Reliability-Aware Power Management Of Multi-Core Systems (MPSoCs)

Klaus Waldschmidt, Jan Haase, Andreas Hofmann, Markus Damm, and Dennis Hauser
Technische Informatik, J. W. Goethe Universität
Post Box 11 19 32, 60054 Frankfurt a. M., Germany
{waldsch|haase|ahofmann|damm|dhauser}@ti.cs.uni-frankfurt.de

Abstract

Long-term reliability of processors in embedded systems is experiencing growing attention since decreasing feature sizes and increasing power consumption have a negative influence on the lifespan. Among other measures, the reliability can be influenced significantly by Dynamic Power Management (DPM), since it affects the processor's temperature. Compared to single-core systems reconfigurable multi-core SoCs offer much more possibilities to optimize power and reliability.

The impact of different DPM-strategies on the lifespan of multi-core processors is the focus of this presentation. It is shown that the long-term reliability of a multi-core system can be influenced deliberately with different DPM strategies and that temperature cycling greatly influences the estimated lifespan. In this presentation, a new reliability-aware dynamic power management (RADPM) policy is explained.

1 Introduction

Systems-on-Chip (SoCs) and Networks-on-Chip (NoCs) will be based on multi-core platforms in future. Besides an increase in performance, reliability and lifespan of these processor cores will become an important issue. Due to the combination of processing cores with other components as Systems-on-Chip, these cores cannot be replaced. Moreover, smaller feature sizes and increasing power densities further stress the lifetime, as they lead to a higher vulnerability to wear-out based failure mechanisms like electro- or stress migration. Therefore reliability adds to the essential problems like performance and power management that must be addressed during the design of an embedded system.

The approaches to tackle the new problem are mostly design-centric. RAMP [9], for example, is a model to determine lifespan estimates depending on the architecture of a processor. In a subsequent paper, however, the authors extend their approach to a so called Dynamic Reliability Management [8], 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.

It has been noted in [7] that DPM schemes affect a processor's reliability, since they directly influence 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 [6]. 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.

Therefore our remedy to balance the trade-off between power consumption and reliability is *Reliability-Aware Dynamic Power Management* (RADPM). Compared to single-core systems, DPM on multi-core SoCs has a lot more ways to scale the energy consumption of a chip if tasks can migrate between cores: Aside from clock frequency reduction, whole cores can be switched off without disrupting the execution of applications.

The workload of a parallel computing environ-

Dagstuhl Seminar Proceedings 06141
Dynamically Reconfigurable Architectures
http://drops.dagstuhl.de/opus/volltexte/2006/745

ment depends on the parallelizability of the applications. Since DPM reacts to this workload, this obviously can be realized efficiently only with a dynamic approach.

The SDVM (Self Distributing Virtual Machine) [3] as a middleware for 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 particular, it supports adding and removing computing resources at runtime, making the implementation of the aforementioned dynamic power management on multi-core processors possible.

To permit DPM on an SDVM-driven multi-core SoC, an appropriate power managing mechanism has been drafted and implemented, which scales the performance of the cores according to the current workload. The reliability-awareness is then achieved by a new power management policy.

The simulations described in this presentation show the potential of reliability-aware power management strategies for multi-core SoCs. In section 2, the concept of the SDVM and its application to reconfigurable multi-core SoCs is presented. Power management, its relevance for reliability, and the modeling thereof is illustrated in section 3. After some simulation results in section 4, this presentation closes with a conclusion in section 5.

2 SDVM - A middleware for MPSoCs

The Self Distributing Virtual Machine (SDVM) [4] is a dataflow-driven parallel computing middleware. It was designed to feature undisturbed parallel computation flow while adding and removing processing units from computing clusters. Applications for the SDVM must be cut to convenient application fragments, which will be spread throughout the cluster depending on the data distribution. Each processing unit which is encapsulated by the SDVM virtual layer and thus acting as an autonomous member of the cluster is called a *site*. The sites communicate by message passing.

The SDVM has several distinguishing features which support the aforementioned reliability-aware power management for SoCs. These features include:

- undisturbed parallel computation while resizing the cluster
- distributed dynamic scheduling and thereby automatic load balancing
- participating computing resources may have different architectures and processing speeds
- support for any connection network topology

Due to the above mentioned features, the SDVM offers the convenient mechanisms to support different power states of processing units in a SoC.

The SDVM is currently implemented as a prototype [1] running as Linux daemons on a workstation cluster creating a site on each system. For this project it has been used to simulate a multi-core processor system.

2.1 Integration of power management

The aforementioned power management capabilities were integrated into the SDVM by implementing a new module, the *energy manager*, which controls the energy state of the local site. A method where any site may freely decide for itself its energy state may result in a situation where all sites simultaneously decide to shut down; therefore the energy managers use an *election* algorithm to define a master which then is the only one to decide.

The master regularly defines the new power configuration of each core based upon the temperature and the mean workload of each core. This information is distributed through the cluster by the SDVM's message mechanism. The master may even decide to shut down its own site or to quit being the master; then the election is simply started again among the remaining sites. 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.

3 Reliability Aware Power Management

Sophisticated power management can scale the power consumption of a system using a variety of measures like frequency scaling, dynamic voltage scaling (affecting dynamic power), adaptive body

biasing (affecting leakage power), and clock-gating. Unlike static power management dynamic power management reacts dynamically and continuously to workload variation at runtime.

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 [7], 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 [6]. Hence, it is not obvious how a certain power management policy affects the lifetime, if all influences are taken into account. Therefore a model is required to evaluate power management policies.

3.1 Modeling Reliability Aware Power Management

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. The MTTF (Mean-Time-To-Failure) can be estimated by the following formula:

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

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.

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 [9]) for the processor's reliability. As no assumptions on the internal structure of the processors or the materials used are made in this presentation, it would make no sense to use those detailed models for our purposes. Instead, equation 1 is used as a generic temperature-dependant reliability measure for processors. For E_a a value of 0.9 eV is used.

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 [6]

$$N_f = C_0 \cdot (\Delta T)^{-q} \tag{2}$$

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; important mechanisms are lifted wire bondings, solder fatigue, or package failures. In this presentation, the reliability of the package is chosen, therefore a value of 1.9 [9] is used.

For our purposes, equations 1 and 2 are used for a comparitive analysis of each of the PM-strategies described below to the non-powermanaged case. The *acceleration factors* AF_T and AF_{TC} represent the acceleration of the time to failure due to temperature and temperature cycling, respectively. Thus lower values are better; values below 1 denote a prolonged lifespan compared to the non-powermanaged case. The acceleration factor of the temperature dependant MTTF is calculated using equation 3.

$$AF_T = \frac{MTTF_{noPM}}{MTTF} \tag{3}$$

To calculate the acceleration factor of the Coffin-Manson relation no value for C_0 in equation 2 needs to be chosen, since it then cancels out. As different PM-strategies not only alter the magnitude of thermal cycling, but also the cycle frequency f, the ratio thereof has to be taken into account, too. Hence, equation 4 is used to calculate AF_{TC}.

$$AF_{TC} = \frac{f}{f_{noPM}} \cdot \left(\frac{\Delta T}{\Delta T_{noPM}}\right)^{1.9} \tag{4}$$

3.2 Power management policies

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.

In this simulation, two reliability aware dynamic power management strategies are considered: 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. Furthermore, the low-temperature-policy aims to limit the temperature of a core to a given maximum t_{max}. 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. Figure 1 shows a diagram describing the smooth-temperature policy.

The simulated computing environment is a homogenous multi-core-processor with four cores. Each core has four different Power-Management states as shown in Table 1.

Table 1: Simulated PM-states

PM-state	clock frequency	supply voltage	time to HFM
HFM	max.	max.	—
LFM	reduced	reduced	short
SLEEP	stopped	reduced	long
OFF	stopped	off	very long

For a more detailed description of the three power management policies and the PM-states of the cores see [2].

4 Results

The test setup simulates a homogenous multi-core processor with four cores. To each PM-state, a typical power consumption value is assigned (see Table 2). These values serve as examples and are based on the power consumption of an Intel Pentium M processor [5].

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

$$T_J = T_A + \theta_{AJ} \cdot P_{DISS} \tag{5}$$

Where T_A is the environmental temperature, θ_{AJ} is the thermal resistance of the core and its cooling system, and P_{DISS} is the power consumption. For θ_{AJ}, a value of $4.5°C/W$ is used. However, the real temperature of the core T_{real} furthermore depends on the heat capacity of the package including the cooling system. This is modeled using the formula

$$T_{real} = T_{real_{old}} + \frac{T_J - T_{real_{old}}}{15} \tag{6}$$

which is evaluated every second by the simulation environment for each core.

Table 2: PM-states and their power consumption

PM-state	HFM	LFM	SLEEP	OFF
power consumption	15 W (10 W idle)	7.5 W (4 W idle)	3 W	0.2 W

With this setup, each PM-strategy (and the "no-PM strategy" as a reference) was simulated using identical workloads composed of multiple instances of a parallelized application. As an example, an algorithm performing the Romberg integration was used. The results of the simulations of the PM-strategies are given in Figures 2, 3, and 4. The figures show the workload (area chart) and the temperature (black line) of each of the four cores.

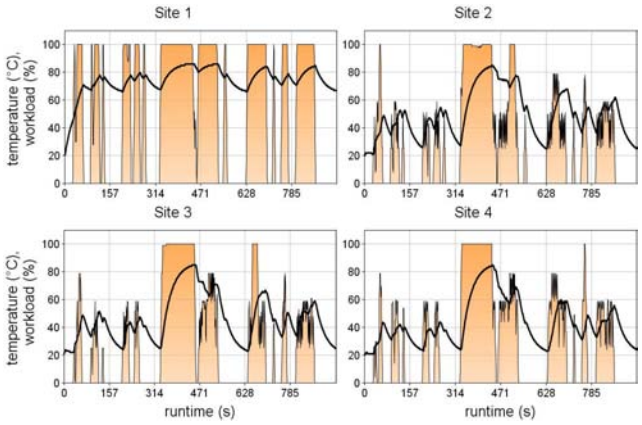

Figure 2: Fast-upgrade policy

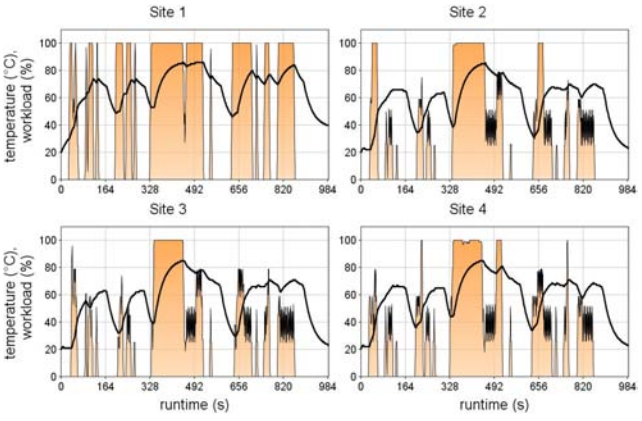

Figure 3: Smooth-temperature policy

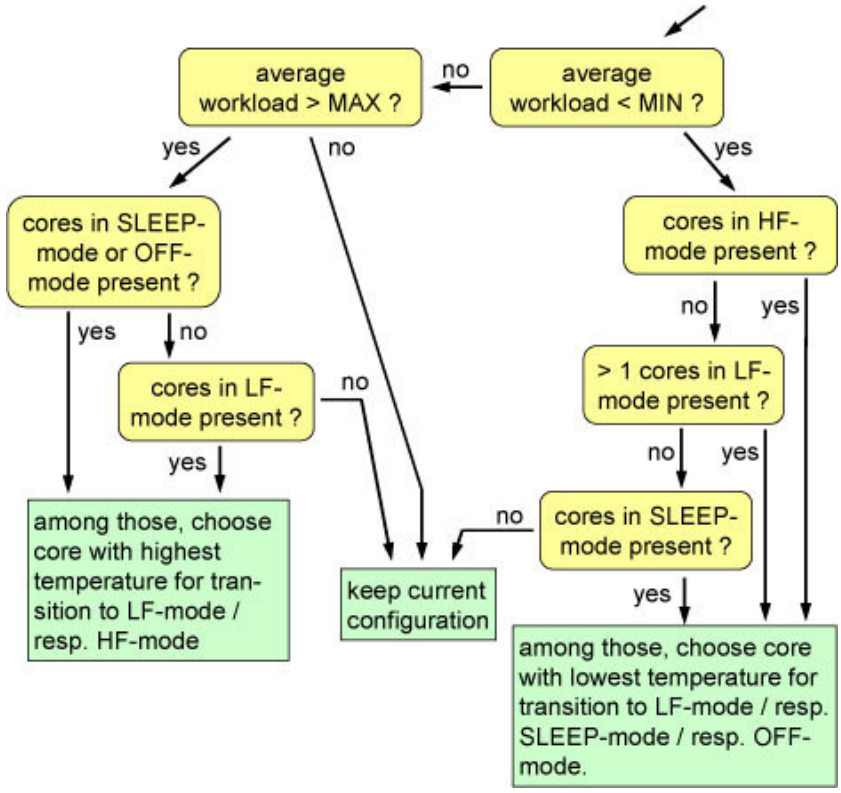

Figure 1: The smooth-temperature policy.

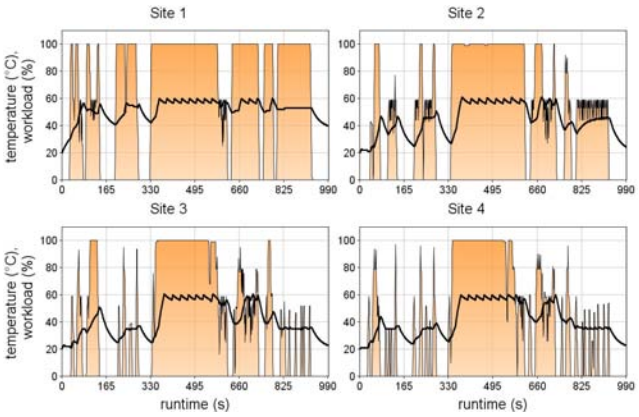

Figure 4: Low-temperature policy

Figures 2, 3, and 4 show a clear difference between the three policies. The low-temperature policy restricts the maximum temperature to $61°C$, while with the other two policies a maximum temperature of $86°C$ is obtained. The higher mean temperature of core 1 in Figure 2 is caused by the fact that the fast-upgrade policy always leaves one core in HF-mode.

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

Using the models described in section 3.1, for each policy the acceleration factors AF_T and AF_{TC} are computed. Table 3 gives the means of these values over all cores, together with the mean CPU time and the mean power consumption for the given workload. The acceleration factors of the non-powermanaged policy are 1, since this case is the reference. 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 extended computation durations of the first whereas the latter has shorter computation durations which leaves more time for cooling down. In terms of thermal cycling related failure the smooth-temperature policy is the most reliable

Table 3: 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

power management policy. Compared to the other two policies its acceleration factor is over 2 times smaller.

The results clearly show 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. Incorporating reliability awareness into compile-time power management schemes seems to be almost infeasible.

5 Conclusion

In this presentation, *reliability-aware dynamic power management* (RADPM) is described, which targets lifespan-controlling goals. The usability of RADPM to extend system-lifetime was demontrated by simulating a multi-core chip on the SDVM. The SDVM was augmented for this purpose to implement 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 rather serve as a proof of concept, that the long-term reliability of a multi-core chip can actually be improved. Further implementations for RADPM on multi-core chips could include a "reliability account" for each core or consider the chip's topology to optimize the temperature distribution. In combination with the smooth-temperature policy a reliability-aware scheduling could further reduce thermal cycling. One of our next goals is to measure the impact of dynamic reconfigurations on the reliability. Furthermore, it would be beneficial to explore the impact and optimization of power-management strategies on the reliability of the external power supply circuits.

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

References

[1] The SDVM homepage, 2006. `http://sdvm.ti.cs.uni-frankfurt.de`.

[2] Jan Haase, Markus Damm, Dennis Hauser, and Klaus Waldschmidt. Reliability-aware power management of multi-core processors. In *5th IFIP Working Conference on Distributed and Parallel Embedded Systems (DIPES 2006)*, Braga, Portugal, October 2006. To be published.

[3] Jan Haase, Frank Eschmann, Bernd Klauer, and Klaus Waldschmidt. The SDVM: A Self Distributing Virtual Machine. In *Organic and Pervasive Computing – ARCS 2004: International Conference on Architecture of Computing Systems*, volume 2981 of *Lecture Notes in Computer Science*, Heidelberg, 2004. Springer Verlag.

[4] Jan Haase, Frank Eschmann, and Klaus Waldschmidt. The SDVM - an Approach for Future Adaptive Computer Clusters. In *10th IEEE Workshop on Dependable Parallel, Distributed and Network-Centric Systems (DPDNS 05)*, Denver, Colorado, USA, April 2005.

[5] Intel. Pentium M Processor Datasheet, April 2004. `http://www.intel.com/design/mobile/datashts/252612.htm`.

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

[7] K. Mihic, T. Simunic, and G. De Micheli. Reliability and power management of integrated systems. In *DSD - Euromicro Symposium on Digital System Design*, pages 5–11, 2004.

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

[9] Jayanth Srinivasan, Sarita V. Adve, Pradip Bose, Jude Rivers, and Chao-Kun Hu. Ramp: A model for reliability aware microprocessor design. In *IBM Research Report, RC23048 (W0312-122)*, December 2003.