Early Power Exploration - A World Wide Web Application

David Lidsky, Jan M. Rabaey
University of California, Berkeley

Abstract: ***Exploration at the earliest stages of the design process is an integral component of effective low-power design. Nevertheless, superficial high-level analyses with insufficient accuracy are routinely performed. Critical drawbacks of current high-level design aids include a limited scope of application, inaccuracy of estimation, inaccessibility, and steep learning curves. This paper introduces an approach to alleviate these limitations, thus enabling more effective high-level design exploration. A World Wide Web (WWW) based prototype tool called PowerPlay, which encapsulates and enhances these techniques, is presented.***

Introduction

In the earliest stages of the design process, power, area, and timing estimates are usually effected through first-order manual analyses performed on flow chart representations and/or simplified hardware sketches. These analyses involve the extraction of some critical information, which may include cursory estimates of operation and hardware counts, critical path length, and the manipulation of power, area, and timing models for each hardware module. For example, a high-level power analysis may be obtained by multiplying the number of hardware accesses with the estimated power dissipation per access.

Unfortunately, the accuracy of these estimates is likely to be far lower than desired. Models may be nonexistent, difficult to access, or imprecise. Simulation is usually too time consuming to justify for any but the most critical components and is only feasible after the completion of a considerable portion of the design process. Furthermore, because this process is so tedious and time consuming, alternative implementations may often be dismissed without a scrupulous examination of the potential performance enhancements they may provide.

Effective power exploration at the early phases of the design process requires the following enablers to be present:

1. a well-documented and characterized library of modules,
2. a set of modeling techniques, which makes it easy to characterize and introduce new modules into the library,
3. a spread-sheet-like work sheet, which presents the design-under-exploration and allows the study of the impact of parameter variations (such as supply voltage and clock frequency),
4. a simple and universally available access mechanism.

This paper describes a framework, called PowerPlay, that addresses these issues. While existing exploration aids often require compilable algorithm descriptions which may not exist in the early phases of design, the proposed approach performs high-level estimation based solely upon the manipulation of power, timing, and area models of functional blocks. Each model is parameterized (for example, by bit-width) and is scalable with supply voltage and technology. Existing hardware models are shared among all users, and new models are easily created and integrated.

Effective high-level performance estimation consists primarily of data gathering and management, and routine spreadsheet calculations. As the World Wide Web (WWW) has become the de facto standard for information gathering, it is the most natural choice for a design exploration environment. Besides the relatively seamless data gathering and push-button tool launching capabilities (through hyperlinks — textual pointers to files or programs) that a WWW browser provides, the WWW is publicly accessible and in widespread use, encouraging shared library models and design re-use. Furthermore, because each user can access the tool with her/his favorite browser, there is little overhead involved in learning how to interface with the tool.

Since traditionally, timing and area analysis has received a lot of attention, this paper focuses on power estimation techniques. The remainder of the paper is organized as follows: The following section illustrates high-level design exploration through a vector quantization decoder example. A survey of modeling techniques and a detailed description of several of PowerPlay's models is then provided. A system-level power analysis of a portable multimedia terminal demonstrates the system modeling capabilities of PowerPlay. Details about the implementation of PowerPlay are given in the final section.

Design Example: Video Decompression

This section illustrates the methodology for high-level power estimation and exploration. Two alternative designs of the luminance sub-component of a custom real-time video decompression chip [4],[8] are explored and compared.

The vector-quantization decompression scheme (Figure 1) decodes an 8-bit input into 16 6-bit words, each of which represent the luminance of a video pixel. The decompression is accomplished using a memory look-up table (LUT), where the 8-bit input specifies the address of a 16-word block of luminance values. Incoming data are buffered using a ping-pong memory access scheme. Figure 1 shows that the current frame's data is being stored in memory Bank 0, while the previous frame's data is being read from Bank 1. With each new video frame, the read/write roles of the memory banks are reversed.

The system has a 256 x 128 pixel video screen which requires updates at a minimum rate of 60 frames/second. Since the incoming video arrives at 30 frames/second, each read buffer is decompressed and displayed twice. In other words, a buffer is read twice as often as it is written. These requirements set the minimum frequency, *f*, at which pixels are sent to the screen to 2 MHz, and the read and write buffer access rates to *f/16* and *f/32*, respectively [4].

An estimate of the power consumption of this proposed archi-

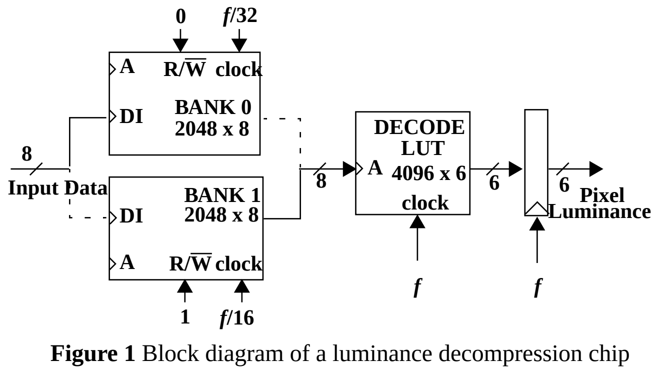
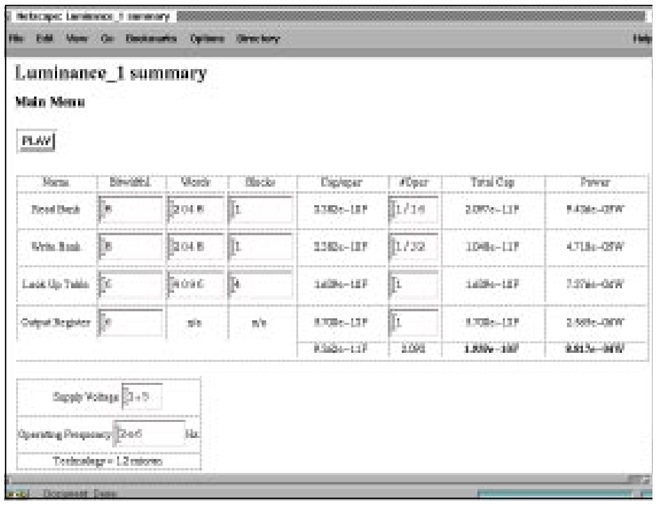


Figure 1 Block diagram of a luminance decompression chip



Netscape: Luminance_1 summary

File Edit View Go Bookmarks Options Directory Help

Luminance_1 summary

Main Menu

PLAY

Name	Bitwidth	Words	Blocks	Cap/oper	#Oper	Total Cap	Power
Read Bank	8	2048	1	3.382e-12F	1/16	2.097e-13F	8.43[illegible]e-05W
Write Bank	8	2048	1	3.382e-12F	1/32	1.048e-13F	4.718e-05W
Look Up Table	6	4096	4	1.628e-12F	1	1.628e-12F	7.37[illegible]e-05W
Output Register	6	n/a	n/a	8.708e-13F	1	8.708e-13F	2.58[illegible]e-05W
				9.5[illegible]e-12F	2.0[illegible]	1.8[illegible]e-12F	9.815e-05W

Supply Voltage	1.5
Operating Frequency	2e6 Hz
Technology = 1.2 microns	

Document: Done

Figure 2 PowerPlay's spreadsheet power analysis

tecture was produced as follows: Hardware modules were selected from a library of pre-characterized components, and were customized by defining the model parameters, such as bit-width, memory-block organization, and signal-correlation characteristics. PowerPlay multiplied the resulting energy/operation by the estimated number of accesses of each resource (or the activity). Based on this information, it generated a spreadsheet that contains the power estimates per module as well as total power consumption. The table is parameterized; that is, parameters such as bit-widths and supply voltages can be varied dynamically. The whole process, including the selection of the library elements and the composition of the architecture, was executed through a standard WWW browser, Netscape, in less than three minutes. No other tool interfaces are needed.

Note that the clock capacitance is included in the model of each block. In this example, signal correlations are neglected, yielding a conservatively high power estimate. Note however, that the lack of interconnect analysis in this example neglects some switching capacitance. A later section details how these factors can be introduced into the estimation. The important point is that one should be able to decide which components are important and provide estimates as accurately as possible. In this way, the estimate can take seconds to perform and account for everything (or more than) that would normally be taken into consideration in manual analysis.

This estimation strategy enables a quick comparison of alternative design choices. Figure 3 shows another implementation of the same algorithm, which exploits the locality-of-reference of vector quantization by addressing groups of four words. In this implementation, each memory access yields four times as many bits as in the previous implementation. The question is whether the overhead of the larger memory accesses and extra multiplexors outweighs savings earned by reducing the total number of accesses. In this implementation, only one multiplexor and register are switching at the

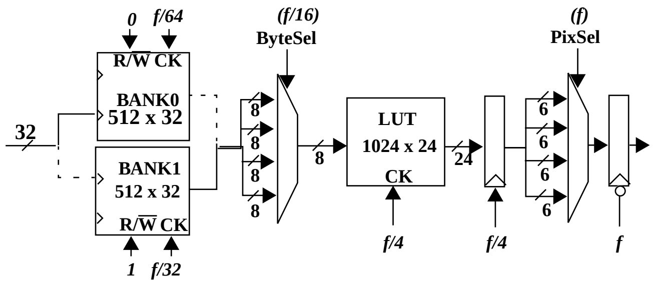


Figure 3 Alternate implementation of the decompression chip

full 2 MHz. The memories switch more capacitance per access, but at a fraction of the frequency of the initial implementation.

At this level of abstraction, accuracy should be within an octave of the actual value. This enables power budgeting at an early stage and gives a good basis for making architectural and algorithmic decisions. In this example PowerPlay estimated the power dissipation of the second implementation (Figure 3) to be ~150 µW, or 1/5 that of the original design (Figure 1). The final implementation of the chip used this second architecture and had a measured average power dissipation of 100 µW [4]. The models that enable such quick and relatively accurate estimates are presented in more detail in the following section.

Models

Providing good estimates of power, area, and/or speed, requires accurate, parameterized models of the components of the chip, such as computational blocks, controllers and interconnect. The strength of a modeling environment lies in the richness of its library, the availability of pre-defined models, and the ease of introducing new elements and models. Models for each element in the University of California's low-power cell library are provided [1], but more importantly, PowerPlay enables easy integration of any user-defined model. Due to its flexibility in model introduction and analysis, the framework can be used for any class of digital, analog or mixed-mode components at any abstraction level. This section discusses PowerPlay's power model template and provides examples which illustrate how different classes of primitive functional blocks are mapped to this template.

PowerPlay Model Template. Electronic power dissipation can be most generally described by the sum of static and dynamic components,

$$P = \sum_i C_{sw,\,i} V_{swing,\,i} V_{DD} f + I V_{DD} \qquad \text{(EQ 1)}$$

where $C_{sw,i}$ is the average capacitance at node i, switching over a voltage range $V_{swing,i}$, at a frequency f. I is the static current drawn from voltage supply V_{DD}, used to model leakage, bias currents, or any other static component. In digital complementary CMOS, nodes generally have rail-to-rail swings; in this case, $V_{swing,i}$ is equal to V_{DD}.

Landman [12] uses empirical analysis to provide a "black box model" — no knowledge of the internal workings is required — of the capacitance switched in a digital hardware module. This approach accounts for glitching and does not require complex analysis of library blocks. Svensson [18] models switching capacitance analytically without requiring extensive simulations. The input and output capacitance of each stage in a functional block are calculated and the switching activity at the input and output of each stage is determined a function of the input. A stage is a single PMOS pull-up / NMOS pull-down configuration. A bit slice may have one or more of these stages.

PowerPlay allows any block to be modeled using any combination of $C_{sw,i}$, $V_{swing,i}$, and I as a function of any input parameters to give maximum flexibility. In turn, larger systems may be modeled using compositions of these blocks. (This is discussed in more detail in a subsequent section.)

The following subsections illustrate how various classes of electronic components are modeled using PowerPlay's template of (EQ 1). Most digital circuit models can be divided into four elements: computation, storage, controllers, and interconnect. The specific models in this section are for static CMOS logic, and bipolar analog; however, this general modeling technique is valid for nearly any electronic circuit.

Computational blocks. The computation elements perform the numerical computations required by an algorithm. Landman's approach characterizes each library cell with capacitance coefficients which relate the complexity of a library element (e.g. bit-width, memory size, etc.) to capacitance switching over a voltage range V_{DD}. For example, a ripple adder has a single coefficient relating the inputs' bit-width to the total capacitance switched,

$$C_T = \sum_{i}^{bitwidth} \alpha_i C_i \qquad \text{(EQ 2)}$$

where α_i is the activity/bit and C_i is the average capacitance switched per bit. Assuming a constant activity per bit, $\alpha = \text{constant}$, each bit switches the same capacitance, $C_o = \alpha C_i$, and

$$C_T = bitwidth \bullet C_0 \qquad \text{(EQ 3)}$$

More complex modules (e.g. multipliers or logarithmic shifters) require additional capacitive coefficients.

The capacitance of a unit can also be derived analytically if the layout of the block is known. Svensson's model for a single pull-up/pull-down stage of logic can be generalized to:

$$C_S = \alpha_{in} C_{in} + \alpha_{out} C_{out} \qquad \text{(EQ 4)}$$

where α_{in} is the probability of an input transition, α_{out} is the probability of an output transition, and C_{in} and C_{out} are the physical input and output capacitance, respectively. The total capacitance per bit-slice is found by summing the capacitance switched in each stage:

$$C_{ST} = \sum_{j}^{stages} \alpha_{in,\,j} C_{in,\,j} + \alpha_{out,\,j} C_{out,\,j} \qquad \text{(EQ 5)}$$

where C_{ST} *is* the summation of the capacitance in each stage of logic in a bit-slice. Assuming random activity, the total effective capacitance switched in the entire block, C_T, is:

$$C_T = Bitwidth \bullet C_{ST} \qquad \text{(EQ 6)}$$

Storage. Small memories, such as pipeline registers or register files, can use the same modeling strategy as that used for computational elements. In larger memories (e.g. SRAM, DRAM), the modeling becomes more intricate due to the complex architectures typically used in these structures. The switching capacitance of the SRAM module in the U.C. Berkeley library, for instance, is a function of the word-width (*bits*) and the number of words (*words*), but contains a constant factor as well. This suggests the following model:

$$C_T = C_0 + C_1(words) + C_1(bits) + C_2(words)(bits) \qquad \text{(EQ 7)}$$

Using this model for memories with reduced bit-line swings may prove to be inaccurate (as a function of voltage). Landman's model calculates power by finding the effective capacitive coefficients (defined at one voltage level) and then simply multiplying by V_{DD}^2 to find the energy per operation. When the capacitance is not switching rail-to-rail, however, the dynamic power is more accurately modeled by:

$$P = \alpha \left\{ C_{fullswing} V_{DD}^2 + C_{partialswing} V_{swing} V_{DD} \right\} f \qquad \text{(EQ 8)}$$

This equation fits PowerPlay's template model (EQ 1). Therefore in modeling memories (or any logic with reduced swing) it is important to characterize them at more than one voltage level to extract $C_{partialswing}$ and V_{swing}. If short-circuit currents are non-negligible, charge dissipated due to direct-path power consumption needs to be characterized as well. The direct path charge from V_{DD} can be modeled as an effective capacitance and voltage swing and fits into (EQ 1).

Using an analytical method for calculating the capacitance switched in each stage of logic may become cumbersome for complex memories. However, short-circuit currents and reduced swing switching can be evaluated analytically. Power due to direct path current may be analytically quantified if the input rise and fall times, τ, are known [20]. V_{swing} and $C_{partialswing}$ of (EQ 8) may also be determined analytically to compute power dissipation on reduced swing bit-lines.

Controllers. Controller power estimation is particularly difficult in the rudimentary stages of design. Given a control algorithm, the specific combinational logic implementation platform (e.g. PLA,ROM, random logic) may be unknown, and the size and complexity of the controller may not be fully characterized. Deriving this information directly from an algorithmic description is non-trivial, and often requires feedback from the designer.

As examples, power dissipation models of two implementation platforms, random logic-based and ROM-based controllers, are described below. The two important parameters, N_I (the number of inputs, including state and status bits), and N_O (the number of outputs, including state bits and status signals), can often be accurately estimated at an early stage.

A **random logic** controller consists of two or more levels of boolean gates. The average capacitance switched in the controller is modeled as the sum of two terms which represent the input and output logic planes [12]:

$$C_T = C_0 \alpha_0 N_I N_O + C_1 \alpha_1 N_M N_O \qquad \text{(EQ 9)}$$

Here, C_0 and C_1 are library-specific coefficients, α_0 and α_1 are switching probabilities, and N_M is the number of minterms (which, in turn, is related to the complexity of the controller). Switching probabilities are difficult to determine without knowledge of input signal correlations, but for quick estimates may be assumed to be a randomly distributed set of input vectors, $\alpha_0 = \alpha_1 = 0.25$.

In a **ROM**-based controller, N_I address bits are decoded, enabling one of 2^{NI} word lines. N_O sense amplifiers amplify the output signals on N_O bit-lines to full-rail. If the word and bit-lines are precharged high before each access, energy is consumed while precharging only those bit lines that evaluated low in the previous cycle. Thus, an appropriate switching probability, P_O, is the average fraction of low output bits. The average capacitance switched in the ROM is modeled by [12]:

$$C_T = C_0 + C_1 N_I 2^{N_I} + C_2 P_O N_O 2^{N_I} + C_3 P_O N_O + C_4 N_0 \qquad \text{(EQ 10)}$$

where C_0, C_1, C_2, C_3, and C_4 are library-specific coefficients. Other implementation platforms (e.g. PLAs) may be modeled in a similar way.

Because there is a high level of abstraction involved with these models, their results should be interpreted with caution. Once the control path is more carefully characterized, many of the parameters in (EQ13) and (EQ14) can be more accurately estimated (using other tools through PowerPlay's hyperlinks), ultimately providing a more accurate power dissipation estimate.

Interconnect. Unlike the activity of computational blocks, the amount of interconnect activity is not inherent to an algorithm. Therefore, determining the amount of capacitance switched in interconnect at an early stage of design is a challenging problem. In the earliest stages of design, the best one can do is a quick estimate. Donath [6] and Feuer [7] propose methods of estimating total interconnect area from the amount of active area using Rent's rule [11], which relates block count in a region to the number of external connections to the region (area estimates of the modules are easily provided). Once the physical interconnect area is determined, capacitance on the line can be parameterized by feature size and capacitance per unit area.

As the user gets further along in the design process, architectural estimators may be used to improve accuracy. As the design process is iterated, these values should be back-annotated to the design to give more accurate results.

Programmable Processors. The first-order model of power dissipation of a programmable processor is derived from the average power consumption, P_{AVG}, as supplied by data books or measured through experimentation. The total power due to a specific process is:

$$P = \alpha P_{AVG} \qquad \text{(EQ 11)}$$

where $\alpha(\leq 1)$ is the activity factor of the processor. This model does not take into consideration the types of computations in an algorithm. It assumes that the processor is either consuming P_{AVG} when active or nothing during shutdown. A processor with no power-down capability has an activity factor of one.

To get a more accurate model of a processor requires modeling each instruction in terms of energy per instruction, E_{inst}. Total energy dissipation for a particular algorithm is the summation of the energy dissipated in each instruction [19]:

$$E_T = \sum_i N_i E_{inst,i} \qquad \text{(EQ 12)}$$

where $E_{inst,i}$ is the energy for instruction i, and N_i is the total number of instructions i in the algorithm. Power is this total energy divided by the time to process the algorithm. Ong and Yan have used this methodology on a fictitious processor to determine that there can be orders of magnitude variance in power consumption for different sorting algorithms [15].

These models tend to underestimate power because factors such as cache and branch misses are neglected. More detailed information can be obtained by using a coded algorithm and profilers (e.g. SPIX, Pixie) and cache simulators(e.g. Dinero).

Analog Integrated Circuits. The power dissipation of most analog circuits is dominated by static bias currents rather than the dynamic charging of capacitance. Estimating the power of analog circuits, therefore, requires the summation of the bias currents and multiplying *linearly* with supply voltage:

$$P_{ANALOG} = V_{supply} \sum_i I_{Bias,i} \qquad \text{(EQ 13)}$$

For op-amp circuits, information about input/output impedances and circuit gain can be parameterized in terms of bias currents. For example, in a bipolar emitter-coupled transconductance amplifier, the transconductance, G_m and the input and output impedance, R_{id}, R_o are defined by [9]:

$$G_m = g_m = \left(\frac{q}{kT}\right)\frac{I_{bias}}{2} \qquad \text{(EQ 14)}$$

$$R_{id} = 2r_\pi = 2\frac{\beta_0}{g_m} = \left(\frac{4kT\beta_0}{q}\right)\frac{1}{I_{bias}} \qquad \text{(EQ 15)}$$

$$R_o \approx \frac{r_o}{2} = \frac{V_A}{I_{bias}} \qquad \text{(EQ 16)}$$

Since the bias current can be defined by characteristics of the amplifier, this differential pair may be parametrized by G_m, R_{id}, and/or R_o, much like a digital adder is parameterized by bit-width. The power consumption of this amplifier, for example, can be parameterized by transconductance:

$$P_{ANALOG} = V_{supply} I_{bias} = 2V_{supply}\left(\frac{kT}{q}\right)G_m \qquad \text{(EQ 17)}$$

DC-DC Converters. A DC-DC converter may be specified by the power it supplies to its load, P_{Load}, and its conversion efficiency, η [14]:

$$\eta \equiv \frac{P_{Load}}{P_{in}} = \frac{P_{Load}}{P_{Load} + P_{diss}} \qquad \text{(EQ 18)}$$

The efficiency of the converter is a function of temperature, input voltage, and load power for varying loads, but in many applications, it can be assumed constant to the first order. Under this assumption, the average power dissipation of the converter is

$$P_{diss} = P_{Load} \times \frac{1-\eta}{\eta} \qquad \text{(EQ 19)}$$

This is an example of intermodel interaction; the output from other models is used to calculate the dissipation in the converter.

Example Model: PowerPlay currently contains a library characterized using Landman's empirical approach. The multiplier in UC Berkeley's low-power library, for example, is modeled as a capacitance switching rail-to-rail. This capacitance is parameterized by a coefficient multiplied by the bit-widths of the inputs:

$$C_T = bitwidthA \bullet bitwidthB \bullet 253\,fF \qquad \text{(EQ 20)}$$

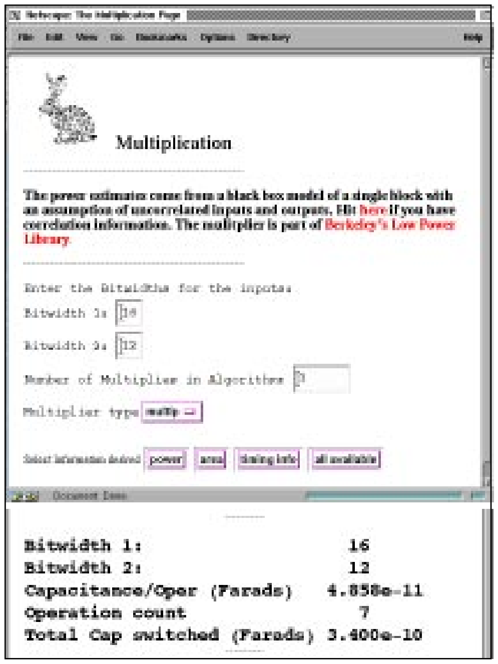
The capacitive coefficient, 253fF, is for non-correlated inputs. PowerPlay also contains models for correlated inputs which has the same format of equation but with different coefficients. Figure 4 shows the interface for calculating the capacitance dissipated in a computation unit. The user has the option on the input form of setting bit-widths and multiplier type. The feedback is virtually instantaneous, so the user may cycle through many options. When satisfied, the user saves the results to a design space spread-sheet (e.g. Figure 2).

PowerPlay will accept **any** model and in fact will support paths to estimation tools in lieu of an equation. The goal is to provide a framework to allow the most accurate estimation possible given the current state of a design.

System Design

One of the main pitfalls in low-power design is that a great deal of effort is concentrated on a part of the system that consumes only a small percentage of the power budget. When working on power-minimization, it is important to identify both the major power consumers and the point of diminishing returns. To do so requires modeling not only of integrated circuits, but also of components such as FPGAs, memory sub-systems, embedded processors, displays and servos. Systems are mixed-mode (digital, analog, electro-mechanical) and may use a range of frequencies and supply voltages.

Power analysis of complex systems is only possible when good models are available for each of the components. Due to its flexible



Multiplication

The power estimates come from a black box model of a single block with an assumption of uncorrelated inputs and outputs. Hit here if you have correlation information. The multiplier is part of Berkeley's Low Power Library.

Enter the Bitwidths for the inputs:

Bitwidth 1: 16

Bitwidth 2: 12

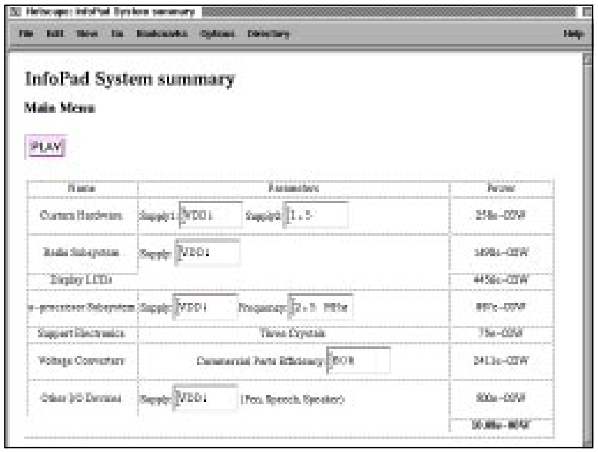
Number of Multiplier in Algorithm: 7

Multiplier type: multip

Select information desired: power | area | timing info | all available

```
Bitwidth 1:                      16
Bitwidth 2:                      12
Capacitance/Oper (Farads)   4.858e-11
Operation count                   7
Total Cap switched (Farads) 3.400e-10
```

Figure 4 Multiplier input form and result excerpt



InfoPad System summary

Main Menu

PLAY

Name	Parameters	Power
Custom Hardware	Supply1: VDD1 Supply2: 1.5	258e-03W
Radio Subsystem	Supply: VDD1	1698e-03W
Display LCDs		4458e-03W
u-processor Subsystem	Supply: VDD1 Frequency: 2.5 MHz	897e-03W
Support Electronics	Three Crystals	75e-03W
Voltage Converters	Commercial Parts Efficiency: 80%	2413e-03W
Other I/O Devices	Supply: VDD1 (Pen, Speech, Speaker)	800e-03W
		10.88e+00W

Figure 5 InfoPad system power breakdown

modeling approach, the proposed framework is not restricted to integrated circuits but can be used to provide power estimations for other elements as well. The power information for commodity components is, for instance, readily available from data-sheets. On the other hand, providing high-level macro-models for other elements, such as FPGAs, is non-trivial and is the subject of further research.

Crucial to the success of the proposed exploration technique at the system level is the availability of hierarchical macro-modeling. It should be possible to lump a modeled design, such as the video-decompression sub-system described earlier, into a single macro, that can be used at higher levels of the system design, or re-used in other designs. This capability is easily provided in the spread-sheet-like environment of PowerPlay, which allows for the introduction of variables at any level in the design hierarchy and where any parameter can be expressed as a function of these parameters. For instance, the parameters for a video-compression module are the bit-width, the supply voltage and the pixel rate. Even more intricate usages of system parameters are possible: the dissipation of a DC-DC converter can be expressed as a function of the power dissipation of connecting modules, the power dissipation of interconnect is a function of the active area of the design (and thus of its composing modules).

To illustrate the usefulness of such a scheme, we use the example of a portable multi-media terminal called InfoPad [16]. We demonstrate how the use of hierarchy and parameterized modeling allows for the analysis of the power break-down of such a complex system.

Each subsystem of the InfoPad terminal is a row entry in the spreadsheet of Figure 5. The user has the option of selecting different models from the spreadsheet and recomputing power. The power consumption in each section of the figure may be computed independently, using whatever models, tools, or level of abstraction is available at the time. For example, the power dissipation data for the LCDs came from actual measurements, the data for the custom hardware is modeled for one configuration and measured for another, and the data from the voltage converters is measured in one case and results of hand estimates of an optimal convertor in another. Figure 5 shows the measured power breakdown for the entire InfoPad system. Each of the subsystems listed in the first column are hyperlinked to the power breakdown of that unit. For example, the luminance chip discussed earlier is a subcircuit of the custom hardware subsection. By clicking on the subsystem name, the custom hardware spreadsheet is called.

The user interacts with the entire design through the design spreadsheet. All subcircuit parameters are given in the second column of the spreadsheet so the user can change any parameter from the top page. When the Play button is pressed power is calculated for the entire design and the spreadsheet is updated. Each component of the design has a specified model or tool path regardless of implementation platform. PowerPlay calculates the specified power for each component and sums the results. (Compositional techniques for delay estimation are currently being examined.) Hyperlinks from the spreadsheet to the various subcircuits enable the user to access and optimize the entire design space from one location, as well as facilitate full interactive documentation. The following section details the implementation of PowerPlay, including said hyperlink capabilities.

World Wide Web

The WWW enables seamless access to data and software on file systems throughout the world. This property has made the WWW the de facto standard for information gathering and distribution. PowerPlay utilizes the WWW to provide library re-use, integrated documentation and universal access.

Libraries of primitives (e.g. multipliers, memories) as well as macro cells (e.g. video decompression) may be shared and reused. If a library is characterized and put on the web in Massachusetts, it can be used for estimates in California. Likewise, macros built from said libraries are also automatically made available for re-use unless specified as proprietary.

PowerPlay incorporates hyperlinks, which are textural pointers to scripts or files, to encourage and automate integrated documentation. For example, from the spreadsheet playground, every subcircuit or primitive instantiation has links to relevant documentation. When new primitives or macros are put into PowerPlay, simple input windows are presented to encourage documentation. PowerPlay then automatically generates appropriate documentation links whenever the primitive/macro is used. Documentation, therefore, is appended to the design process virtually effortlessly. Hyperlinks are also provided to PowerPlay's tutorials and help pages.

PowerPlay's accessibility by any browser has two large benefits. Firstly, the learning curve for the tool is reduced since users can use their favorite web browser. More importantly, since PowerPlay is local to one server, it can be accessed by any machine on the web. There is no need to port, recompile and install the tool. PowerPlay is accessible at anytime from any machine.

The network architecture is depicted in Figure 6 showing how a user at MIT can simultaneously access tools and models from remote sites (e.g. Motorola, and the server site, Berkeley). At the time of publication, access of models across the network has been demonstrated on a small scale using a modification of the protocols described by Silva [17]. The top of Figure 7 illustrates Silva's method which uses the mail protocol SMTP and relies on hubs on each machine to interpret requests for information. This method is modified for WWW using the HyperText Transfer Protocol (bottom of Figure 7). The key is using secure scripts at Universal Resource Locators to handle information transfer on demand. Models which

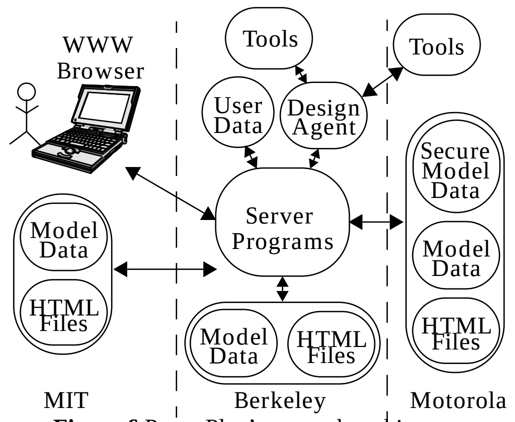


Figure 6 PowerPlay's network architecture

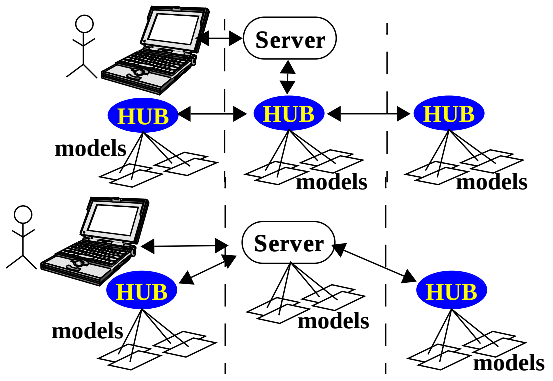


Figure 7 Model access across the network

require tool invocations are implemented through a dynamic design-flow manager called the *Design Agent* [1], which translates the hyperlink request for data into a sequence of appropriate tool invocations determined by the chosen design context.

When designing across the internet, protection of proprietary information is a concern. Proprietary designs can be protected in a number of ways. PowerPlay can provide password-restricted access plus WWW programs enable file access to be restricted to specific machines. For full security, a private version of PowerPlay may be run within a company's firewalls.

PowerPlay Implementation

PowerPlay is implemented using a combination of WWW pages and custom Perl scripts. A WWW page is written in HyperText Markup Language (HTML). HTML pages enable hyperlinks to other pages and calls to programs located on the WWW. A program can be written in any language. Perl was chosen due its strong file manipulation capabilities. Output from the program is printed to the user's WWW browser.

User identification is necessary to ensure privacy and to enable PowerPlay to track each individual's designs and preferences. Since WWW browsers do not supply user names, when PowerPlay is initially accessed the user must identify her/himself. The username is passed to a Perl script which retrieves the individual user's defaults from the PowerPlay server's local file system. These user defaults include the relevant hardware libraries and any previously generated designs. The script then generates an HTML formatted menu page and sends it to the browser. From the menu page the user may select library elements. Library elements are primitives (e.g. multipliers, memory units) and/or groups of primitives called subsystems (e.g. video chip set, multiply-accumulate macros). The selected primitive/subsystem and the username is passed to a script that uses the relevant user default parameters along with information about the library element, to produce a primitive/subcircuit input page (top of Figure 4). The user may then alter parameters (e.g. bitwidths, gain). A Perl script updates the user defaults and calculates variables relating to power, area and/or timing. The user may then choose to repeat this process or add the results to the design spreadsheet.

From the design spreadsheet, the user may alter parameters of subcircuits and recalculate power, area and/or timing numbers. Subcircuits may be defined to inherit global parameters. When the Play button is hit, the entire design is passed to a Perl script. This script calculates the power for each subcircuit hierarchically (through specified models or tools) using the parameters that are passed from the top level. There is no fundamental limit to the levels of hierarchy.

PowerPlay also provides a simple method for users to define models for their own primitives using an interactive HTML page. The user is prompted for names, equations, and documentation information. The new model is incorporated into PowerPlay with links to appropriate pages, and an entry page with integrated documentation and hyperlinks similar to Figure 4.

Summary

A methodology, which combines precharacterized and user-defined models to provide quick and "as accurate as possible" power estimation in a spreadsheet-like format at the earliest stages of a design has been presented. Though not detailed in this paper, parameterized models are also used for area and timing analysis. The estimation strategy has been incorporated into a prototype tool that leverages off the features of the World Wide Web. PowerPlay, which is under continuing development, is available at http://infopad.eecs.berkeley.edu/PowerPlay.

References

[1] O. Bentz *et al*,"Information-based Design Environment", *IEEE VLSI Signal Processing VIII*, pp. 237-246, Nov. 1995.
[2] T. Burd, *Low-Power CMOS Library Design Methodology*, Masters Thesis, Engineering Researh Laboratory, UC Berkeley, CA, June 1994.
[3] F. Catthoor *et al*, "Global Communication and Memory Optimizing Transformations for Low-Power Signal Processing Systems," *IEEE VLSI Signal Processing*, VII, pp. 178-187, 1994
[4] A. Chandrakasan, *et al*, "A Low-Power Chipset for Portable Multimedia Applications," *ISSCC*, March 1994.
[5] A. Chandrakasan, *et al*, "Low Power Digital CMOS Design," Kluwer Academic Publishers, Boston, MA, 1995.
[6] W. Donath, "Placement and Average Interconnection Lengths of Computer Logic," *IEEE Transactions on Circuits and Systems*, Vol. CAS-26, No. 4, pp. 272-277, April 1979.
[7] M. Feuer, "Connectivity of Random Logic," *IEEE Transactions on Computing*, vol, C-31, pp. 29-33, Jan. 1982
[8] A. Gersho, *et al*, "Vector quantization and Signal Compression," Kluwer Academic Publishers, Boston, MA, 1992.
[9] P. Gray, *et al*, "Analysis and Design of Analog Integrated Circuits," John Wiley and Sons, Inc., New York, NY, 1993.
[10] K. Keutzer, *et al*, "The Impact of CAD on the Design of Low-Power Digital Circuits," *IEEE Symposium on Low Power Electonics -Digest of Technical Papers*, pp. 342-46, October, 1994
[11] B. Landman *et al*, "On a Pin Versus Block Relationship for Partitions of Logic Graphs," *IEEE Transactions on Computing*, vol. C-20, pp. 1469-1479, Dec. 1971.
[12] P. Landman, *Low-Power Architectural Design Methodologies*, Ph. D. Thesis, Electronic Research Laboratory, University of California, Berkeley, August 1994.
[13] D. Lidsky *et al*, "Low-Power Design of Memory Intensive Functions", *IEEE VLSI Signal Processing*, VII, pp. 378-387, 1994
[14] Maxim, "1994 New Releases Data Book Volume III," Section 4, 1994
[15] P. Ong *et al*, "Power-Conscious Software Design - a framework for modeling software on hardware," *IEEE Symposium on Low Power Electonics -Digest of Technical Papers*, pp. 36-37, October, 1994
[16] S. Sheng *et al*, "A Portable Multimedia Terminal," *IEEE Communications Magazine*, pp. 64-75, December 1992.
[17] M. Silva *et al*, "The Case for Design Using the World Wide Web," *Proceeding of the Design Automation Conference*, pp. 579-585, June 1995.
[18] C. Svensson *et al*, "Low Power Circuit Techniques," from "Low Power Design Methodologies", pp.37-64, Kluwer Academic Publishers, Boston, MA, 1996.
[19] V. Tiwari *et al*, "Power Analysis of Embedded Software," *IEEE Transactions on VLSI Systems*, pp 437-445, December 1994.
[20] H. Veendrick, "Short-Circuit Dissipation of Static CMOS Circuitry and It's Impact on the Design of Buffer Circuits," *IEEE JSSC*, vol.sc-19, pp. 468-473, August 1994.