

System Level Power Estimation Methodology with H.264 Decoder Prediction IP Case Study

Young-Hwan Park, Sudeep Pasricha, Fadi J. Kurdahi, Nikil Dutt
University of California, Irvine, CA
 {younghwp, spasrich, kurdahi, dutt}@uci.edu

Abstract

This paper presents a methodology to generate a hierarchy of power models for power estimation of custom hardware IP blocks, enabling a trade-off between power estimation accuracy, modeling effort and estimation speed. Our power estimation approach enables several novel system-level explorations - such as observing the effect of clock gating, and the effects of tweaking application-level parameters on system power - with an estimation accuracy that is close to the gate-level. We implemented our methodology on an H.264 video decoder prediction IP case study, created power models, and evaluated the effects of varying design parameters (e.g., clock gating, I/P frame ratios, Quantization), allowing rapid system-level power exploration of these design parameters.

1. Introduction

To address the ever widening designer productivity gap, transaction level models (TLM) [18][21] and high-level simulation platforms are increasingly being used for System-on-Chip (SoC) architecture analysis and optimization. The increasing importance of power as a design objective in today's complex systems is making it imperative to address power consumption early in the design flow, at the system-level, where the benefit of performing power optimizing design changes is the greatest. Whereas previous work [9-13][17] has investigated power estimation techniques for processor, memory and communication architecture IPs, few approaches have addressed system-level power estimation of custom hardware IP blocks used in SoCs.

In this work, we present a hierarchical power characterization model that is applicable to any kind of custom IP. The novelty of our approach is the ability to enable rapid, system-level evaluation with gate-level accuracy of several critical design parameters such as clock gating, quantization, I/P frame ratios. To the best of our knowledge ours is the first approach that comprehensively addresses system level power modeling of custom hardware IPs (ASICs). Fig. 1 outlines the basic concept of our hierarchical approach: the power model has several levels that have varying accuracy, design effort and simulation speed. Upper levels (e.g. Level 0, Level 1) represent coarser grained power models, while lower levels (e.g. Level $N-1$, Level N) represent more accurate power models with greater detail. Power nodes at each level represent the power for the behavioral

granularity represented at a particular level. Each node can be typically decomposed into children nodes (at a lower level) that capture power of the hardware block at a finer granularity. These finer grained power models require more understanding of hardware and effort for system designers to categorize and characterize their design properly. Thus there is tradeoff between upper nodes and lower nodes: upper nodes with a coarse grained power model have faster simulation speed, less accurate power prediction and require less designer effort; whereas lower nodes with a finer grained model have slow simulation speed, more accurate power prediction and require more designer effort. For instance, in Fig. 1, Level 0 represents mean value of power for the entire design of the custom IP. It is a very coarse-grained power model which is widely used for power modeling of custom IP (e.g. [4]). At Level 1, the power model has two states, represented by an Active and Idle node. The Active node represents average power of activated function blocks when it is working. The Idle node represents average power when the design is idle (i.e. not executing and disabled by clock-gating). Similarly, nodes at the lowest hierarchical level (Level $N-1$) can be decomposed into detailed power models (Level N) that consider the effect on power of different input data characteristics such as Hamming distance and number of 1s in the data. The number N is determined by considering the inherent hierarchical structure of the application and the need for sufficient decomposition to enable good power prediction. (Section 4 illustrates this on a custom IP block of the H.264 decoder.) Our methodology encompasses various power modeling granularities and offers the designer a trade-off between estimation accuracy, design modeling effort and simulation time.

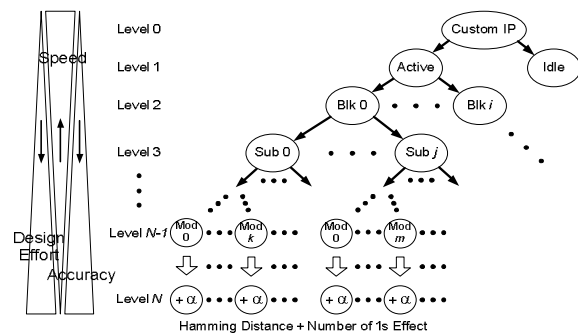


Figure 1. Concept of hierarchical power model

We implemented our methodology on an H.264 [2] video decoder prediction IP. The different power models generated by our methodology are used in a cycle-accurate transaction level (TLM) simulation environment at the system level, to estimate power for the H.264 video decoder prediction IP. We were able to simulate various kinds of videos that have different characteristics at the system level (e.g. different I/P frame ratios and quantization parameters), which is practically not feasible at the gate-level. Most importantly, different hardware configurations and software scheduling schemes can be explored quickly and easily by making changes in the system level model. Our power models enable rapid estimation of performance and power consumption data that increases the designers' confidence in early design decisions. Our approach enables a designer to get fairly accurate power prediction data that is close to gate-level estimates (below 1% average of cycle error and below 10% maximum error). Additionally, our methodology also supports early analysis of power optimizations such as clock gating. The increased usage of clock gating to save power has added complexity to the power behavior of systems and increased its dependence on operating modes, data and states, making it much harder to analyze. Unlike most existing system level power modeling techniques, our power modeling methodology allows a designer to quickly and accurately account for the impact of clock gating at the system level.

2. Related work

There have been numerous studies in literature to estimate and reduce design power at the gate-level [8], which is typically the most reliable but requires impractical simulation time, at the RT-level [5][7][8], and at the system level [3][4][6], which provides more flexible design exploration opportunities but gives relatively unreliable prediction data. Heterogeneous components (e.g. processors, buses, memories, caches, customized ASIC) which constitute SoCs today require different estimation approaches for better power modeling, which best fit the unique characteristics of these specific types of components. There have been several power estimation approaches for processors [9-13], various communication fabrics [1][14-16] and memories [17]. Some researchers have tried to make comprehensive power models for all these components [3][4][6] and power examination tools such as Wattch [9] and SimplePower [5] have also been proposed. Onouchi et al. [4] proposed a power estimation methodology for an embedded SoC case study used in mobile phones (SH-Mobile3AS) which employed power models for each of the IPs in the system. For the H.264 IP, they used only one frame to extract power data. However, this rough approach (which is equivalent to level 0 of our power model) can easily result in large errors depending on the input video type. All of aforementioned researches fail to adequately address a comprehensive methodology for customized ASIC blocks due to their extreme

heterogeneity. On the other hand, we believe our approach is the first to present a comprehensive methodology that enables system-level power modeling of custom IP blocks. We demonstrate this approach with several blocks from an H.264 decoder design.

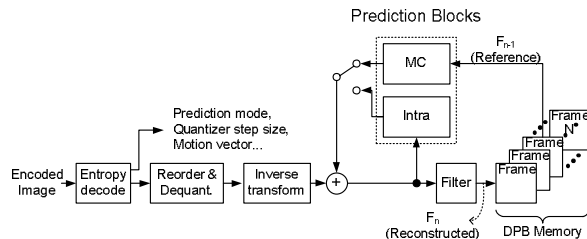


Figure 2. Block diagram of H.264 decoder

3. Design exemplar: H.264 video codec

H.264 is emerging as one of the most promising video standards today, with higher video quality, better compression efficiency and more error robustness for various applications [19]. We use the H.264 decoder as a design exemplar to illustrate our approach, since several blocks of this design need to be implemented as custom hardware (ASICs) to meet performance and power budgets. Fig. 2 illustrates a typical configuration for an H.264 video decoder. In H.264, video streams are compressed in several different ways: entropy coding (VLC or CABAC), quantization, transform and prediction (motion compensation and intra prediction) using redundancy in a video. Prediction is usually the most efficient way for video data compression, but also one of the most computationally demanding tasks in an H.264 decoder. In this study, the motion compensation (MC) or inter-prediction and intra-prediction (Intra) blocks in the figure are implemented as ASIC blocks. A more detailed overview of these two prediction blocks is presented below.

3.1. Motion compensation (inter prediction)

Inter prediction reduces temporal redundancy in a video and creates a prediction model from one or more previously encoded video frames which are stored in the decoded picture buffer (DPB). The H.264 codec uses block-based motion compensation. One of major difference from earlier standards is the support for a range of block sizes (down to 4x4) and fine sub-pixel motion vectors ($\frac{1}{4}$ -pixel in the luma component) [2]. In an inter-coded macroblock, blocks are predicted from the same size area of a reference picture. Even though sub-pixel resolution is allowed, those locations do not exist in reality and they are created by interpolation with adjacent video samples. If the motion vector has integer values, the block is simply copied from an existing area of the reference frame. A block having sub-pel or quarter-pel position is predicted using a combination of 6-tap filter and bilinear filter based on its location. The overall

interpolation of inter prediction depends on motion vector. The chroma components have 1/8 resolution for video sampling 4:2:0 case and are interpolated between integer-pixel samples of chroma.

3.2. Intra prediction

Intra prediction uses spatial coherence of a video and reduces the spatial redundancy with previously encoded and reconstructed blocks. The prediction luma block can have 4x4 sub-block or 16x16 macroblock sizes. There are 9 prediction modes for the 4x4 luma sub-block and 4 modes for the 16x16 luma block. The samples from previously encoded blocks are available as the prediction reference. The predicted samples are calculated by copying or weighted averaging to the direction that gives the most matched predicted video to the original. Chroma component has 8x8 block size and there are 4 modes which are similar to the 16x16 luma prediction mode. Each mode of the intra prediction requires different computation and corresponding power consumption.

3.3. Prediction block ASIC implementation

Hardware/software partitioning is a popular and widely used method in SoC designs today that partitions application functionality into hardware and software. The software part enables flexibility for future design changes, reusability of available design IPs and simplicity of design compared to full custom ASIC, while the hardware part can better handle the more performance critical functionality. Our H.264 decoder prediction IP has been built based on this concept. It has software and hardware components (instead of being completely implemented as custom ASIC hardware because of aforementioned reasons) and follows the ITU-T H.264 video standard [2]. Overall orchestrating and scheduling of functional modules is the role of software which is executed on a DSP or a processor, while the computational intensive work is handled by the ASIC hardware accelerators. A high level block diagram of the implemented prediction IP of the H.264 decoder is illustrated in Fig. 3. It has a virtual master and two major functional modules: Inter and Intra prediction blocks and shared input and output buffers for those functions.

The virtual master handles the software part and ASIC blocks (enclosed by the dotted box) handle the hardware part of this partitioned design. The Inter-prediction module has two sub-function engines which are Luma and Chroma. The Intra-prediction module has three sub-function engines which are Luma 16x16, Luma 4x4 and Chroma. Those sub-function engines receive control signals via their slave controllers. The master reads test vectors and controls overall data flow and activation of function modules by sending data to the proper function blocks and configuring the function register set of each module. The dotted box represents the entire ASIC block which is used for our case study. The physical size of implemented prediction block is provided in Table 1.

Detailed gate count based on two input NAND cell (ND2D1) in a 90nm library for two major prediction blocks, Inter and Intra, input buffers (IB) and output buffers (OB) for those functional modules, bus and total size are shown in the table. Note that like most complex IP blocks, this prediction block has an inherent hierarchical functional decomposition that we can exploit to generate a hierarchical power model as described in the next section.

Table 1. Physical Size of Implemented Prediction Block

Name	Inter	Intra	Bus	IB (Inter)
Gate Count	248636.4	114331.8	2435.5	56812.5
Name	OB(Inter)	IB(Intra)	OB(Intra)	Total
Gate Count	3351.5	54220.7	53844.0	533637.5

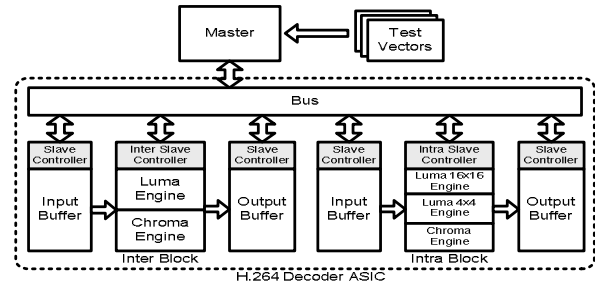


Figure 3. Block diagram of H.264 decoder prediction

4. Power modeling

At the SoC level, an IP block such as the motion compensation (MC) component would be part of a system that would contain a processor running software, several memories, other hardware IP blocks and buses to interconnect all the components. In order to accurately model the power (as well as the performance) of such a system, one must simulate the design under a variety of situations (e.g., various input videos for a video codec). Given the complexity of today's systems and their sheer size, it is impractical to simulate them at low levels of abstraction. Thus gate level or even RTL are considered too low level for simulating anything more than a handful of frames at small resolutions. System level simulation presents itself as the only practical way of exploring realistic scenarios (e.g. complete video sequences in the case of H.264). Furthermore, there is usually a wide range of configuration parameters available at the SoC level that would result in significantly different behavior and potentially, power and performance numbers. In order to get an accurate modeling of the power consumption of a SoC, one must first establish accurate models for all its components. These include IP blocks, processors, memories and the communication infrastructure (e.g., buses). In [1] we addressed the issues of system level modeling of power for on-chip communication infrastructure. In this work, we propose a method for modeling ASIC blocks and apply it to the H.264 motion compensation block as a case study.

A SoC is typically modeled at the system level using languages such as SystemC [22] or SpecC [23]. These languages can model the various components as well as the communication infrastructure. The IP blocks are modeled in a variety of ways. At the low level, a synthesizable model is written in RTL using Verilog or VHDL. At the higher system level, it can be described behaviorally using the same approach with system level languages such as C/C++. When attempting to model the power consumption of an IP block one is faced with some issues:

- (i) how much variability in power consumption is there under different operating conditions?
- (ii) how much do power reduction techniques such as clock gating affect such variability?
- (iii) how do we accurately reflect such variability in a system level power model?

In the following sections, we attempt to address these issues, and in doing so, propose a model that shows good correlation with gate level power simulation.

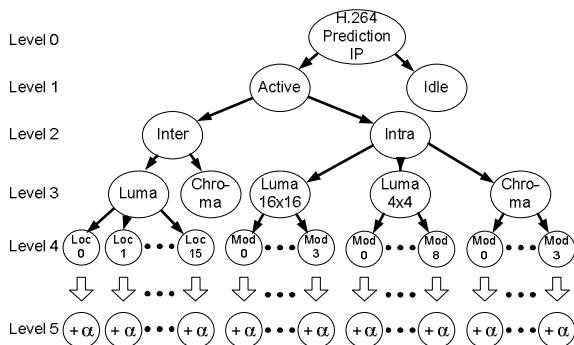


Figure 4. Power model for H.264 decoder prediction IP ASIC

4.1. Hierarchical power model of implemented design

For the hierarchical power model of the H.264 decoder prediction IP, 6 levels are defined, as shown in Fig. 4. The model has mean value of power for H.264 decoder prediction IP at Level 0 and has Active and Idle node at level 1 as described earlier. At Level 2, a more detailed power model is used, with the Active node being decomposed into Inter and Intra prediction nodes, for which power models are created. At Level 3 the Inter block power model is decomposed into two finer granularity power models represented by the Luma and Chroma nodes. The Luma nodes can be further decomposed into sub-modes based on the interpolation pixel location value. These sub-modes are represented as nodes at Level 4, which represent power models for the sub-modes. Unlike the Luma node, the Chroma node does not have children nodes because sub division for the Chroma block does not show much computational and

power related difference. Nodes at Level 4 are decomposed into the most detailed power models as Level 5 which considers the effect of different input data characteristics on power.

4.2. Power estimation methodology

The overall methodology used to build the proposed hierarchical power model is illustrated in Fig. 5. We modeled the prediction blocks of H.264 video decoder at RTL level in Verilog. This RTL design is synthesized to the gate-level net-list with Synopsys Design Compiler [24] using the target technology cell library (Step 1). As part of this step, RC parasitic value files (SPEF), standard delay format (SDF) file and design constraints file (SDC) are also generated to be used for power simulation and gate-level simulation. Then the gate-level net-list file is used for gate-level simulation with Cadence NC-Verilog simulator [25] (Step 2). During the gate-level simulation, a simulation information file that has information about the executed function ID, operating cycles, starting and ending function time, calculated Hamming distance and number of 1s in the input data is generated and stored in a text file. A Value Change Dump (VCD) file which has logic switching information for dynamic power measurement is also generated in this step.

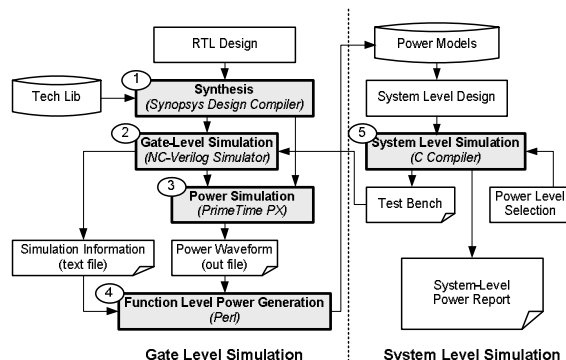


Figure 5. System power modeling methodology

With the VCD and RC parasitic values, gate-level power simulation is done with the Synopsys PrimeTime PX tool [24] to create cycle accurate power waveform file (Step 3). This file is stored as .out file which has a simple text format. The simulation information file from Step 2 and the .out file are provided to the function level power generation phase in Step 4. In this step, cycle-accurate (e.g. Nano second detailed) original gate-level power simulation waveform data are modified (using Perl scripts) to the function-level granularity power waveform data (as shown in Fig. 6) as well as producing the power models described in sub section 4.1.

These functions in Fig. 6 are equivalent to Level 3 in Fig. 4 and functions of system level implementation. For the upper nodes, such as those at Level 0, 1 and 2, we can tie those functions together based on categories we defined for each level. For the lower nodes, such as those

at Level 4, we decompose each function based on its functional modes and calculate power for these different modes separately.

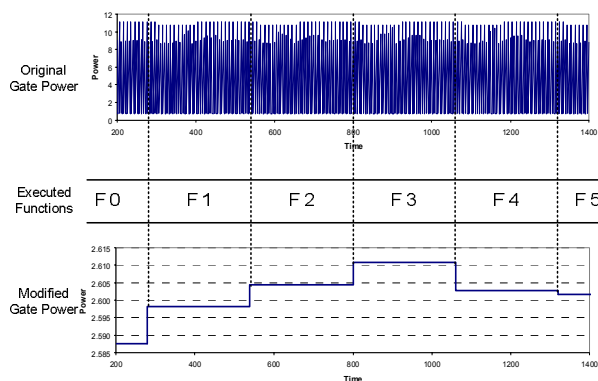


Figure 6. Modifying gate power wave

Once these levels corresponding to our RTL implementation are defined, Perl scripts allow us to automatically generate most of power levels except the most detailed model at Level 5. To generate the Level 5 model, we applied multiple linear regression analysis using the GNU R tool [20] to correlate the effect of Hamming distance and number of 1s in the input to power consumption (described in section 5.1). From now on, gate level simulation data, which will be compared with our system level power data, will have the function-level granularity resolution. Since detailed (ns-level) power modeling is practically impossible and unnecessary at the system-level, lowering the resolution process is needed for efficiency and feasibility. Next, the power model is plugged into the system-level environment, modeled at the cycle-accurate transaction-level abstraction and captured using the C language. The values for power and simulation cycles are stored in a system level power report after the system level simulation process (Step 5). We can adjust the power level during system-level simulation, depending on the requirements for estimation accuracy and the trade-off with modeling effort and simulation time. The system-level power report file provides cycle accurate power transition information, computed average power and peak power numbers just like a gate-level power simulation captures. With the average power, we can compute energy by multiplying with execution time. Peak power is the highest value of power during simulation and knowing its value allows us to determine the electrical and thermal limits of ICs and the system packaging. This also allows us to guarantee correct functionality of ICs within given limits [26].

5. Experiments

We conducted several experiments to determine the factors that have an impact on power consumption (Section 5.1), the power saving through clock gating and

its influence on power modeling (Section 5.2), and the trade-off between different power levels in terms of accuracy, simulation speed and design effort (Section 5.3). Finally, we show the power characteristics for different input videos (Section 5.4). A QCIF input video that has two types of frames (I and P) is used for all the power modeling experiments and TSMC 65nm and 90nm standard libraries are used for synthesis.

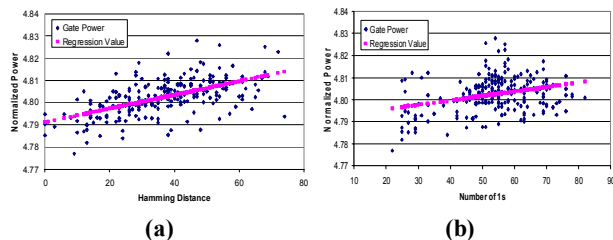


Figure 7. Impact of (a) Hamming distance and (b) number of 1s on power

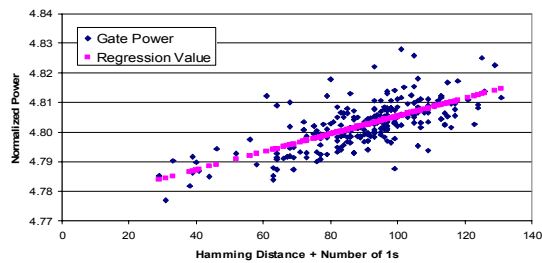


Figure 8. Impact of Hamming distance + Number of 1s on power.

5.1. Impact of data on power

To estimate power more accurately, just considering average power for certain modes of the IP block would not be enough since it does not take into account the impact of different input data. Dynamic power consumption is proportional to switching activity of gates. The Hamming distance (which is the count of flipped bits between two consecutive data) and number of 1 bits in the data can be used to represent this switching behavior.

Initially, the impact of Hamming distance and number of 1s on the power for chosen mode of operation (e.g. Intra 4x4, mode 1) are considered separately and shown in Fig. 7 for the H.264 decoder prediction IP. We observe linear relation between those two numbers with power. When they are increased, corresponding power is also increased. This becomes clearer when we draw regression curves using the scattered power data points. The simple linear curves clearly describe the impact of those two factors on the power. Combining these two numbers by simple addition showed closer correlation between the summed number and the power, and this is illustrated in Fig. 8. The scattered dots of power data are located more closely to the calculated regression value in this case. Standard deviations for those three cases to regression

value are 6.501E-03, 7.603E-03 and 5.743E-03 respectively, showing that the combined number has less deviation and has closer correlation to the gate level power. The same phenomenon is observed for other functional modes as well. Thus, the level N (LN) power (which is the finest granularity) for function 0 and mode 0 as an example is derived using simple equation:

$$P_{LN-f0m0} = P_{0-f0m0} + \alpha_{f0m0} \times (\Psi_{HDin} + \Psi_{N1in}) \dots (1)$$

where P_{0-f0m0} is the base level power for the given mode of function and α_{f0m0} is the coefficient which is the slope of the linear regression curve and represents correlation of the number and the power. Ψ_{HDin} and Ψ_{N1in} are Hamming distance and the number of 1s of the input data respectively.

From the above experiments, it can be seen that taking into account Hamming distance and number of 1s can give a very good power prediction, at the cost of more elaboration time to build the finest grained power level (e.g. Level 5 of H.264 decoder prediction IP case study).

5.2. Clock-gating effect on power model

Clock gating is a well known technique to save dynamic power consumption by preventing clock supply to flip-flops of the unused portion in synchronous logic design. This technique is especially important for mobile embedded systems where power is a crucial design constraint.

The impact of clock gating on the power of the implemented H.264 decoder prediction IP design is shown in Fig. 9. Two different power waveforms for non-clock gating and clock gating enabled design are shown in the figure. We observed approximately 40% power saving with clock gating. In the figure, huge variation of the waveform at the time 480000 ns is shown because it decodes an I frame until that time and then starts decoding a P frame.

We can see some spikes during decoding of the I frame on the clock gating power waveform which are not observed in the non-clock gating case. The spikes occur when it decodes luma 16x16 intra prediction blocks. Peak power consumption is observed at the beginning of each function module (at 30 ns for intra prediction block and at 480000 ns for inter prediction block) because it consumes large power for block initialization. For accurate power modeling, this effect should be considered. Much more waveform variation is shown for clock gating case compared to the non-clock gating case. The symptom occurs because all flip-flops are switching unnecessarily and consuming certain amount of power in the non-clock gating logic. On the other hand, the clock gating enabled logic supplies clocks to only necessary portion of the design and saves large power. Our results clearly show that system-level analysis of dynamic power behavior due to clock gating requires finer-grained power modeling for reasonable accuracy of system level simulation.

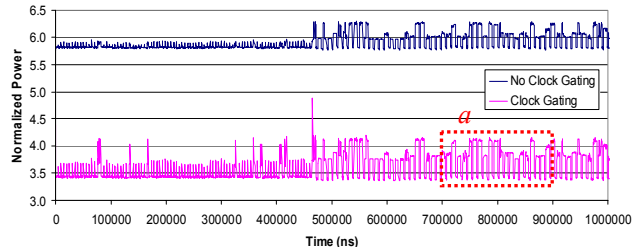
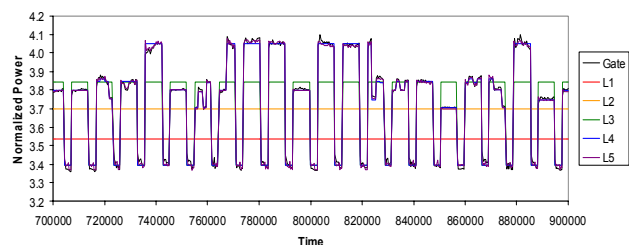
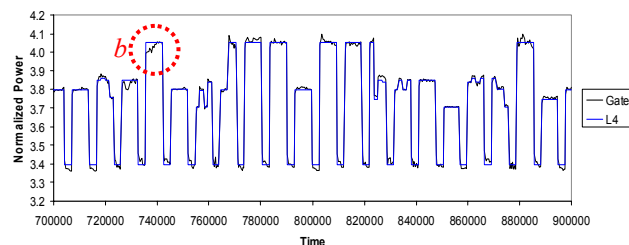


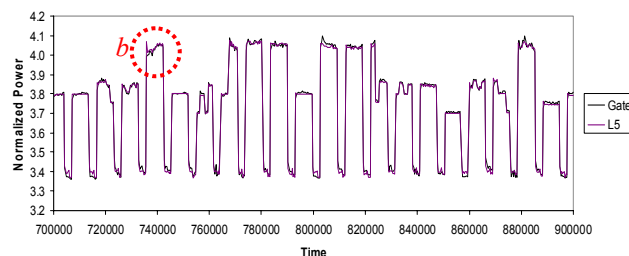
Figure 9. Gate simulation power data of clock gating and non-clock gating design.



(a) Composite power estimate across all levels



(b) Level 4 power vs. gate power



(c) Level 5 power vs. gate power

Figure 10. Comparing waveforms of each level

5.3. Power level trade-off

The power estimates for the various levels of power macro models introduced in Section 4.1 for the H.264 decoder prediction IP are shown in the Fig. 10 (a). The figure shows a magnified region a from Fig. 9. As we expect, finer-grained power models are close to the gate-level simulation power waveform. Level 1 (L1) model can predict better than L0. L2 does better than L1 and so forth. L0, which is the average power of our design without considering active and idle status, has a large error compared to the gate level power data, as well as the data obtained with other power models (such as L1-L5). Thus, the L0 waveform is omitted in the figure. A comparison of the estimates from the most fine-grained models – L4 and L5 with gate power is shown in Fig. 10

(b) and (c) respectively. We can see the difference by looking into the circle, *b*. The L5 waveform is very close to the gate level power. On the other hand the L4 waveform shows less detailed data at the same area. We observed that the L5 model, which considers the low-level factors of input data as described in section 5.1, gives better prediction than coarser-grained power levels (0.33% improvement of average cycle error over L4).

Next, we computed the average cycle error for the Level 0 power model compared to gate level simulation power data while decoding two frames for non-clock gating situation. Interestingly, *the most simple averaged power model can provide pretty accurate power data in such a case (average cycle error is just about 2%)*. The symptom is caused by the power consuming behavior of the IP as depicted in Fig. 9. Since it always consumes certain level of power without much variation, simple averaged power can be a good assumption for the level 0 model. However, this simple model gives a very high error when clock gating is used. Note that previous papers [3][4] did not take into account the effect of clock gating deeply in their study for system level power modeling.

We summarize accuracy of each of the proposed power models for a clock gated H.264 decoder prediction IP design in Fig. 11. It shows average cycle error (Average) and maximum error. Simple models (Level 0) show comparatively large errors and we have to select a more detailed model to acquire accurate power data. The large dynamic variations under clock gating requires considering more detailed power modeling. The same module is synthesized with 90nm and 65nm library and power modeling was done for both cases. The 65nm case shows lower power consumption and lower power variation than 90nm case. Thus, for the 65nm case the same level power model shows comparatively better prediction than 90nm case.

The simulation time for decoding 60 frames of a QCIF video is shown in the Table 2. System level simulation is done on the desktop computer (Intel Pentium 4 3.2 GHz, 32-bit) and gate level simulation is done on a UNIX machine (Sun Fire V240 Server, 1.5GHz UltraSPARC IIIi dual-processor, 64-bit). For the gate level simulation, we could only simulate decoding of 2 frames. Note that for system level simulation, changing the depth of system power model does not show much difference in simulation time. However, gate level simulation takes much longer: *we can achieve a 120000× speed up through system level simulation for reasonably good results that are close to the accuracy of gate level simulations*. Furthermore, in reality, the simulation of many frames is not feasible on a UNIX machine because of huge size of generated simulation related files (total is about 6GB for decoding 2 QCIF size frames) which are described in Section 4.2.

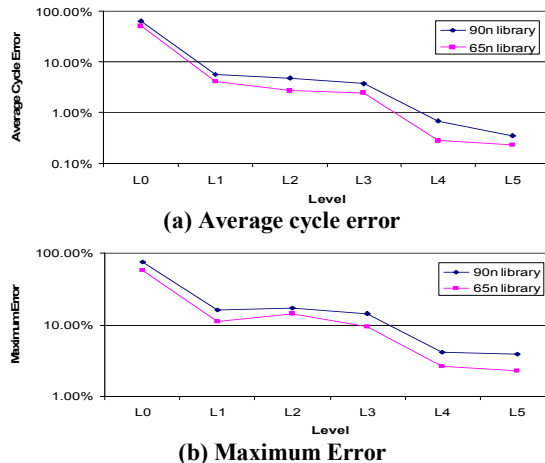


Figure 11. Error comparison for H.264 decoder prediction

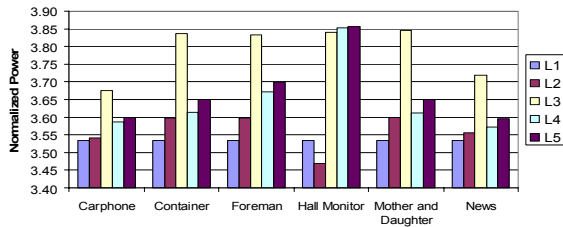
Table 2. Simulation Speed of Different System Level Power Models and Gate Level Simulation

	L0	L1	L2	L3	L4	Gate
Time (s)	20.93	21.55	21.6	21.62	23.38	≈2853000

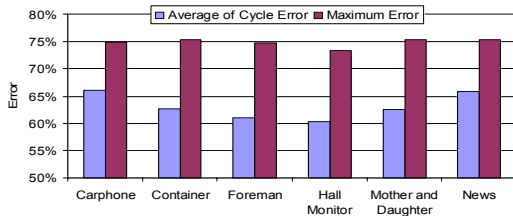
5.4. System level power exploration

In our final experiment, we simulated various input videos at the system level with the built in power models to observe the relation between characteristics of the input video with the power consumption for the H.264 decoder prediction IP, which is not feasible at the gate-level simulation due to the huge simulation time and large generated file sizes. There are many features of input video that can influence power consumption. We observed that inserting I frame vs. P frame, changing quantization parameter (QP) value and dynamic nature of input video have significant influence on power.

We used six different videos and observed that the power consumption significantly depends on characteristics of the input videos. As shown in Fig. 12 (a), the *Hall Monitor* and *Foreman* videos consume relatively more power than *News* and *Carphone* videos. This is due to their different characteristics. Each power level has different predictability. For instance, the L2 power data of *Hall Monitor* video has about 10% error compared to L4 and L5 in Fig. 12 (a). Thus, in order to archive stable and accurate prediction, L4 or L5 model is needed. Power values of L0 compared to L5 show large average cycle error and more serious maximum error with small variation for each input video case as shown in Fig. 12 (b). *With the system level design which has power models enabled by our methodology, we can simulate various inputs having different characteristics and observe their influence on power quickly, which is not practically feasible with gate level simulation.*



(a) Average power consumption



(b) L0 error compared to L5

Figure 12. Impact of video characteristics

6. Conclusion

We proposed a hierarchical power modeling methodology that is applicable to any kind of custom hardware IP block. We applied this methodology on an H.264 decoder prediction IP case study. With our method (using the fine-grained power model) and system level design which has annotated power data, we can achieve below 1% average cycle error and below 10% maximum error compared to gate-level simulation at about 120000× faster simulation speed. The choice of power level to use depends on accuracy needs and design effort required. With our model, we have the ability to rapidly explore clock-gating effects at the system level and observe the influence of various input videos (that vary system parameters such as quantization) which cannot be feasibly explored using traditional gate level simulation. Hamming distance and number of 1s are considered for the most accurate power model with 0.33% gain of average cycle error. To our best knowledge, this work is the first approach to comprehensively address system level power modeling of custom hardware IPs (ASIC). Future work will focus on automating power modeling for custom IP block, without the need of knowing structural details of the IP, to ease designer's effort while building the power model.

7. Acknowledgments

This research was partially supported by a grant from SRC Corp (2005-HJ-1330) and a fellowship from the Center for Pervasive Computing and Communication (CPCC).

References

[1] S. Pasricha, et al, "System-level power-performance trade-offs in bus matrix communication architecture synthesis," in *Proc. CODES+ISSS 2006*.

- [2] ITU-T Rec. H.264/ISO/IEC 11496-10. Advanced video coding, Final Committee Draft, Doc. JVT-G050. Mar. 2003.
- [3] I. Lee, et. al, "PowerViP: Soc power estimation framework at transaction level," in *Proc. ASP-DAC 2006*.
- [4] M. Onouchi, et al. "A system-level power-estimation methodology based on IP-level modeling, power-level adjustment, and power accumulation," in *Proc. ASP-DAC 2006*.
- [5] W. Ye, et al. "The design and use of SimplePower: a cycle-accurate energy estimation tool," in *Proc. DAC 2000*.
- [6] N. Bansal, et al. "Power monitors: a framework for system-level power estimation using heterogeneous power models," in *Proc. ICVD 2005*, pp. 579- 585.
- [7] S. Ravi, et al. "Efficient RTL power estimation for large designs," in *Proc. Int. Conf. on VLSI Design 2003*, pp. 431-439.
- [8] E. Macii, et al. "High-level power modeling, estimation, and optimization," in *Proc. DAC 1997*, pp. 504-511.
- [9] D. Brooks, et al. Martonosi, "Watch: a framework for architectural-level power analysis and optimizations," in *Proc. ISCA 2000*, pp. 83-94.
- [10] M. Sami, et. al, "Instruction-level power estimation for embedded VLIW cores," in *Proc. CODES*, Mar. 2000, pp. 34-38.
- [11] D. Sarta, et al. "A data dependent approach to instruction level power estimation," in *Proc. IEEE Alessandro Volta Memorial Workshop. Low-Power Design*, Mar. 1999, pp. 182-190.
- [12] N. Kavvadias, et. al "Measurements analysis of the software-related power consumption in microprocessors," in *Proc. IMTC 2003*, vol.2, pp. 981- 986.
- [13] C. Chakrabarti et. al, "Instruction level power model of microcontrollers," in *Proc. IEEE ISCAS*, pp. 176-179, 1999.
- [14] M. Gasteier et al. "Bus-based communication synthesis on system level," *ACM TODAES*, vol. 4, no. 1, pp. 1-11, Jan. 1999.
- [15] A. Pinto, et. al, "Efficient Synthesis of Networks on Chip," in *Proc. ICCD 2003*, pp. 146-150, Oct. 2003.
- [16] S. Pasricha, N. Dutt and M. Ben-Romdhane, "Constraint-driven bus matrix synthesis for MPSoC," in *Proc. ASPDAC 2006*.
- [17] M. Mamidipaka, et. al, "Analytical Models for Leakage Power Estimation of Memory Array Structures," in *Proc. CODES + ISSS 2004*, pp. 146- 151.
- [18] S. Pasricha, et al, "Extending the transaction level modeling approach for fast communication architecture exploration," in *Proc. DAC 2004*.
- [19] I. Richardson, H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia, John Wiley & Sons, 2003.
- [20] GNU R, <http://www.gnu.org/software/r/R.html>
- [21] L. Cai and D. Gajski, "Transaction Level Modeling: An Overview," in *Proc. CODES+ISSS 2003*.
- [22] SystemC initiative, <http://www.systemc.org>
- [23] SpecC System, Center for Embedded Computer Systems, <http://www.cecs.uci.edu/~specc/>
- [24] Synopsys Design Compiler, PrimeTime PX, www.synopsys.com
- [25] Cadence NC-Verilog, <http://www.cadence.com/>
- [26] Y. Bonhomme, et al. "Test Power: a Big Issue in Large SOC Designs," delta, p. 447, *Proc. DELTA '02*, 2002.