

Modeling of DRAM Power Control Policies Using Deterministic and Stochastic Petri Nets

Xiaobo Fan, Carla S. Ellis, Alvin R. Lebeck

Department of Computer Science, Duke University, Durham, NC 27708, USA
{xiaobo, carla, alvy}@cs.duke.edu

Abstract. Modern DRAM technologies offer power management features for optimization between performance and energy consumption. This paper employs Petri nets to model and evaluate memory controller policies for manipulating multiple power states. The model has been validated against the analysis and simulation used in our previous work. We extend it to model more complex policies and our results show that DRAM chip should always immediately transition to *standby* and never transition to *powerdown* provided that it exhibits typical exponential access behavior.

Keywords: Control Policy, DRAM, Memory Controller, Modeling, Petri Nets

1 Introduction

Energy efficiency is becoming increasingly important in system design. It is desirable both from the point of view of battery life in mobile devices, and from environmental and economical points of view in all computing platforms. With the introduction of low power processors, novel displays, and systems without hard disks, main memory is consuming a growing proportion of the system power budget. Modern DRAM technologies are making memory chips with multiple power states to offer power management capability. Usually there is an *Active* state to service requests and several degraded states (*Standby*, *Nap* and *Powerdown*) with decreasing power consumption but increasing time to transition back to *Active*. We must design a power control policy to manipulate these states effectively to improve energy efficiency without sacrificing too much performance.

Our work to date has adhered relatively closely to the specifications of RDRAM [4], thus giving our new power aware memory management ideas the credibility of being based on actual hardware. However, our experience with this one design point suggests that alternatives to the current set of RDRAM power states may offer better management possibilities. There is a large space of potential DRAM power states and associated memory controller policies to explore. Identifying the most productive design points is valuable to influence future power aware DRAM development.

Since the design space is so large, it is impractical to use the detailed simulations. Therefore, we need models to allow rapid exploration of the space. As part

of our previous work we investigated memory controller policies for manipulating DRAM power states in cache-based systems [3]. We developed an analytic model that approximates the idle time of DRAM chips using an exponential distribution, and validated our model against trace-driven simulations. Our results show that, for our benchmarks, the simple policy of immediately transitioning a DRAM chip to a lower power state when it becomes idle is superior to more sophisticated policies that try to predict DRAM chip idle time [2].

Although this model served our purpose, it is not easily extended to more power states. Therefore, we propose developing a Petri net model of DRAM power states. The first step is to create a model that mimics our previous analytic model for validation. Then we can extend it to more power states, finally using it to explore the space. We use the Petri net toolkit, DSPNexpress [6], which supports graphical development and performance analysis of deterministic and stochastic Petri nets.

In this paper, we first establish a DSPN (Deterministic and Stochastic Petri Nets) model for the 2-state control policy we used in [3]. We use DSPNexpress [6] to solve the model. In [3] the analysis is done by assuming exponentially distributed *gap* — the idle time between clustered accesses, while here the model is driven by an exponentially distributed inter-access time. By measuring *gap* from the DSPN model, we verify that it is equal to the exponentially distributed inter-access time. From the DSPNexpress solution, we can derive other results of our interest. Using the same DRAM configuration, all results are consistent with those in [3]. Then we extend this model to 4 states and explore other threshold-based policies. Based on the available DRAM configuration, we have the following conclusions for typical exponential memory access patterns: 1) a simple *Active* to *Nap* immediate transition policy is the best 2-state policy if the average *gap* is large enough, 2) chips should always transition to *Standby* if it is available, 3) chips should not transition to *Powerdown* during execution.

The rest of this paper is organized as follows. In the next section, we introduce how our control policy works on a multi-state memory chip and develop a DSPN model for the 2-state power control policy. Section 3 discusses how to derive performance and energy data from DSPN solutions and validates these results against probability based analysis. In Section 4, we extend the model to 4 states and investigate the effect of two other thresholds on memory performance and energy consumption. Finally, Section 5 concludes and describes future work.

2 Methodology

Figure 1 is an example illustrating how a 2-state control policy works. In this case, we use *Active* and *Nap* power states. When a memory chip has any outstanding access, it stays in the *Active* state. After the last access completes and before the start of the next access, the memory chip is idle, and we denote this interval as the *gap*. If the *gap* is greater than a threshold value, the chip transitions down to the low power state, *Nap*, otherwise it remains in *Active*. This is a typical state/event dynamic system which can be modeled by Petri

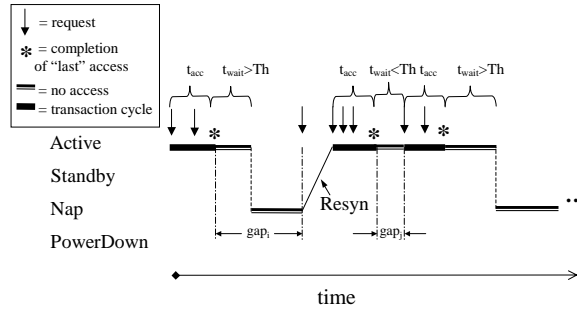


Fig. 1. Power Management

net. Each state of a memory chip (i.e. *Active*, *Nap*, etc.) can be mapped to a marking in the Petri net model. Each state change or transition (i.e. power degradation, resynchronization, etc.) can be described by a Petri net transition. By associating timing information with these transitions, we can model system performance. Furthermore, knowing performance results and power consumption of each state and transition, we can evaluate the system’s energy efficiency. Due to space limitations, we skip the background knowledge about Petri nets and Stochastic Petri nets. Detailed information can be found in [10], [7], [8], [1] and [9].

Based on this 2-state control policy, we can develop a Petri net model, as shown in Figure 2. There are three situations a memory chip could be in when a request arrives: 1) in the *Active* state and currently servicing other requests, 2) in the *Active* state and idle-waiting, or 3) in the *Nap* state.

In the first case, it only takes a small amount of additional time to service this request because most DRAM technologies offer “bursting” optimizations and we assume requests are serviced by independent internal DRAM banks. In the Petri net model, we use the place labelled *active* to represent this state and a deterministic transition *service* with delay, *serviceDelay*, to represent a service of the request. In the second case, represented by the place labelled *idle*, this initial access incurs delay to initiate the access. We use the transition *activate* and its delay *activateDelay* to depict the initial access cost. In the last case, a resynchronization cost is incurred to transition out of the low power state, *nap*, denoted by transition *resync2* with *resync2Delay* as its cost.

After completing the last access, the memory chip goes to *idle* through the immediate transition *rest*, and further into *nap* through a timed transition *sleep* with delay equal to the waiting threshold, *napTh*. Both these two degrading transitions can be disabled or interrupted by an inhibitor arc from place *buffer*, where outstanding requests are buffered. It means that when there is an outstanding request, if the chip is in *active*, it stays in *active* servicing the request; if it is in *idle* and waiting for the threshold, the timer is canceled and the transition *activate* will fire after *activateDelay*.

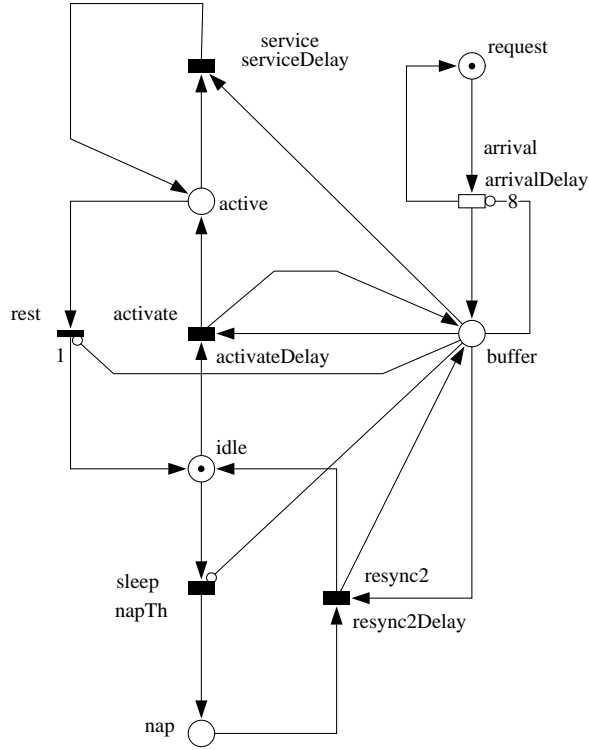


Fig. 2. DSPN model for 2 state power control policy

Based on analysis from our previous studies, we use an exponentially distributed memory access pattern to drive this power management model. The transition *arrival* is an exponential stochastic transition with mean firing delay *arrivalDelay*. Since we assume all requests come from a cache hierarchy with 8 outstanding misses, there is an inhibitor arc with multiplicity 8 from *buffer* to *arrival* to model this aspect. We allocate one token in place *request* to generate accesses and one token in the set of places $\{active, idle, nap\}$ to simulate power states. Table 1 are the parameter values for the multi-state DRAM which we use to do all the following computation. Transitions *resync1* and *resync3* are going to be used later in the 4-state model.

3 Validation

After running the DSPNexpress steady-state solution, we obtain the throughput for all deterministic and stochastic transitions, and average token number for all places. We use X_T to denote the throughput of transition T and N_P to denote the average token number of place P . Then we can derive other performance

Power State	Power (mW)	Time (nS)
Active	$P_a = 300$	$serviceDelay=60$
Standby	$P_s = 180$	-
Nap	$P_n = 30$	-
Powerdown	$P_p = 3$	-
Transition		
activate	$P_a = 300$	$activateDelay = 60$
resync1	$P_{s \rightarrow a} = 240$	$resync1Delay = 6$
resync2	$P_{n \rightarrow a} = 165$	$resync2Delay = 60$
resync3	$P_{p \rightarrow a} = 152$	$resync3Delay = 6000$

Table 1. DRAM Power State and Transition Values

and power consumption values. Since we define *gap* as the time interval between two clustered accesses, we want to determine the period of each *gap* cycle and the average *gap*. Then, for each gap cycle, we can compute how much time is spent in each state and how much energy is consumed.

Because each firing of transition *activate* terminates the current *gap* cycle and creates a new *gap* cycle, the mean period of the *gap* cycle is

$$T_{total} = \frac{1}{X_{activate}}$$

Then we can compute the time spent on each place (all the following computations are for one *gap* cycle with mean period T_{total})

$$\begin{aligned} T_{active} &= T_{total} N_{active} \\ T_{idle} &= T_{total} N_{idle} \\ T_{nap} &= T_{total} N_{nap} \end{aligned}$$

Since the time elapsed in place *nap* also includes time spent on transition *resync2* where the power consumption is different from that in the *nap* state, we need to determine how much time is spent on resynchronization:

$$T_{resync2} = T_{total} \cdot X_{resync2} \cdot resync2Delay$$

Therefore the power consumption is (assuming P_a , P_n and $P_{n \rightarrow a}$ are power consumptions for the *active* power state, *nap* power state and resynchronization, respectively)

$$E_{total} = P_a(T_{active} + T_{idle}) + P_n(T_{nap} - T_{resync2}) + P_{n \rightarrow a}T_{resync2}$$

Finally we have the per gap energy•delay product and the relative product (compared to the always-active policy)

$$\begin{aligned} e \bullet d &= E_{total}T_{total} \\ \Delta(e \bullet d) &= E_{total}T_{total} - P_a(T_{total} - T_{resync2})^2 \end{aligned}$$

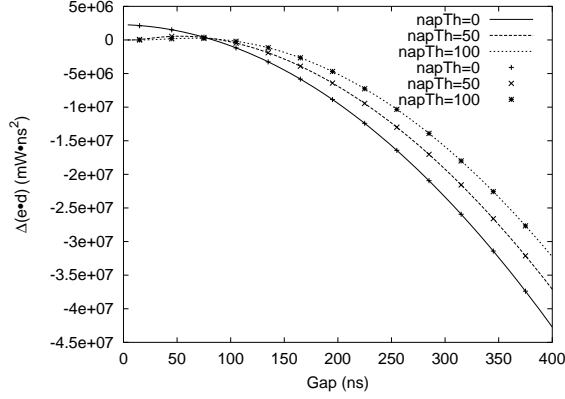


Fig. 3. $\Delta(e \bullet d)$ computed from analysis and modeling (lower is better)

Before solving the model and comparing it with the analytical results in [3], there is one more problem we need to deal with. The memory access pattern we use to drive this model, as shown in Figure 2, follows exponential distribution on the inter-access-arrival time instead of the inter-clustered-access idle time (*gap*), which we used to develop the analytical model in [3]. In theory we can prove that if inter-access-arrival time follows exponential distribution, *gap* should follow the same distribution. As we can see from Figure 1, *gap* is the interval between the completion of the last access and the arrival of next access. Because of the memoryless property of exponential distribution, the time lapsed from the arrival to the completion of the last access doesn't affect the probability of the next access's arrival. Therefore *gap* follows the same distribution as inter-arrival time. To verify this, we can also compute the average gap μ from the DSPN model using one of the following equations:

$$\begin{aligned} \mu &= T_{total} - T_{active} - T_{resync2} - activateDelay \\ \mu &= T_{idle} - activateDelay + T_{nap} - T_{resync2} \end{aligned}$$

In fact each *gap* value computed from above equation is equal to the corresponding *arrivalDelay* value used in the model.

Using the same DRAM parameters as in our previous analysis [3], we run DSPNexpress to obtain results for this DSPN model. These results are shown in Figure 3, as the points, together with the solid lines derived from our previous analysis. As we can see, they exactly match, validating the DSPN based modeling and probability based analysis against each other. Recall, our probability based analysis was validated against simulations [3].

Because the DSPN model is much easier to develop and extend than mathematical analysis, we propose to use it to investigate more complex policies and to explore a larger parameter space. In particular, we plan to explore the impact

of: 1) changing the threshold values for transitioning to lower power states, 2) changing the power consumed during each DRAM power state, 3) changing the power consumed and delay incurred to transition between power states, and 4) changing the number of available power states. This paper only covers the first case.

4 Model Extension

In the previous section, we validated the DSPN model against an analytical model, which is validated in [3] against simulation. In this section, we will extend the 2-state model to 4 states and investigate how the other two thresholds affect energy efficiency.

Figure 4 is the DSPN model for a 4-state power control policy. We add two more low power states – *standby* and *powerdown*. When the memory chip is in *idle* for time *standbyTh*, it first transitions down to *standby*. Then it transitions down to *nap* if it stays in *standby* for *napTh*, and further down to *powerdown* if it stays in *nap* for *powerdownTh*. If a memory access arrives during any of these downward transitions, the transition is disabled by one of the three inhibitor arcs. Then the relevant upward resynchronization and/or activation is fired. This is consistent with the real hardware.

Since we already know the immediate *active* to *nap* transition is the best 2-state policy when average *gap* is greater than 75ns and no *nap* transition should be made when average *gap* is less than 75ns, we want to know what the appropriate *standby* threshold should be in both of the two cases. Therefore, for memory access patterns with average *gap* greater than 75ns, we use 0 as *napTh* and, for those with average *gap* less than 75ns, we use infinity. In order to avoid the effect of the *powerdown* transition, we set *powerdownTh* to infinity. Then we observe how the energy•delay product changes as we vary *standbyTh*.

Figure 5 shows the relative energy•delay product for different *standbyTh* values when *gap* increases from 15ns to 375ns. Unlike the *nap*-based transition, zero threshold is always the best even when the *gap* is very small. Therefore even with a very high cache miss rate the memory chip should always transition down to *standby* right after the completion of outstanding accesses. For contrast only when the cache hierarchy generates high enough hit rate so that the average *gap* is large enough (e.g. 75ns) should the memory chip transition down to *nap* immediately. This justifies the fact that *standby* is the default power state for Rambus DRAM when the chip is idle.

Knowing *standbyTh* should always be 0 and *napTh* should be 0 when *gap* is greater than 75ns, we next want to know what *powerdownTh* should be for our exponential memory access pattern. *Powerdown* is an extreme state in that memory chip consumes very low power (3mW) and it incurs huge delay to get out of the state (6000ns).

We run the model with *gap* starting from 75ns, because this is the boundary beyond which transitioning down to *nap* starts giving benefit and *powerdown* becomes reachable. Figure 6 shows the relative energy•delay product for differ-

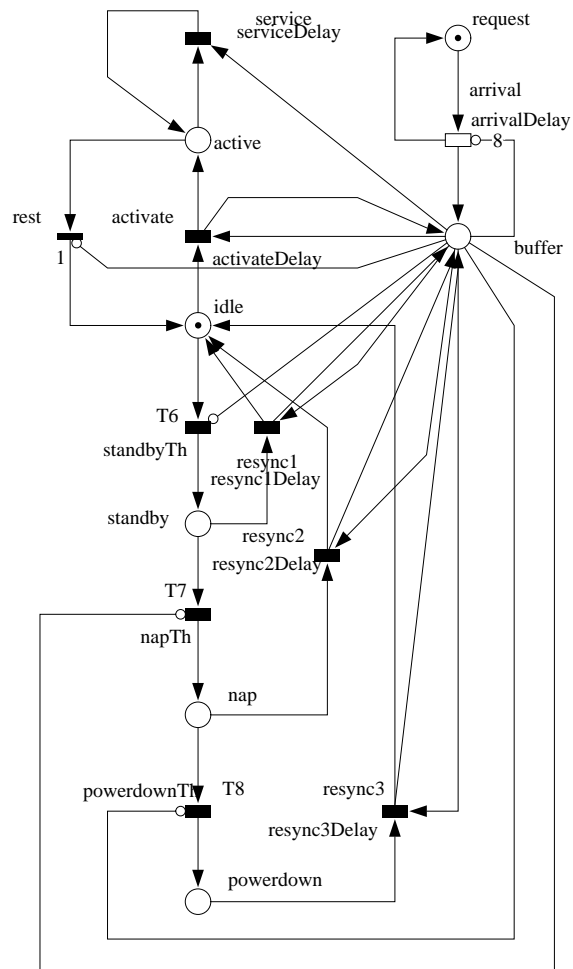


Fig. 4. DSPN model for 4-state power control policy

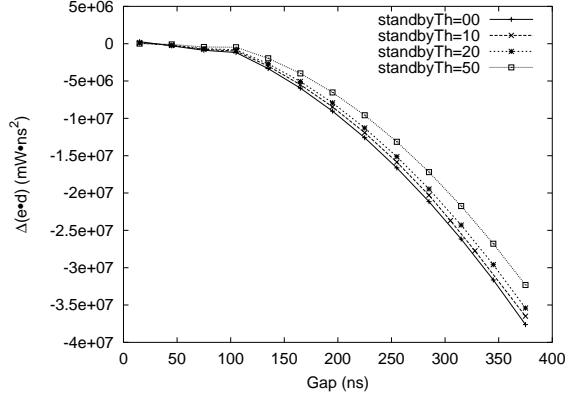


Fig. 5. $\Delta(e \bullet d)$ computed from 4-state model for different *gap* and *standbyTh* values

ent *powerdownTh* values. We didn't include results for *powerdownTh* values less than 500ns because they are out of the graph range, making other parts undistinguishable.

When *powerdownTh* is small (less than 2000ns), the chip could more easily get to the *powerdown* state and incur very long resynchronization delay when an access comes. Therefore it performs much worse than the always-*Active* policy. The larger the average *gap*, the bigger the probability that it goes to *powerdown* and the more penalty. When *powerdownTh* is large enough (greater than 5000ns), the probability of getting into *powerdown* becomes so small that the policy is similar to the immediate *nap* transition. The larger the *powerdownTh*, the smaller the probability and the better the policy performs. Therefore when the memory access pattern imposed on one memory chip exhibits exponential or close to exponential distribution and has average *gap* on the order of 100ns, we need to set an infinity *powerdown* threshold to prevent any *powerdown* transition.

The above conclusion does not hold on the *unoccupied* chips which are intentionally created by some page allocation or page movement algorithms [5]. The *gaps* of these idle chips are usually several magnitude larger than those of the "active" chips we have studied above and do not follow an exponential distribution anymore. Hence in those cases we still need a *powerdown* threshold to shut down the "idle" chips to get maximum energy efficiency.

5 Conclusion

Power management features provided by modern DRAM technologies can be exploited to develop power efficient memory systems. Petri net is a powerful tool that can be used to model and evaluate different memory power control policies.

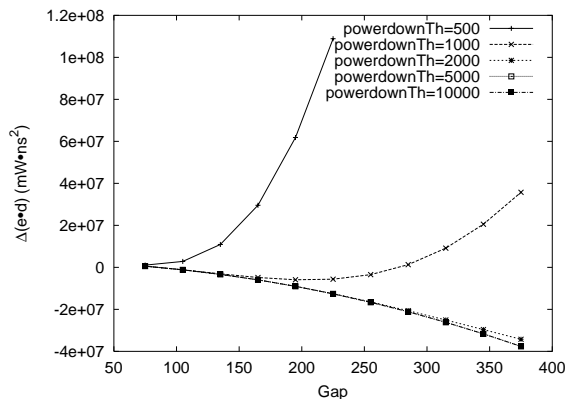


Fig. 6. $\Delta(e \bullet d)$ computed from 4-state model for different *gap* and *powerdownTh* values

In this paper, we consider a 2-state control policy model, derive our energy efficiency metric, and validate the model against probabilistic analysis. Then we extend our model to 4 states and investigate the effects of these additional states on energy efficiency. The results reveal that memory chips should always immediately transition to *standby*, and for chips that exhibit exponential-like access pattern without extremely long average *gap* values they should not transition to *powerdown* state.

All our studies to date are investigations of the appropriate control policy based on a certain available DRAM platform which provides power management features. As future work, in order to obtain some valuable power aware memory design points, we plan to explore the design space of alternative potential DRAM power/performance features (i.e., the number of available power states, the power consumption for each power state, the power consumption and delay for transition between power states, etc.).

Acknowledgments

This work supported in part by NSF Grants CCR-0082914, EIA-99-72879, EIA-99-86024, NSF CAREER Award MIP-97-02547, Duke University, and equipment donations from Intel and Microsoft. We thank Professor Christoph Lindemann and his group at the University of Dortmund for providing the DSPNexpress software.

References

1. G. Chiola, M. Ajmone Marsan, G. Balbo, and G. Conte. Generalized stochastic petri nets: A definition at the net level and its implications. *IEEE Transactions*

- on Software Engineering*, 19(2):89–107, February 1993.
2. V. Delaluz, M. Kandemir, N. Vijaykrishnan, A. Sivasubramaniam, and M.J. Irwin. DRAM Energy Management Using Software and Hardware Directed Power Mode Control. In *HPCA 2001*, January 2001.
 3. Xiaobo Fan, Carla S. Ellis, and Alvin R. Lebeck. Memory controller policies for dram power management. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, August 2001.
 4. D. Lammers. IDF: Mobile Rambus spec unveiled. *EETimes Online*, February 1999. [//www.eetimes.com/story/OEG19990225S0016](http://www.eetimes.com/story/OEG19990225S0016).
 5. Alvin R. Lebeck, Xiaobo Fan, Heng Zeng, and Carla S. Ellis. Power aware page allocation. In *Proceedings of Ninth International Conference on Architectural Support for Programming Languages and Operating System (ASPLOS IX)*, November 2000.
 6. Christoph Lindemann. *Performance Modelling with Deterministic and Stochastic Petri Nets*. John Wiley and Sons, 1998.
 7. M. Ajmone Marsan, G. Balbo, and G. Conte. *Modeling with Generalized Stochastic Petri Nets*. J. Wiley, 1995.
 8. M. Ajmone Marsan and G. Chiola. On petri nets with deterministic and exponential transition firing times. In *Proceedings of the 7th European Workshop on application and Theory of Petri Nets*, pages 151–165, June 1986.
 9. T. Murate. Petri nets: properties, analysis, and applications. *Proceedings of IEEE*, 77(4):541–580, April 1989.
 10. J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, 1981.