High Level Metrics for Power Dissipation and Signal Interference: An Integrated Methodology

Carlos Krieghoff, c.krieghoff@computer.org

Ravi Shankar, ravi@cse.fau.edu

Hari Kalva, hari@cse.fau.edu

John Perret, johnperret@bellsouth.net

Computer Science & Engineering, Florida Atlantic University, Boca Raton, FL

Abstract. High level evaluation of performance, power dissipation, and signal interference can significantly accelerate real-time embedded system design. Tradeoffs between software and hardware, architectural variations, and selection of appropriate application software or of the most effective algorithm, will be easier. Impact of technology on the product can be assessed early on. We have integrated, into our modeling and design flow, two recently reported abstract level methods to estimate power dissipation and cross-talk. This will allow one to make appropriate design decisions at an early stage. We present results for two profiles of the MPEG4 Decoder application running on an ARM processor.

Introduction

High level evaluation of performance, power dissipation, and signal interference can significantly accelerate real-time embedded system design. There are several trends that are necessitating this: desire to reduce the time-to-market (TTM), increasing number of applications and mobile use, and increasing SOC (system-on-a-chip) design complexity and clock rate. The underlying DSM (deep submicron) semiconductor technology, while enabling the integration of more powerful systems, can also lead to poor battery life and quality of service (QOS). However, software-hardware codesign can provide better functional integration and performance, while enhancing the battery life and QOS metrics, if such metrics can be incorporated into the codesign flow.

Estimation of the power dissipation and signal interference at the circuit and VLSI (very large scale integration) levels can be fairly accurate. The process can, however, be very slow, especially for software profiling to compare and contrast alternative algorithms, applications, and architectures. Performance estimation at the architectural level, using an ISS (Instruction Set Simulator), is both feasible and reliable [1]. We thus set out to extend the integration at the architectural level to include the estimation of power dissipation and signal interference. For this, we

adapted two recently reported methodologies, used at the RTL (register-transfer language) level [2, 3], to our architectural level model. Both have been shown to correlate well with the lower level more detailed models.

Our power dissipation and interference analysis methodology will eventually permit one to perform what-if scenarios and build a more powerful and intelligent spreadsheet-like package to determine the overall impact on the system by the suite of software and applications that are under consideration. Specifically, and more narrowly, this paper addresses the comparative evaluation of application and other software, on a virtual model of the actual architecture, but at a high level of abstraction; hence, rough (or no) estimates of today can be replaced with more objective measures.

For power monitoring, we analyze the temporal switching activity on a given signal line. For Interference monitoring, we analyze spatial and simultaneous switching activity across many adjacent signal lines and used them to measure the effect of cross-talk.

For our test case, we used an MPEG4 Decoder application and simulated two different input streams: standard rectangular video and object based video.

Methodology

We present here an integrated methodology based on earlier publications [2, 3]. We have limited the analysis to bus activity here, but the method can be extended to include other hardware modules, written in a system language such as SystemC [4].

We used an Instruction Set Simulator (ISS) of the ARM940T [5] processor, which includes a cache, to generate information profiles for various applications. The figure below shows how an application is fed to the ARM ISS. The application can be an ARM image generated by the ARM compiler or the source code. This application is simulated on the ISS. The simulator generates a trace file with all the bus transactions performed by the application. The trace file is then examined by the power and interference analyzers.

For the power dissipation analysis, we calculated the total number of logic transitions as a measure of the activity factor. We obtain this by adding the transitions on each of the bus lines [2] for the entire execution of the application. For the interference monitor, we calculated two glitch and two delay parameters [3]. We identified each bus line as a potential victim and repeatedly determined a weighted sum of simultaneous transitions on other bus lines. We incremented appropriate parameter's count if the sum exceeded a threshold value. Both these metrics depend on electrical parameters, such as voltage, capacitance(s), and frequency of operation. We assumed these as constant since we were interested in relative comparisons only.

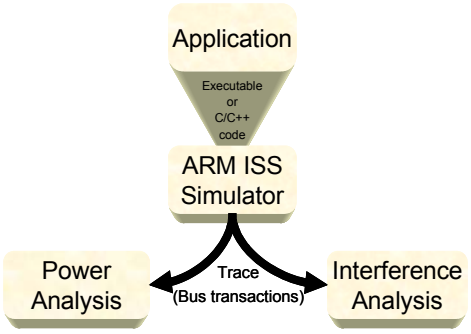

Fig. 1. High Level Power Dissipation and Interference Analysis Flow

1. Power Dissipation Algorithm

As mentioned previously, for power dissipation we analyze the temporal switching activity on a given signal line. The ISS (ARM emulator) generates the profile information for the application running on the ARM CPU; this includes all the instructions that initiate memory transactions. Bus transactions typically account for a major part of the dynamic power consumption for executing a particular software application. We thus focused, to begin with, on the bus contribution only.

In order to estimate the power dissipation of a particular application, we developed an application in C++ that takes the profile information as input and then calculates the number of bus transactions and the number of transitions on the address bus and the data bus. Figure 2 shows the actual temporal activity on the lines of the bus. For instance, compare the activity on the bus at state t+5 against its previous state (t+4); one notices that there were two transitions: one positive transition (from 0 to 1) and one negative transition (from 1 to 0). Table 1 shows the results.

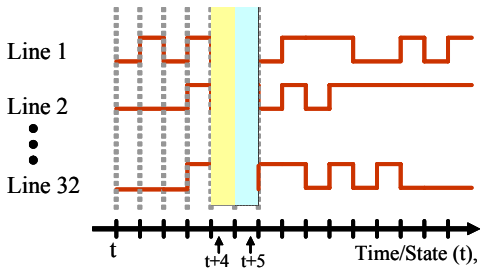

Fig. 2. Power dissipation calculation

	Line 1	Line 2	Line 32	Total # of transitions
t + 5	0	1	0	
t + 4	0	0	1	
XOR	0	1	1	2

Table 1. Power Dissipation calculation results

2. Interference Analysis Algorithm

Signal interference can be simply explained as the undesirable effect that one line carrying electrical signals has on another line running parallel to it, because of its high activity. The line that causes interference is called the aggressor and the one that is affected is called the victim.

The probability that a line (the victim) will pick up interference is high when other lines (aggressors) are running parallel to it and are close to it. The interference can impact the voltage level and the transition delay. For example, if we have line 1 at state '0' and there are three other lines 2, 3, and 4, making transition from state '0' to state '1' then the lines 2, 3, and 4 are potential aggressors and line 1 is a potential victim, if located physically close to them. The interference in this case is a positive glitch which can be misinterpreted by down stream logic as a positive transition, if the interference is high enough.

These are the steps we used to identify possible signal interference:

1. Identify the parallel lines running close to each other
2. Identify the number of lines making similar transitions, i.e. from '0' to '1' and vice versa.
3. If the number of lines making similar transitions is greater than a defined threshold, they are potential aggressors.
    a. If the number of lines making the transition from '0' to '1' is greater than this threshold, then they are potential aggressors; the potential victims are the lines carrying a '0' and the lines making a '1' to '0' transition.
    b. If the number of lines making the transition from '1' to '0' is greater than this threshold, then they are potential aggressors; the potential victims are the lines carrying a '1' and the lines making a '0' to '1' transition.
4. Assign weights to aggressors and defenders depending upon how close they are from the potential victim. Defenders are the lines making transitions opposed to those of the aggressors (This weight assignment can be improved later with annotation).
    a. Aggressors and defenders that are closer to the victim are assigned higher weights.
    b. Add the total weights of the aggressors and subtract the total weight of defenders for a particular victim.

5. If the cumulative aggression score (sum of the weights of the aggressors minus the sum of the weights of the defenders) for a potential victim is greater than a threshold value called "Interference Threshold", then the probability of it being a victim is extremely high and it is considered as a victim. The type of interference can be decided as follows:
    a. If the aggressors are making '0' to '1' transitions
        i. Victims that are making a '1' to '0' transition will get a positive delay
        ii. Victims that are at state '0' will get a positive glitch
    b. If the aggressors are making '1' to '0' transitions
        i. Victims that are making a '0' to '1' transition will get a negative delay
        ii. Victims that are at state '1' will get a negative glitch
6. Repeat step 4 and 5 for all the probable victims.

See Figure 3 for the different interference effects that can be potentially induced.

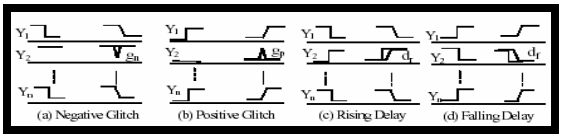

Fig. 3. Four different effects induced by cross-coupling capacitances [3]

Figure 4 depicts the method for our signal interference analyzer, with an example.

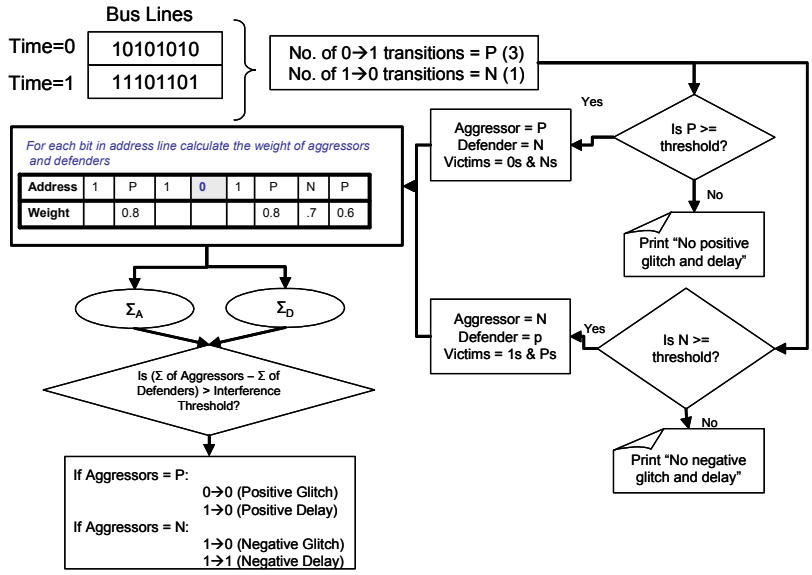

Fig. 4. Signal interference analysis algorithm

The output of the signal interference analyzer is the number of possible positive glitches, negative glitches, negative delays, and positive delays for a given application.

Results

We used the test case of the MPEG4 Decoder for our methodological evaluation. MPEG-4 video coding is a complex standard with over a dozen profiles and several levels for each profile. Mobile phones today typically use MPEG-4 Simple Profile video. Even though the MPEG-4 video decoding algorithms are standardized, implementations and performance vary from one implementation to another. There is no objective way to compare and evaluate solutions from different vendors. We have developed a methodology and a set of metrics to evaluate such applications.

In this project, we first determined and then compared the two metrics (power dissipation and signal interference) for a rectangular video (Simple Profile) and an arbitrary shaped object based video (Core Profile) on a given architecture. Instead of considering two different vendor implementations, we considered two different MPEG-4 video profiles. Using two different profiles allowed us to examine the core MPEG-4 functionality and develop representative metrics. Only video decoding was analyzed. We expect the methodology can be extended to compare and contrast multimedia and other applications running on a given architectural platform. This is useful for rapid and objective evaluation of vendor application software for a given architecture, or vice versa.

The details of the MPEG4 decoder application were as follows: There were 250+ files written in C; the total number of lines of code exceeded 60 K (excluding comments and blank lines); the input bit streams were encoded either for traditional rectangular video or low-bandwidth object-based video; the video size was 176 x 144 pixels.

For the rectangular video we used bits streams with the following bit rates: 100 Kbits/sec, 50 Kbits/sec, and 15 Kbits/sec. The resolution of the different bit rates can be appreciated in the figures below.

Fig. 5. Rectangular video at 100Kbit/sec (left) and 15Kbit.sec (right)

Figure 6 and 7 show the results of the power dissipation and signal interference analyzers for different standard rectangular video input bit streams.

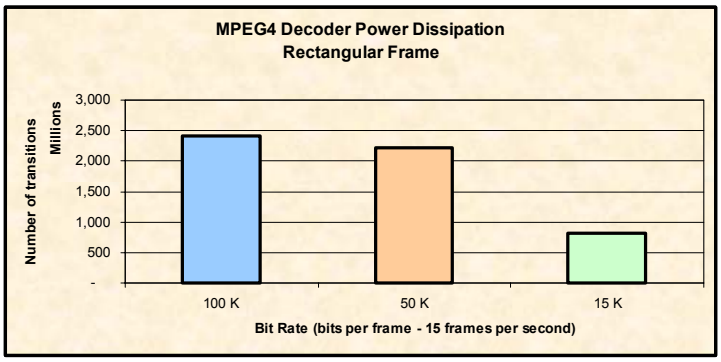

Fig. 6. Results: power dissipation for rectangular video at different bit rates

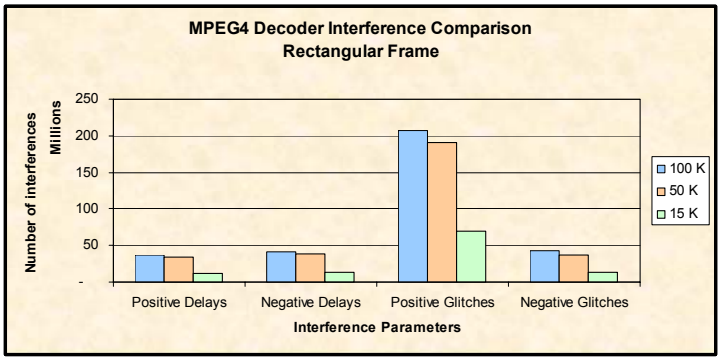

Fig. 7. Results: signal interference for rectangular video at different bit rates

For object-based video we used bits streams with the following bit rates: 60 Kbits/sec, 40 Kbits/sec, and 20 Kbits/sec. The resolution of the different bit rates can be appreciated in the figures below.

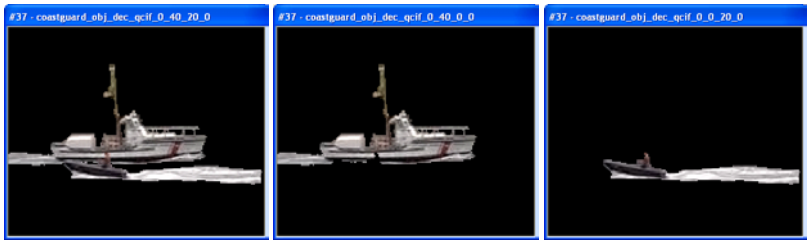

Fig. 8. Object-based video at different bit rates (Left: bit boat and little boat at 60Kbit/sec, center: big boat at 40Kbit/sec, and right: little boat at 20Kbit/sec)

Figure 9 shows the results of the power dissipation analyzer for different object based video input bit streams.

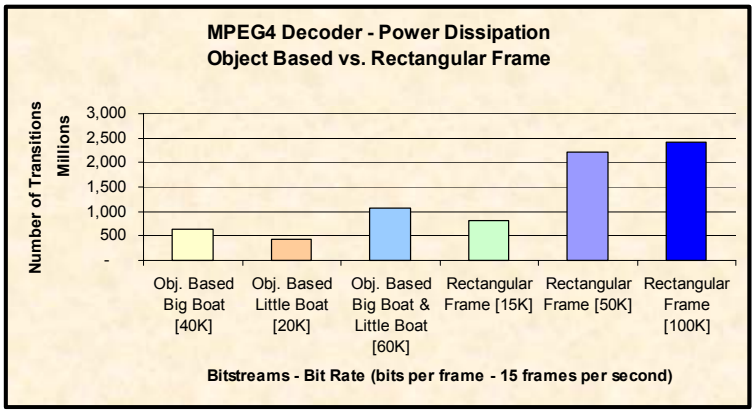

Fig. 9. Results: Power Dissipation for object-based video vs. rectangular frame

Figure 10 shows the results of the signal interference analyzer for different standard object-based video input bit streams.

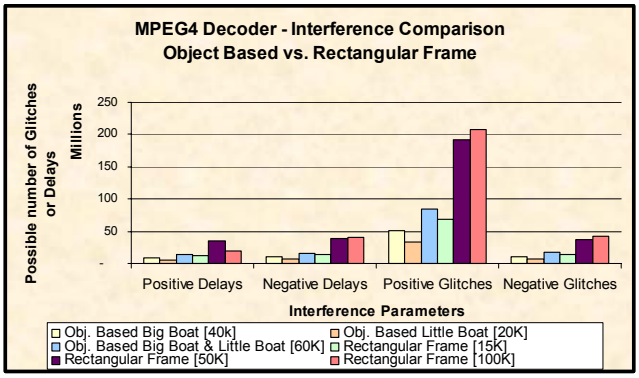

Fig. 10. Results: Signal Interference for rectangular video at different bit rates

As can be seen, the number of positive glitches is about 5 times larger than the number of negative glitches. To understand this 'anomaly', we conducted extensive experimentation with a sort routine, which also yielded similar results. We were able to conclude that this results from the relatively high number of zeros (the non-active state) on the bus lines, a tendency brought about because of the conventional coding style, locality of code, and lower address ranges; this increased the number of

positive glitch transitions occurring on the buses with respect to the other kinds of transitions.

Discussion

We have developed and integrated abstract level methods for the estimation of power dissipation and signal interference, so one can perform what-if scenarios at the software-hardware co-design level, on a relative basis.

For our specific test case, the bandwidth savings due to object based content are well known [6]; but there is no existing work on the effect of object based coding on power consumption and interference. Our results show that there is substantial reduction in both power consumption and interference when object based video is used, as compared to traditional rectangular video.

Our software profiling methodology can be used for selection (that is, comparative evaluation) of application and other software for a given architecture. Back annotation of the electrical parameters (from the VLSI level) and algorithmic improvements can lead to both better relative and better absolute estimates.

Conclusions

Software performance profiling with an ISS (instruction set simulator) is feasible today. We extend such an ISS-based methodology for software profiling to include power dissipation and signal interference, which have become additional critical parameters because of technological and consumer trends. Comparative vendor software profiling, for a given architecture, to cover all major electrical parameters (performance, power, and interference) thus becomes feasible. We present results for two profiles of the MPEG4 Decoder application running on an ARM processor, which show that the low bandwidth object-based video is also more power efficient and more immune to noise, relative to the traditional rectangular video.

Our methodology can be extended to lead to better relative and absolute estimates, as well as more global estimates, so true software-hardware tradeoffs across different suites of applications, algorithms, architectures, and implementations can be undertaken early on in the design. This will enhance design productivity and provide for more predictable system behavior.

References

1. ARM Ltd.: Software Development Kit Version 2.50 User Guide, Chapter 11: Benchmarking, Performance Analysis, and Profiling, (1998) 399-426, available at http://www.arm.com/pdfs/sdt250usrman.pdf

2. Mizuno, H., Kobayashi, H., Onoye, T., Shirakawa, I.: Power Estimation at Architecture Level for Embedded Systems, Proc. IEEE International Symposium on Circuits and Systems (ISCAS2002) (May 2002) 476--479
3. Bai, X., Dey, S.: High-level Crosstalk Defect Simulation for System-on-Chip Interconnects. IEEE VLSI Test Symposium (2001) 169-177
4. www.systemc.org
5. www.arm.com
6. Kalva, H.: Designing Object-Based Audio-Visual Content Representation Format for Mobile Devices, Proceedings of the 47th IEEE International Midwest Symposium on Circuits and Systems (July 2004) III-479 - III-482.

Acknowledgments

Abhijit Ajmera and Jigisha Goswami developed the early version of this monitor.