

# A Formal Framework for the Analysis of Dynamic Power Management Policies

Shrirang M. Yardi and Michael S. Hsiao ([yardi,mhsiao}@vt.edu](mailto:{yardi,mhsiao}@vt.edu))

The Bradley Department of Electrical and Computer Engineering, Virginia Tech

The primary concern in today's portable systems is to manage system-wide power consumption and not just that of the processor subsystem. Due to decreasing feature size and aggressive power reduction techniques, the processor may not be the dominant power consumer in the system. Further, processors such as Intel™ XScale® and IBM™ PowerPC405LP® have the ability to scale both voltage and frequency with a very small latency and without interruption of the executing tasks. Due to the support of such dynamic voltage and frequency (DVFS) capabilities, designers can now think about new, highly aggressive, fine-grained dynamic power management (DPM) policies. One such architecture, called *policy-based DPM*, to support such policies has been proposed in [1].

Policy-based DPM is based on the observation that the most effective way to manage energy consumption is highly dependent on the design as well as the applications running on an embedded system. Hence, fixed DPM policies are likely to provide only limited energy savings. On the other hand, a flexible power management architecture is much better suited to support different platforms and “pluggable” DPM policies that are specific to the device and/or the application. Policy-based DPM enables this by defining policies as named data structures within the OS kernel that can be accessed through an abstract API. This API allows the specification of the component and device-state transitions according to a DPM policy. A policy provides a mapping from the *operating state* of the system to a particular physical parameter setting of the components termed as the *operating point*. Multiple DPM policies can be supported by using a *policy manager* which is an executable program that implements a user-level/application-specific DPM strategy [1].

Policy-based DPM requires that a complete DPM strategy be defined in advance for each application. The DPM strategy can be effective only if a thorough analysis during the design stage can be made as to the behavior of the complete system in the context of this strategy. However, to assess the impact of the designed strategy on the energy consumption and system performance, the designer has to account for all possible interactions of other user-level/application-specific policies, device constraints, system tasks and execution environments. Simulation will only be able to cover a small sub-set of this search space and may result in a sub-standard implementation of the DPM strategy. Further, such time-consuming simulation may have to be repeated to tune the design.

To enable the designer to effectively perform such exhaustive analysis in a formal manner, we propose a state-machine based modeling approach that combines the functional and power behavior, and the interaction between the two, for the complete system. The model consists of two layers which represent two different abstractions - functional and power. Figure 1(a) provides an overview of our modeling strategy.

The first layer, termed as the *Operation Layer (OL)* abstracts the functional behavior of the system as concurrent Extended Finite State Machines (EFSMs). Each state in the OL represents a functional operation and a state transition represents one step in the execution sequence of the modeled function. A state transition can take place only if a control condition associated with it is satisfied. The control condition is related to the availability of a requested resource that is modeled in the

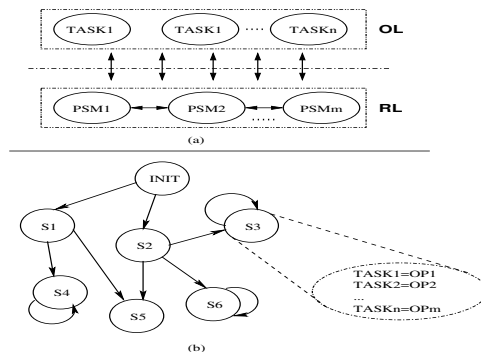


Figure 1. (a) Overview of the Model, (b) System Product FSM

second layer. All the EFSMs operate concurrently at every control step but they cannot communicate with each other. Due to such an abstraction, the designer is free to model the EFSMs in any amount of detail as required. For example, the EFSMs can represent a user-level application or OS tasks or even ISAs of the cores used in an SoC or processor(s) in the underlying hardware.

The second layer, termed as the *Resource Layer (RL)* models the power behavior of the power-managed system components as concurrent Power State Machines (PSMs). Each PSM state models a particular power mode of the component. An enumeration of all the current power states gives the current operating point. State transitions in the RL are controlled by the current active DPM policy. The PSMs can communicate with each other through output signals which can be used to model the interaction and constraints between multiple components.

The *power manageability (PM)* of such a system can then be calculated using formal reachability analysis on the *Product FSM* of the complete system. The product FSM is obtained by a synchronous composition of the individual FSMs in the OL and the RL. Figure 1(b) shows an example of the product FSM. Each state of the product FSM represents the *operating state* of the entire system. The PM is computed by performing implicit state enumeration using symbolic simulation and calculating the power for each state in the product FSM. The minimum, maximum and average power values thus calculated can be used to compare candidate DPM strategy implementations. The power values can be viewed as properties for each state. Since the entire state space is enumerated, all environmental inputs that drive the system functionality are implicitly considered. Hence the calculated value will hold true for any environment that the system is operated in.

We are currently in the process of implementing a prototype model using the SMV language for a PowerPC405LP® based system. The model includes the processor, a display, a HDD and a system bus as the power-managed components. The OL is modeled as tasks executing on the system that are scheduled by the OS using a simple round-robin scheduler. The RL is modeled as PSMs of each component with *Active*, *Idle* and *Sleep* power modes.

[1] IBM and MontaVista Software, “Dynamic Power Management for Embedded Systems”, <http://www.research.ibm.com/arl/projects/dpm.html>.