

Run-time Modeling and Estimation of Multimedia System Power Consumption

Yu Hu

Integrated Media Systems Center
and Electrical Engineering
University of Southern California
Los Angeles, California, 90089
Email: yhu0@usc.edu

Qing Li

Integrated Media Systems Center
and Electrical Engineering
University of Southern California
Los Angeles, California, 90089
Email: qingli@sipi.usc.edu

C.-C. Jay Kuo

Integrated Media Systems Center
and Electrical Engineering
University of Southern California
Los Angeles, California, 90089
Email: cckuo0@sipi.usc.edu

Abstract—The power dissipation of a VLIW processor is investigated across a series multimedia benchmarks, from grayscale image compress algorithm to the most advanced speech codec H.264, to characterize their behavior. Based on this study, a run-time power modeling is proposed to achieve fast and accurate estimation. The model shows its ability to predict the validating sequence’s cumulative energy consumption with little error. Finally, this model is validated against an industry VLIW processor by comparison of their power behavior character.

I. INTRODUCTION

With the significant growth in the mobile Internet and portable media devices, the embedded multimedia system (EMS) has become a major platform in the digital media era. The pursuit of higher clock rates and better CPU performance has driven power dissipation and energy consumption more and more so that they become the key issue in the EMS design and the software implementation. Additionally, the total “energy consumption” for battery-powered portable devices decides the length of usage time. The increasing gap between the battery life and the energy consumption requirement for multimedia applications has imposed a great challenge on today’s embedded processor design.

Embedded media processors possess a specific energy dissipation pattern, which is quite different from that of general purpose processors (GPPs), due to the different micro-architecture (VLIW) and the multimedia instruction extension (SIMD). It is critical to understand the energy dissipation phenomena of embedded multimedia applications for battery-powered portable devices so as to get power-efficient implementations. The energy consumption profiling of a wide spectrum multimedia applications is conducted in this research to characterize their power behavior. Based on this investigation, a run-time power modeling is proposed to achieve fast and accurate estimation. The model shows its ability to predict the validating sequence’s cumulative energy consumption with little error. Finally this model is validated against an industry VLIW processor by comparing the power behavior characteristics provided by the manufacturer.

The rest of this paper is organized as follows. Related previous work on power estimation and modeling is briefly reviewed in Sec. 2. The experimental frameworks, like the

methodology, benchmarks and test sequences, are described in Sec. 3. Sec. 4 presents the routine level multimedia application model based on the profiling sequence result. Sec. 5 illustrates the model’s ability to predict the cumulative energy consumption of the validating sequence by comparing the the experiment data with the theoretical data. The model is furthermore validated against an industry processor by the comparison of their power dissipation character in Sec. 6. Finally, we draw a conclusion of this work in Sec. 7.

II. PREVIOUS WORK ON POWER ESTIMATION

The overall goal in power analysis is to understand the power consumption phenomenon and minimize the power/energy consumption under hardware constraints. Thus, this research includes two tightly coupled research topics: power / energy modeling and power/energy optimization. The former is to build up power dissipation models for circuits, architectures, instructions and even programs while the latter is to develop algorithms to reduce power dissipation or energy consumption over different layers.

In this section, we provide a review of previous work on power analysis. Power analysis techniques can be categorized into two types: the model-based approach and the measurement-based approach. The measurement-based approach examines power consumption at a certain level based on actual power measurements. In contrast, the model-based approach analyzes the system power performance using simulation tools. Most commercial simulators use low-level tools with the under-layer available information (*i.e.*, the physical layer). However, in past decades, researchers have kept exploring higher level simulation systems and tools, and achieved some success.

A. Cycle-accurate Architecture-Level Simulation

Commercial tools have long been developed for power dissipation analysis at the gate/circuit level, *e.g.*, PowerMill [1] and SPICE [2]. With years’ effort, there are also quite a few academic architecture-level simulators available now, most of which are based on the renowned performance simulator SimpleScalar [3] (*e.g.*, Wattch [4], SimplePower [5] and SimAnalyzer [6]).

With full details of the physical layer, it is possible to undergo fine-grained analysis for a specific application, *e.g.* the branch decision in multimedia standards. In the meanwhile, since the architecture of simulator SimpleScalar is a superscalar one, whose instruction is dynamically scheduled and all the power analysis constructed on it has to be for the super-scalar system, too. Due to the structure's difference, it might not be helpful in analyzing the performance and power dissipation of another type of microstructure such as VLIW. Furthermore, the simulation process is fairly slow, which motivates the study of higher level simulation to get accurate online estimation.

B. Instruction-Level Power Modeling

An instruction level model is derived from measure-based experiments, where details of lower-level models are concealed. The main goal is to provide a power consumption analysis of a given software. The basic idea is to associate the consumed power with each individual executed instruction. An empirical instruction-level energy model for each instruction can be given as

$$E = \sum_i (B_i \times N_i) + \sum_i \sum_j (W_{ij} \times N_{ij}) + \sum_k (S_k), \quad (1)$$

where E is the total energy consumed by the underlying software program, B_i is the base energy consumed by instruction N_i , W_{ij} reflects the dissipated energy due to circuit switching between each pair of consecutive instructions (i, j) and S_k accounts for the extra energy consumption introduced by the CPU stall caused by cache miss and/or bank conflict of the k th instruction. The variation in the total energy comes from two aspects: the inter-instruction effect of switching circuit states and some resource constraints that may lead to CPU stalls.

Although the modern microprocessor is a very complex system, its complexity is hidden behind a simple interface, *i.e.* the instruction set architecture (ISA). The parameter B_i could be obtained by an exhaustive measurement of the entire ISA in advance. Similarly, W_{ij} can be determined through the measurement of any possible instruction pair. Thus, both B_i and W_{ij} can be pre-determined for certain micro-architecture and ISA. However, the workload of exhaustive measurement for W_{ij} may vary in a large scale depending on the size of ISA. For example, TI TMS320C64X owns an ISA of 167 instructions, the total possible pairs are $167 \times 166 = 27,722$. For Intel's Itanium 2 with 331 instructions, the number could increase to $109,230 (= 331 \times 330)$. Parameter S_k is largely dependent on the available processor resource such as the cache and the register file during the program execution. Thus, it varies from one case to another, and should be determined by on-line measurements.

C. VLIW Micro-Architecture Power Estimation

Power analysis follows the development of microprocessors. The target platform evolves from the general purpose processor (GPP) RISC CPU to a specific purpose processor like media processors and communication processors. Since all media

processors adopt the VLIW microarchitecture and the single instruction multiple data (SIMD) ISA with no exemptions, researchers have made a lot of effort in the field of power/energy analysis for VLIW processors.

VLIW processors issue a fixed number of instructions as one large instruction package with multiple regular instructions inside. The compiler statically schedules issued instructions and, therefore, the VLIW compiler plays an important role in the whole system's performance. The power model in Sec. II-B has been extended to the VLIW micro-architecture. Intuitively, it is known that the measurement workload is even heavier to get the inter-instruction-package energy. For each instruction package of an n -issue VLIW processor, the fully or partially issued case in each instruction package has to be considered. Let m ($0 \leq m \leq n$) denote the actually issued instruction number. The complexity is high.

A comprehensive instruction-level power model for the VLIW-based embedded processor was presented in [[7]]. It can be written as

$$E(W) = \sum (E(w_n|w_{n-1})) + E_c + c_0, \quad (2)$$

where $E(w_n|w_{n-1})$ is the energy dissipated by the data path associated with the execution of instruction w_n with w_{n-1} preceding, E_c is the total energy consumed by the control unit and c_0 is a constant associated with the energy to initialize the processor. To validate this power/energy model, a VLIW processor working at 100M Hz described with VHDL was constructed in [7] and PowerMill was used to verify the theoretical values.

A VLIW-based simulator was developed by Ascia *et al.* [8], [9] for power/energy performance simulation. Their work is similar to that in [4] by dividing the complex CPU into several structured components: function units, register files, memory hierarchy, buses, etc. The main drawback of this work is that some other important components are not considered yet they may play a significant role in the functionality and energy consumption (*i.e.*, clocking). In the same time, the architecture level simulation is still slow in the speed.

D. Run-time System-level Power Estimation

The system level simulation tools and models aim at flexible treatment of different low-level components and programs. Most research in this level cannot be applied in the design stage but to available processors only. A built-in source meter of variable power supply was used to measure the current with the help of a subroutine in [10], where the average current value was probed from different addressing modes and inputs on the test-bed of a StrongARM processor and a Hitachi SH4 processor. Consequently, a first-order and a second-order models are constructed based on these measurements.

III. METHODOLOGY

In this section, first we describe the experimental environment used to compile and execute the benchmarks, like the simulator and its configuration. Then the multimedia benchmarks and their test sequences will be introduced briefly.

TABLE I: Target processor configuration

Processor core parameters	
Frequency	200 MHz
Technology	0.25 μ m
Voltage	3.3V
Fetch/Issue/Retire Width	8/8/8
Instruction window size	128
Recorder buffer size	
integer FUs	4
Integer FUs latencies	1/3/8 add/multi/div
float FUs	2
float FUs latencies	3/3/8 add/multi/div
branch unit/penalty	1/7
general register	64
float point register	64
Memory hierarchy	
L1 cache size	0.5 KB for L1D and L1I each
L1D hit/miss latency	1/2
L1P hit/miss latency	1/2
L2U cache size	8KB
L2U hit/miss latency	2/7
Memory	7/35

A. Simulation Environment

The experiments were carried out over a cycle-accurate, instruction level EPIC (explicitly parallel instruction computing) /VLIW simulator: Trimaran [11]. It is a parameterized processor architecture, providing a vehicle to explore the instruction level parallelism (ILP) in compilation. EPIC-explorer [8] calculates the energy consumption value using the collected profiling result produced by Trimaran, behaving similar to other low-level power simulators like Wattch [4]. By dividing the processor architecture into a set of function block units (FBUs), EPIC-explorer captures the power and energy characteristic using an adapted model of the Cai-Lim's.

Table I describes the target configuration of Trimaran which is used for our experiments. It should be pointed out here is that this basic configuration is based on the observation that most media applications require a relatively small cache size, especially for the L1P, to obtain a high hit rate. *please give out the scale of the L1D L1P hit and miss rate later please check the value's validation or not*

B. Benchmarks

Our choice of benchmarks is intended to model next generation multimedia (speech, audio, digital still image and video) applications over embedded processors. Although Mediabench provides a suite of media benchmarks, some of them are out of time due to the fast development in multimedia algorithms. For example, the most advanced video compression standard has evolved from MPEG2, the one included in Mediabench suite, to MPEG4 to today's H.264. The latter two represent the future video compression trend because they support higher rate distortion character than MPEG2.

We consider totally 7 sets of multimedia benchmark codecs, covering both high and low bit rate applications. Here is a brief review of these 14 applications. Advanced audio coding (AAC) is a wide band coding algorithm first specified in MPEG2. MPEG 4 audio compress technology evolved from MPEG2 AAC by strengthening the effectiveness at low bit rate

TABLE II: Benchmarks

Codec name	Profiling sequence	Validation sequence
Image compression		
EPWIC	grayscale lenna 512x512	grayscale baboon 512x512
JPEG	color lenna 512x512	color baboon 512x512
Video compression		
MPEG2		foreman cif
MPEG4	foreman qcif	mobile qcif
H.264	foreman cif	mobile cif
Speech & audio		
AMR		
AAC		

and additional error resilience capability. Adaptive multi-rate (AMR) speech codec is introduced by 3G and widely used in GPRS and W-CDMA. Embedded predictive wavelet image coder (EPWIC) is a grayscale image compression algorithm incorporating wavelet pyramid decomposition. Its predecessor EPIC (embedded pyramid image coding) is included in Mediabench suite. JPEG is the technology generally employed in digital cameras, working on color image compression. MPEG2 is the most popular compress format used in DVD system nowadays. MPEG released MPEG4 standard in 1999, with the most important features to support higher levels interactions between users and the contents, controlled by the content developers. It is further extended in low bandwidth networks to support mobile applications. As the advanced multimedia coding standard jointly developed by MPEG and ITU, H.264 provides better video quality and a higher compression ratio as compared to MPEG series. Several new coding techniques such as edge loop filtering, quarter pixel interpolation and content based binary arithmetic coding (CABAC), etc. have been incorporated to enhance coding efficiency and image quality.

The execution of a multimedia application is largely dependent on its algorithm complexity. Beyond that the input sequence parameters, like the size (width and height), frame length and the correlation of the consequent frames, also have an important impact on the program's execution. To do the experiment and verify our conclusion, the test sequences are divided into two groups, *e.g.*, profiling and validating. The profiling group is used to generate data to construct the power and energy model. On the other hand, the validating group is employed to check the accuracy of proposed models. Lenna 512 \times 512 is the profiling sequence and the same size baboon image is for validation purpose. The difference is grayscale images are used for EPWIC and color images are for JPEG. For the video compress algorithms, a sequence of 4 different frame lengths *i.e.*, 10, 25, 50, 100 are tested. The profiling and validation sequences for MPEG2 and MPEG4 are the same: foreman qcif size for training purpose and the validation sequences are foreman and mobile cif size and mobile qcif size. Considering H.264's video quality advantage, only cif size sequences are tested here. Please refer to Table II for more details. *Adding introduction for the speech and audio*

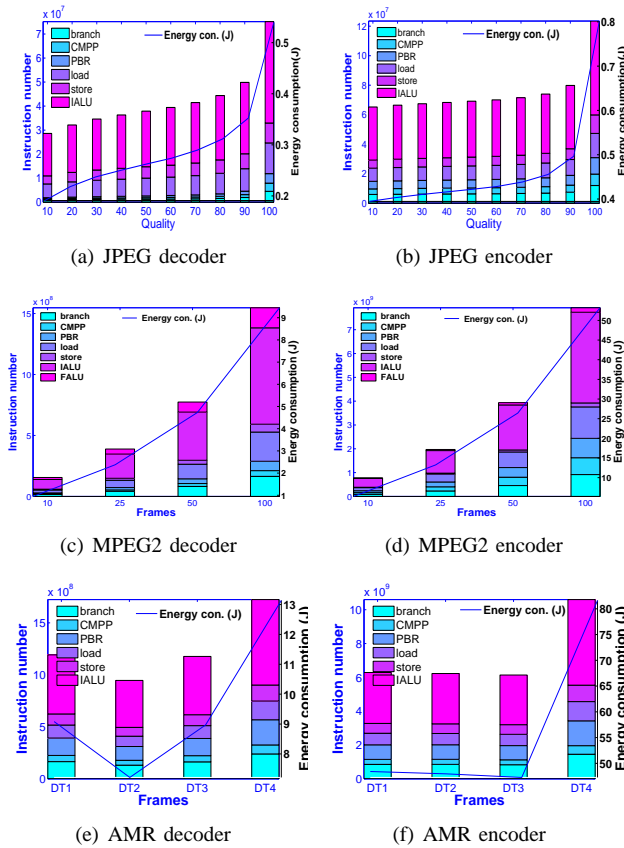


Fig. 1: Execution and energy characters of multimedia application

test sequence here

IV. ROUTINE LEVEL MULTIMEDIA APPLICATIONS CHARACTERIZATION

The behavior of modern multimedia program is rather unpredictable because of its heterogeneous structure. In general, the compress algorithms try to eliminate all kinds of redundancies inside the original data. Thus the image codec consists of three main blocks: DCT/IDCT, quantization / dequantization and the entropy coding to remove spatial redundancy and coding redundancy. Beyond these fundamental modules, motion search and compensation are included respectively in video coder and video decoder to get rid of the temporal redundancy. Nevertheless, there are a bunch of algorithms for each module. Thus the program's performance depends largely on the employed algorithm, the platform and the implementation. The execution and energy character is so hard to predict that no general rules could be applied to all the programs to get an accurate estimation. In this section, we will present the characterizations of multimedia execution and power behavior.

A. Execution and Energy Character of Multimedia Standards

To explore multimedia standards' execution and energy characteristics, first we investigate the data related to the

performance, like total instruction number consumed by each multimedia program with the profiling input. The energy consumption of the whole multimedia standards is also collected as well, as shown in Fig. 1. The general trend in Fig. 1 shows that as the complexity and requirement of the test sequence increase, more instruction and energy are used to execute a program. Nevertheless, it is inaccurate to conclude that the relationship between instruction number and energy consumption is a simple linear function. In addition, a further look at the average power of the whole standards, which is obtained by:

$$\text{Average Power} = \frac{\text{energy}}{\text{execution time}}$$

it reveals the more characteristics of multimedia applications execution by looking at the average power and its standard deviation, which is shown in Fig. 2. The feature of the average power is that its value fluctuates in some certain scale for each specific application. It reaches an extreme prototype for video codec, the encoders' average power swings in a very small space: ± 0.005 , independent of the input sequence's frame length. Meanwhile, video codec's average power also possesses a smaller standard deviation compared to the image codec's. For example, the power scale of JPEG decoder is 2 to 2.3 and 2.25 to 2.6 for JPEG encoder. At the same time, the standard deviation of image codec is always above 20%, almost the double value of that of the video codec. By continuing our observation further into the routine level, we could explain why the difference exists for the image and video codec.

Fig. 3 can well explain the difference in the average power and its standard deviation between image and video codec. Due to the limited space, here shows only image quality 10 and 100 for JPEG codec and frame length 10 and 100 for MPEG2 codec. In our experiments, on the image encoder side, although it has the same input, the encoder routines are called by different times on the request to encode the image into different size bitstream. Since each routine dissipates different power and energy, the whole encoder's power and energy consumption vary correspondingly. The decoder takes into variable size bitstream to decode a 512×512 image. The fundamental blocks are responsible for most of the variance. On the contrary, the situation is totally different for the video codec: the routine energy distribution of codec for 10 frames are almost the same as that of 100 frames, which infers the number each routine (or the most frequently referred routines) is called is proportional to the frame length. The reasons lie in three facts:

- video codec is much larger in the code size and execution time than image codec. Thus the influence of overhead on the whole program is less important;
- because of the much larger execution time, the energy difference is easy to get offset;
- the strong correlation between sequential video frames makes the function call numbers of the main modules proportional to the frame length.

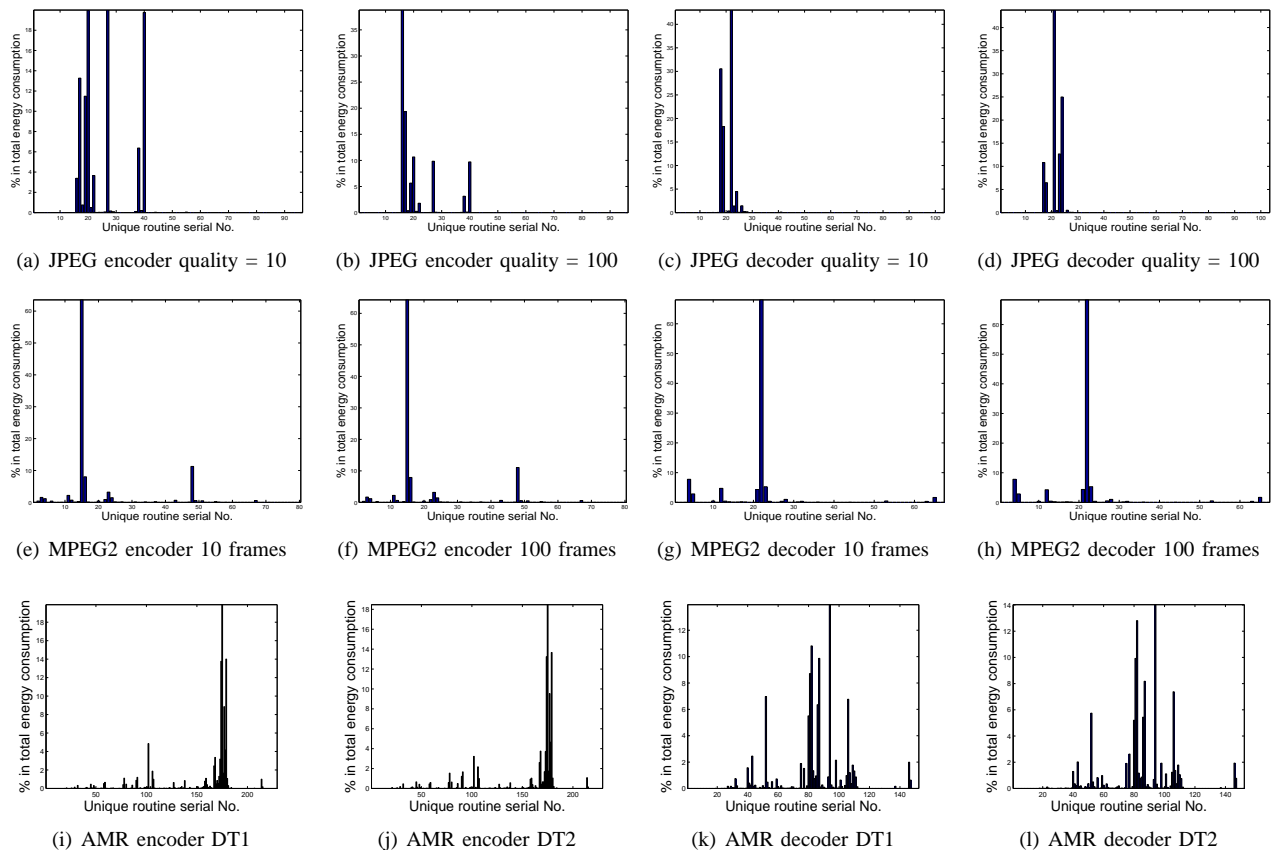


Fig. 3: Routine energy distribution

The above observations, combined with the fact that individual multimedia application shows different power and energy behavior from each other implies that: (1) Overall, the multimedia power and energy behavior vary from one application to another; (2) A rough approximation of using one single average power across various routines will lead to significant estimation errors.

Pay more attention to the sub-conclusion later

B. Power-Performance Correlation

Most of nowadays high-performance processor tries to take full advantage of the program parallelism. Multiple-instruction issue, branching in superscalar and VLIW two directions, is a popular design to employ the instruction level parallelism (ILP). Consequently, instruction per cycle (IPC) becomes a metric of processor activity. The higher IPC, the busier processor is. The average power and average IPC for each routine are measured and shown in Fig. 4. In Fig. 4(a) and 4(b), it is shown that the power and IPC of JPEG codec with compression quality 100. The character of MPEG2 codec with frame length 100 is given in Fig. 4(c) and 4(d). *add the speech codec explanation later.* By observing Fig. 4, we could draw a conclusion that the average power for each routine is strongly correlated to its IPC, which we could explain in this way: In general, the dynamic power at a specific moment is proportional to the IPC. The higher IPC, the more resources

in processor get involved in program execution. Consequently, the higher dynamic power is. Nevertheless, most routines' execution is so short that the dynamic power through this duration could be treated by a constant average power with small error. This suggests that the average power always correlate with IPC.

By breaking the total energy consumption of MPEG2 decoder down we could clearly see the contribution of each component to the total energy consumption in Fig. 5(a). The datapath and pipeline, the core of a processor, take charge of instruction decoding, instruction issue and executing the instructions. It consumes around 65% of total energy. The clock system, due to the frequently wire switch in each tick, is responsible for 23% energy cost. Since most video codecs are not intensive memory-access application, the energy consumption of the memory hierarchy occupies only 12%, including the contribution of L1, L2 and main memory. A more detailed data about each function unit's energy consumption is shown in Fig. 5(b). The active energy consumption of instruction decode unit plays a important role due to its continuous instruction decoding job, implying that video decoder is an instruction-intensive application. The register file consumption is too trivial compared to any other components. Moreover, the static and active energy consumption of VLIW-structured processor performs quite different from that of superscalar one. Power minimization only pays attention to the active energy

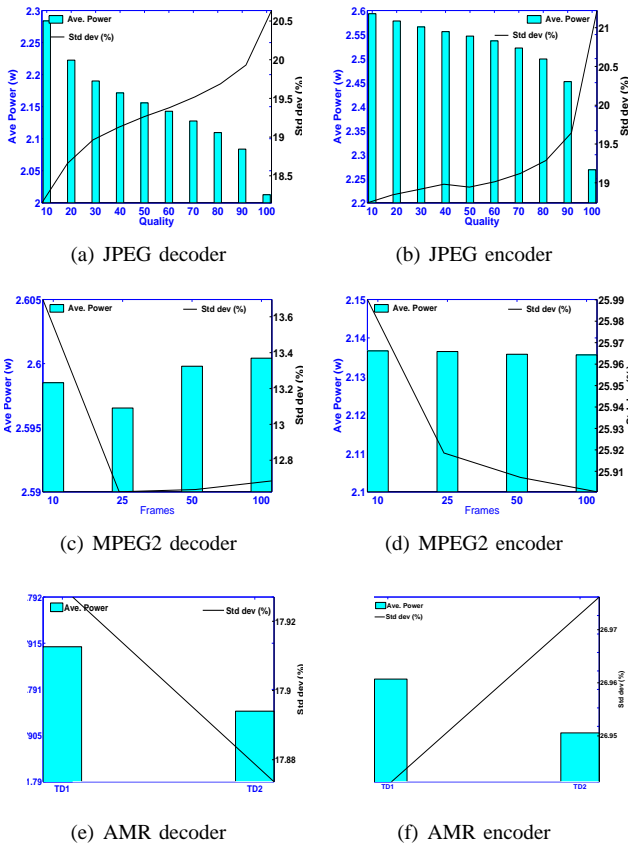


Fig. 2: Average power and its standard deviation

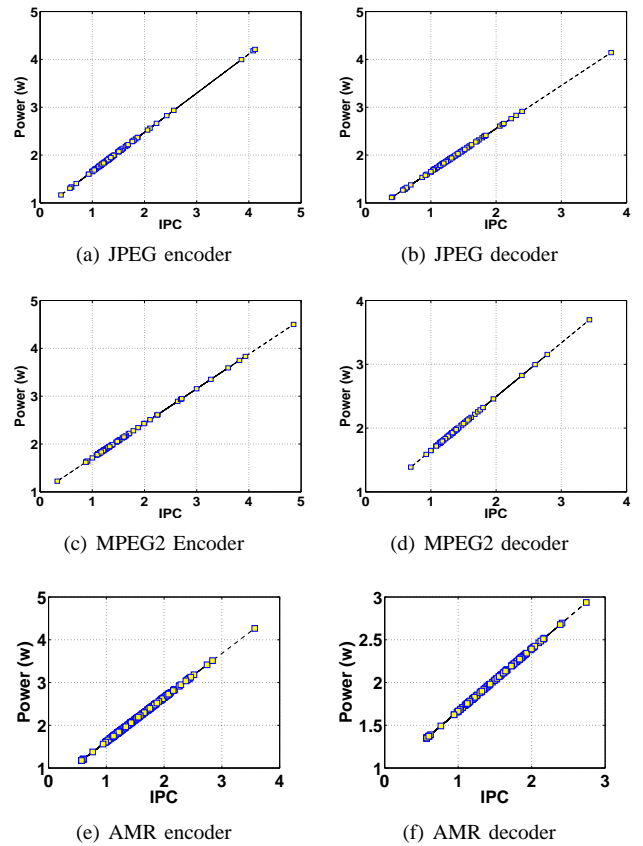


Fig. 4: the relationship between average power and IPC

consumption on superscalar processors. However, due to the higher and higher weight static leakage power takes, power optimization job also has to work on static power to achieve more progress. Meanwhile for the "expensive" and rarely used units like float point and branch units, the static power is even higher than the active one, where the clock-gating technique is necessary. *seems a little far away from the topic*

The discussion in section 4.1 and 4.2 hints that one linear relation model could be applied to the routine power and its IPC by:

$$P = k_0 + k_1 \times IPC \quad (3)$$

where k_0 and k_1 are constant for a specific application. By a further extension, k_0 is a parameter to describe the static power consumption and k_1 is related to the percentage that a processor is involved into an instruction's execution.

V. ROUTINE LEVEL MULTIMEDIA POWER MODEL

The profiling-result-based routine level energy model for multimedia application is presented in this section. The objective is aiming at providing a simple and accurate enough method to meet the run-time estimation requirement. The validation result will be presented as well.

The total energy consumed by a program could be calculated by:

$$\text{Energy} = \int_0^{ET} \text{Power}(t) dt \quad (4)$$

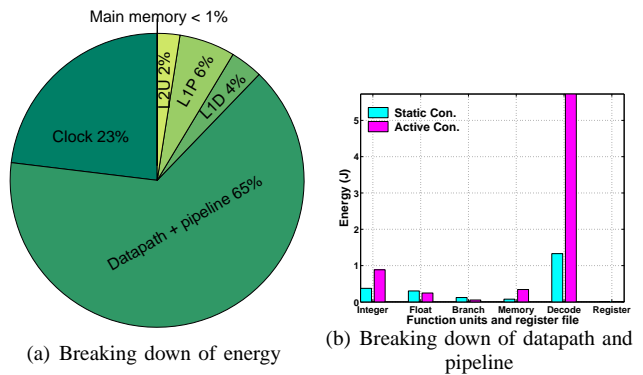


Fig. 5: Energy component of MPEG2 decoder

where $Power(t)$ represents the dynamic value of power and ET is the duration to execute the program. Nevertheless, the dynamic value is practically unavailable due to the measurer's limited sensitivity: the ampere meter could not follow the fast fluctuation of current. Thus it is impossible to provide delicate data for the integration. Meanwhile, faster and faster has the high performance processor become to make most routines' execution fairly quick, usually in the order of millisecond. Naturally it is reasonable to substitute the dynamic power with the average one of each routine while introducing slight error. Therefore a whole program's energy consumption could be

TABLE III: Energy character of multimedia applications

Benchmarks	Power model		
	$P = k_0 + k_1 \times IPC$		
	k_0	k_1	$\epsilon(\%)$
EPWIC decoder	.9472	.7176	3.928
EPWIC encoder	1.703	.595	16.2
JPEG decoder	.8996	.7536	1.9
JPEG encoder	.8418	.817	0.5
MPEG2 decoder	.8643	.7588	1.717
MPEG2 encoder	1.3738	.4802	0.649
MPEG4 encoder	1.4871	.3642	0.7685
MPEG4 decoder	.9709	.7336	0.6
H.264 encoder	.9924	.7429	0.4
H.264 decoder	.9599	.6428	2.05

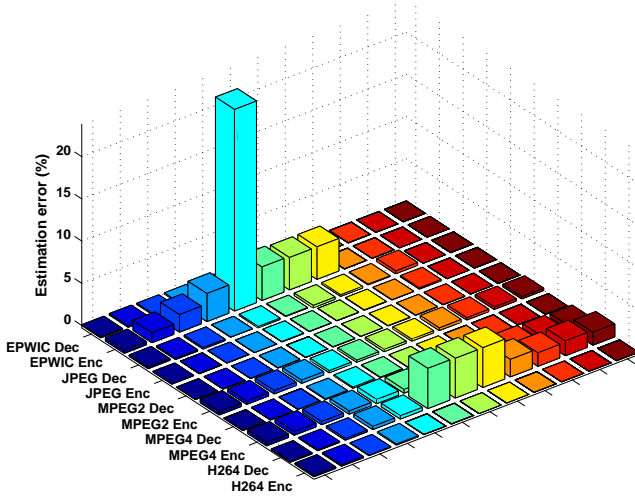


Fig. 6: Error in validation sequence

calculated by summing up each routine’s energy consumption and Eq. (4) could be changed to:

$$\begin{aligned}
 Energy_i &= APower_i \times ET_i \\
 Energy &= \sum_{i=0}^N Energy_i
 \end{aligned} \tag{5}$$

where $Energy_i$ is the energy consumption of the i th routine, $APower_i$ is the average power during the i th routine’s execution ET_i , calculating with Eq. (3). If a flat average power used throughout the whole application’s execution instead of the routine level average power, the error will be much higher.

Table III presents the values of k_0 and k_1 got from each profiling sequence for every benchmark. Also we give out the standard deviation of the validation result. The error by applying the result of profiling sequence into validation sequence is show in Fig. 6. Please refer to Appendix for more detail about the validation error and the execution characteristics.

VI. MODEL VALIDATION

The validation step is crucial because a fast estimation is only effective when it is relatively accurate. Compare the low level capacitance value should be the most directive and accurate means to validate a power model. However, we have no way to access the detail of any industry hardware design

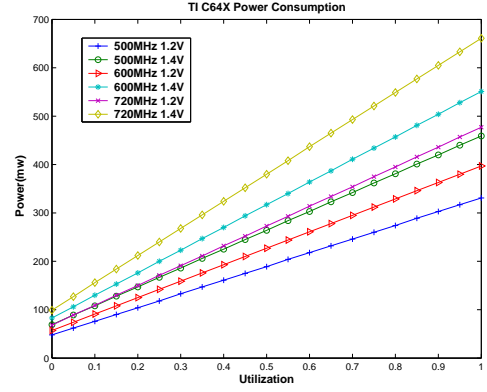


Fig. 7: C64x Power Consumption

and its detailed capacitance values. To validate our model against the industry processors, we here present a way to prove the model’s properness in a high level. The downside of this rough comparison is that each unit energy consumption could not be known.

A. Power Model Comparison

TI C6416, one of the fastest DSPs is chosen as a prototype to check the relative quantity of our model. As one of the newest members of the TI DSP family, the $C64x$ series [12] could achieve a higher clock rate and increase the CPU throughput via two duplicate data paths and register files, with 32 registers for each. Besides the VLIW structure, it also supports SIMD to enhance the media processing data throughput by the use of data level parallelism (DLP).

A power dissipation model provided by TI [13], is described by

$$\begin{aligned}
 \text{Power} &= \text{Baseline Power} + \text{Activity Power} \\
 &= a(f, v, t) + b(f, v) \times \alpha
 \end{aligned} \tag{6}$$

where f is the clock frequency, v the supply voltage, t the environment temperature while utilization rate α reflects the activity level of the system. In Eq. (6), the power dissipation is broken into two parts: the baseline power and the active power. The baseline power, which is a function of v , f and t , includes the PLL power and the clock tree power [14]. The baseline power corresponds to the static dissipation part of a processor. The active power is related to active energy consumption of the processor, and it is also a function of v , f and α . The voltage and frequency can be easily measured while the utilization rate α is usually obtained by empirical measurements or via statistics from architectural-performance simulators.

Fig. 7 [13] depicts the relationship of the power and the utilization rate. It is easy to get the total instruction and CPU cycles (execution time) from the profiling result but difficult to know the utilization rate of each cycle. Since IPC is a good metric of processor’s engagement into instruction execution, it is used here to calculate utilization rate α in Eq. (6), namely,

$$\text{Power} = a(f, v, t) + b(f, v) \times \frac{\text{IPC}}{\text{Full Issue}} \tag{7}$$

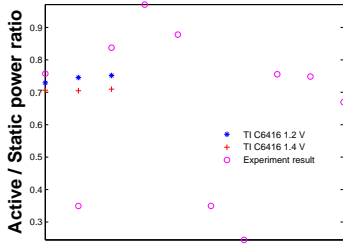


Fig. 8: Active/Static power ratio

The full issue for TI C6416 is 8 which means it could issue and execute 8 instructions in a cycle. By substituting the corresponding data to Eq. (7), the power and energy formulas for TI C6416 working at 500 ~ 720 MHz with 1.2V or 1.4V voltage supply become

$$\text{Power} = \begin{cases} 48 + 35 \times \text{IPC} & \dots & 500\text{MHz } 1.2\text{V} \\ 69 + 48.75 \times \text{IPC} & \dots & 500\text{MHz } 1.4\text{V} \\ 57 + 42.5 \times \text{IPC} & \dots & 600\text{MHz } 1.2\text{V} \\ 83 + 58.5 \times \text{IPC} & \dots & 600\text{MHz } 1.4\text{V} \\ 68 + 51.125 \times \text{IPC} & \dots & 720\text{MHz } 1.2\text{V} \\ 99 + 70.25 \times \text{IPC} & \dots & 720\text{MHz } 1.4\text{V} \end{cases} \quad (8)$$

By comparing Eq. 8 and Eq. 3, it is found that both of them are in the form of linear form. Hence the model we proposed is correct in the form first. Next we will prove that the properness of k_0 and k_1 .

B. Relative Components Validation

Because of different layout and manufacture technology, e.g., the transistors number per die, power consumption will vary from each other even when the utilization rate is 1. Nevertheless, the same VLIW structure suggests that the ratio of active power consumption over the static one should be of the similar order. Fig. 8 plots the ratios of $\frac{k_1}{k_0}$ in Table III and those in Eq. 8. The statistics of C6416 is in the scale of 0.73 ~ 0.75 for 1.2V supply and that of 1.4V is around 0.70. Most of the experiment results fall into a space 0.67 ~ 0.84, with an exception that the value of JPEG encoder, a value of almost 1. Thus from the power saving point of view, the static consumption should be paid more attention when processor is rather idle. In the meanwhile, as the wide-issue processor becomes busier, the active component dominates again, similar with superscalar processors' behavior.

Therefore the system power model for VLIW structure processor is verified by comparing its power behavior with the industry processor's power consumption characteristics.

VII. CONCLUSION

Modern embedded multimedia applications are characterized by the emergence of resource-limited VLIW processors with SIMD instruction extension. Power and energy models for the VLIW-structured processors, which is validated by the TI C6416 chip-set, were proposed in this work. The profiling

result suggests a strong correlation between IPC and power dissipation. By exploiting this relationship, the power and energy consumption for a multimedia application could be easily modeled by several test sequences' profiling result, yielding fairly low error. This simple model not only leads to run-time efficient power estimation for various multimedia systems, but also provides insight into VLIW-structured processor energy saving. For instance, the model points out that the static power dissipation plays an important role in the total consumption whereas its influence is previously believed to be ignorable.

REFERENCES

- [1] SYNOPSYS. "PowerMill,". <http://www.synopsys.com/products/etg/powermill.ds.html>.
- [2] EECS of UC Berkley. "SPICE,". <http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/>.
- [3] Todd Austin. "SIMPLESCALAR,". <http://www.simplescalar.com/>.
- [4] David Brooks. "Watch,". <http://www.eecs.harvard.edu/dbrooks/watch-form.html>.
- [5] CE Univ. Penn. "Simple Power,". <http://www.cse.psu.edu/mdl/software.htm>.
- [6] EECS UMICH. "Sim Panalyzer,". <http://www.eecs.umich.edu/panalyzer/>.
- [7] M. Sami, D. Sciuto, C. Silvano, V. Zaccaro, "An Instruction-Level Energy Model for Embedded VLIW Architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **21**, pp. 998-1010, September 2002.
- [8] G. Ascia, V. Catania, M. Palesi and D. Patti, "EPIC-Explorer: A Parameterized VLIW-based Platform Framework for Design Space Exploration," in *First Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia)*, Newport Beach, California, USA, October 2003.
- [9] CS Univ. Catania, Italy. "EPIC Explorer,". <http://epic-explorer.sourceforge.net/>.
- [10] A. Sinha and A. P. Chandrakasan, "JouleTrack-a web based tool for software energy profiling," in *Proceedings of Design Automation Conference*, pp. 220-225, June 2001.
- [11] HP Lab, IMPACT UIUC, Georgia Inst.of Tech. "Trimaran,". <http://www.trimaran.org>.
- [12] TI. "C6000 Architecture,". http://dspvillage.ti.com/docs/catalog/generation/details.jhtml?path=templatedata/cm/dspdetail/data/c6000_architecture.
- [13] TI, "TMS320 C641x Power Estimation Spreadsheet," in *6414_5.6.power.xls*, 2004.
- [14] TI. "TMS320DM64x Power Consumption Summary,". <http://focus.ti.com/lit/an/spra962e/spra962e.pdf>.