Power Emulation: A New Paradigm for Power Estimation

Joel Coburn, Srivaths Ravi, and Anand Raghunathan
NEC Laboratories America, 4 Independence Way, Princeton, NJ 08540
{jcoburn,sravi,anand}@nec-labs.com

ABSTRACT

In this work, we propose a new paradigm called **power emulation***, which exploits hardware acceleration to drastically speedup power estimation. Power emulation is based on the observation that most power estimation tools typically perform the following sequence of operations: simulating the circuit to obtain value traces or statistics for the inputs of its constituent components, evaluating power models for each circuit component based on the input values seen during simulation, and aggregating the power consumption of individual components to compute the circuit's power consumption. We further recognize that the steps involved in power estimation (power model evaluation, aggregation) can themselves be thought of as synthesizable functions and implemented as hardware circuits. Thus, any given design can be enhanced by adding to it "power estimation hardware", and the resulting power model enhanced circuit can be mapped onto a hardware prototyping platform. While drastic speedups in power estimation (orders of magnitude) are possible using this approach, a significant challenge arises due to the increase in circuit size as a result of adding power estimation hardware. We propose a systematic methodology to reduce the size of the power model enhanced circuit through a suite of techniques, including power model reuse across different circuit components, regulating the granularity of components for power modeling, exploiting inter-component power correlations, resource sharing for power model computations, and the use of block memories for efficient storage within power models. We demonstrate the benefits of the proposed power emulation paradigm by applying it to register-transfer level (RTL) power estimation for industrial designs, resulting in speedups from around* 10*X to over* 500*X compared to state-of-the-art commercial power estimation tools.*

Categories and Subject Descriptors

B.5.2 [**Hardware**]: Register-Transfer-Level Implementation - *Design Aids - Simulation*; B.8.2 [**Hardware**]: Performance and Reliability - *Performance Analysis and Design Aids*; C.4 [**Computer Systems Organization**]: Performance of Systems - *Modeling techniques*

General Terms

Design, Measurement, Performance, Algorithms, Experimentation

Keywords

Power Estimation, Emulation, Design, Design Methodologies, Macromodels, FPGA, Hardware Acceleration, Register-Transfer Level, Simulation

1. INTRODUCTION

Power consumption has emerged as a primary design metric for a wide range of electronic systems, ranging from battery-powered appliances (*e.g.*, cell phones, PDAs, and networked sensors), to high-performance computing and communication systems. In fact, power consumption is regarded a likely limiting factor to the increasing scales of integration predicted by Moore's law [1]. Minimizing and managing power consumption requires appropriate tool support for power estimation and optimization at various stages in the design flow. Extensive research in the low power design area has addressed the problem of power estimation for circuits described at varying levels of abstraction, including the transistor level, logic (or gate) level, register-transfer level, behavioral level, and system level. These technologies have been incorporated into various commercial power estimation tools [2, 3, 4, 5].

Advances in fabrication technologies have led to shrinking device sizes, and consequently increasing chip complexities. The increase in circuit sizes and testbench complexities is straining the capabilities of conventional power estimation tools. For example, RTL power estimation for a 1.25 million transistor MPEG4 decoder circuit when decoding just 4 frames of a video stream required 43 minutes and 55 minutes using two different state-of-the-art commercial tools that we evaluated [3, 6]. Gate- and transistor-level power estimation tools are even slower (10 to 100X). The poor speed of power estimation tools limits their utility in the design flow. Clearly, such estimation tools cannot be used in an iterative manner for architectural exploration. Hence, efficient power estimation for large designs is a critical challenge. Raising the level of abstraction to the system level can lead to substantial efficiency improvements, but accuracy is significantly compromised.

This work addresses the problem of efficient power estimation by exploiting hardware accelerators (that have been commonly used for accelerating functional simulation). Power estimation is typically performed by evaluating power models for different circuit components, based on the input values seen to each component during circuit simulation. We observe that on the power models can themselves be thought of as hardware circuits. Thus, any design can be enhanced with power estimation hardware, and mapped onto a hardware prototyping platform (such as an FPGA). While the basic idea of power emulation is applicable at any level of abstraction, in this paper, we explore it in the context of RTL power estimation. We propose a systematic power emulation methodology, and demonstrate that it can facilitate *orders of magnitude* speedup in power estimation (by factors of 10X to over 500X). We identify and address the major challenges involved in power emulation, including the increase in size of the circuit when power estimation hardware is added to it. We demonstrate the feasibility and benefits of the proposed power emulation methodology through experiments on several industrial designs. Our results show that power emulation has the potential to significantly extend the capabilities of current power estimation tools. Much like functional emulation, power emulation can enable the investigation of hardware power consumption characteristics in the context of realistic system environments and workloads (*e.g.*, booting up an OS) — a task that can often be achieved only after fabrication with current design flows.

1.1 Related Work

Extensive work has been performed in power estimation techniques at the transistor, gate, and register-transfer levels of design abstraction [7, 8, 9]. At the transistor level, power estimation is typically performed as a by-product of circuit simulation. Gate-level power estimation requires the computation of signal statistics for the signals in the circuit, which can be performed through simulation, probabilistic analysis, or simulation with statistical sampling [7, 8]. Of these, simulation with a comprehensive testbench is the most commonly used in practice, due to its accuracy and the ability to produce detailed feedback such as power breakdown *vs.* time for different circuit components. At the register-transfer level, approaches to power estimation include analytical techniques [10, 11], characterization-based macromodels [12, 13, 14], and fast synthesis into gate-level descriptions [15]. While a few attempts have been made to perform hardware power estimation at the behavioral level [16, 17, 18], their accuracy is limited due to the lack of structural circuit information in behavioral descriptions. At the system level, most research has focused on developing power models for different system components, including processors [19], memories [20, 21], on-chip buses [22, 23], *etc.*

In practice, most current commercial design flows utilize RTL and gate-level power estimation tools. However, due to their poor efficiency for large designs, their applicability is limited until late in the design flow, or they are

DAC 2005, June 13–17, 2005, Anaheim, California, USA.

applied only to small parts of a design. Speedup techniques such as statistical sampling [24] and circuit partitioning for parallel mixed-level simulation [25] offer useful improvements in efficiency, but are not sufficient in the face of ever-increasing circuit complexities.

We believe that the idea of leveraging emulation platforms for power estimation is complementary to most previous work, and can be applied at different levels of abstraction. In [26], we proposed the basic concept of power emulation and offered an estimate of the speedup possible using this technique. However, realizing the benefit of power emulation requires addressing the main challenge outlined in this work, namely, reducing the area overhead due to power estimation hardware. We propose a systematic methodology that incorporates various area reduction techniques and demonstrate the feasibility of this technology for large designs in the context of RTL power estimation.

2. POWER EMULATION: PRELIMINARIES

While the concept of power emulation is applicable at multiple levels of abstraction, we discuss it here in the context of register-transfer level (RTL) power estimation. Since RTL descriptions in practice can contain an arbitrary combination of macroblocks (arithmetic units, registers, multiplexers, *etc.*) and random logic gates, the proposed techniques apply to gate-level descriptions as a special case.

For subsequent explanations, we consider a characterization-based power estimation methodology, wherein a "power macromodel library" is obtained by characterizing the power consumption of a universal library of RTL components using gate- or transistor-level implementations. These power macromodels are used to compute the power consumption of each component in the RTL circuit, based on the values of the component's input/output signals during simulation.

The basic idea in power emulation is to identify the operations performed during power estimation, express them as hardware circuits, and append them to the circuit under consideration. Fig. 1 illustrates this process with the help of an example RTL circuit, which is used to perform binary search. The circuit, which we call the *power model enhanced circuit*, has several new components specifically added for power estimation, which are shaded in Fig. 1. The enhancements include (i) *power models* that are instantiated for each RTL component to track the component's input/output signals and compute a power value whenever triggered, (ii) a *power strobe generator* to trigger the evaluation of the power models, and (iii) a *power aggregator* to accumulate the power consumption of individual RTL components to compute the total power consumption.

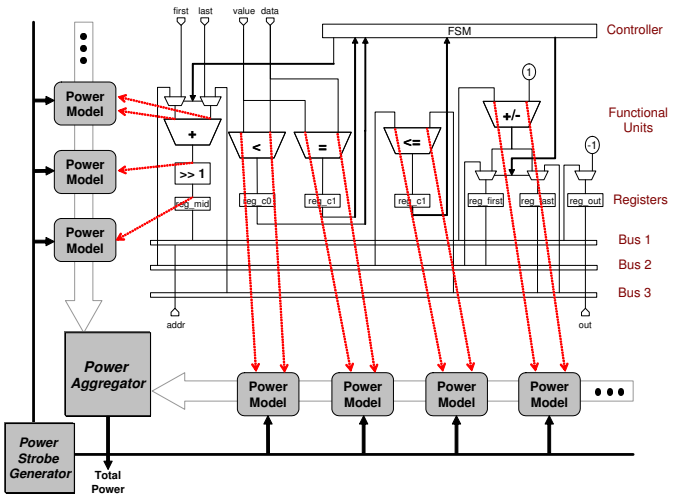


Figure 1: An example RTL circuit enhanced for power emulation

Power strobe generation is similar to clock generation and is done separately for each clock domain in the design. Power aggregation is simply implemented as a sequence of additions to accumulate the total power from the outputs of the power models. The power models themselves represent the most significant portion of the power estimation hardware, hence we examine them in further detail. Since the power model is just a hardware manifestation of the corresponding RTL component's power macromodel, its internals depend on the nature of the power macromodel function. For the sake of the following explanations, let us consider the cycle-accurate linear regression based macromodel [13, 14], which expresses the power consumed in an RTL component with n input/output bits as $\sum_{i=1}^{n} Coeff_i * T(x_i)$, where $Coeff_i$ represent the power model coefficients, and $T(x_i)$ is the transition count (0 or 1) at each input/output bit. The hardware implementation of this power model used for the purpose of power emulation is shown in Fig. 2. The inputs to the power model include the input/output bits of the component being monitored, and a power strobe. The output of the power model is the component's power consumption. The power model performs the following computation

$$Power = tc(queue\_x_1(0), queue\_x_1(1)) * Coeff_1 + \cdots + tc(queue\_x_N(0), queue\_x_N(1)) * Coeff_N$$

where, tc represents the transition count (XOR) function. The inputs to tc come from a set of internal queues that maintain the previous and current values of each component input/output. Since the transition count is a binary value, the multiplications in the power model equation are simply implemented using vector AND gates, as shown in Fig. 2. The products of the coefficients and respective transition counts are added to obtain the power consumed by the component in the current strobe period.

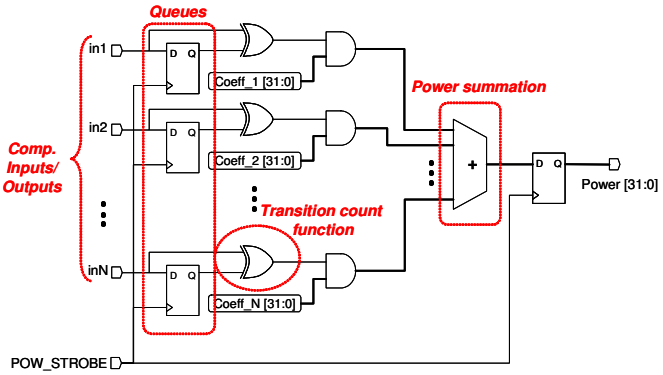


Figure 2: Structural netlist of a power model for a component with N input/output bits

3. MOTIVATION

In this section, we discuss the performance of conventional RTL power estimation for a large design and the benefits of using power emulation in accelerating this process. We also present the challenges involved in the use of power emulation.

Let us consider an MPEG4 decoder that is used in chips for mobile handsets. The design consists of seven sub-designs, with a total of 26,048 RTL components. The sub-designs include an input buffer (Readbit), a DCT coefficients block (Dct_coeff), a variable length decoder (Vld), an inverse quantization block (Ispq), an inverse DCT (Idct_da), a motion vector block (Mv), and a motion compensation block (Mc). Table 1 compares the execution times for RTL power estimation and power emulation for the MPEG4 decoder, while decoding 4 frames of an input video stream. For RTL power estimation, we report the time taken using (a) NEC's internal power estimation tool [6] and (b) PowerTheater [3], both running on a 900MHz UltraSparc-III workstation with 8GB RAM. For power emulation, we present the results when we consider a target platform consisting of an FPGA with unlimited resources, clocked at the design's functional emulation speed of 30 MHz. The results show that 524X (411X) speedup is possible using hardware emulation with respect to NEC's RTL power estimator (PowerTheater).

Table 1: RTL power estimation *vs.* power emulation for the MPEG4 design

	NEC-RTPower	PowerTheater	Power Emulation
Run Time	**3300sec**	**2587sec**	**6.3 seconds**

However, the gains of power emulation can be realized only if the enhanced RTL description can be fit into an emulation platform. Fig. 3 presents the area overheads of adding power models to each sub-design in the MPEG4 design. Each sub-design was synthesized with Synplicity's Synplify Pro [27] and targets the largest Xilinx Virtex-II FPGA, the XC2V8000 [28]. The area reports indicate the following: (i) on an average, the power model enhanced RTL description increased the design area by as much as 18.2X, (ii) the enhanced version of Vld itself exceeded the capacity of the XC2V8000 FPGA and (iii) fitting the power model enhanced MPEG4 design on an emulation

platform would require a capacity nearly 4.5 times larger than the XC2V8000 FPGA.

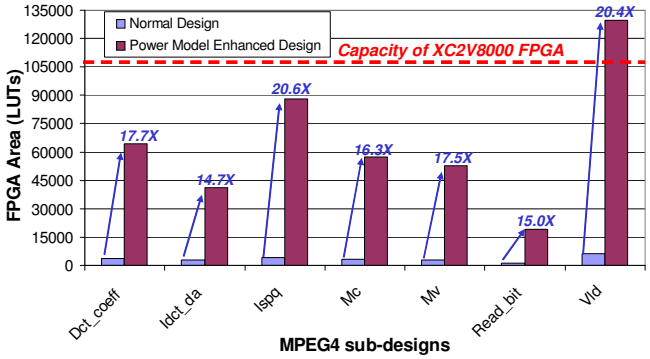


Figure 3: Area requirements for the original and power model enhanced RTL descriptions of various blocks in the MPEG4 design

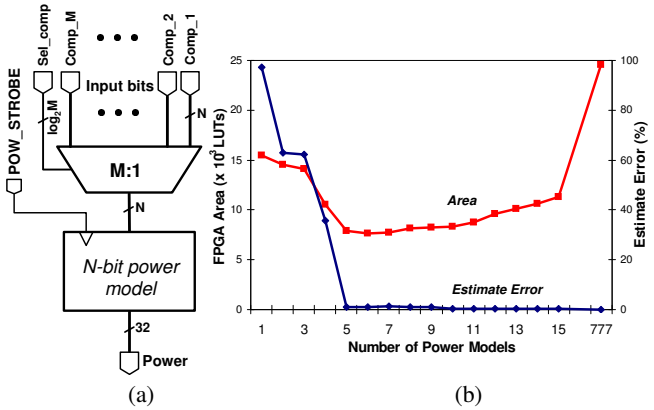
The results clearly indicate that the area overheads of power model enhanced circuits are unreasonably high and can prevent the deployment of power emulation in many designs. Thus, for power emulation to be practical, we require various techniques that can reduce the hardware requirements of enhanced RTL designs, without compromising power estimation accuracy.

4. DESIGN TECHNIQUES FOR POWER EMULATION

In this section, we present various techniques that reduce the area requirements of power model enhanced circuits. We base our techniques on the observation that power models dominate the overall area, since they are instantiated for every component in the design. Therefore, our suite of optimizations attempts to reduce the number of power models in a design, and also to enable area-efficient implementations of the power model logic, without a significant loss of estimation accuracy.

4.1 Power Model Re-Use Through Clustering

The number of power models required for a design can be reduced by grouping components into clusters, and using a single power model to service all components in a cluster. In effect, a component may be considered by the power model (or "sampled") only once in several cycles, similar to statistical sampling [24]. The architecture of a generic power model that services a cluster of M components is shown in Fig. 4(a). It consists of (i) an input multiplexer that selects the component inputs being monitored in a cycle, and (ii) a basic N-bit power model for calculating the component power consumption value, where N is the maximum bitwidth among all components in the cluster. The area of the generic power model is chiefly governed by trade-offs between the number of components being serviced (which decides the multiplexer size) and the largest bitwidth component (which decides the size of the remaining power model logic such as the adder tree).



(a) (b)

Figure 4: (a) Generic power model for M components with maximum component bitwidth N, and (b) Area/accuracy *vs.* number of power models for the Bubble_Sort design

EXAMPLE 1.: Fig. 4(b) analyzes the impact of clustering on area reduction and estimation error for an example design, Bubble_Sort. The design contains 777 RTL components, and we consider various clustering solutions by varying the number of generic power models allowed. At one extreme, we have 777 power models (one power model per component) and this baseline configuration results in the highest area cost with no additional estimation error beyond the inherent error from RTL macromodels. When the number of generic power models reduces to six, the area curve has reached a minimum value of 7,615 LUTs (3X smaller), and there is an additional estimation error of 1% due to clustering.

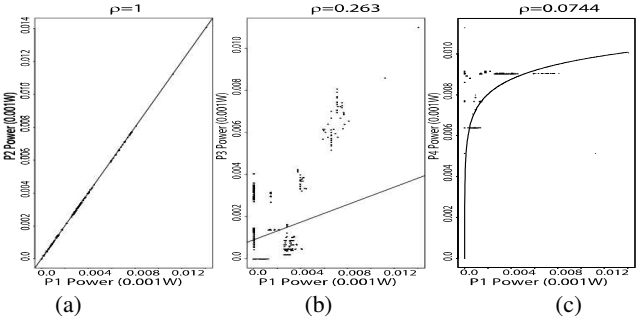
As the number of power models is decreased further, we first note that the estimation error increases sharply. This is to be expected, since the estimation error depends on the frequency with which a component is sampled for power consumption, and sampling frequency decreases as the number of components serviced by a model increases. Secondly, we observe that area requirements start increasing again. The concave nature of the area curve in Fig. 4(b) is explained by tradeoffs between multiplexer and adder area costs. Note that queues do not add area because flip-flops are paired with LUTs in the FPGA architecture [28]. Decreasing the number of power models means that each model services more components, requiring larger multiplexers which begin to outweigh the benefits of having fewer power models. Thus, we must carefully consider the conflicting trends imposed by the multiplexer and adder costs of a generic power model, while performing clustering. ∎

4.2 Exploiting Inter-Component Power Correlations

The power consumptions of several components in a design are often correlated due to the circuit topology. Correlations can be exploited to reduce the number of components being *explicitly* monitored, since the power consumption of a set of correlated components can be potentially inferred by monitoring a subset of them. For example, if P_x and P_y are power variables correlated by a function f such that $P_y = f(P_x)$, then we can monitor only component x to obtain P_x, and apply f to compute P_y.

For power emulation, since the correlation function will also be implemented in hardware, we additionally require function f to be simple, requiring very few hardware resources. For example, a linear fitting function meets these requirements. Additionally, the linear correlation must be strong and this can be captured by the statistical correlation coefficient (ρ) [29] between two power variables P_x and P_y. A high value of ρ indicates strong linear correlation.

EXAMPLE 2.: Fig. 5 plots the correlation between the power profiles of various component pairs in the Bubble_Sort design. Using a 12-to-1 multiplexer as our reference component (power variable P_1), we examine its correlation with two other 12-to-1 multiplexers (power variables P_2 and P_3), and a register that forms an input to our reference component (power variable P_4). Fig. 5(a) shows that P_1 and P_2 are perfectly correlated with $\rho = 1$ (it turns out that they are a duplication of the same component to improve timing). Fig. 5(b) shows that components P_1 and P_3 are weakly correlated with $\rho = 0.263$, while Fig. 5(c) shows that P_1 and P_4 are correlated non-linearly, but weakly correlated linearly. Thus, in this example, we monitor P_1, P_3 and P_4 and use P_1 to infer P_2. ∎



(a) (b) (c)

Figure 5: Scatter plots from the Bubble_Sort design showing (a) perfectly linearly correlated, (b) weakly linearly correlated and (c) strongly non-linearly correlated power variables

4.3 Changing Component Granularity

A power model enhanced RTL description contains power models at the granularity of basic RTL components. We can modify this policy by increasing the granularity of the components for which power models are constructed

and instantiated. In other words, we can construct a new entity comprising several RTL components, characterize this entity and use the resulting power model. Thus, by increasing the component granularity, we lower the number of power models, leading to a decrease in area. However, as shown by the following example, increasing component granularity has a significant impact on estimation accuracy.

EXAMPLE 3.: We consider the design DES that implements the popular DES encryption algorithm and contains several chains of two-input *OR* gates. In the enhanced RTL description, a power model is dedicated to each *OR* gate, but we can combine several consecutive gates in a chain to form a wide-OR entity and construct the corresponding power model. Fig. 6 plots the impact on estimation accuracy as the size of the coalesced gate increases (from 3 inputs to 11 inputs). The plot shows that the absolute error increases monotonically. This trend can be explained by the fact that when several 2-input gates are coalesced and subsumed by a large power model, the internal signals are no longer explicitly visible to the new power macromodel. This implies that it is often only practical to group small numbers of components into a single entity. ■

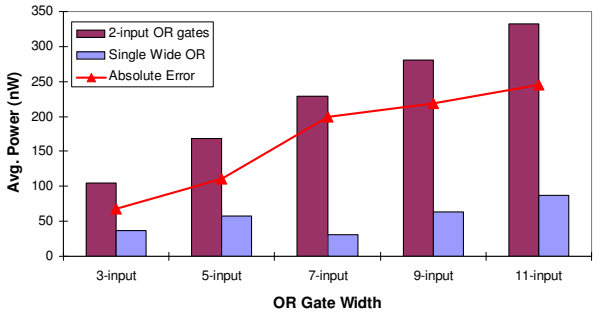


Figure 6: Power estimates from a single wide OR macromodel compared to estimates from a cascade of 2-input macromodels

4.4 Resource Sharing Within Power Models

Classical resource sharing techniques can also be employed to make the computation within each power model area-efficient, by extending the power model computation across multiple cycles, and sharing hardware across clock cycles. At the same time, the estimation error increases, since higher power model latencies translate to less frequent sampling of component inputs and outputs. Fig. 7 plots the area and estimation error for the Bubble_Sort design as a function of the number of adders allowed per power model. As expected, estimation error decreases as we increase the number of adders per power model. At the same time, area exhibits an interesting trend by descending rapidly, reaching a minimum, and then rising slowly. When the number of adders is small, the same adder is re-used in multiple cycles of a given power model computation. Consequently, large multiplexers are placed at the input of each adder to select the correct coefficient during each cycle. In fact, multiplexer overhead dominates power model area so much that there is a drastic drop in area when the number of adders increases from 1 to 2. Also, adders are area-efficient because the FPGA architecture contains dedicated carry-chain logic [28]. Thus, for a growing number of adders beyond the optimal value of 8, we see a slowly increasing area curve.

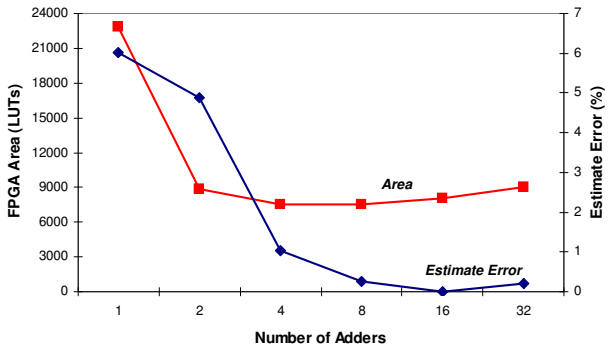


Figure 7: Area and accuracy *vs.* number of adders within each power model for the Bubble_Sort design

4.5 Using Block Memories

When clustering is applied to create a generic power model, there must be a coefficient array for each type of component supported in the cluster. The size of each array increases to match the maximum bitwidth of the generic model (to avoid extra control logic). If implemented directly in LUTs on an FPGA, the coefficient arrays are a major contributor to the area overhead. Fortunately, FPGAs provide block memories, which are ideal for storing coefficients. Xilinx's CORE Generator tool [28] offers the ability to configure a block memory macro with parameters such as width and depth. Since block RAM has at best a one cycle latency, it is essential to read multiple coefficients per cycle. This is achieved by packing coefficients into wide words and fetching the data appropriately for the power model computations.

5. ALGORITHM

Figure 8 shows the overall flow for power emulation. The inputs to this flow consist of the RTL design, the power model library, and the target FPGA platform. The power model library has been optimized for area through resource sharing (based on a predfined adder limit per power model). For a given RTL design, step **1** infers the power models needed for every component in the design and inserts the necessary estimation code to produce the enhanced RTL description. Step **2** then optimizes the power model enhanced RTL description so that it can meet a target area budget (based on the capacity of the emulation platform), while minimizing any loss in estimation accuracy. The output of this step is an RTL description ready for power emulation that can be fed to any FPGA synthesis, place and route tool flow (step **3**). Finally, the netlist can be downloaded to the FPGA and executed with the given testbench to produce the design's power profile.

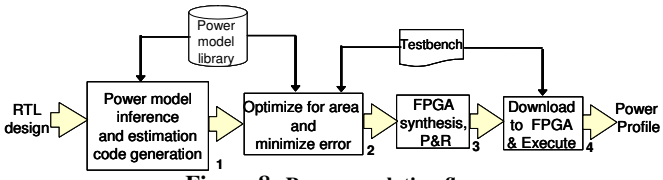


Figure 8: Power emulation flow

The key step in emulation is step **2** of Figure 8, which is described in detail in Figure 9. The methodology takes as its input the power model enhanced RTL design and its testbench, the power model library (which is automatically pre-constructed for a given resource sharing and block memory configuration) and various parameters including a target area constraint (*target_area*) and a clustering algorithm control factor (*k*). The output of the algorithm is a power emulation ready RTL description that can meet the area constraint with a minimum loss of estimation accuracy. Step **2.1** in the algorithm involves running RTL simulation for a short, user-specified interval to generate the power profiles for all the components. The power profiles are then used to generate various statistics such as (i) mean and (ii) variance of each component's power profile, and (iii) inter-component power correlation factors. These statistics are used by the area reduction techniques that follow (steps **2.3-2.8**).

Step **2.3** combines components whose power consumption statistics are strongly and linearly correlated. We utilize a user-specified threshold on the inter-component power correlation factors to identify such component sets and create a combined power model for each set. The new power model can

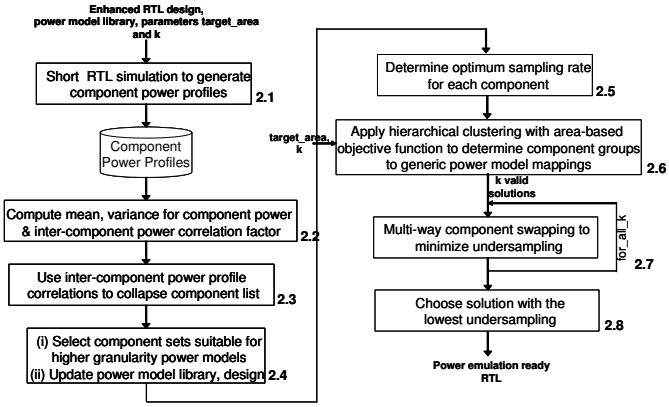


Figure 9: An accuracy-aware area optimization methodology for power emulation

estimate the power consumption for all the components by monitoring the inputs of any one of the correlated components. This reduces the number of components with explicit power models.

Step **2.4** identifies sets of components for which we can construct higher granularity power models. Since the number of such sets is exponential, based on our empirical evidence, we consider only connected components (higher potential of area savings) and small sets with upto three components (likely to have lower loss of estimation accuracy). Finally, if the fitting error for the resultant power model is higher than a user-specified limit, then the new power model is not a good choice and is not utilized.

The task now is to reduce the number of power models further by determining component clusters that can be mapped to generic power models. Steps **2.5-2.8** provide a two-phase strategy in order to meet the target area constraint with a minimum loss of accuracy. In the first phase (step **2.5**), we apply a hierarchical clustering algorithm that determines k possible clustering solutions that meet the target area constraint. In the second phase, we first compute a measure of the relative significance of each component based on the component's power mean and variance (step **2.6**). This allows us to compute a desirable sampling rate for each component. Since the area-optimized solutions of step **2.5** can result in undersampling for some components and oversampling for others, we perform classical multi-way component swapping between clusters in step **2.7** to minimize the undersampling. Step **2.8** then chooses the solution with the lowest undersampling to generate an RTL description ready for power emulation. These steps are detailed in the following sub-sections.

5.1 Hierarchical Clustering For Area Reduction

We use a hierarchical clustering algorithm [30] that takes as its input the list of components, and outputs several candidate clustering solutions that meet the specified target area constraint. Starting from an initial state wherein every component forms a distinct cluster and each cluster is associated with a power model, the algorithm proceeds as follows:

1. We evaluate pairwise the cost of combining two clusters into a single cluster. The cost is given by the size in LUTs of a generic power model that will be used to service all the components in the combined cluster. In other words, if CL_i and CL_j are two clusters, the area cost of a generic power model that services the cluster $CL_i + CL_j$ is approximately given by
$$Area(CL_i + CL_j) \approx (max(BW_{CL_i}, BW_{CL_j}) - 1) * Area_{add} + max(BW_{CL_i}, BW_{CL_j}) * Area_{mux}(|CL_i| + |CL_j|)$$
where the first term denotes the contribution due to the power model computation and the second term denotes the contribution due to the input multiplexer. BW_{CL_i} denotes the bitwidth of the largest component in cluster CL_i (with cardinality $|CL_i|$) , $Area_{add}$ denotes the size of a basic adder required to add the products of the power model coefficients and transition counts, and $Area_{mux}(n)$ is the area of a n-to-1 multiplexer.
2. We now choose the pair of clusters that can be combined to result in the best area savings ($Area(CL_i)$ + $Area(CL_j)$ - $Area(CL_i, CL_j)$) and update the bitwidth of the resultant cluster as $max(BW_{CL_i}, BW_{CL_j})$.
3. We repeat the above steps until k solutions meet the target area constraint are found or all components are in a single cluster.

5.2 Determining Optimum Component Sampling Rates

We derive the optimum sampling rates for each component based on the observation that *components whose power consumption characteristics are associated with a higher mean or variance must be sampled more frequently.* Let $comp_1, comp_2 \ldots comp_n$ denote n RTL components of a design. Assuming that we are sampling this set of components, the objective is to minimize the aggregate error due to sampling. If δP_i represents the estimated error due to sampling a component $comp_i$, then the aggregate error for the entire design is given by
$$\Delta P = \sum_{i=1}^{n} \delta P_i$$
Furthermore, during minimization, the errors associated with components with higher power should be more significant compared to other components. Therefore, we weigh the estimated error δP_i by the fractional power f_i for the corresponding component, which is computed as follows:
$$f_i = \overline{P}_{comp_i} / \sum_{i=1}^{n} \overline{P}_{comp_i}$$
Therefore, the objective function being minimized can be written as
$$Minimize\ \ \Delta P\_weighted = \sum_{i=1}^{n} f_i * \delta P_i$$

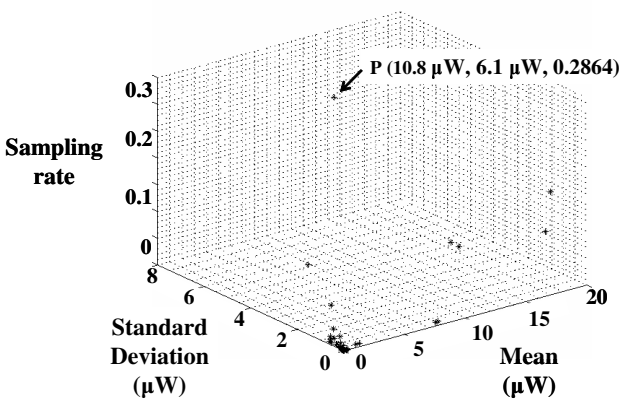


Figure 10: Sampling rates computed based on the power consumption characteristics (mean and standard deviation) of different RTL components in the design DES

For normally distributed power profiles of an RTL component $comp_i$, δP_i is governed by the following equation [24].
$$\delta_{comp_i} \approx t * s_{comp_i} / \sqrt{N_i}$$
In the above equation, s_{comp_i} refers to the standard deviation of the power profile of $comp_i$, N_i is the number of samples for the component $comp_i$ and t is a positive constant. Therefore, the objective function can be re-written as
$$Minimize\ \ \Delta P\_weighted = \sum_{i=1}^{n} f_i * s_{comp_i} / \sqrt{N_i}$$
The constraints that must be obeyed during minimization can be formulated as follows. We sample one component per sampling period from the cluster of n components for the duration of the simulation. If we denote N_{tot} to be the total number of simulation samples,
$$N_1 + \cdots + N_n \leq N_{tot},$$
$$and, N_i \geq 1, \ \ \forall i = 1 \cdots n$$
Since the above constraints are linear and the objective function is nonlinear, our minimization problem is a linearly constrained optimization problem. There are many well-known solvers such as MINOS [31] that can be used to determine the values of N_i. Once $N_1, \cdots, N_n$ are determined, the sampling rate for each component R_i can simply be written down as follows.
$$R_i = N_i / N_{tot}$$
Figure 10 shows the results of the above optimization procedure for the design DES. The design contains 1520 RTL components, and for each component, we plot the sampling rates computed based on the mean and standard deviation of the component's power consumption characteristics. For example, point P denotes the highest sampling rate of 0.2864 and corresponds to a component characterized by high mean power (10.8μW) and high standard deviation (6.1μW).

5.3 Minimizing Undersampling

Let clusters $CL_1, CL_2 \ldots CL_m$ denote a solution output by the hierarchical clustering algorithm. Assuming a uniform sampling rate for all the components in a given cluster, we can determine a measure of the estimation error introduced for a component $comp_j$ in cluster CL_i with a computation latency of τ_i by computing the distance from its optimum sampling rate (denoted by the undersampling factor δR_{ji}):
$$\delta R_{ji} = R_j - 1/(\tau_i * |CL_i|), \ \ if\ R_j > 1/(\tau_i * |CL_i|)$$
$$= 0, \ \ if\ R_j \leq 1/(\tau_i * |CL_i|)$$
where, (a) $1/(\tau_i * |CL_i|)$ denotes the uniform sampling rate for all components in a cluster CL_i with cardinality CL_i, (b) R_j is the optimum sampling rate given in Section 5.2, and (c) the undersampling is zero if the optimum component sampling rates are met by the clustering solution, *i.e.*, if $R_j \leq 1/(\tau_i * |CL_i|)$. Therefore, the aggregate undersampling for the present clustering solution is given by
$$\Delta R(CL_1, CL_2 \ldots CL_n) = \sum_{i=1}^{n} \sum_{comp_j \in CL_i} \delta R_{ji}$$
We minimize $\Delta R(CL_1, CL_2 \ldots CL_n)$ by using an iterative improvement algorithm based on the Kernighan-Lin heuristic [32] to select components and move them to other clusters in order to reduce undersampling, while ensuring that the target area constraint is not violated.

6. EXPERIMENTAL RESULTS

We present the results of applying the proposed power emulation flow to various industrial designs. Our experimental set-up is as follows. Given a

Table 2: Accuracy, area and speed comparisons for power emulation

Design	Benchmark Statistics		Execution time (*sec*)				Estimated power (*mW*)			FPGA area (*LUTs*)		
	Comp	{Gate,FU,Mux,Reg}	NEC	PT	Emu.	Speedup	RTL	Emu.	Error	Orig.	Emu.	A.O.
Bubble_Sort	777	{660,2,78,37}	11.6	80.2	1.2	{9.7X,66.8X}	0.33	0.31	6.1%	1605	5665	3.5X
HVPeakF	1526	{1364,10,101,51}	120.3	136.8	1.7	{70.8X,80.5X}	0.14	0.14	0.0%	3192	9016	2.8X
DCT	1632	{1021,5,328,278}	172.9	173.3	3.7	{46.7X,46.8X}	8.22	7.76	5.6%	6121	19242	3.1X
IDCT	819	{623,11,135,50}	130.9	108.7	2.5	{52.4X,43.5X}	0.62	0.60	3.2%	2803	9640	3.4X
Ispq	3111	{2537,16,345,213}	900	1266.3	2.6	{346.2X,487.0X}	1.24	1.21	2.4%	4282	12180	2.8X
Vld	11094	{10133,21,704,236}	62.2	227.8	2.3	{27.0X,99.0X}	1.08	1.06	1.9%	6363	18800	3.0X
MPEG4	26048	{23191,93,1944,820}	3300	2587	6.3	{524X,411X}	4.74	4.51	4.9%	24907	72351	2.9X

C behavioral description of a design, we run the behavioral synthesis tool CYBER [33] to synthesize RTL implementations. We perform RTL power estimation by using (a) PowerTheater [3], and (b) NEC's in-house power estimation tool [6]. For power emulation, we used the flow described in Section 5 to generate the power emulation ready RTL. RTL power estimates and power emulation results are obtained for NEC's CB130M [34] standard cell based technology. The power enhanced circuits were synthesized using Synplify Pro [27]. Power emulation times include the time required to simulate the testbench using the Modelsim HDL simulator, and the time to perform power emulation on a PC-based emulation platform using the Xilinx Virtex-II FPGA [28].

Table 2 summarizes the results of our experiments. Major column *Design* lists the benchmarks, including the MPEG4 and Bubble_Sort designs described earlier. We also report results for the following sub-designs of MPEG4: IDCT, Ispq, and Vld. Other designs include HVPeakF (a peaking image filter used in HDTV applications) and DCT (an implementation of the DCT signal processing algorithm). Minor column *Comp* represents the total number of RTL components in the design, while column {*Gate,FU,Mux,Reg*} provides a component breakdown in terms of the number of logic gates, functional units, multiplexers, and registers.

For the two RTL power estimation tools and the proposed power emulation technique, we report the execution times and the corresponding power estimates. These numbers are reported in major columns *Execution time* and *Estimated power*, respectively. Minor columns *NEC*, *PT* and *Emu.* represent RTL power estimation using NEC's power estimator, PowerTheater, and the proposed power emulation, respectively. Minor columns *Speedup* and *Error* denote the estimation time speedups (with respect to the two RTL power estimation tools) and percentage error due to power emulation. The last major column *FPGA area* reports the size of the original design (*Orig.*), power emulation ready RTL (*Emu.*), and the relative overhead (*A.O.*).

The results given in Table 2 indicate that power emulation achieves significant performance improvements over existing RTL power estimation tools (average speedups of 161X and 225X over NEC's RTL power estimator and PowerTheater, respectively). The cost of power emulation in terms of estimation accuracy averages 3.4%, and the power emulation ready design is, on an average, 3.1 times the area of the original design. The largest benchmark (MPEG4) shows the best performance improvement from using power emulation because its size and complexity make RTL power estimation very slow.

7. CONCLUSIONS

In this work, we discussed a new paradigm for efficient power estimation called power emulation. Power emulation harnesses hardware prototyping platforms for power estimation, leading to orders of magnitude efficiency improvements. We proposed a systematic methodology to realize power emulation in the context of a high-level design flow and a suite of area optimization techniques that makes power emulation feasible for large designs. Finally, we demonstrated the efficacy of the proposed techniques for several industrial designs.

8. REFERENCES

[1] "International Technology Roadmap for Semiconductors, 2001 Edition." http://public.itrs.net/Files/2001ITRS/Home.htm.

[2] "Galaxy Power Products, Synopsys Inc. (http://www.synopsys.com).".

[3] "PowerTheater, Sequence Design Inc. (http://www.sequencedesign.com).".

[4] "PowerChecker, BullDAST s.r.l. (http://www.bulldast.com).".

[5] "ORINOCO, ChipVision Design Systems. (http://www.chipvision.com).".

[6] S. Ravi, A. Raghunathan, and S. Chakradhar, "Efficient RTL power estimation for large designs," in *Proc. Int. Conf. VLSI Design*, Jan. 2003.

[7] J. Rabaey and M. Pedram (Editors), *Low Power Design Methodologies*. Kluwer Academic Publishers, Norwell, MA, 1996.

[8] E. Macii, M. Pedram, and F. Somenzi, "High-level power modeling, estimation, and optimization," in *Proc. Design Automation Conf.*, pp. 504–511, June 1997.

[9] A. Raghunathan, N. K. Jha, and S. Dey, *High-level Power Analysis and Optimization*. Kluwer Academic Publishers, Norwell, MA, 1998.

[10] D. Marculescu, R. Marculescu, and M. Pedram, "Information theoretic measures for energy consumption at the register-transfer level," in *Proc. Int. Symp. Low Power Design*, pp. 81–86, Apr. 1995.

[11] M. Nemani and F. Najm, "High-level area and power estimation for VLSI circuits," *IEEE Trans. Computer-Aided Design*, vol. 18, pp. 697–713, June 1999.

[12] P. Landman and J. M. Rabaey, "Architectural power analysis: The dual bit type method," *IEEE Trans. VLSI Systems*, vol. 3, pp. 173–187, June 1995.

[13] L. Benini, A. Bogliolo, M. Favalli, and G. De Micheli, "Regression models for behavioral power estimation," in *Proc. Int. Wkshp. Power & Timing Modeling, Optimization, and Simulation*, 1996.

[14] Q. Wu and Q. Qiu and M. Pedram and C. -S. Ding, "Cycle-accurate macro-models for RT-level power analysis," *IEEE Trans. VLSI Systems*, vol. 6, pp. 520–528, Dec. 1998.

[15] R. P. Llopis and F. Goossens, "The Petrol approach to high-level power estimation," in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 130–132, Aug. 1998.

[16] R. Mehra and J. Rabaey, "Behavioral level power estimation and exploration," in *Proc. Int. Wkshp. Low Power Design*, pp. 197–202, Apr. 1994.

[17] F. Ferrandi, F. Fummi, E. Macii, M. Poncina, and D. Sciuto, "Power estimation of behavioral descriptions," in *Proc. Design Automation & Test Europe (DATE) Conf.*, pp. 762–766, Feb. 1998.

[18] L. Kruse, E. Schmidt, G. jochens, A. Stammermann, A. Schulz, E. Macii, and W. Nebel, "Estimation of lower and upper bounds on the power consumption from scheduled data flow graphs," *IEEE Trans. VLSI Systems*, vol. 9, pp. 3–14, Feb. 2001.

[19] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: A first step towards software power minimization," *IEEE Trans. VLSI Systems*, vol. 2, pp. 437–445, Dec. 1994.

[20] M. Kamble and K. Ghose, "Analytical energy dissipation models for low power caches," in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 143–148, Aug. 1997.

[21] M. Chinosi, R. Zafalon, and C. Guardiani, "Automatic characterization and modeling of power consumption in static RAMs," in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 112–114, Aug. 1998.

[22] T. D. Givargis, F. Vahid, and J. Henkel, "Evaluating Power Consumption of Parameterized Cache and Bus Architectures in System-on-a-Chip Designs," *IEEE Trans. VLSI Systems*, vol. 9, pp. 500–508, Aug. 2001.

[23] K. Lahiri and A. Raghunathan, "Power Analysis of System-Level On-Chip Communication Architectures," in *Proc. Int. Conf. Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pp. 236–241, Sept. 2004.

[24] R. Burch, F. N. Najm, P. Yang, and T. Trick, "A Monte Carlo approach for power estimation," *IEEE Trans. VLSI Systems*, vol. 1, pp. 63–71, Mar. 1993.

[25] M. Chinosi, R. Zafalon, and C. Guardiani, "Parallel mixed-level power simulation based on spatiotemporal circuit partitioning," in *Proc. Design Automation Conf.*, pp. 562–567, June 1999.

[26] J. Coburn, S. Ravi, and A. Raghunathan, "Hardware Accelerated Power Estimation," in *Proc. Design Automation & Test Europe (DATE) Conf.*, pp. 528–529, Mar.

[27] "Synplify Pro, Synplicity Inc. (http://www.synplicity.com).".

[28] "Virtex-II FPGA and tools, Xilinx Inc. (http://www.xilinx.com).".

[29] G. G. Roussas, *A Course in Mathematical Statistics, Second Edition*. Academic Press, London, UK, 1997.

[30] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[31] "Using AMPL/MINOS (http://www.ampl.com/BOOKLETS/ampl-minos.pdf).".

[32] B. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *The Bell System Tech J.*, vol. 49, pp. 291–307, Feb. 1970.

[33] K. Wakabayashi, *C-Based High-Level Synthesis System, "CYBER"-Design Experience-*, vol. 41, pp. 264–268. July 2000.

[34] *CB130 Family 0.13um CMOS CBIC* http://www.necel.com/cbic/en/cb130/cb130.html. NEC Electronics, Inc., Sept. 2000.