Learning-Based Power Modeling of System-Level Black-Box IPs

Dongwook Lee*, Taemin Kim†, Kyungtae Han†, Yatin Hoskote†, Lizy K. John* and Andreas Gerstlauer*
*The University of Texas Austin †Intel Corporation
{dongwook.lee, ljohn, gerstl}@utexas.edu {taemin.kim, kyungtae.han, yatin.hoskote}@intel.com

***Abstract*—Virtual platform prototypes are widely utilized to enable early system-level design space exploration. Accurate power models for hardware components at high levels of abstraction are needed to enable system-level power analysis and optimization. However, the limited observability of third party IPs renders traditional power modeling methods challenging and inaccurate. In this paper, we present a novel approach for extending behavioral models of black-box hardware IPs with an accurate power estimate. We leverage state-of-the-art-machine learning techniques to synthesize an abstract power model. Our model uses input and output history to track data-dependent pipeline behavior. Furthermore, we introduce a specialized ensemble learning that is composed out of individually selected cycle-by-cycle models to reduce overall complexity and further increase estimation accuracy. Results of applying our approach to various industrial-strength design examples shows that our models predict average power consumption to within 3% of a commercial gate-level power estimation tool, all while running several orders of magnitude faster.**

I. INTRODUCTION

Energy efficiency has become a critical design concern. Fast and accurate system-level power estimation approaches are needed to drive associated validation and optimization. Virtual platform models capable of simulating whole systems are widely utilized to enable rapid system-level design space exploration. Within this context, fast functional transaction-level models (TLMs) of hardware components are utilized as an alternative to slow co-simulation with low-level RTL. At the same time, a continued increase in system complexities has brought an increasing reuse of pre-designed hardware components acquired from third party vendors rather than being developed from scratch. Such black-box IPs are not usually well documented, and only functional simulation models without detailed architecture descriptions are provided together with pre-synthesized gate-level implementations. This limited observability makes power modeling for black-box IPs challenging.

Previous work in high-level power estimation has relied on accurate, data-dependent activity estimation using a fine-grain micro-architecture, register-transfer level (RTL) or IR level model [1–6]. Unfortunately, models provided for black-box IPs are usually only functionally equivalent ones, and necessary architectural information for fine-grain modeling is usually not available. The absence of internal architecture information limits its power estimation to coarse-grained simulation techniques using state-based models. Such models only support capturing average power transitions between different operating modes, which makes them inherently inaccurate.

In this paper, we propose a novel power model for black-box IPs that is aimed at capturing accurate power consumption using state-of-the-art-machine learning techniques. We extract a data-dependent, invocation-by-invocation power model from gate-level cycle-by-cycle power traces, which enables fast yet accurate fine-grain data-dependent power estimation. To synthesize the power model, we develop a specialized ensemble learning approach in which power models are decomposed into individual cycle-by-cycle models for efficient training and accurate prediction. The decomposed models use transaction-level I/O activity to estimate cycle-level behavior, which results in an overall high accuracy with low prediction overhead.

The rest of the paper is organized as follows: following a discussion of related work, Section III introduces an overview of our proposed methodology, while Section IV elaborates on each step in more detail. Section V shows experimental results of applying the flow to a set of industrial-strength design examples. Finally, Section VI concludes the paper with a summary and an outlook on future work.

II. RELATED WORK

To generate higher level power estimation models of white-box IPs, learning based approaches have most recently been utilized [4, 6–8]. In such approaches, power traces and activity data are obtained from a lower-level and higher-level simulation, respectively, to train a regression-based model. A key concern in learning-based approaches is managing model complexities to reduce generalization error and training overhead. Many previous approaches rely on a manual or trial-and-error based sampling of key signals or state variables [4, 6]. Other approaches decompose the full power model into several parts based on architectural information or manual decisions [7, 9]. To apply existing decomposition and feature sampling approaches, detailed architectural knowledge or designer insight is required, which is not usually available for black-box IPs. By contrast, we decompose the power model without any pre-assumed knowledge of the architecture, which also leads to better accuracy while reducing learning and estimation overhead.

Most of the previous work in high-level power modeling for black-box has relied on a coarse-grain estimation using a state-based component model [8, 10–12]. The projection of either given, documented states [10, 12], or state information estimated from external transaction events [8, 11] only supports capturing coarse-grain power transitions between different operating modes, such as read and write modes in memories or buses. To take in account data-dependent effects in power estimation of black-box components, a corresponding extension of coarse-grain state-based models was recently proposed [13]. In this approach, the authors first identify and refine states in which significant data-dependent power variations are observed. Cycle-level input switching activity information is then utilized to estimate data-dependent power consumption using a simple linear regression. This requires augmenting state-based models with the ability to capture

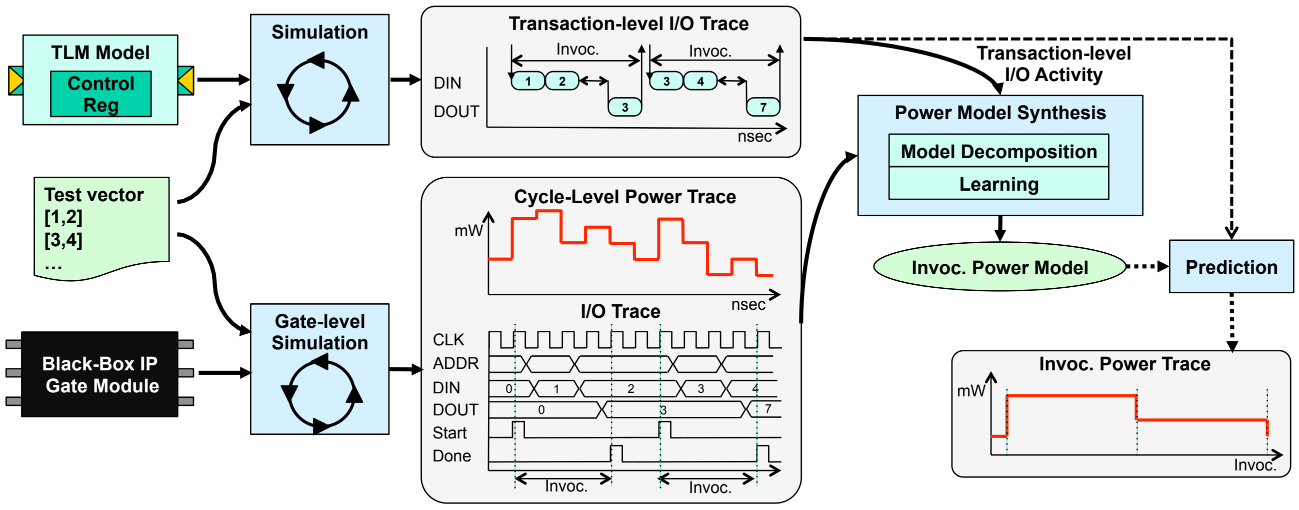


Fig. 1: Overview of power model synthesis flow.

cycle-by-cycle activity, which introduces a significant overhead in the simulation. Furthermore, a simple linear regression is inherently limited in accuracy. By contrast, our approach only requires capturing transaction-level activity and instead uses advanced learning methods to estimate component-internal power at cycle-level accuracy.

III. Power Model Synthesis Flow

Figure 1 shows an overview of our proposed power model synthesis flow. A given black-box gate-level model and corresponding TLM of a hardware IP are simulated with the same input vectors. Power synthesis then utilizes data I/O and control signal traces from TLM simulation together with cycle-level power traces from gate-level estimation to learn a power model. Based on the captured traces, we extract an invocation-by-invocation power model that enables fast yet accurate fine-grain data-dependent power estimation. Instead of building a single invocation-by-invocation power model, the synthesis flow decomposes power models into multiple models and individually trains them. In the prediction phase, the decomposed models are combined into an ensemble estimation model that predicts invocation-level power traces based on transaction-level I/O activity vectors. The synthesized power models are able to compute data-dependent power consumption estimates from online or offline activity traces captured in high-level TLM simulations.

Internal signal switching activity estimation is a key for data-dependent power modeling. In system-level black-box IPs, the internal signal activity is indirectly observed from switching activity of input and output signals. Internal signal activity for pipelined and multi-stage hardware architectures in the current cycle can be approximated from future and past switching activities of output and input ports, respectively. In a system-level model, re-arrangement of transactions and cycle-by-cycle I/O tracking is usually required to estimate cycle-level switching activity on input and output ports. However, such Hamming distance and switching activity computation is typically the most significant bottleneck for power estimation, and it is often much slower than actual functional simulation [14]. By contrast, our approach directly computes power estimates from unmodified high-level transaction-by-transaction activity, which significantly reduces computational overhead.

We assume that I/O interface mapping information between system-level transactions and the black-box data ports is given. System-level hardware models are usually written in system-level design languages (SLDLs), such as SystemC or SpecC. In such TLMs, communication interfaces are approximately modeled, and the detailed computation architecture is fully abstracted out. Models can also be purely functional, where no timing information is available. However, even a functional model has interfaces that map to corresponding data I/O ports. In general, we can find such mapping information in documents or test benches for gate level simulation. We assume that data port mapping, bitwidths and information about control signals is given, but internal architecture details are not available. Another assumption regarding observability in the system-level hardware model is that some important control registers or control ports are available. Such control dependencies are also necessary to model functional or performance behavior. The activities of control signals, such as mode selections, do not by themselves affect power consumption. However, their value is utilized to estimate operating mode dependent power variations.

IV. Power model Synthesis

In the following, we describe our power model and the proposed power model synthesis process utilizing state-of-the-art machine learning techniques. As indicated in Figure 1, this process consists of two steps: decomposing power models and learning with training data.

A. Power Model

Many previous approaches for power estimation at the RTL or micro-architecture level choose a linear function to model relation between the internal signal switching activity and power consumption of a hardware component. Given the internal and external signal switching activity vector $\mathbf{a}(t)$ at time t, power consumption $p(t)$ can be modeled as

$$p(t) = \mathbf{c}^{\mathbf{T}} \cdot \mathbf{a}(t), \tag{1}$$

where $\mathbf{c}$ denotes a coefficient vector. Note that, to simplify the model, we assume that related pins, e.g. of buses are

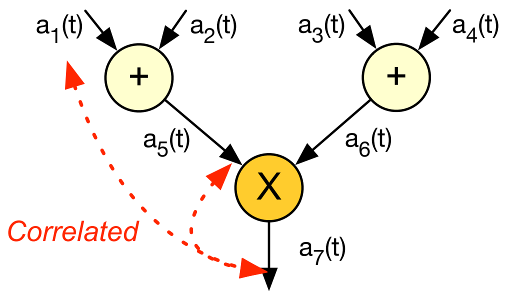


Fig. 2: I/O switching activity correlation.

grouped and Hamming distances within a group are utilized as alternative to individual bit-wise switching activity.

Ignoring glitching or asynchronous activities, we can convert the continuous power function into a discrete cycle-level model. In general, average power consumption $p_C(n)$ in cycle n can be modeled as

$$p_C(n) = \frac{1}{T}\int_{(n-1)T}^{nT} p(t)dt = \mathbf{c^T} \cdot \mathbf{a}(nT) = \mathbf{c^T} \cdot \hat{\mathbf{a}}(n), \quad (2)$$

where $\hat{\mathbf{a}}(n)$ is a discrete activity vector. The activity vector can be further refined into

$$\hat{\mathbf{a}}(n) = HD(\mathbf{d}(n), \mathbf{d}(n-1)), \quad (3)$$

where $HD()$ is the Hamming distance function and $\mathbf{d}$ denotes the column vector of signal groups.

Instead of internal signal switching activities not available in a black-box model, we only utilize switching activity of I/O signals for power modeling. The input and output switching activity of arithmetic operators such as adders and multipliers are linearly correlated [13]. This means that input switching activity of an operator can be modeled as a linear function of the input switching activity of the driving ancestor. For example, following equation (1), the power consumption of the dataflow graph in Figure 2 can be formulated as $p(t) \approx \sum_{i=1}^{7} c_i^T \cdot a_i(t)$. Using such a linear input-output relationship, we can simplify this equation to $p(t) \approx \sum_{i=1}^{1,2,3,4,7} c_i'^T \cdot a_i(t)$.

For pipelined or multi-stage architecture, we leverage past input and future output switching activities for prediction. Input activity and activity of the first pipeline stage register are linearly correlated. We can therefore propagate input switching activities through the pipeline and leverage such linear correlations across multiple stages by considering the input activity history. However, activity of inputs and registers deep in the pipeline, i.e. far away from the input are weakly correlated or not correlated at all. Instead, they are likely to be correlated to activity at the outputs leaving the pipeline. Hence, to handle deeply pipelined logic and improve accuracy, we also similarly consider future output activities for prediction. For a given pipeline of depth D, we can formulate its power as

$$p_C(n) = \sum_{i=0}^{D-1} \mathbf{ci}_i^{\mathbf{T}} \cdot \mathbf{a}_I(n-i) + \mathbf{co}_i^{\mathbf{T}} \cdot \mathbf{a}_O(n+i), \quad (4)$$

where $\mathbf{a}_I(n)$ and $\mathbf{a}_O(n)$ denote the input and output activity vectors, respectively, and $\mathbf{ci}_i$ and $\mathbf{co}_i$ are corresponding stage-specific coefficient vectors.

In this paper, we target prediction of average power consumption per invocation. As such, power estimation does not have to compute cycle-level information, which reduces estimation overhead. Given a per-invocation execution latency L and assuming that, without loss of generality, the invocation starts in cycle 0, we can compute the invocation power

$$\frac{1}{L}\sum_{j=1}^{L} p_C(j) = \frac{1}{L}\sum_{j=1}^{L}\sum_{i=0}^{D-1} \mathbf{ci}_i^{\mathbf{T}} \cdot \mathbf{a}_I(j-i) + \mathbf{co}_i^{\mathbf{T}} \cdot \mathbf{a}_O(j+i) \quad (5)$$

To simplify the equation, we assume the following initial condition of the concatenated input and output data:

$$\mathbf{d}_{IO}(n) = \vec{0}, n <= 0 \text{ or } n > L \quad (6)$$

Switching the first and second summation and considering the initial condition, we can remove the summation over the pipeline by introducing a new coefficient vector $\dot{\mathbf{c}}$ in the following way:

$$\frac{1}{L}\sum_{j=1}^{L} p_C(j) = \frac{1}{L}\sum_{j=1}^{L} \dot{\mathbf{c}}_{\mathbf{j}}^{\mathbf{T}} \cdot \mathbf{a}_{IO}(j) = \frac{1}{L}\sum_{j=1}^{L} \dot{\mathbf{c}}_{\mathbf{j}}^{\mathbf{T}} \cdot HD(\mathbf{d}_{IO}(j), \mathbf{d}_{IO}(j-1)) \quad (7)$$

Given the I/O data matrix $\mathbf{D}_{IO} = (\mathbf{d}_{IO}(0), \ldots, \mathbf{d}_{IO}(l))$, we can define the invocation-level power model $p_I(\mathbf{D}_{IO})$ as

$$p_I(\mathbf{D}_{IO}) = \frac{1}{L}\sum_{j=1}^{L}\sum_{p=0}^{P-1} \dot{c}_{j,p} \cdot HD(d_{p,j}, d_{p,j-1}), \quad (8)$$

where P denotes the total number of I/O signal ports. Equation (8) indicates that the invocation-level power model does not need pipeline-level or cycle-accurate data transition information.

If there is no transition in cycle n, the $HD(d_{p,n}, d_{p,n-1})$ result will be zero, which indicates that we can model the invocation-level power consumption by finding the contributed coefficient factors ($\ddot{\mathbf{c}}$) purely from transaction-level activity vectors ($\mathbf{a_{tr}}$):

$$p_I(\mathbf{a_{tr}}) = \ddot{\mathbf{c}}^T \cdot \mathbf{a_{tr}} \quad (9)$$

Transaction-level activity vectors are computed using Hamming distances over transaction data traces in a similar manner as in (3) and (8), where $\mathbf{a_{tr}}$ is a concatenated vector composed over all transactions in an invocation. The worst case dimension of $\mathbf{a_{tr}}$ is a $P \cdot L$, which may create generalization errors in learning processes. In the following section, we will reduce the model complexity by decomposition.

B. Power Model Decomposition

The complexity of the power model in (9) is directly proportional to the total number of transactions during a single invocation. To avoid overfitting, feature sampling, which reduces model dimensions by selecting a key subset of signals, can be a good approach. Unfortunately, a global power model using primary I/O signals as model features is not suitable to apply feature selection. The importance of various ports for estimating the power consumption can be a time-varying function depending on the internal IP state in a given cycle, which precludes feature selection to be applied at the port level without a significant loss in accuracy.

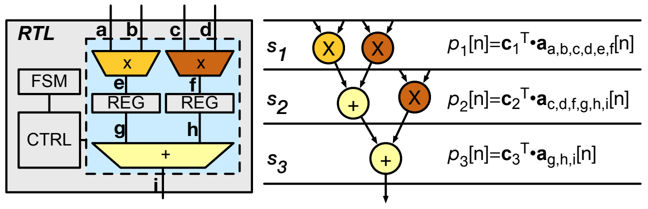


Fig. 3: Example of power model decomposition.

As an alternative to traditional feature selection, model decomposition can be applied by utilizing architectural information to reduce unnecessary signals. Such decomposition based on architectural information has previously been applied in white-box hardware models [9]. In white-box models, the hardware can be described as a finite state machine with datapath (FSMD). Given a finite set of FSMD states $S = \{s_1, \ldots, s_n\}$, the FSM state in cycle n can be defined by a trace function $t : n \to S$. The power consumption in a given cycle n is thereby dependent on resource utilization in FSM state $t(n)$. Given a finite set of hardware resources $R = \{r_1, \ldots, r_m\}$, a resource scheduling and binding function can be defined as $m : S \times R \to \{0, 1\}$. For instance, $m(r_1, s_1) = 1$ indicates that resource r_1 is utilized in state s_1. With such scheduling and binding information, we can formulate the power consumption in a given cycle n in the following manner:

$$p_C(n) = \sum_{i \in R} \mathbf{c}_{t(n),i}{}^{\mathbf{T}} \cdot \mathbf{a}_i(n) \cdot m(i, t(n)), \tag{10}$$

where $\mathbf{c}_{t(n),i}$ denotes the coefficient vector corresponding to resource i in state $t(n)$, and $\mathbf{a}_i(n)$ denotes the switching activity vector corresponding to resource i at time n. In this equation, the coefficient vector $\mathbf{c}_{t(n),i}$ is not only resource dependent, but also state-dependent. It is possible that the operand of the resource as well as connected glue and control logic is dependent on the state.

With resource scheduling and mapping information corresponding to $m(r_i, s_i)$, we can decompose equation (10) based on the given state information $t(n)$. We illustrate this with the help of a small example. Figure 3 shows a hardware where three resources are allocated. The power consumption of the complete hardware processor can be estimated using equation (1) with all switching vectors connecting to the resources. The power model $p_i(n)$ of a given control state s_i thus utilizes the much smaller subset of signals connecting the resources scheduled in the given state only. For example, the power consumption of state s_3 ($p_3(n)$) can be estimated with three signals instead of all 9 switching vectors.

Based on this idea, we can decompose the power model into separate and independent models for each state. However, this is not possible in black-box IP components, where state composition as well as scheduling and binding information is not available. Instead, since the current state is a function of the cycle count n, we can indirectly capture the state based on n and additional control data $\mathbf{d}_C$. We thereby assume that control signals $\mathbf{d}_C$, if any, determine the IP operating mode on a per invocation basis, but remain constant over one invocation. With this, we decompose the power model into separate and independent cycle-level models. In the process, we convert coefficients $\ddot{\mathbf{c}}$ of equation (9) into $\ddot{\mathbf{c}}_j(\mathbf{d}_C)$:

$$p_E(\mathbf{a_{tr}}, \mathbf{d}_C) = \frac{1}{L} \sum_{j=1}^{L} p_j(\mathbf{a_{tr}}, \mathbf{d}_C), \tag{11}$$

$$p_j(\mathbf{a_{tr}}, \mathbf{d}_C) = \ddot{\mathbf{c}}_j(\mathbf{d}_C)^{\mathbf{T}} \cdot \mathbf{a_{tr}} \tag{12}$$

The dimension of each cycle-level model p_j is the same as the invocation-level power model, i.e. the models use the complete transaction activity $\mathbf{a_{tr}}$ at their input. However, only a small part of the transaction activities actually contribute to the power consumption in any given cycle, which results in most of the elements in $\ddot{\mathbf{c}}_j(\mathbf{d}_C)$ being zero or small. To prune away such unimportant features, we additionally leverage a decision tree based feature selection for each decomposed model [15]. As a result, the uncertainty of the models is improved and there is less chance to run into generalization errors.

In (11), we assume that the execution latency L is a fixed value. In reality, however, hardware can have varying execution delays across invocations. In this case, we first decompose the power model based on execution latencies and then hierarchically perform the cycle-level decomposition.

Overall, the total number of models to learn is increased. However, each decomposed model uses the same input vectors, which enables parallel learning and prediction. Nevertheless, the decomposed models are not in linear form, which requires a proper model selection for learning.

C. Model Selection and Training

In general, linear regression over a set of training vectors has been widely employed to find coefficient of power models. If there is a linear correlation between the power consumption trend and control data, equation (12) can be converted into a linear form. However, the control data may have non-linear correlations with power consumption. Depending on how control signals are assigned to each mode, control data and power consumption trends can be linear or not. To handle such problems, models could be further decomposed along control inputs. By decomposing based on control signals, the power consumption behavior of each model would potentially become a linear function of the activity. The control data space, however, is exponential in the number of control signals, which results in tremendous learning overhead. Moreover, power behavior of complex arithmetic units is generally still not linear [2].

To find the individual model coefficient vectors for given control data $\mathbf{d}_C$, we instead leverage a decision tree based learning model. Decision tree learning is a technique commonly used in data mining to predict a target by learning decision rules from given training data. Training builds a tree by choosing features and splitting data based on selected feature values to reduce the standard deviation of the total training set. One can then predict the target value by traversing the trained tree along paths where most of the important features are located, where leaves then represent the predicted value. A decision tree learning model can find non-linear correlations, which is suitable for learning of non-linear power consumption patterns. Moreover, the diversity of the model is an important factor for reducing the total error in the final ensemble-based invocation-level model, as will be discussed in detail in the following section. A decision tree learning model provides more diversity than linear regression even with the same training vectors, which is also one of the reasons why we choose such a model [16].

D. Ensemble Estimation Model

Trained cycle-level models are combined to predict the invocation-level execution power of the hardware IP. In equation (12), the individual models take the same transaction-level I/O activity vector $\mathbf{a_{tr}}$ to predict each corresponding cycle power. By averaging the predicted cycle powers, we can estimate the data-dependent execution power consumption, which is a form of ensemble learning. Ensemble learning utilizes diversity over multiple learning models to achieve better accuracy. Traditional ensemble methods introduce diversity by dividing the training set, training each model with the partitioned training set, and then predicting the target value as the average over the prediction values of each model. By contrast, we introduce diversity by decomposing the model into cycle-level models. As such, we train each individual model with the same feature vectors, but inherently achieve diversity since individual model targets (cycle powers) are different.

Ensemble models are well known for providing better performance than single models in many cases [17]. In our case, we can prove that the proposed ensemble model in (11) shows better performance than the single invocation model from (9). We can define the error-free perfect target function as $h(\mathbf{a}_{tr})$. The sum-of-square errors of the single invocation model (E_I) can then be defined as

$$E_I = \mathbb{E}_{\mathbf{a}_{tr}}[\{p_I(\mathbf{a}_{tr}) - h(\mathbf{a}_{tr})\}^2] = \mathbb{E}_{\mathbf{a}_{tr}}[\varepsilon(\mathbf{a}_{tr})^2], \qquad (13)$$

where $\mathbb{E}_{\mathbf{a}_{tr}}$ denotes the expectation with respect to the distribution of the input vector $\mathbf{a}_{tr}$, and the control parameter is ignored for simplification. In the same way, the sum-of-squared error of the ensemble model (E_E) can be given by

$$\begin{aligned} E_E &= \mathbb{E}_{\mathbf{a}_{tr}}\left[\{\tfrac{1}{L}\sum_{j=1}^{L} p_j(\mathbf{a}_{tr}, \mathbf{d}_C) - h(\mathbf{a}_{tr})\}^2\right] \\ &= \mathbb{E}_{\mathbf{a}_{tr}}\left[\{\tfrac{1}{L}\sum_{j=1}^{L} \varepsilon_j(\mathbf{a}_{tr})\}^2\right] \end{aligned} \qquad (14)$$

To simplify the problem, we assume that errors have zero mean and are uncorrelated,

$$\mathbb{E}_{\mathbf{a}_{tr}}[\varepsilon_m(\mathbf{a}_{tr})] = 0, \quad \mathbb{E}_{\mathbf{a}_{tr}}[\varepsilon_m(\mathbf{a}_{tr})\varepsilon_l(\mathbf{a}_{tr})] = 0, \quad m \neq l \qquad (15)$$

We further assume that all models are trained well and the sum-of-square errors of individual models are the same for simplification. We can obtain

$$E_E = \frac{1}{L} E_I \qquad (16)$$

Hence, the error of the ensemble model can be reduced by a factor of L when the assumption that errors are uncorrelated is satisfied. In general, each decomposed model predicts a corresponding cycle power, which implies that individual cycle errors may not be highly correlated. Moreover, we utilize decision tree learning to prevent correlations between models. As such, we can expect that the ensemble model always provides better accuracy than the single invocation one.

In the same manner, we can prove that the ensemble model has better accuracy than averaging over a single cycle-level model. A single cycle-level model estimates cycle-by-cycle power behavior using a single model instead of multiple decomposed ones. The sum-of-squared errors of the single cycle-level model can be formulated as equation (14). However, since each error is generated from the same model, i.e. is highly correlated, it cannot satisfy the assumption in (15). As a result, we can also expect that the ensemble model shows better prediction accuracy than a single cycle-level model.

TABLE I. Benchmark summary.

	Gates	Total I/O Ports	I/O Delay	Exec. Cycles
GEMM	964	2/1	12	436
DCT	6309	4/4	64	96
QUANT	1456	3/1	4	410
R2Y	1757	4/1	6	806/742
HDR	7887	11/1	57	825

TABLE II. Training and test summary.

	Train Invoc.	Test Invoc.	Total Test Cycles
GEMM	1250	5000	2,180,000
DCT	2700	10800	1,015,200
QUANT	10000	36864	5,038,080
R2Y	1200	3600	2,786,400
HDR	900	1300	1,072,500

V. Experimental Results

We have implemented our power synthesis flow utilizing the scikit-learn [18] machine learning library. We applied the flow to generate models for black-box hardware designs of a 6x6 general matrix- matrix multiplication (GEMM), a 2D discrete cosine transform (DCT), a quantizer (Quant), an RGB to YUV color converter (R2Y), and a weight computation block of a high dynamic range (HDR) imaging application [19]. The quantizer has two control inputs for choosing a quantization table and the image scaling quality. The RGB to YUV converter has a control input for switching the sampling mode. All benchmarks were synthesized using industry-standard EDA tools with the Nangate 45nm Open Cell Library [20] at 200Mhz clock frequency. I/O traces were collected from corresponding high-level functional implementations. Gate-level power was estimated using a commercial EDA tool with VCD files generated from full gate-level simulation. All experiments were performed on a quad-core Intel i7 linux workstation running at 3.5 GHz. Table I summarizes benchmarks and synthesis results including the total input and output port numbers, input to output delays, and execution cycles for a single invocation. Note that the invocation latency for the R2Y benchmark varies as a function of the chosen sampling parameter.

To generate test vectors, GEMM and HDR designs were simulated with 5000 random test matrices and a 200x100 24-bit RGB image, respectively. The R2Y design converts a 512x512 24-bit RGB image to a 512x512 YUV image for two different sampling modes. A 640x320 24-bit RGB image are used to generate DCT and QUANT test vectors. Three different quality factors and two different table setting are utilized to generate the test set for the QUANT design. To learn each power model, we used training sets generated from different random seeds or images. All models are trained with sufficient training data. In each case, we were able to synthesize power models within 21 minutes including trace generation. Depending on the trace length, model synthesis takes between 6 and

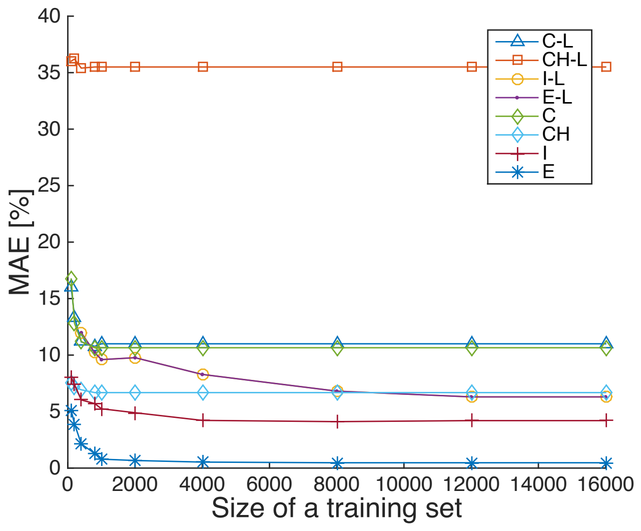


Fig. 4: Learning overhead and model accuracy comparison for QUANT.

19 minutes for one-time gate-level simulation plus 20 to 120 seconds for total training time. Table II summarizes the size of training and test sets.

Figure 4 shows the learning overhead and accuracy of the proposed ensemble power model (E) as compared to other learning models. We measure the data-dependent invocation-by-invocation mean absolute error (MAE) of values predicted by each model compared to gate-level simulations, normalized against average power over the full simulation. The major learning overhead is collecting gate-level simulation results to construct the training vectors. By increasing the size of training sets, we explore trade-offs between learning overhead and accuracy of the models. We compare between a single cycle-level power model utilizing the current input switching activity (C), a single cycle-level power model utilizing the Hamming distance input history (CH), a single invocation-level power model (I), and the proposed ensemble power model (E). All of these models utilize either the decision tree regression or a least squared linear regression (-L).

In most cases, power models considering the pipelined structure (CH, I, E) show better accuracy compared to the model only considering current I/O activity (C). Models utilizing decision tree regression always show better results than simple linear ones. We can observe that ensemble learning utilizing the decision tree regression (E) provides the best accuracy for the same size of the training set, reaching more than 99% accuracy for a training set with 4000 vectors. Ensemble and single-invocation models with linear regression (E-L and I-L) show the same accuracy, which indicates that individual linear regression models in the ensemble model are highly correlated. We utilize CH, I and E power models for further analysis.

Figure 5 and Figure 6 compare model accuracy and speed across various benchmarks. For our ensemble power model, we compared a decision tree regression (E-DT), a linear Bayes ridged model (E-BR), a gradient boosting regression composed of multiple decision trees (E-GB) and a non-linear support vector regression with a Gaussian radial basis function kernel (E-SVR). An SVR with a high order polynomial kernel function was not able to generate a stable model (producing errors of

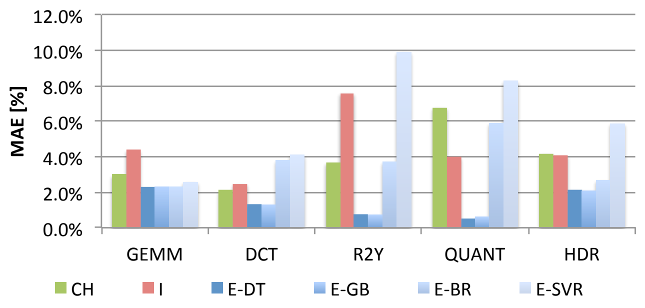


Fig. 5: Invocation-by-invocation power model accuracy.

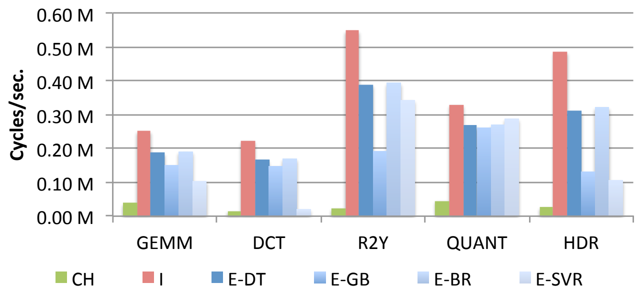


Fig. 6: Estimation speed of models.

more than 50%). For all benchmarks, ensemble power models utilizing the decision tree (E-DT) or the gradient boosting (E-GB) show better accuracy than others. The support vector regression shows the worst results among all ensemble models. When comparing speed (Figure 6), the single invocation model is faster than the others. Among ensemble power models, linear Bayes ridged (E-BR) and decision tree (E-DT) regressions are fastest. The decision tree model is thereby faster than a gradient boosting (E-GB) one at similar accuracy. Overall, when comparing different regression methods and models, results confirm that the ensemble power model utilizing a decision tree regression provides the best trade-off between accuracy and speed.

Table III and Table IV further summarize and detail accuracy and speed of models across benchmarks. We measure the data-dependent invocation-by-invocation mean absolute error (MAE), maximum error, and total average error across a full estimation. All error metrics are normalized against average power. Overall, we can observe that the ensemble model using decision tree regression (E) improves accuracy over the single invocation model (I) by a factor of 3x on average across all error metrics. The single invocation and cycle-level models show the worst estimation result in R2Y and QUANT benchmarks, respectively. By contrast, the ensemble model is able to capture cycle- and invocation- level power variations, which results in a better accuracy for R2Y and QUANT power estimation than the others. Estimation accuracy for GEMM, DCT and HDR designs with large input to output (I/O) delays is worse than for others, which indicates that the remaining errors in the ensemble model are related to the inherently limited external observability for designs with deep logic. Overall, our ensemble models estimate invocation-level power consumption to within 3% MAE and 15% maximum error compared to gate-level power results. In all cases, average errors across the whole simulation are below 2%.

Table IV summarizes the estimation speeds of high-level

TABLE III. Accuracy of modeling.

	MAE			Max. Error			Avg. Error		
	CH	I	E	CH	I	E	CH	I	E
GEMM	3.0%	4.4%	2.3%	19%	23%	12%	0.6%	0.5%	0.2%
DCT	2.1%	2.5%	1.4%	26%	31%	13%	0.5%	0.3%	0.0%
R2Y	3.7%	7.6%	0.9%	13%	37%	5%	1.5%	3.0%	0.8%
QUANT	6.8%	4.0%	0.5%	40%	28%	4%	0.5%	1.8%	0.1%
HDR	4.2%	4.1%	2.2%	18%	28%	13%	3.8%	2.0%	1.7%
Avg.	4.0%	4.5%	1.5%	23%	29%	9%	1.4%	1.5%	0.6%

TABLE IV. Estimation speed [Cycles/Sec.].

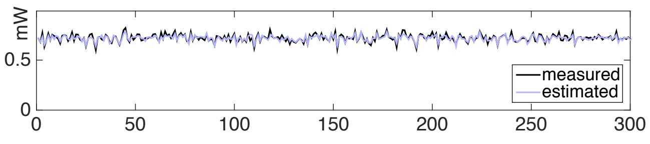
	CH	I	E	Gate
GEMM	38.5K	252.8K	189K	378
DCT	16.0K	224.3K	168K	188
R2Y	24.6K	548.8K	378K	2099
QUANT	46.3K	330.7K	304K	1518
HDR	27.4K	319.1K	319K	199
Avg.	30.5K	368.7K	273K	876

models as also compared to a gate-level simulation. Overall, both the ensemble (E) and cycle-level (CH) power models estimate at a cycle-by-cycle level, but the ensemble model is on average 9 times faster. This shows that transaction-level estimation improves simulation throughput significantly. Compared to the single invocation model (I), the ensemble power model (E) is on average 1.3 times slower. It is, however, nearly 312 times faster than gate-level estimation.

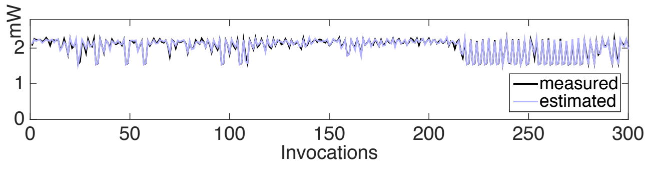
Finally, Figure 7 shows the invocation-by-invocation power traces of estimated versus measured power waveforms for the benchmark designs using the ensemble model. As traces show, our synthesized models can accurately track power behavior within each invocation, as well as data-dependent effects across different invocations of the same design.

VI. Summary and Conclusions

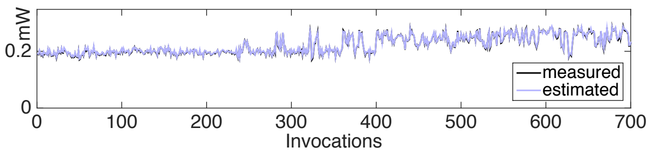
In this paper, we presented a novel approach for extending behavioral, transaction-level models of hardware IPs with accurate power estimates. Our power model synthesis flow leverages state-of-the-art machine learning techniques to synthesize an invocation-level power model. The power model directly utilizes transaction-level I/O activity and control information for fast estimation. The proposed model decomposition and ensemble estimation enable accurate data-dependent power prediction. Our flow has been evaluated on several industry-strength benchmark designs and generated models. Results show that our proposed power model is able to achieve 9x faster prediction compared to cycle-level power models, and orders of magnitude speedup compared to gate-level power simulation, all while estimating power with less than 3% invocation-by-invocation and less than 2% average error. In future work, we plan to integrate such fast and accurate power models with virtual platform or full-system simulators to support system-level architecture exploration.



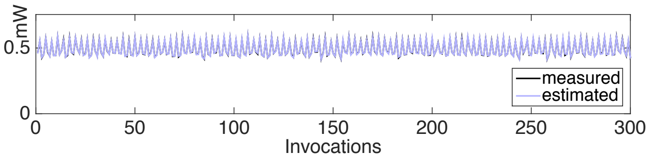
(a) GEMM simulation



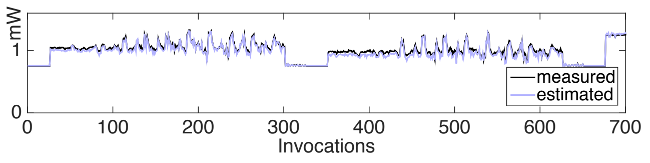
(b) DCT simulation



(c) R2Y simulation



(d) QUANT simulation



(e) HDR simulation

Fig. 7: Invocation-by-invocation power traces.

References

[1] S. Ravi *et al.*, "Efficient RTL power estimation for large designs," in *VLSID*, 2003.
[2] A. Bogliolo *et al.*, "Regression-based RTL power modeling," *TODES*, vol. 5, no. 3, pp. 337–372, Jul. 2000.
[3] S. Gupta and F. Najm, "Power modeling for high-level power estimation," *TVLSI*, vol. 8, no. 1, pp. 18–29, Feb. 2000.
[4] D. Sunwoo *et al.*, "PrEsto: An FPGA-accelerated power estimation methodology for complex systems," in *FPL*, Aug. 2010.
[5] Y. Park *et al.*, "A multi-granularity power modeling methodology for embedded processors," *TVLSI*, vol. 19, no. 4, pp. 668–681, Apr. 2011.
[6] C.-W. Hsu *et al.*, "PowerDepot: integrating IP-based power modeling with ESL power analysis for multi-core SoC designs," in *DAC*, 2011.
[7] Y.-H. Park *et al.*, "System level power estimation methodology with H.264 decoder prediction IP case study," in *ICCD*, Oct. 2007.
[8] S. Schürmans *et al.*, "Creation of ESL power models for communication architectures using automatic calibration," in *DAC*, May 2013.
[9] D. Lee *et al.*, "Dynamic power and performance back-annotation for fast and accurate functional hardware simulation," in *DATE*, 2015.
[10] I. Lee *et al.*, "PowerViP: SoC power estimation framework at transaction level," in *ASP-DAC*, 2006.
[11] E. Copty *et al.*, "Transaction level statistical analysis for efficient micro-architectural power and performance studies," in *DAC*, 2011.
[12] C. Trabelsi *et al.*, "A model-driven approach for hybrid power estimation in embedded systems design," *EURASIP*, vol. 2011, no. 1, Mar. 2011.
[13] D. Lorenz *et al.*, "Data-and state-dependent power characterisation and simulation of black-box rtl ip components at system level," in *DSD*, 2014.
[14] M. Pedram, "Advanced power estimation techniques," in *Low power design in deep submicron electronics*, W. Nebel and J. Mermet, Eds. Kluwer Academic Publishers, 1997.
[15] C. Ratanamahatana and D. Gunopulos, "Feature selection for the naive bayesian classifier using decision trees," *AAI*, vol. 17, no. 5-6, pp. 475–487, 2003.
[16] M. S. Gashler *et al.*, "Decision tree ensemble: Small heterogeneous is better than large homogeneous," in *ICMLA*, Dec. 2008.
[17] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2007.
[18] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *JMLR*, vol. 12, pp. 2825–2830, 2011.
[19] T. Mertens *et al.*, "Exposure fusion," in *PG*, Oct. 2007.
[20] Nangate, "Open Cell Library," http://www.nangate.com.