HYPE: A Hybrid Power Estimation Method for Programmable Systems

Xun Liu Marios C. Papaefthymiou
Department of Electrical Engineering
and Computer Science
University of Michigan
Ann Arbor, Michigan 48109

Abstract

In this paper, we present a novel power estimation scheme for programmable systems consisting of predesigned datapath and memory components. The proposed hybrid methodology yields highly accurate estimates within short runtimes by combining high-level behavioral simulation with analytical macromodeling of circuit characteristics. Given a system of predesigned components along with its initial state and a program to execute, behavioral level simulation is used to derive and analyze all control signals and all data signals at the datapath/memory interface. Within the datapath itself, an iterative procedure is used in conjunction with analytical output macromodeling to compute signal statistics, as opposed to actual signals, at the inputs of its constituent logic components. These statistics are then applied to analytical power macromodels to obtain the dissipation of the datapath and the global interconnects.

Our approach is theoretically sound and yields fast and accurate estimates in practice. Using Banach's fixed point theorem, we prove that, under certain sufficient conditions on the output macromodels, the iterative procedure always converges to a unique point. We have implemented our hybrid technique into a power estimation tool called HYPE *and used it to explore various architectural alternatives in the design of a 256-state Viterbi decoder and a Rijndael encryptor. For designs with close to 1 million transistors, our estimator terminates within 10 seconds. Compared with state-of-the-art industrial gate-level power estimators, the proposed methodology is about 1,000 times faster with 5.4% error on average.*

1. Introduction

The design cost and time-to-market of electronic systems can be greatly reduced through the reuse of predesigned circuits. In this approach, components from possibly different intellectual property (IP) vendors are combined to form complete programmable systems. To support the exploration of the numerous architectural alternatives that can arise, fast and accurate EDA tools are needed for evaluating key design objectives such as timing, area, and power dissipation.

Although it is possible for IP vendors to capture timing and area by a single number, the characterization of power in a simple manner has remained elusive, since power dissipation is input data dependent [16]. Accurate circuit-level power estimation tools require detailed design simulations and therefore have unacceptably high runtimes for real-time design space exploration. On the other hand, fast high-level power tools suffer from relatively large estimation errors. Thus, efficient and accurate power estimation for IP-based systems remains a challenging task.

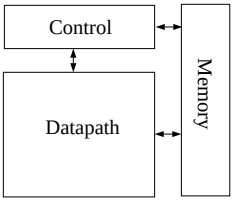

Figure 1. System Model

In this paper, a novel hybrid power estimation methodology is presented for programmable systems. To obtain fast and accurate estimates, our method combines high-level behavioral simulation with analytical macromodeling that abstracts circuit-level characteristics. At a high level, our scheme relies on a 3-part system view, as shown in Figure 1. Given a program and/or a primary input sequence, behavioral simulation of the system is performed to derive all data signals at the datapath/memory interface as well as all control signals. This simulation does not explicitly use the detailed structure of the datapath block. Based on the control signals from behavioral simulation, different possible datapath topologies are identified. For each such topology, or *mode*, an iterative procedure is applied in conjunction with analytical output macromodeling to calculate signal statistics among the various circuit components. These statistics are then applied to analytical power macromodels to obtain the power dissipation of the entire datapath. The power dissipation of global interconnects is also calculated using these statistics. For the control and memory, power estimates are obtained using the signals from the behavioral simulation.

Our estimation scheme can be used to explore architectural alternatives of general programmable systems consisting of pre-designed components. Our approach can handle systems of arbitrary topology that execute any given program. It encompasses the dissipation of memory, control, and datapath components. Unlike a lot of previous work in this area, our method is not exclusively targeted to general purpose microprocessors or specific microarchitectures.

Our power estimation procedure relies on a firm theoretical basis and yields fast and accurate estimates in practice. Using Banach's fixed point theorem, we prove the convergence and robustness of the iterative procedure that computes signal statistics in the datapath. Fast runtimes and high estimation accuracies are accomplished through the judicious combination of high-level behavioral simulation and macromodeling of circuit-level characteristics. Our simulation focuses on the interfaces among control, datapath, and memory blocks and does not consider the details of the datapath structure. Since it does not compute signals on an individual component basis, it is very fast. At the same time, it accurately captures the control signals that affect the dataflow and, therefore, utilization and power dissipation of hardware.

We have implemented our hybrid scheme in a power estimation tool called HYPE. We used our tool to explore several architectural alternatives in the design of two programmable systems. Specifically, we investigated the power impact of computation parallelism on the design of a 256-state Viterbi decoder. We also explored the power effect of loop unrolling in the implementation of a Rijndael encryptor. In both cases, full system layouts were generated from circuit components which were created using industrial standard-cell libraries and synthesis/layout tools.

Our experimental results demonstrate the high effectiveness of our approach. For designs with 100k logic gates and close to 1 million transistors, excluding memory, our estimator terminates within 10 seconds. Compared with state-of-the-art industrial gate-level power estimation tools, the proposed methodology is about 1,000 times faster with 5.4% power estimation error on the average.

The remainder of this paper has 6 sections. We discuss previous related research on power estimation in Section 2. In Section 3, we give background on analytical power and output macromodeling. Our hybrid system-level power estimation methodology is presented in Section 4. In Section 5, we prove the robustness of our signal statistics estimation procedure using Banach's fixed point theorem. Specifically, we give a sufficient condition under which our iterative estimation procedure always converges to a unique solution. In Section 6, we present two case studies, applying our approach to the design of a 256-state Viterbi decoder and a Rijndael encryptor. These experiments show the effectiveness of our estimation scheme for architectural exploration. Section 7 summarizes our paper.

2. Previous Research

A growing volume of research has been devoted to high level power modeling and estimation, since early design decisions often have a large impact on system power consumption [15, 19, 21]. Two major research directions are simulation-based cycle-accurate power estimation and power macromodeling. Cycle-accurate approaches yield fine grain information about power dissipation in each cycle. On the other hand, power macromodeling relies on signal statistics to estimate average power consumption.

A plethora of cycle-accurate power estimation frameworks has been proposed [5, 6, 13, 14, 17, 22, 23, 24, 26]. In these approaches, the power consumption of each individual RTL block is characterized with power-relevant events such as instructions or input vector pairs. Given a system architecture and a sequence of input signals/instructions, architectural level simulation is applied to count these events, and total power is computed by summing up the dissipation of all blocks. These research efforts are primarily targeted at high-end microprocessors [5, 6, 14, 26] or certain specific microarchitectures [22, 23]. Consequently, the resulting simulators tend to be domain-specific and require modification in case of significant architecture changes. In addition, the generation of cycle-accurate power models for the circuit components is challenging. The authors of [5] proposed capacitance models for common blocks in high-end microprocessors. It is unclear, however, how similar models can be applied to custom-tuned circuit designs. The work in [13, 17, 24] used instruction type to model power, without considering operand values.

Although promising results were shown for instruction sets on given microprocessors and DSP cores, estimation accuracy could widely vary on applications such as filters, where power dissipation is strongly data dependent. In [25], a cycle-accurate power model was proposed for general circuits, but it was only applied to single components.

Power macromodeling of predesigned components is a promising approach for accurate yet simulation-free power estimation. In this approach, macromodels are derived from circuit-level characterizations which capture physical details like parasitic capacitance. Each such macromodel contains a mapping between the power dissipation of a circuit and certain statistics of its input signals such as the average signal probability or average transition density [20]. Power macromodeling for single IP components has been investigated using various signal statistics and different mapping approaches in [2, 3, 7, 8, 11, 12]. Power macromodeling for pre-synthesized designs was investigated in [4, 9, 27]. The application of power macromodeling at the system level was explored in [18], where output macromodels were used to compute the signal statistics among IP blocks. That work encompassed only datapath modules, however, and did not consider control and memory components.

3. Background

3.1 Analytical power and output macromodeling

In analytical power macromodeling, a function g maps the space of input signal properties to the power dissipation of a circuit. When the input parameters of the macromodeling function are solely determined by the input signals, the computation of power estimates is a straightforward and fast function evaluation. The key challenges in analytical macromodeling are the choice of appropriate input parameters for the macromodel and the derivation of the macromodel function.

Since realistic circuits typically contain too many input bits to be characterized individually, power macromodels usually rely on input signal statistics such as the average input signal probability P_{in}, the average input transition density D_{in}, and the input spatial correlation S_{in}. To obtain the power function g of a given IP component, the component is first simulated under sample input streams with various P_{in}, D_{in}, and S_{in}. The set of power dissipation points $\mathcal{P}$ obtained by this procedure is subsequently curve-fitted to derive an analytical expression g using a minimum mean-square error criterion so that

$$\mathcal{P} = g(P_{in}, D_{in}, S_{in}) \; . \tag{1}$$

In the estimation procedure, the actual signal statistics are derived and applied to g to compute the power consumption.

Similarly, in analytical output macromodeling, functions are generated to map input signal statistics to those of the output signals. In the characterization step, functional simulations of a circuit are performed with different input sequences to obtain data points for the metrics P_{out}, D_{out}, and S_{out}. Using a minimum mean-square error criterion, analytical functions f_1, f_2, and f_3 are derived so that

$$P_{out} = f_1(P_{in}, D_{in}, S_{in}) \; , \tag{2}$$

$$D_{out} = f_2(P_{in}, D_{in}, S_{in}) \; , \tag{3}$$

$$S_{out} = f_3(P_{in}, D_{in}, S_{in}) \; . \tag{4}$$

3.2 Macromodeling based static power estimation

Static system-level power estimation based on macromodeling [18] relies on both power and output macromodel functions of each circuit component. The estimation procedure can be described using the sample system in Figure 2, where the nodes represent predesigned circuits. The signal statistics of the inter-component nodes $n1$, $n2$, and $n3$ are initialized to arbitrary values. These statistics and statistics of the primary input are then applied to the output macromodels of A, B, and C iteratively until the statistics of $n1$, $n2$, and $n3$ converge. The power consumption of the entire system is calculated by applying the signal statistics to the power macromodel functions of the corresponding components. This procedure is very fast, since it avoids time consuming cycle-by-cycle simulation.

Although this approach has been shown to be quite accurate in practice, it has a number of limitations. First, it cannot handle memory components, because memory output cannot be estimated using input statistics. Second, it cannot handle control signals efficiently. Using signal statistics as the input parameters, power macromodeling implicitly assumes that each input affects the power dissipation roughly in the same way. However, control signals such as clock gating signals, generally have much larger impact on power than data signals. Furthermore, control signals can alter the dataflow through components like multiplexers, reconfiguring system topology and, therefore, changing the power dissipation. Not distinguishing control and data signals can potentially result in large estimation error. In this paper, we propose a hybrid estimation approach which utilizes a high-level behavioral simulation to address the above limitations.

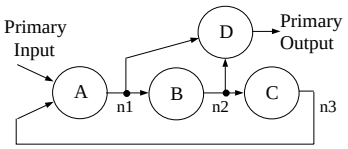

Figure 2. A System Example

4. Hybrid Power Estimation Procedure

In this section, we present our hybrid power estimation method. We first give an overview of our approach. We then describe the three phases of the procedure in more detail. Specifically, in Subsection 4.1, we discuss system partition and simulation issues. In Subsection 4.2, we describe the computation of datapath modes and corresponding active times. In Subsection 4.3, we present the power computation of the datapath, control, memory, and global interconnects.

The flow diagram of our hybrid power estimation procedure HYPE is given in Figure 3. Given a computing system consisting of predesigned components, a sequence of input signals, and a program stored in memory, HYPE reports the total power dissipation of the system when running the program.

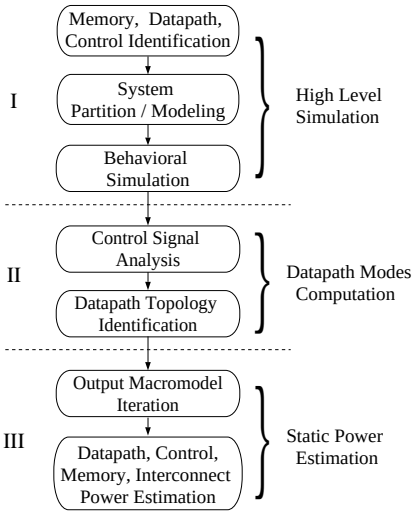

Figure 3. HYPE **Flow Diagram**

Our power estimator HYPE operates in three phases: high level simulation, datapath modes computation, and static power estimation. In the first phase, HYPE identifies the control signals, both at component and system level, and memory components, including the register files. It then partitions the system into memory, control, and datapath. Behavioral simulation is performed using the actual primary input and the program in memory to accurately compute the data signals at the datapath/memory interface and all control signals. This simulation does not require architecture details of the datapath or its constituent IP components and is therefore very fast.

After behavioral simulation, HYPE analyzes the control signal patterns and identifies all the topologies, or modes, which the datapath could operate in. The percentage of runtime for each datapath mode is computed at the same time.

In the third phase, HYPE infers the signal statistics among datapath components by iteratively evaluating output macromodel functions on each datapath topology. These statistics are then applied to power macromodels of the datapath components to obtain the dissipation. The power dissipation of control, memory, and global interconnect is also computed in the same step. The overall power consumption is finally computed by combining all the estimates of the different modes.

4.1 System partition and behavioral simulation

In this phase, the given programmable system is partitioned into three parts, control, memory, and datapath. A high-level behavioral simulation is performed to accurately compute the control signals, memory access information, and data signals at the

datapath/memory interface.

To partition the system, each of its constituent components is assigned to one of the three function blocks. The memory components including register files are easy to be identified based on their functionality. The separation of control and datapath components is a more challenging task.

Control components are chosen by identifying control signals, which can be classified into two groups: system level and component level. System level controls can reconfigure datapath topology, resulting in different dataflow and therefore signal activities. They are usually connected to the control pins of circuits like multiplexers and demultiplexers such as the *Select* signal in Figure 4.

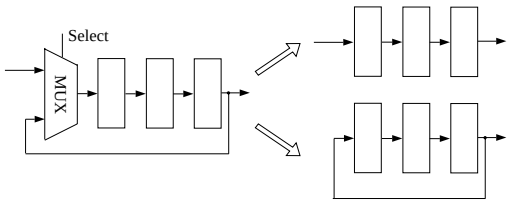

Figure 4. System Level Control Signals Can Change System Dataflow

Component level control signals can be identified from the functionality of individual circuit components. Figure 5(a) shows a dual-function ALU. The control *Select* decides the utilization of the hardware and can thus change the power dissipation significantly. For such circuits, different power macromodel functions g_i are characterized for each possible control value i. Accordingly, the total power is calculated by:

$$\mathcal{P} = \sum_{i \in M} g_i(P_{in}, D_{in}, S_{in}) \cdot p_i \ , \tag{5}$$

where M is the set of all control values and p_i is the fraction of time for which the control value is i.

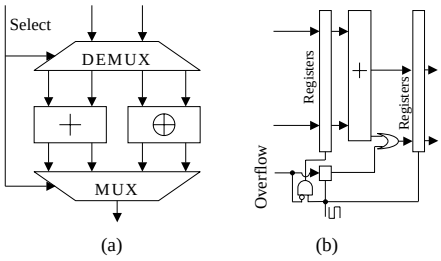

Figure 5. Component Level Control Signals

Once all control signals are identified, the control block of the system is derived. Control components generate all but data dependent controls. Figure 5(b) shows an example of data dependent control in precomputation [1], where the signal *Overflow* from some previous module can turn off the clock signal, resulting in power savings. In such cases, these control signals are modeled as parts of the datapath.

After system partitioning, behavioral descriptions are generated for each of the three parts of the given system without considering structure details. Behavioral simulation is then performed to accurately compute all control signals, memory access information, and data signals at the datapath/memory interface. This information is used in the two subsequent phases to compute the power dissipation of the system.

4.2 Datapath modes computation

After behavioral simulation, our estimation procedure performs profiling of the derived control signals. Based on the values of the system-level control signals, we record the ensuing datapath topologies or modes. These modes can activate completely different hardware and thus give rise to significant variations in power consumption. Figure 6(a) shows a system example with 4 controls. Based on the values of $S1$, $S2$, $S3$, and $S4$, the system can operate in a sequential topology with A and C given in Figure 6(b), or an iterative topology with B in the body of the loop as shown in Figure 6(c). Identifying datapath modes is crucial for high accuracy power estimation.

In addition to the datapath modes, the fraction of time during which each mode is active is also computed using the control signal analysis. These time percentages indicate the extent to which each mode can affect the overall system power.

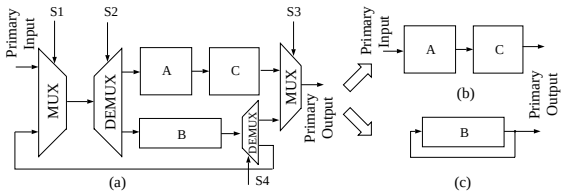

Figure 6. Different Datapath Modes

4.3 Static power estimation

In the third phase, static power estimation based on macromodeling is performed in each mode. The data signal statistics at the datapath/memory interface are first computed. Signal statistics among the circuit components are subsequently computed by iteratively evaluating output macromodel functions on each datapath topology. These signal statistics are then applied to power macromodels to obtain the dissipation results. The average datapath power consumption is finally computed by combining power estimates of all modes based on the runtime percentages.

Global interconnects among the predesigned components can consume a large amount of power. We treat interconnect as a special type of component. The power macromodel of interconnect contains one extra parameter: total capacitance. Its value can be extracted after system global routing or estimated quickly using the size information of the predesigned components. Power dissipation can be computed using the capacitance and signal statistics of the interconnect.

The power dissipation of memories primarily depends on the access rate and does not change significantly with the data stored. It is computed using memory access information from the behavioral simulation and the power related parameters from the IP vendors. The power dissipation of the control components is computed by applying the control signals from behavioral simulation to the corresponding power macromodels.

5. Condition for Unique Convergence

Theorem 1 gives a sufficient condition under which the iterative computation of the signal statistics in the static power estimation step of our method is guaranteed to converge to a unique fixed point, regardless of initialization.

Theorem 1 *Let $\mathcal{S}$ be a system of arbitrary topology, consisting of m predesigned components. For each component n, let $\vec{F}_n$ be the output macromodel functions, and $\left|\overline{\overline{F'_n}}\right|$ be the output sensitivity matrix [18]. For arbitrary initialization, the iterative estimation of signal statistics based on output macromodeling always converges to a unique solution if*

$$\left|\overline{\overline{F'_n}}\right| \cdot \vec{1}_3 \leq \delta \cdot \vec{1}_3 \ , \tag{6}$$

where δ is a real number such that $0 \leq \delta < 1$ and $\vec{1}_3$ denotes the 3-dimensional vector of 1's.

Proof. First, we recast the iterative signal statistics computation as a fixed point problem. The variables are the signal statistics of *all* the internal nodes, i.e.

$$\vec{x} = (P_1, D_1, S_1, ..., P_m, D_m, S_m) \ . \tag{7}$$

The iterative computation can be written as

$$\vec{x}_{k+1} = \vec{\mathcal{F}}(\vec{x}_k) \ , \tag{8}$$

where $\vec{\mathcal{F}}$ contains all the output macromodel functions, and k is the iteration times. It remains to show that, under the condition described in Inequality (6), Relation (8) will converge to a unique solution.

According to Banach's theorem, the iterative evaluation of Relation (8) results in a unique solution, if, for every pair of $3m$-dimensional vectors $(\vec{a}, \vec{b})$ and an infinite $3m$-dimensional vector sequence $\vec{x}_j, j = 1, 2, ...\infty$, there exists a function d which

satisfies

$$d(\vec{a},\vec{b}) = d(\vec{b},\vec{a}) \geq 0,\ with\ d(\vec{a},\vec{b}) = 0\ iff\ \vec{a} = \vec{b}, \tag{9}$$

$$\forall \vec{c},\ \ d(\vec{a},\vec{b}) + d(\vec{b},\vec{c}) \geq d(\vec{a},\vec{c}), \tag{10}$$

$$d(\vec{x}_p,\vec{x}_q) \to 0\ if\ p,q,\to\infty \tag{11}$$
$$\Rightarrow \exists\ \vec{x}:\ d(\vec{x}_j,\vec{x}) \to 0,\ j\to\infty\ ,$$

and $\vec{\mathcal{F}}$ satisfies

$$d(\vec{\mathcal{F}}(\vec{a}),\vec{\mathcal{F}}(\vec{b})) \leq \delta \cdot d(\vec{a},\vec{b})\ . \tag{12}$$

We define function d as follows:

$$d(\vec{a},\vec{b}) = \max|a_i - b_i|\ ,\ i \in \{1,2,...,3m\}\ . \tag{13}$$

It is shown in [10] that this definition satisfies Relations (9) to (11). We next prove that Inequality (12) holds.

Without loss of generality, we assume the first elements of $\vec{\mathcal{F}}(\vec{a})$ and $\vec{\mathcal{F}}(\vec{b})$ give the maximal difference $d(\vec{\mathcal{F}}(\vec{a}),\vec{\mathcal{F}}(\vec{b}))$. We denote the function computing these two elements as F and the corresponding inputs P_a, D_a, S_a, P_b, D_b, and S_b. Therefore, we have

$$\begin{aligned} d(\vec{\mathcal{F}}(\vec{a}),\vec{\mathcal{F}}(\vec{b})) &= |F(P_a,D_a,S_a) - F(P_b,D_b,S_b)| \\ &= \left|\int_{\vec{l}_a \rightsquigarrow \vec{l}_b} \nabla F d\vec{l}\right| \\ &\leq \max(|\partial F/\partial l_1| + |\partial F/\partial l_2| + |\partial F/\partial l_3|) \\ &\quad \cdot\ \max(|P_a - P_b|\ ,|D_a - D_b|\ ,|S_a - S_b|) \\ &\leq \delta \cdot \max(|P_a - P_b|\ ,|D_a - D_b|\ ,|S_a - S_b|) \\ &\leq \delta \cdot d(\vec{a},\vec{b}) \end{aligned} \tag{14}$$

where $\vec{l}_a$ and $\vec{l}_b$ are the 3-dimensional points defined by (P_a, D_a, S_a) and (P_b, D_b, S_b). □

6. Experimental Results

In this section, we present two case studies, applying our power estimator HYPE to design a 256-state Viterbi decoder and a Rijndael encryptor. Our scheme enables fast exploration of important architectural alternatives such as degrees of parallelism and loop unrolling. Our experimental results show the effectiveness of our approach. For designs close to 1 million transistors, HYPE terminates within 10 seconds. Compared with state-of-the-art industrial gate-level power tools, our method achieves a speedup of 1,000 times with only 5.4% error on average.

6.1 256-state Viterbi decoder

Viterbi decoders are widely used in portable wireless digital communication. Often running on batteries, such devices require very low power consumption. At the same time, they need to perform a large amount of computation, i.e. use maximum likelihood criterion to decode convolutional codes, in short time to meet performance objectives. In this section, we discuss the design of a Viterbi decoder for IS95 standard encoding scheme. The encoder contains 8 registers and the coding rate is 1/3. Accordingly, the decoder has to compute 256 cost values at each decoding cycle. Since all cost computations can be performed simultaneously, parallel architectures have been designed to meet performance requirements. In the following paragraphs, we describe the application of HYPE to investigate the impact of parallelism on power dissipation.

Figure 7 shows the block diagram of our Viterbi decoder. The CPU interface synchronizes the whole system. The lock control unit analyzes the datapath status and triggers the backtrace control unit, which performs survivor path tracing and reads the decoded data out. The core of the system contains n computation units (subcores) connected by address/data buses. In Figure 7, the nth subcore is enlarged to show its internal architecture. The cost calculation is performed by an addition-comparison-selection (ACS) circuit, which contains two adders, one comparator, and one 2-to-1 multiplexer. Due to the modular nature of their implementations, Viterbi decoders are prime candidates for design methodologies based on reuse.

We designed 6 different Viterbi decoders, containing 2, 4, 8, 16, 32, and 64 subcores. The circuit components were generated in the following way. Behavior description of each component was written in the hardware description language *Verilog*® and

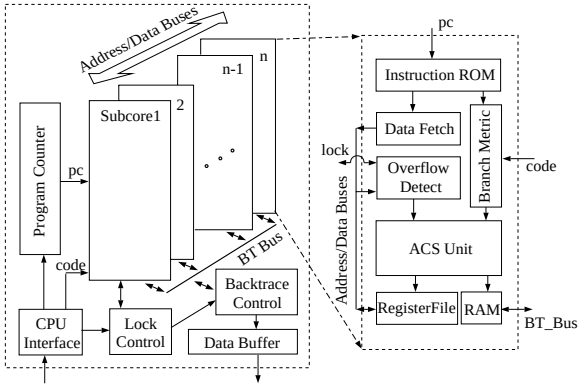

Figure 7. Viterbi Decoder

synthesized using $Design\_Analyzer^{TM}$ from *Synopsys, Inc*. A 0.25 μm standard-cell library from *Cadence* was used. The layouts of the components were generated using $Silicon\ Ensemble^{TM}$ from *Cadence* and were saved as LEF files. Output and power macromodels were computed as described in Section 3 using the methodology and tools from [18].

The overall systems were generated in a similar fashion. The *Verilog*® descriptions of the systems were synthesized using $Design\_Analyzer^{TM}$. The results were then loaded into $Silicon\ Ensemble^{TM}$, together with the layout of predesigned components. Manual placement was applied to each system to achieve minimal area. Finally, automatic global routing was used to connect all components.

We applied the following methodology to evaluate our power estimation scheme. We set the clock frequency of each design so that the decoding speed is 35 Mbps. For each estimation run, we randomly generated 2,000 binary values with probability 0.5 and encoded them using the IS95 encoder. We sent the codes through a noisy channel modeled by a fixed code error rate of 0.01. The received codes were fed into our Viterbi decoders. The programs in each subcore contained the cost computation order and the data fetch target information. They were generated by a detailed analysis of the data transfer volume between different subcores.

We applied HYPE to estimate the power dissipation of all 6 Viterbi decoders when executing our program and decoding the inputs. For each design, 40 estimation runs were performed. For comparison, we used *PrimePower*, a state-of-the-art gate-level power estimation tool from *Synopsys, Inc.*, to estimate power dissipation using the same programs and inputs.

Figure 8 shows the power estimation results of HYPE as well as those of *PrimePower*. The average estimation difference is 5.4% with a standard deviation of 0.04. HYPE finished each estimation run within 10 seconds and executed about 1,000 times faster than *PrimePower*.

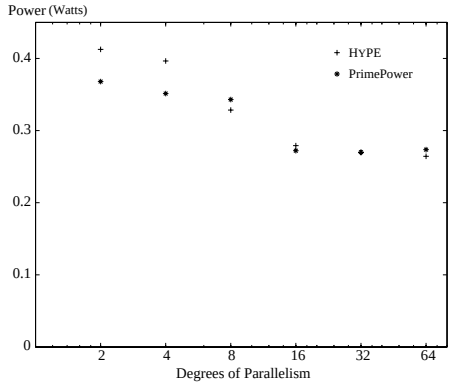

Figure 8. Viterbi Decoder Power Estimation Results

Figure 8 reveals an interesting relation between system power dissipation and the degree of computation parallelism. The highly parallel structures result in low clock frequency but high capacitance, two factors with opposite impacts on power consumption. Our results show that increasing parallelism reduces power dissipation. However, no significant reductions are achieved for degrees greater than 16 due to the significant increase in the global routing capacitance.

6.2 Rijndael encryptor

Low power encryptors/decryptors are required for mobile communication security. Such devices often perform a huge amount of computation for decipher protection and need high performance to operate in real time. The Rijndael cipher algorithm has been recently chosen as the advanced encryption standard by the U.S. National Institute of Standards and Technology. In this study, we apply our power estimation scheme to explore architectural alternatives in Rijndael encryptor design.

In the Rijndael cipher algorithm, several rounds of data manipulations are performed. In each round, very similar operations are executed. The only difference is the value of an input *key*. The hardware implementation of Rijndael encryptor often contains duplicated modules for parallel computation to increase the throughput. It is therefore well suited for IP-based design.

In our study, we analyzed the power impact of loop unrolling. Figure 9 shows two extreme cases of the Rijndael encryptor architecture. Figure 9(a) gives the loop structure. A control unit switches the multiplexer so that new input data can only be processed until the previous loop operations are finished. Multiple loop structures can be replicated to increase the system throughput. In Figure 9(b), the loop is *unrolled*, i.e. the hardware is repeated, and encryption steps are performed sequentially.

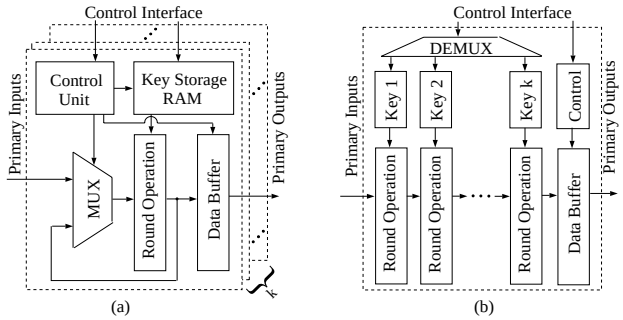

Figure 9. Rijndael Encryptor Diagrams

We applied the same procedure as in Subsection 6.1 to generate the layout and macromodels of all circuit components that were used to build two Rijndael encryptors. Specifically, for the loop-based encryptor, we duplicated 12 loop structures which operated in parallel. For the unrolled encryptor, 12 *round operation* blocks were connected in sequence. As a result, the two systems had the same throughput when clocked at the same frequency. We set both system clocks at 25MHz.

For each estimation run, we generated 1,000 input vector sequences with switching activities from 0.05 to 0.85. The granularity was 0.1. We applied our power estimator to calculate the dissipation of the two Rijndael encryptors across the entire range of input switching activities.

Figure 10 shows our results. For comparison, the estimation results from *PrimePower* are also presented. The dotted and solid lines represent the estimates of unrolled and loop-based encryptors, respectively. HYPE gives highly accurate estimates as compared with *PrimePower*. The average estimation error is 5%, with standard deviation of 0.05.

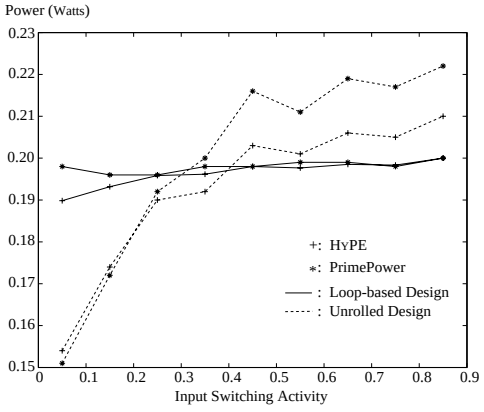

Figure 10. Rijndael Encryptor Power Estimation Results

Figure 10 shows that the power consumption of the unrolled structure is closely correlated with its input signals. The loop-based architecture, on the other hand, has very stable power dissipation. The choice of architecture thus depends on the statistics of the application signals. The unrolled architecture provides a low power solution when the input switching activity is below 0.3. However, the power performance of the loop-based design is superior when the input switching activity exceeds 0.5.

7. Conclusion

In this paper, we have presented a novel hybrid power estimation methodology for programmable systems. Our approach is applicable to general computing systems made of predesigned components including memory, control, and datapath circuits. We have provided a theoretical framework for our approach based on Banach's fixed point theorem. Specifically, we have given a sufficient condition under which our iterative signal statistics estimation procedure always converges to a unique solution regardless of initialization. Our power estimation tool, called HYPE, executes very fast and computes accurate power estimates through a balanced combination of high-level behavioral simulation and macromodeling of circuit-level characteristics. We have performed two case studies, demonstrating the application of HYPE to the exploration of different architectural alternatives in the design of a 256-state Viterbi decoder and a Rijndael encryptor. Our power estimator reveals interesting relations between power consumption and architectural characterizations such as degrees of parallelism and loop unrolling. Compared with state-of-the-art industrial gate-level power estimators, our method is about 1,000 times faster with 5.4% error on average.

References

[1] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou. Precomputation-based sequential logic optimization for low power. *IEEE Trans. VLSI Systems*, Dec. 1994.

[2] M. Barocci, L. Benini, A. Bogliolo, B. Ricco, and G. D. Micheli. Lookup table power macro-models for behavioral library components. In *Proc. IEEE Alessandro Volta Workshop on Low Power Design*, Mar. 1999.

[3] G. Bernacchia and M. Papaefthymiou. Analytical macromodeling for high-level power estimation. In *Proc. IEEE International Conference on Computer Aided Design*, Nov. 1999.

[4] A. Bogliolo, R. Corgnati, E. Macii, and M. Poncino. Parameterized RTL power models for combinational soft macros. In *Proc. IEEE International Conference on Computer Aided Design*, pages 284–287, Nov. 1999.

[5] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architecture-level power analysis and optimizations. In *Proc. 27th International Symposium on Computer Architecture*, pages 83–94, June 2000.

[6] R. Chen, M. Irwin, and R. Bajwa. An architecture level power estimator. In *Power-Driven Microarchitecture Workshop*, June 1998.

[7] Z. Chen and K. Roy. A power macromodeling technique based on power sensitivity. In *Proc. 35th Design Automation Conference*, June 1998.

[8] Z. Chen, K. Roy, and T. L. Chou. Power sensitivity–a new method to estimate power dissipation considering uncertain specifications of primary inputs. In *Proc. of IEEE International Conference on Computer Aided Design*, Nov. 1997.

[9] F. Ferrandi, F. Fummi, E. Macii, M. Poncino, and D. Sciuto. Power estimation of behavioral descriptions. In *Design, Automation, and Test in Europe*, pages 762–766, Mar. 1998.

[10] J. Franklin. *Methods of Mathematical Economics*. Sringer-Verlag New York, Inc, Jan. 1980.

[11] S. Gupta and F. N. Najm. Power macromodeling for high level power estimation. In *Proc. 34th Design Automation Conference*, June 1997.

[12] S. Gupta and F. N. Najm. Analytical model for high level power modeling of combinational and sequential circuits. In *Proc. IEEE Alessandro Volta Workshop on Low Power Design*, Mar. 1999.

[13] C. Hsieh, L. Chen, and M. Pedram. Micorprocessor power analysis by labeled simulation. In *Design, Automation, and Test in Europe*, pages 182–189, Mar. 2001.

[14] R. Joseph and M. Martonosi. Run-time power estimation in high performance microprocessors. In *Proc. 2001 International Symposium on Low Power Electronics and Design*, pages 135–140, Aug. 2001.

[15] P. Landman. High level power estimation. In *Proc. International Symposium on Low Power Electronics and Design*, Aug. 1996.

[16] P. Landman and J. M. Rabaey. Activity-sensitive architectural power analysis. *IEEE Trans. CAD*, June 1996.

[17] M. T. Lee, V. Tiwari, S. Malik, and M. Fujita. Power analysis and minimization techniques for embedded DSP software. *IEEE Trans. VLSI Systems*, pages 123–135, Mar. 1997.

[18] X. Liu and M. C. Papaefthymiou. A static power estimation methodology for IP-based design. In *Design, Automation, and Test in Europe*, pages 280–287, Mar. 2001.

[19] E. Macii, M. Pedram, and F. Somenzi. High-Level power modeling, estimation, and optimization. *IEEE Trans. CAD*, pages 1061–1079, Nov. 1998.

[20] F. N. Najm. Transition density: A stochastic measure of activity in digital circuits. In *Proc. 28th Design Automation Conference*, June 1991.

[21] F. N. Najm. A survey of power estimation techniques in VLSI circuits. *IEEE Trans. VLSI Systems*, 2(4):446–455, Dec. 1994.

[22] T. Šimunić, L. Benini, and G. D. Micheli. Cycle-accurate simulation of energy consumption in embedded systems. In *Proc. 36th ACM/IEEE Design Automation Conference*, pages 867–872, June 1999.

[23] P. Stanley-Marbell and M. S. Hsiao. Fast, flexible, cycle-accurate energy estimation. In *Proc. 2001 International Symposium on Low Power Electronics and Design*, pages 141–146, Aug. 2001.

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

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

[26] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin. The design and use of simplepower: A cycle-accurate energy estimation tool. In *Proc. 37th ACM/IEEE Design Automation Conference*, pages 340–345, June 2000.

[27] R. Zafalon, M. Rossello, E. Macii, and M. Poncino. Power macromodeling for a high quality RT-level power estimation. In *Proc. International Symposium on Quality Electronic Design*, pages 59–63, 2000.