementation complexity of the decoder and does not address the receiver resources and the receiver complexity description [18]. The H.264 video profile and level information represent the upper bound on the amount of resources required and do not reflect the actual complexity. The actual complexity is typically much less than the upper bound complexity implied by the particular profile and level used. Furthermore, even when a decoder is designed to process video at a certain profile and level, the current available resources may not be sufficient to process the video. A good estimate of complexity is necessary that allows a good estimation of resource requirements. For the resource estimation model we abstract the bitstream complexity with the following metrics given per frame:

1. Average IntraMB 16x16 – the average number of Intra 16x16 MBs per frame
2. Average IntraMB 4x4 – the average number of Intra 4x4 MBs per frame
3. Average Inter MB 16x16 – the average number of Inter 16x16 MBs per frame
4. Average Inter MB 16x8 – the average number of Inter 16x8 MBs per frame
5. Average Inter MB 8x16 – the average number of Inter 8x16 MBs per frame
6. Average Inter MB 8x8 – the average number of Inter 8x8 MBs per frame
7. The average number of skipped MBs per frame
8. Average Non Zero Coefficients per frame

These metrics for a H.264 video bitstream depend on the content being encoded and the quality of the content. Resources required to decode a H.264 bitstream vary with the quality of the bitstream since the bitrate can be assumed to be proportional to quality for rate-distortion optimized encoding. The variation in the metrics has a strong correlation to the variation in the resources consumed by the decoder and hence decoder resource requirements can be estimated based on the Bitstream Abstraction.

3.5. Decoder Abstraction

The Decoder Abstraction is the process of representing the decoder complexity with target platform independent metrics. We abstract the decoder complexity by abstracting the complexity of the components in the decoder. The component complexity description should enable resource estimation for a given architecture. Since the resources

needed for running a tool vary depending on the target architecture, the tool complexity has to be described using target-independent metrics. One approach is to use the number and type of instructions that are required to implement the component functionality – instructions per component (IPC). The number of instructions required to achieve a given functionality are typically independent of a given architecture. The target dependencies can be described by specifying the average number of instructions retired per cycle (IRC) that are target specific.

We are currently using three metrics: instructions, reads, and writes. For the high level resource estimation we are considering, further sub-classifying the instructions is not necessary [19]. We will explore the issue of sub-classification in the future work. Since memory has significant impact on performance, the complexity description should also include the memory requirements. Since most of the video compression algorithms are block based and use macro blocks as a basis for coding, a block based metric, i.e. complexity per block, can be used to describe component complexity.

The time complexity of a component on a target architecture is given by, $C_T = (IPC \times IRC \times f) \times F_C$. F_C is the frequency of component use and f is the clock frequency. The time complexity however represents the upper bound as it does not take the content dependencies into consideration. The component use frequency, F_C, also depends on the content being encoded. The data dependencies usually result in a lower complexity. For example, consider the inverse discrete cosine transform (IDCT) that is applied on decoded DCT coefficients. As the number of zeros in the decoded DCT coefficients increase, the complexity of the IDCT decreases because of the multiplications involving a zero. This data dependency has to be considered in resource estimation. Data dependencies represented using the Bitstream Abstraction process are used to estimate the actual resources. For the case of the IDCT component, the number of non-zero coefficients per block characterizes the dependency well and IPC can be described as a function of the number of non-zero coefficients per block.

4. Integrated Methodology

The Bitstream Abstraction and Decoder Abstraction form the key elements of the methodology for resource estimation. The methodology is used to develop a resource estimation model; the resource estimation model is then used to estimate the resources based on user level requirements. The process flow in the proposed methodology is shown in Figure 3. The steps followed in this methodology are:

1. User requirements are given in terms of video resolution, frame rate, and minimum/maximum expected quality
2. H.264 bitstreams are generated at various bitrates for typical test video sequences for the domain
3. The bitstream are decoded and complexity is estimated in the Bitstream Abstraction process
4. The bitstreams are decoded using the VTune performance analyzer
5. Based on the component model and the VTune estimates, component level resources are derived
6. The bitstreams are decoded with the decoder running in the Armulator
7. The bus transitions gathered in the Armulator are used to determine power dissipation
8. A parametric model for resource estimation is derived exploiting the correlation between resource consumption and component complexity

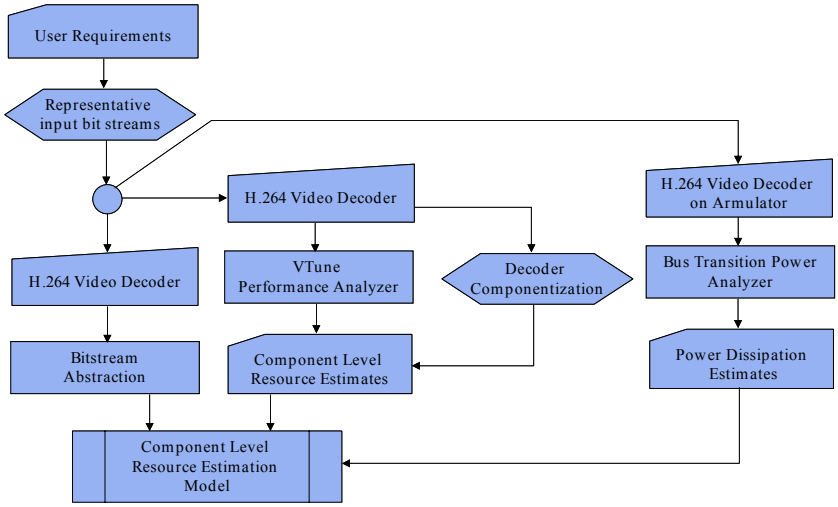

Fig. 3. Process flow for the resource estimation methodology: model development

We are currently working on developing parametric models (step 8). The results presented in this paper show a strong correlation between the resources consumed and bitstream

abstraction. Once the model is developed, the parametric model is used to estimate the resources. Figure 4 shows the methodology for estimating the resources using the model developed. The steps followed in this methodology are:

1. User requirements are given in terms of video resolution, frame rate, and minimum quality
2. H.264 bitstreams are generated at various bitrates for typical test video sequences for the domain
3. The bitstream are decoded and complexity is estimated in the Bitstream Abstraction process
4. Traget platfom dependencies are modeled (e.g., IRC, clockspeed, cache)
5. Component level resource estimates are obtained.

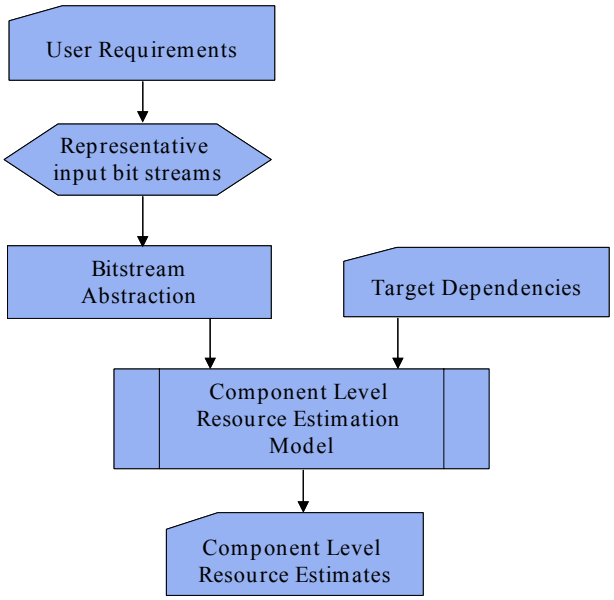

Fig. 4. Process flow for the resource estimation methodology: model application

One of the key tasks at architectural level design is the design space exploration to understand the resource consumption by a component on different targets. For example, estimating the resource consumption by the deblocking filter component of the H.264 video decoder on the processor vs. DSP. Design space exploration is performed by using the same model and updating the target dependencies. This approach allows quick evaluation of available alternatives.

5. RESULTS AND DISCUSSION

Nokia H.264 baseline encoder and decoder were used at different bit rates to develop a model for resource estimation. Nokia H.264 employs frame and macroblock level rate control. The initial QP of the encoded frame is decided at the frame level, and QP is updated at the macroblock level whenever necessary. For constant QP case, where a rate control is not employed, the encoded video quality is constant, but the number of bits of each frame varies significantly depending on the frame's complexity. When the rate control is in use, the number of bits for each frame is kept within a certain limit and resulting in variable quality streams. The same decoder was ported to run on ARM 9 processor in the Armulator environment to perform power dissipation estimates. Three different videos, Akiyo, Foreman, and Football, with same resolution of 176x144 and 15 Fps were used for the experiment. These three vides represent typical video that one can expect in mobile devices. The video sequences were encoded using H.264 baseline profile, with one I and all P frames, at 9 different bitrates from 15 Kbps to 740 Kbps. The metrics used and results shown are per frame and results can be extended to other frame rates assuming a linear relationship between frame rate and resource consumption.

The results of the proposed approach are shown in Figures 5 – 8. Figure 5 shows the instructions per frame for each of the five components of the H.264 video decoder. The five components account for over 95% of the resources consumed and the remaining are reported as "Other" in the plot. As expected, the resources consumed (instructions) increases as the bitrate increases for all the components except for the deblocking filter; this is due to increase in complexity because of the increase in the quality and bitrate. For the deblocking filter the complexity drops at higher bitrates since the quality of video is very high at higher bitrates and the deblocking strength is reduced. The plot also shows the relative complexity of the components. Inter frame decoding takes up the most resources because of large number of inter macroblocks in the sequence. The relative component complexity is also useful in design space exploration and resource allocation based on the resource requirements. Figure 6 shows the Bitstream Abstraction for the same bitstreams. These bitstream characteristics vary with the bitrate and give a measure of the bitstream complexity and hence resource requirements. These component level resource estimates and bitstream abstraction can be exploited to estimate the resources. The units on the Y-axis are normalized to see how closely the metrics track each other and understand the correlation between the metrics.

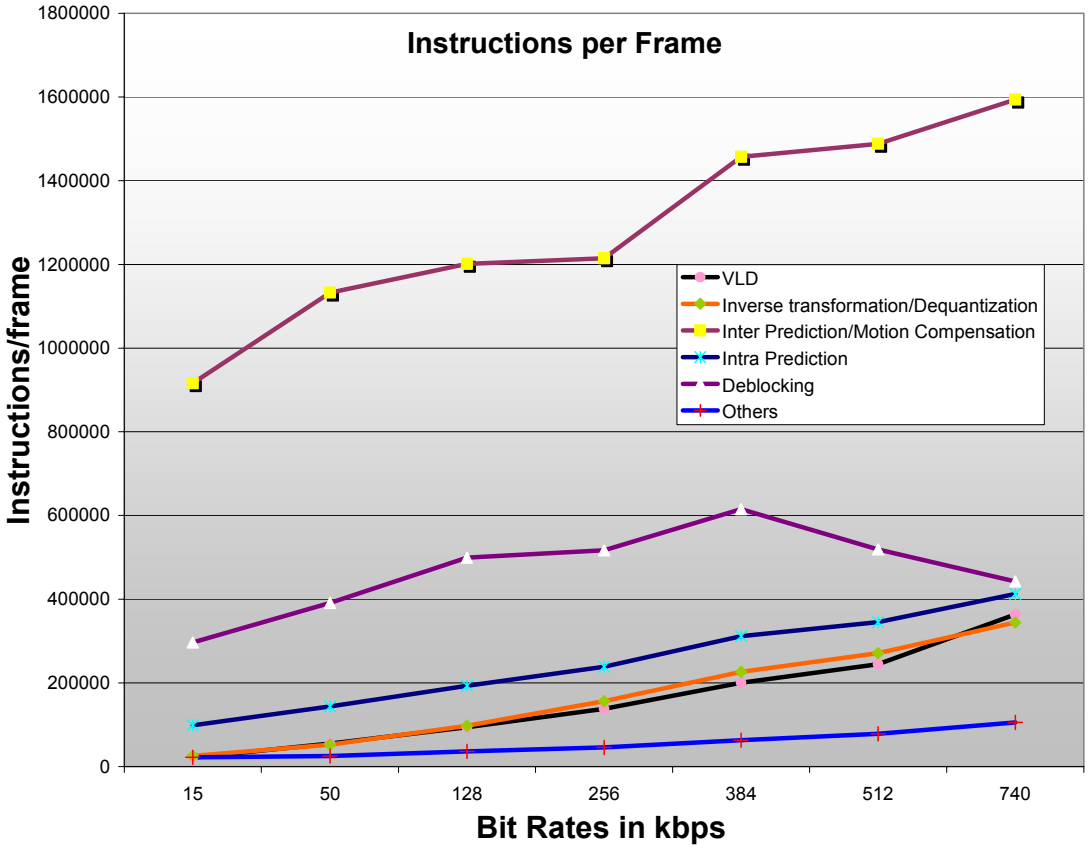

Figure 5. Instructions per frame per component for the Football sequence

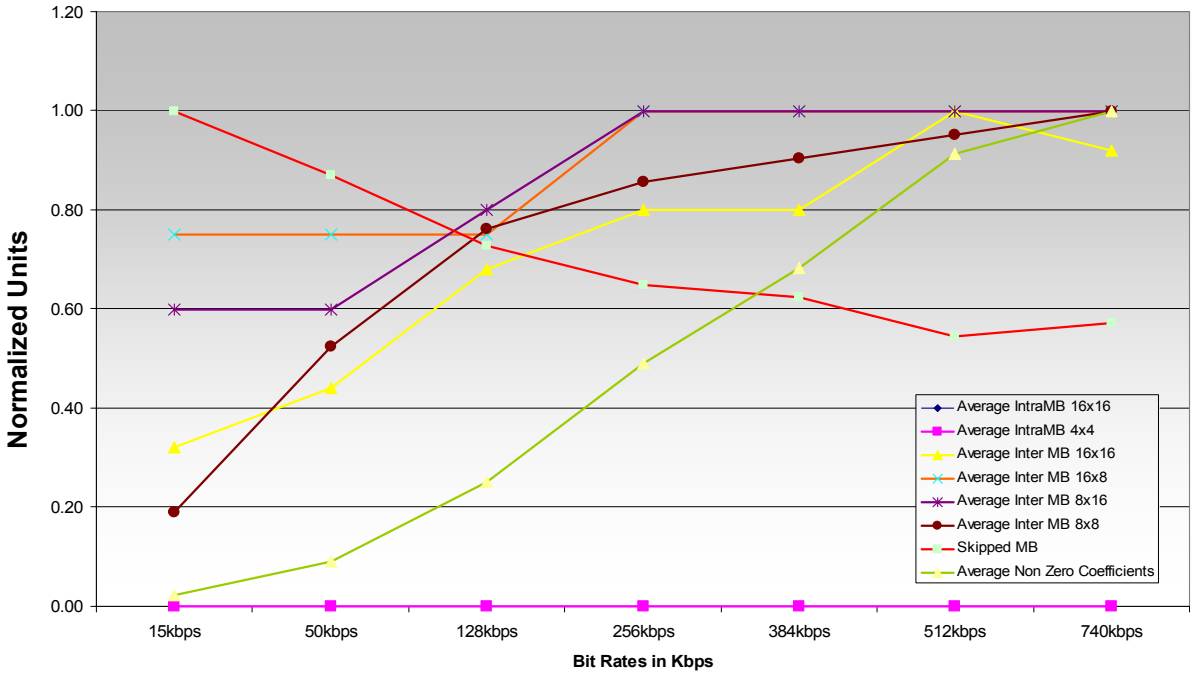

Figure 6. Bitstream Abstraction for the Football sequence

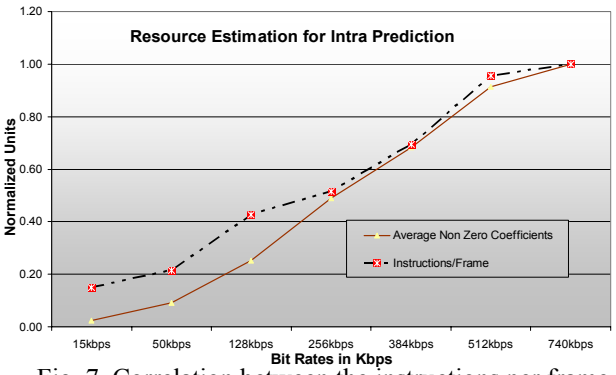

Fig. 7. Correlation between the instructions per frame and the average number of non-zero coefficients per frame

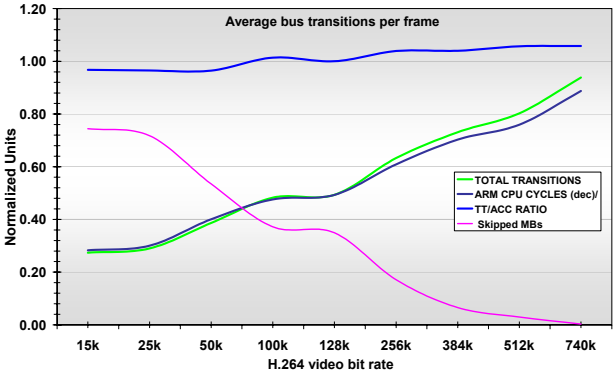

Fig. 8. Correlation between the bus transitions per frame, and the average number of skipped macroblocks per frame

Figure 7 shows the normalized plots for the instruction per frame and the average non-zero coefficients for the Football sequence. The average non-zero coefficients vary from 149 at 15Kbps to 6635 per frame at 740 Kbps the corresponding instructions per frame vary from 29,666 at 15 Kbps to 199,733 at 740 Kbps. Such a strong correlation can be exploited to develop a parametric model for resource estimation. A simple linear model can estimate the instructions per frame at a given bitrate B based on the non-zero coefficients:

$$InstructionPerFrame(B) = 199733 \times NZ(B) / 6635 = 30.2 \times NZ(B)$$

NZ(B) is the number of non-zero coefficients at a given bitrate. We are currently working on developing more sophisticated parametric models that can estimate the resources better. Figure 8 shows the bus transitions measured on the Armulator which are proportional to the power dissipation due to switching capacitance. The Figure also has a plot of the skipped macroblocks per frame. The units are normalized to show the strong correlation between skipped macroblocks and the total transitions. This relationship can be exploited to estimate power dissipation due to bus activity.

5.1. Applying the Model for Reducing Product Development Cycle

The proposed model allows estimating the resources required by a video decoder on a given platform. This model is expected to be used at architects' level where high level design decisions are made. Based on the user requirements for video support, a set of video streams representing the support space are created. These bitstreams are then input to the model to estimate the resources required. The resource estimates can be obtained at a component level allowing resource estimates for multi-core architectures where each component could be run on a separate core. The output of this process gives architects quick answers to questions on resource requirements. With mobile devices expected to run broad range of applications, the proposed high level modeling tools are necessary to quickly evaluate resource needs of all the supported applications.

6. CONCLUSIONS

This paper presents a methodology for architectural level resource estimation for multimedia applications. Ability to quickly determine resource requirements of applications is essential for reducing the product development cycle time. We consider the problem of resource estimation at architect's level. At this level of abstraction, an architect is conducting a design exploration to build a system of sub-systems and speed outweighs the higher accuracy of resource estimation. We developed a methodology for estimating the resource requirements of data intensive applications such as video. The H.264 video decoder with functionally separate components was used to develop the methodology. The proposed methodology abstracts the data dependencies using Bitstream Abstraction. The component complexity is modeled using target independent metrics using Decoder Abstraction. A resource estimation model that combines Bitstream Abstraction, Decoder Abstraction, and target architecture dependencies can be used to quickly determine resource requirements. The paper also presented a methodology for estimating the power consumed due to bus activity. The results showed that bitstream complexity is strongly correlated to the resource consumption and Bitstream Abstraction is sufficiently accurate to estimate resource consumption. The proposed methodology results in resource estimation without a need for extensive simulations.

REFERENCES

1. Brooks, D.M.; Bose, P.; Schuster, S.E.; Jacobson, H.; Kudva, P.N.; Buyuktosunoglu, A.; Wellman, J.; Zyuban, V.; Gupta, M.; Cook, P.W., "Power-aware microarchitecture: design and modeling challenges for next-generation microprocessors," Micro, IEEE , vol.20, no.6pp.26-44, Nov/Dec 2000
2. Milder, P. A., Ahmad, M., Hoe, J. C., and Püschel, M. 2006. Fast and accurate resource estimation of automatically generated custom DFT IP cores. In Proceedings of the International Symposium on Field Programmable Gate Arrays (Monterey, California, USA, February 22 - 24, 2006). FPGA'06. ACM Press, New York, NY, 211-220.
3. Tsui, C., Chan, K., Wu, Q., Ding, C., and Pedram, M. 1997. A power estimation framework for designing low power portable video applications. In Proceedings of the 34th Annual Conference on Design Automation (Anaheim, California, United States, June 09 - 13, 1997). DAC '97. ACM Press, New York, NY, 421-424.
4. Degalahal, V.; Tuan, T., "Methodology for high level estimation of FPGA power consumption," Design Automation Conference, 2005. Proceedings of the ASP-DAC 2005. Asia and South Pacific , vol.1, no.pp. 657- 660 Vol. 1, 18-21 Jan. 2005
5. Kalla, P.; Henkel, J.; Hu, X.S., "SEA: fast power estimation for micro-architectures," Design Automation Conference, 2003. Proceedings of the ASP-DAC 2003. Asia and South Pacific , vol., no.pp. 600- 605, 21-24 Jan. 2003
6. A. Sinha and A. Chandrakasan, "JouleTrack – A Web based tool for software energy profiling," in Proc. 38th Design Automation Conf., June 2001, pp. 220-225.
7. E. Senn, , et al., "Power Consumption Estimation of a C Program for Data-Intensive Applications," Integrated Circuit Design Power and Timing Modeling, Optimization and Simulation: Proceedings 12th International Workshop PATMOS 2002, Seville, Spain, September 11-13, 2002, Lecture Notes in Computer Science 2451, Springer Verlag, 2002, pp. 332-341.
8. M Schneider, H Blume, TG Noll," Power estimation on functional level for programmable processors," Advances in Radio Science (2004) 2: 215–219.
9. K. Choi, K. Dantu, and M. Pedram, "Frame–based dynamic voltage and frequency scaling for a MPEG decoder," Proc. of the IEEE/ACM International Conference on Computer Aided Design (ICCAD 2002), Nov. 2002, pp. 732–737.
10. E.–Y. Chung, L. Benini, and G. De Micheli, "Contents provider–assisted dynamic voltage scaling for low energy multimedia applications," Proc. of the 2002 International Symposium on Low Power Electronics and Design, Aug. 2002, pp. 42–47.
11. S. Pasricha, S. Mohapatra, M. Luthra, N. Dutt, and N. Subramanian,"Reducing backlight power consumption for streaming video applicationson mobile handheld devices," Proc. of the 1st Workshop on Embedded Systems for Real-Time Multimedia, Oct. 2003, pp. 11–17.
12. ISO/IEC JTC1/SC29/WG11, "Text of ISO/IEC FDIS 14496-10: Information Technology – Coding of audio-visual objects – Part 10: Advanced Video Coding," MPEG Document N5555, March 2003.
13. T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra. "Overview of the H.264/AVC Video Coding Standard," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, July 2003.
14. J. Reinders, "VTune Performance Analyzer Essentials: Measurement and Tuning Techniques for Software Developers," Intel Press 2004.
15. H. Mizuno, H. Kobayashi, T. Onoye, and I. Shirakawa, "Power Estimation at Architecture Level for Embedded Systems," Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS2002), Vol. II, pp. 476 – 479.
16. K Tatas, M Dasygenis, A Argyriou, et al. Address Bus Power Exploration in Programmable Processors for Realization of Multimedia Applications[C]. IEEE International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS'2001), YverdonLes-Bains, Switzerland, IEEE Computer Society, 2001.
17. Kalva, Hari; Furht, Borko, "Complexity estimation of the H.264 coded video bitstreams," Computer Journal. Vol. 48, no. 5, pp. 504-513. 2005.
18. Horowitz, M., Zoch, A., and Kossentini, F. (2003) H.264/AVC Baseline Decoder Complexity Analysis. IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, pp. 704-716.
19. Wolf, F., "Behavioral Intervals in Embedded Software," Kluwer Academic Publishers, Boston, MA, 2002.