RELIABILITY-AWARE POWER MANAGEMENT OF MULTI-CORE PROCESSORS *

Jan Haase, Markus Damm, Dennis Hauser, Klaus Waldschmidt
J. W. Goethe-Universitt Frankfurt/Main, Technical Computer Sc. Dep.,
Box 11 19 32, D-60054 Frankfurt/Main, Germany
{haase|damm|dhauser|waldsch}@ti.informatik.uni-frankfurt.de

Abstract Long-term reliability of processors is experiencing growing attention since decreasing feature sizes and increasing power consumption have a negative influence on the lifespan. The reliability can also be influenced by Dynamic Power Management (DPM), since it affects the processor's temperature.

In this paper, it is examined how different DPM-strategies for Multi-Core processors alter their lifespan. By simulating such a Multi-Core system using the Self Distributing Virtual Machine (SDVM), thus exploiting dynamic parallelism, it is shown that its long-term reliability can be influenced actively with different DPM strategies.

Keywords: Adaptivity; Power Management; Reliability; SDVM.

1. INTRODUCTION

The long-term reliability resp. lifespan of microprocessors hasn't been much of an issue in the past, since a processor was usually obsolete due to technological aging (and has been replaced) before it began to fail. This is about to change for several reasons. First, microprocessors and multi-core processors are nowadays combined with other components as complete systems on chip (SoCs) or networks on chip (NoCs). Therefore the processor cannot be replaced easily. Secondly, smaller feature sizes and increasing power densities lead to a higher vulnerability to wear-out based failure mechanisms like electromigration or stress migration. The international technology roadmap on semiconductors (ITRS) sees a trend that is threatening "the nearly unlimited lifetime and high level of reliability that customers have come to expect" [1]. The approaches to tackle this problem are mostly design-centric. RAMP [2], for example, is a model to determine lifespan estimates depending on the architecture

*Parts of this work have been supported by the Deutsche Forschungsgemeinschaft (DFG).

of a processor. In a subsequent paper, however, the authors extend their approach to a so called *Dynamic Reliability Management* [3], whose idea is to adjust a processor at runtime (e.g. by voltage scaling) to meet a certain reliability target, though no algorithms for this cause are proposed. Apart from this, no further concepts or algorithms for dynamic reliability management do yet exist.

Our remedy to this problem has two ingredients: Dynamic power management (DPM) and parallel computing. It has been noted in [4] that DPM schemes affect a processor's reliability, since it directly affects a processor's temperature. The essential failure mechanisms like electromigration, corrosion, time-dependent dielectric breakdown (TDDB), hot carrier injection (HCI), surface inversion, and stress migration are more or less temperature dependent [5]. While DPM tends to lower a processor's temperature, which is beneficial, it also leads to the unfavorable effect of temperature cycling, i.e. frequent heating up and cooling down.

Dynamic power management on multi-core processors, however, has a lot more possibilities to scale the power consumption of a chip: Aside from clock frequency reduction (along with dynamic voltage scaling or adaptive body biasing), whole cores can be switched off without disrupting the execution of applications. Since the workload and thus DPM in parallel computing environments also depends on the parallelizability of an application, it seems to be obvious that this can be done efficiently only with a dynamic approach.

The SDVM (Self Distributing Virtual Machine) as a middleware for the dynamic, automatic distribution of code and data over any network of computing resources seems to be an ideal choice to be run on multi-core processors in NoCs. In particular, it supports adding and removing of computing resources at runtime, making the implementation of the aforementioned dynamic power management on multi-core processors possible in the first place. To permit DPM on an SDVM-driven multi-core processor, an appropriate power managing mechanism has been implemented, which scales the performance of the cores according to the current workload. The reliability-awareness is then achieved by appropriate power management policies.

The goal of this paper is to examine the potential of such reliability-aware power management strategies for multi-core processors by simulation.

2. SDVM - A MIDDLEWARE FOR POWER OPTIMIZED SOCS

The Self Distributing Virtual Machine (SDVM) [6] was designed to feature undisturbed parallel computation flow while adding and removing processing units from computing clusters. These clusters may consist

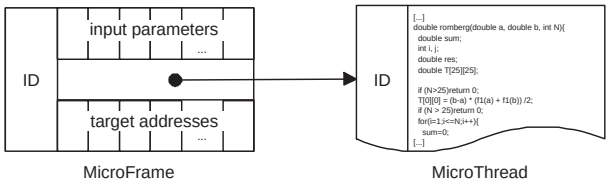


Figure 1. The Micro*thread* contains a code fragment whereas the Micro*frame* contains the parameters needed to execute the corresponding code fragment, as well as the IDs of other Microframes the results of the execution then should be sent to.

of several processing cores or even full-grown computers, and any connection network topology is supported.

The SDVM is a middleware, implemented as a daemon to be run on each participating machine or processor, creating a *site* each. The sites communicate by sending messages. Applications must be cut to convenient application fragments, the *microthreads*, which can be executed on any site. The SDVM follows the dataflow principle, therefore a microthread is executable if it has received all its needed input parameters. These parameters are collected in a special data structure, the *microframe* (see Fig. 1). Data, and code, is automatically sent to the sites where it is needed. Therefore, the SDVM is actually real parallel processing though it may look like a multi-threaded concept.

The SDVM supports growing and shrinking the cluster at runtime. When a site is out of work, it requests executable microframe/microthread pairs from other sites automatically. This behavior provides automatic load balancing, even between processing units with different processing speeds, and offers the addition of new sites at runtime. For a detailed description of the scheduling see [6]. If more processing power is available than needed, the local memory data and possibly microthreads are pushed out to another site and then the site can be safely shut down.

The SDVM daemon is organized in several modules with different tasks, which are part of one of three layers:

- The execution layer, where the actual calculationd are performed. It contains the memory (containing data and microframes), the code storage (which contains the needed microthreads or requests them from other sites), the processing manager, the scheduler (see Fig. 2), and a unit for possible input and output.
- The network layer is the part of the daemon which is related to sending messages over the network. Messages are encrypted by a security manager to avoid eavesdropping. The *energy manager* which is described in section 4 is located here.
- The maintenance layer is concerned with the organization of the cluster and the (local) site. Modules are located here which know

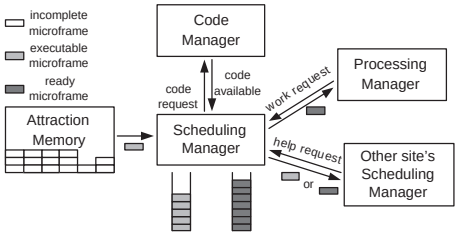


Figure 2. The SDVM's scheduling mechanism: The scheduling manager receives executable microframes (which have got all their parameters) from the attraction memory. It then sends a code request to the code manager. When the corresponding microthread code is locally available (possibly after getting it from another site), the code manager informs the scheduling manager, which then puts the microframe from the executable-queue into the ready-queue. The processing manager is always given a ready microframe. Help requests from other site's scheduling managers are answered by sending ready or—preferably—executable microframes.

the current composition of the cluster, the physical (IP) addresses of other sites, data about the local site (e.g. performance data), and the list of currently running applications and where to find their microthreads.

The SDVM can be used to simulate a multi-core processor on a computer cluster, but it may even be run on a real multi-core processor, as well. Due to these features, the SDVM offers the convenient mechanisms to support different power states of processing units in a SoC. The thereby realized power management is described in the following section.

3. RELIABILITY AND TEMPERATURE

The long-term reliability of a processor is affected by its operating temperature as well as thermal cycling. The effect of the temperature can be modeled by the Arrhenius equation, which describes the influence of the temperature on the rate of chemical reactions. In terms of MTTF(Mean-Time-To-Failure), we then have [2]

$$MTTF \sim e^{\frac{E_a}{kT}} \tag{1}$$

where T is the operating temperature in Kelvin, k is Boltzmann's constant, and E_a is the activation energy in electron volts of the precise failure mechanism considered. The Arrhenius equation is the basis for modeling the temperature-dependence of several failure mechanisms. For instance, failure due to electromigration in interconnects can be modeled with the equation [5]

$$MTTF_{EM} \sim A_0(J - J_{crit})^{-N} e^{\frac{E_a}{kT}} \tag{2}$$

where J is the current density, J_{crit} is the critical current density for electromigration and A_0 and N are empirically determined constants. The activation energy E_a then depends on the material used for the interconnect and varies from 0.5 to 0.9 eV [5]. Other failure mechanisms like stress migration or hot carrier injection have different activation energies.

With the knowledge of the physical and structural construction of a chip, the models for different failure mechanisms can be combined to get a model (like RAMP [2]) for the processor's reliability. As we make no assumptions on the internal structure of the processors or the materials used, it would make no sense to use those detailed models for our purposes. Instead, we use equation 1 as a generic temperature-dependant reliability measure for processors. For E_a we use a value of 0.9 eV.

The temperature of a processor depends on its power consumption, and since dynamic power management lowers the average temperature, it should contribute to the chip's lifespan. But, as it was noted in [4], the switching between different power consumption levels leads to thermal cycling, which can cause various types of failures like lifted bonds, solder fatigue or even a cracked die [5]. The effect of thermal cycling on the reliability of a chip can be modeled by the Coffin-Manson relation, which computes the number of cycles to failure, N_f, as [5]

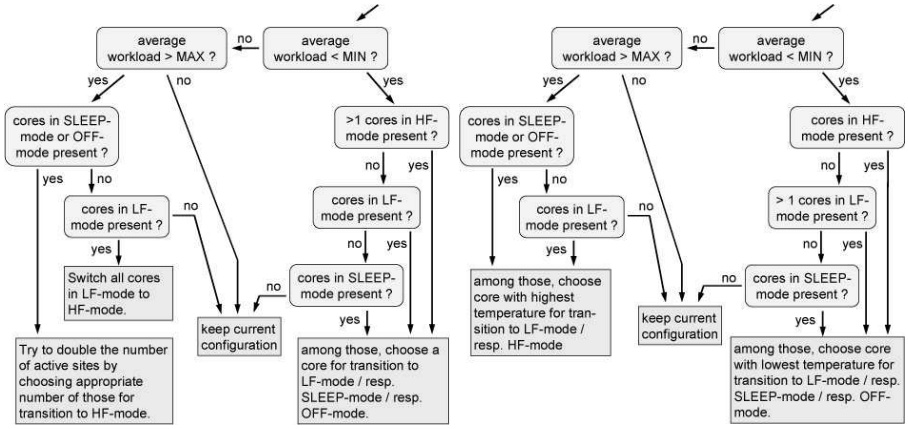
$$N_f = C_0 \cdot (\Delta T)^{-q} \tag{3}$$

where ΔT is the magnitude of thermal cycling, C_0 is a material-dependant constant, and q is the empirically determined Coffin-Manson exponent. This exponent depends on the failure mechanism considered; we use a value of 1.9, which focuses on the reliability of the package [2].

For our purposes, we use equations 1 and 3 for a comparitive analysis of different power management strategies to the non-powermanaged case to get an *acceleration factor* (i.e. the ratio) for each of the PM-strategies described below. Therefore, we don't need to choose a value for C_0 in equation 3, since it then cancels out.

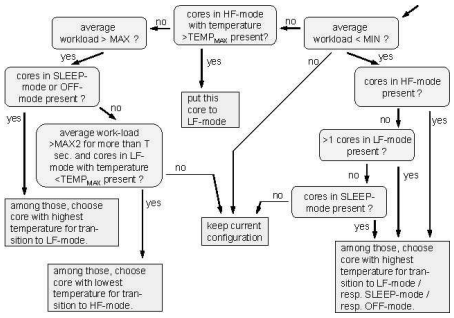
3.1 RELIABILITY AWARE POWER MANAGEMENT

In view of the previous section, a power management strategy which is aware of reliability issues should limit the temperature as well as temperature changes. While the first is a side effect of usual power management strategies, the latter might involve keeping a processor "powered up", although this might not be necessary regarding performance, and is definitely not desirable regarding power consumption. So obviously, there's a trade-off between power consumption, performance and reliability.



(a) fast-upgrade policy

(b) smooth-temperature policy



(c) low-temperature policy

- In the PM-state "HFM" (High Frequency Mode), the core runs with maximum clock frequency and supply voltage.
- The PM-state "LFM" (Low Frequency Mode) has lower clock frequency and supply voltage, yielding less performance and power consumption, but full functionality.
- If a core is in the "SLEEP" state, it is switched off while its state is stored, so it can resume computation pretty quickly. The power consumption is reduced.
- The PM-state "OFF" is self-explanatory; the state of the core is not stored and the power consumption is minimal. Resumption of the computation takes more time than in SLEEP-mode.

(d) four PM-states

Figure 3. Power management policies.

In our simulation, we consider two reliability aware dynamic power management (RADPM) strategies: The low-temperature-policy, which tries to keep the temperature as low as possible, and the smooth-temperature-policy, whose goal is to restrict thermal cycling. These policies are compared to the (reliability unaware) fast-upgrade-policy, which tries to optimize performance and serves as a representative of usual power management strategies. The simulated computing environment is a homogenous multi-core-processor with four cores. Each core has four different Power-Management states (see Figure 3(d)).

Figure 3 shows diagrams describing the three PM-policies in detail. Note that the parallelization of the applications is influenced *indirectly* by altering the PM-states and thus the performance of the different cores.

4. IMPLEMENTATION AND RESULTS

The aforementioned power management capabilities were integrated into the SDVM by implementing the so-called *energy manager*. This energy manager has a master mode and a slave mode. Only one core's energy manager is in master mode (the *master core*), which then controls the PM-states of all cores. The main task of the energy managers in slave mode is to listen to the master core and to implement its orders, setting the local site to the desired PM-state. If a slave energy manager observes the absence of the master core (due to a crash or shutdown), it starts an *election* of a new master core.

The basis for the decision for a new power configuration is the temperature and the mean workload of each core. This information is distributed through the cluster by the SDVM's cluster manager's message mechanism.

The test set-up simulates a homogenous multi-core processor with four cores. To this end, the SDVM runs on a cluster of four identical computers. To each PM-state, a typical power consumption value based on an Intel Pentium M processor [7] is assigned (see Table 1).

Table 1. PM-states and their power consumption

PM-state	HF	HF_{idle}	LF	LF_{idle}	SLEEP	OFF
power consumption	15 W	10 W	7.5 W	4 W	3 W	0.2 W

The temperature T_J of a core is determined out of its power consumption by the formula

$$T_J = T_A + \theta_{AJ} \cdot P_{DISS} \tag{4}$$

Where T_A is the environmental temperature, θ_{AJ} is the thermal resistance of the core, and P_{DISS} is the power consumption. For θ_{AJ}, a value of $4.5°C/W$ is used.

With this set-up, each PM-strategy (and the "no-PM strategy" as a reference) was simulated using identical workloads composed of multiple instances of a parallelized example application (Romberg integration [6]). The results of the simulations of the PM-strategies are given in figures 4, 5, and 6 showing the workload (area chart) and the temperature (black line) of the four cores for the fast-upgrade, smooth-temperature and low-temperature policy respectively.

The figures 4, 5, and 6 show a clear difference between the three policies. The low-temperature policy restricts the maximum temperature to

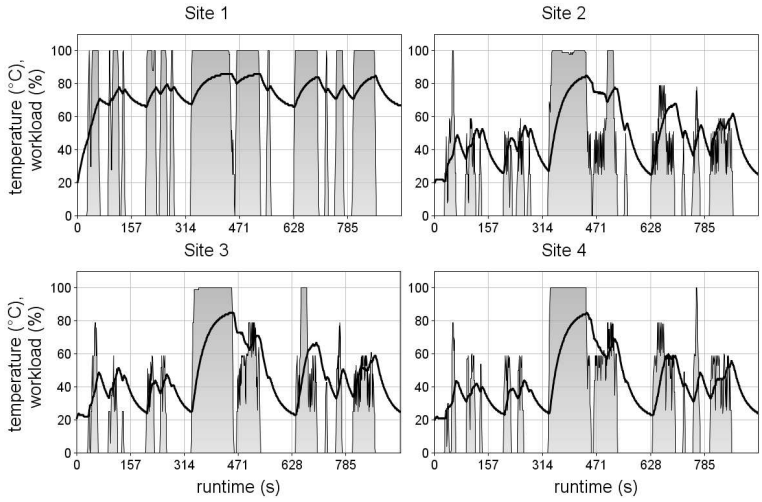


Figure 4. Fast-upgrade policy.

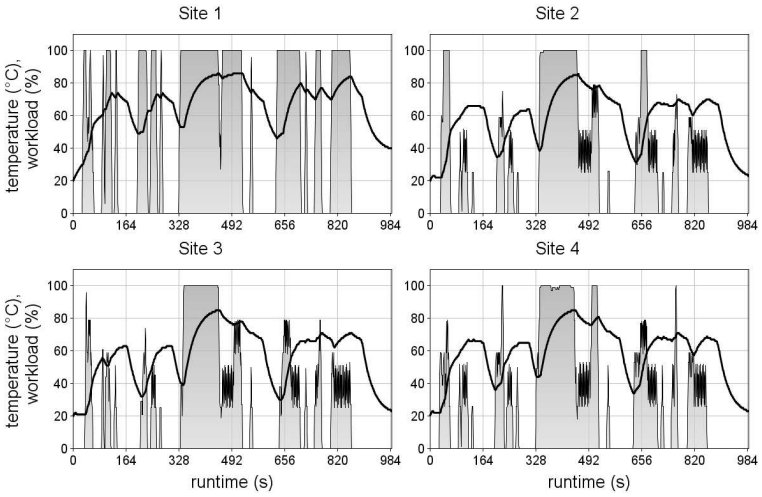


Figure 5. Smooth-temperature policy.

$61°C$, while with the other two policies a maximum temperature of $86°C$ is obtained. The higher temperature of core 1 in figure 4 is caused by the fact that the fast-upgrade policy always leaves one core in HF-mode.

Regarding thermal cycling, we see a reduction both in frequency and magnitude by the smooth-temperature policy compared to the fast-upgrade policy. Because of the temperature limitation, the low-temperature policy causes thermal cycling of lower magnitude, but also with higher frequencies, expecially when the temperature limit is reached.

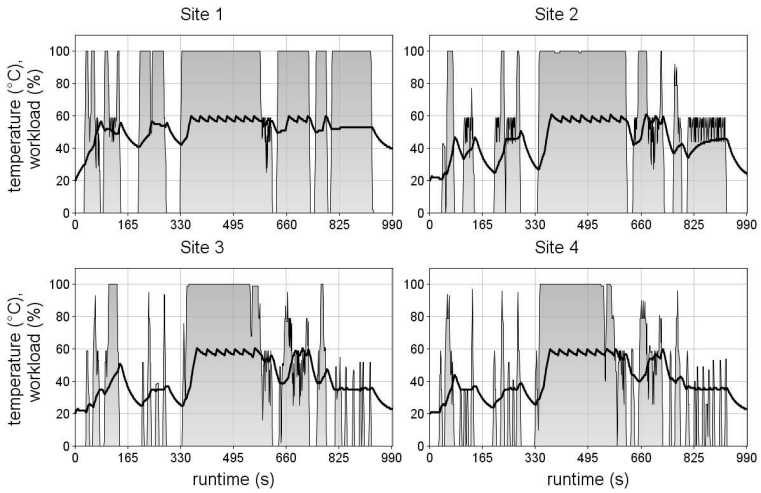


Figure 6. Low-temperature policy.

Table 2. AF_T, AF_{TC}, mean runtime, and mean power consumption of all cores

PM-policy	AF_T	AF_{TC}	mean runtime	power consumption
no PM	1	1	32.7s	48.15 W
fast-upgrade	0.12	3.27	35.0s	31.55 W
smooth-temperature	0.19	1.2	35.9s	37.29 W
low-temperature	0.1	3.28	64.8s	23.18 W

Using the models described in section 3, we computed for each policy the acceleration factors AF_T and AF_{TC} giving the acceleration of the time to failure due to temperature and temperature cycling respectively. Table 2 gives the means of these values over all cores, together with the mean runtime and the mean power consumption.

The acceleration factors without power management are 1, since this case is the reference. We see that all PM-strategies are beneficial regarding failure due to temperature, especially the low-temperature and the fast-upgrade policy. The fact that the low temperature policy is not much better than the fast upgrade policy regarding AF_T (despite the lower maximum temperature) is owed to prolonged computation durations of the first whereas the latter has shorter computation durations which leaves more time for cooling down. In view of thermal cycling, the smooth-temperature policy is the clear winner, while the other two policies show obvious acceleration.

This clearly shows that the reliability of a multi-core chip can be influenced actively with PM-strategies. It should be pointed out that such an approach is only possible using dynamic power management,

which in turn can be implemented only within a system which distributes the workload dynamically, as the SDVM does. Incorporating reliability awareness into compile-time power management schemes seems to be almost infeasible.

5. CONCLUSION

In this paper, we proposed *reliability-aware dynamic power management* (RADPM), which incorporates lifespan-controlling goals. The usability of RADPM to prolong system-lifetime was demontrated by simulating a multi-core chip on the Self Distributing Virtual Machine (SDVM). The SDVM was augmented for this purpose with the so-called *energy manager*, which implements different PM-policies. The basic approach, however, could be implemented on any multi-core system which distributes the workload dynamically.

The PM-policies presented are no final solutions for RADPM, but serve as a proof of concept, that the long-term reliability of a multi-core chip can actually be altered deliberately with RADPM. Real implementations for RADPM on mult-icore chips could include, for example, a "reliability account" for each core or could consider the geometric configuration of the cores on the chip to optimize the temperature distribution.

A new insight, however, is that parallelism may not only be used to improve performance, but to improve reliability as well.

The SDVM's homepage containing its complete source code and documentation can be found at: `http://sdvm.ti.cs.uni-frankfurt.de`.

REFERENCES

[1] ITRS, "Critical reliability challenges for the international technology roadmap for semiconductors," 2003, international Sematech Technology Transfer document 03024377A-TR.

[2] J. Srinivasan, S. V. Adve, P. Bose, J. Rivers, and C.-K. Hu, "Ramp: A model for reliability aware microprocessor design," in *IBM Research Report, RC23048 (W0312-122)*, Dec. 2003.

[3] J. Srinivasan and et al., "The case for lifetime reliability-aware microprocessors," in *Proc. of the 31st Annual Intl. Symp. on Comp. Architecture*, 2004.

[4] K. Mihic, T. Simunic, and G. D. Micheli, "Reliability and power management of integrated systems," in *DSD - Euromicro Symposium on Digital System Design*, 2004, pp. 5–11.

[5] JEDEC, "Failure mechanisms and models for semiconductor devices," 2003, jEDEC Publication JEP122-B, Jedec Solid State Technolgy Association.

[6] J. Haase, F. Eschmann, B. Klauer, and K. Waldschmidt, "The SDVM: A Self Distributing Virtual Machine," in *Organic and Pervasive Computing – ARCS 2004: International Conference on Architecture of Computing Systems*, ser. Lecture Notes in Computer Science, vol. 2981. Heidelberg: Springer Verlag, 2004.

[7] Intel, "Pentium M Processor Datasheet," Apr. 2004, http://www.intel.com/design/mobile/datashts/252612.htm.