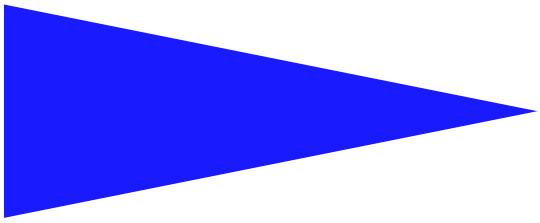
Periodic activity migration for fast sequential execution in future heterogeneous multicore processors

Pierre Michaud

I R I S A
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

PUBLICATION
INTERNE
N° 1909



PERIODIC ACTIVITY MIGRATION FOR FAST SEQUENTIAL EXECUTION IN FUTURE HETEROGENEOUS MULTICORE PROCESSORS

PIERRE MICHAUD

ISSN 1166-8687

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

Periodic activity migration for fast sequential execution in future heterogeneous multicore processors

Pierre Michaud

Systèmes communicants
Projets CAPS

Publication interne n ˚ 1909 — Novembre 2008 — 18 pages

Abstract: On each new technology generation, miniaturization permits putting twice as many computing cores on the same silicon area, potentially doubling the processor performance. However, if sequential execution is not accelerated at the same time, Amdahl's law will eventually limit the actual performance. Hence it will be beneficial to have asymmetric multicores where some cores are specialized for fast sequential execution. This specialization may be achieved by architectural means, but it may also be achieved by specializing transistors, voltage, and clock frequency. In the latter case, one of the main constraints is that the power consumption of fast cores is not increased across technology generations. Yet this implies that the instantaneous heat flux in fast cores be potentially doubled on each new generation. A high instantaneous heat flux can be tolerated by doing periodic activity migration. This requires to double the number of fast cores on each new generation, even though only a single fast core can be used at a given time. To keep the chip temperature below the limit, the migration interval must be divided approximately by four on each new generation. We show with an analytical model that this will eventually decrease the apparent level-2 cache size. We show that this problem can be tackled by preparing a certain number of cores before they become active.

Key-words: Heterogeneous multicore processor, power consumption, heat flux, periodic activity migration

(Résumé : tsvp)

Centre National de la Recherche Scientifique
(UMR 6074) Université de Rennes 1 – Insa de Rennes

Institut National de Recherche en Informatique
et en Automatique – unité de recherche de Rennes

Migration périodique d'activité pour exécution séquentielle rapide sur les futurs processeurs multi-coeurs hétérogènes

Résumé : À chaque nouvelle génération technologique, la miniaturisation de l'échelle de gravure permet de doubler le nombre de coeurs de calcul sur une surface de silicium donnée, doublant ainsi la performance potentielle. Cependant, si l'exécution séquentielle n'est pas accélérée simultanément, la loi d'Amdahl finira par limiter la performance effective. Il sera donc intéressant de considérer des multi-coeurs asymétriques comportant des coeurs spécialisés pour l'exécution séquentielle rapide. Cette spécialisation peut être obtenue par des moyens architecturaux. Elle peut aussi être obtenue en spécialisant les transistors, la tension électrique et la fréquence d'horloge. Dans ce cas-ci, une des contraintes principales est que la puissance électrique des coeurs rapides n'augmente pas d'une génération technologique à la suivante. Mais cela implique que le flux de chaleur instantané dans les coeurs rapides soit potentiellement doublé à chaque nouvelle génération. Un flux de chaleur élevé peut être toléré en faisant de la migration périodique d'activité. Cela nécessite de doubler le nombre de coeurs rapides à chaque nouvelle génération, cependant qu'un seul coeur rapide ne peut être utilisé à un instant donné. Afin de maintenir la température en-deçà de la limite, l'intervalle de migration doit être approximativement divisé par quatre à chaque nouvelle génération. Nous montrons à l'aide d'un modèle analytique que cela finira par réduire la taille apparente du cache de deuxième niveau. Nous montrons que ce problème peut être combattu en préparant un certain nombre de coeurs avant qu'ils ne deviennent actifs.

Mots clés : Processeur multi-coeur hétérogène, consommation électrique, flux de chaleur, migration périodique d'activité

1 Introduction

Moore's law has been the main reason behind the huge processor performance gains in the last four decades. Smaller transistors and wires were faster, more energy efficient, and it was possible to build more complex circuits using the same silicon area. Performance gains came both from architectural improvements and from shorter logic-gate delays. However, during the 2000's, the processor performance has not increased as quickly as in the previous three decades. Architectural improvements have been slow because processors have reached a high degree of complexity and because the simplest and most effective architectural techniques have already been used. Nevertheless, the continuation of Moore's law would have led to expect a continuous increase of the processor clock frequency that normally comes with miniaturization. However, for several reasons, this has not been the case. Instead, the processor industry went to multicores. Initially, the move to multicores was a plan B solution to the unexpected leakage power consumption experienced with the 90 nm and 65 nm technologies. After this initial move, the processor industry adopted the multicore as the main paradigm. It is assumed that the software will be able to take advantage of the potential performance offered by multicores. The multicore paradigm has indeed a lot of potential but it is important not to neglect sequential programs, which are still prevalent today.

Section 2 emphasizes the importance of accelerating sequential execution in order to overcome Amdahl's law. Consequently, as advocated by some other researchers (e.g., [11, 14, 9, 19]) we propose that future multicores be heterogeneous and dedicate a part of the chip area for fast sequential execution.

In Section 3, we consider two different scaling scenarios, constant power density [1] scaling in the parallel chip area and constant power scaling in the sequential chip area. We show that constant power density scaling and constant power scaling lead to dramatically different situations : constant power scaling allows the clock frequency to be increased across technology generations whereas constant power density scaling strongly limits the extent to which this is possible. A consequence of constant power scaling is that the instantaneous power density doubles on each new technology generations. Activity migration is a possible way to manage very high instantaneous power densities [17, 8].

We explain in Section 4 that, on each technology generation, the sequential chip area must remain approximately constant and the migration interval must be divided by four in order to keep temperature below the limit.

Using simple analytical models, we show in Section 5 that, because the clock cycle duration is unlikely to decrease as quickly as the migration interval, the apparent level-2 (L2) cache size is going to decrease as we decrease the migration interval. To combat this phenomenon, we propose that a certain number of cores on which the execution will migrate next be prepared before becoming active. This means that updates that are done in the local L2 cache are replicated in the L2 caches of soon-to-be-active cores.

[1] In this paper, power density (aka heat flux) is the power consumption per unit of area, in W/m^2.

2 The importance of fast sequential execution

Let us consider a multicore consisting of one fast core and several slow cores. We assume that the number of slow cores doubles on each technology generation and that their clock frequency remains constant. We assume that the clock frequency of the fast core is multiplied by a on each generation, with $a \geq 1$. We model applications as follows. We assume that an application exhibits a periodic pattern, where the period consists of a parallel phase of duration $T_p(g)$ followed by a sequential phase of duration $T_s(g)$. $T_p(g)$ and $T_s(g)$ are in seconds and $g = 0, 1, 2, \cdots$ represents the technology generation (a technology generation is 2 to 3 years). We define the sequential fraction s as

$$s = \frac{T_s(0)}{T_p(0) + T_s(0)}$$

The fast core is used during sequential phases and we assume the performance is proportional to the clock frequency, that is,

$$\frac{T_s(g)}{T_s(0)} = \frac{1}{a^g}$$

We assume that the parallelism of the parallel phase is perfect, that is,

$$\frac{T_p(g)}{T_p(0)} = \frac{1}{2^g}$$

At generation g, the global speedup is

$$\frac{T_p(0) + T_s(0)}{T_p(g) + T_s(g)} = \frac{1}{\frac{1-s}{2^g} + \frac{s}{a^g}}$$

The graph in Figure 1 shows the global speedup as a function of s after $g = 5$ generations for different values of a. So far, most existing applications are on the right side of the graph ($s > 0.1$), i.e., they are either purely sequential or have a small number of parallel threads. Processor makers are currently increasing the number of cores, hoping that more and more applications will move toward the left side of the graph (small s). Nevertheless, many applications will remain on the right side. Increasing the clock frequency of the fast core will allow to obtain substantial speedup for these applications.

3 Impact of technology scaling

Let us consider the following classical formula [2] for the delay of a NOT CMOS logic gate loaded with a capacitance C : [3]

[2] This formula is simplistic but it allows to understand important trends.

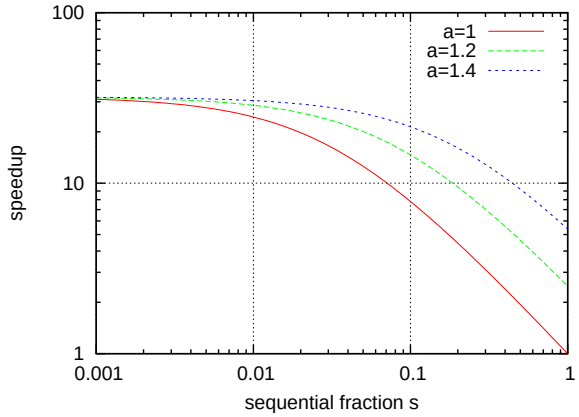


Figure 1: Global speedup as a function of the sequential fraction s after 5 generations ($g = 5$) for different values of the fast-core clock frequency acceleration a.

$$\text{gate delay} = \frac{C \times V_{dd}}{I_{dsat}} = \frac{2}{\mu} \times \left(\frac{L}{W}\right) \times \left(\frac{t_{ox}}{\epsilon_{ox}}\right) \times C \times \frac{V_{dd}}{(V_{dd} - V_t)^2} \quad (1)$$

and the energy per transition

$$\text{energy} = \frac{1}{2} C V_{dd}^2$$

Here, V_{dd} is the supply voltage, V_t is the transistor threshold voltage, I_{dsat} is the transistor saturation current, μ is the charge carrier mobility, L and W are the length and width of the transistor channel, t_{ox} and ϵ_{ox} are the thickness and permittivity of the transistor gate dielectric. In the remaining, we assume that the maximum clock frequency is proportional to the inverse of the gate delay. Leakage power is the main reason why clock frequency has remained stuck at a few gigahertz for several years. Before the 90 nm technology, decreasing all dimensions in the x,y and z directions decreased C and t_{ox}, hence the gate delay and the energy per transition. During the 1990's, the voltages V_{dd} and V_t were scaled down while still allowing a smaller gate delay, as predicted by the classical scaling theory [5]. Because the energy per transition was decreasing faster than the gate delay, it was possible to benefit from the clock frequency boost coming with miniaturization and, at the same time, to enhance the microarchitecture (more pipelining for further clock frequency boost, more transistors crammed on the same chip area) while staying within the limits of the allowed power consumption. But leakage currents were increasing exponentially. Subthreshold leakage increased as V_t became smaller. Because of this, it became difficult to decrease V_t. But a fixed V_t limits the extent to which V_{dd} can be scaled down (cf. formula (1)). As for gate leakage, it increased because the transistor gate dielectric became extremely thin. This has prevented a proper scaling of t_{ox} in the 90 nm and 65 nm technology nodes. Further scaling of the dielectric required the introduction

of new material with a higher ϵ_{ox} than silicon dioxide. Intel is using a new dielectric material in its 45 nm technology [2], which allows a thicker dielectric layer. It took several years for the industry to find a solution because replacing silicon dioxide incurred some difficulties [20], but it is expected that these new materials will allow to continue scaling the gate dielectric, either by scaling t_{ox}, ϵ_{ox} or both.

3.1 Constant power density scaling

Assuming the core area is divided by 2 on each generation, it is possible to put twice as many cores in the same area. But to prevent an increase of power, the power density in each core must not increase. If we assume that the clock frequency is inversely proportional to the gate delay and if we neglect static power, the power density at generation g is proportional to

$$\frac{\text{energy}}{\text{core area} \times \text{gate delay}} \propto V_{dd} \times (V_{dd} - V_t)^2 \times \left(\frac{\epsilon_{ox}}{t_{ox}}\right) \times 2^g \quad (2)$$

Keeping power density constant requires to decrease V_{dd} as g increases. We used formula (2) to determine the value of V_{dd} that keeps power density the same for all g. We assume $V_t = 0.3V$ remains constant and $V_{dd} = 1V$ at $g = 0$. Then we compute the clock frequency from V_{dd} and formula (1). We considered two scenarios : one in which $\epsilon_{ox}/t_{ox} = K_0$ remains constant and one in which $\epsilon_{ox}/t_{ox} = K_0(\sqrt{2})^g$. Figure 2 shows the relative frequency across generations. In the constant dielectric scenario, V_{dd} decreases by $\sim$ 10-16% on each technology generation, while the decrease for the scaled dielectric scenario is $\sim$ 12-20% (V_{dd} decreases more slowly when approaching V_t). Maintaining the dielectric constant prevents the clock frequency to be increased. This is what has happened since the introduction of multicores so far. With a scaled dielectric, it is possible to increase the clock frequency but only up to a certain point. The limitation in frequency comes from our assumption that the threshold voltage remains constant, but more importantly from the constraint that power density must not increase.

In summary, it is possible to double the number of cores on each generation but, if we want to use all these cores simultaneously, the power density constraint limits the extent to which the clock frequency can be increased. In its recent dual-core processors, Intel has introduced a new execution mode that increases the voltage and frequency automatically when the second core is not used. But power density increases quickly with voltage, and the temperature constraint allows only a modest frequency boost. If we want significant sequential speedup in future multicore processors, it will be necessary to have a special core optimized for sequential execution.

3.2 Constant power scaling

Now let us consider a core specialized for sequential execution. For constant power density scaling, we assumed that all the cores are used simultaneously For sequential execution, it is possible to migrate the execution periodically to a different core so that, even though the instantaneous power density increases across technology generations, it is possible to maintain the **time-averaged** power density at a reasonable value [17, 8, 15, 13]. Here, a single core is used at a time. It is assumed that

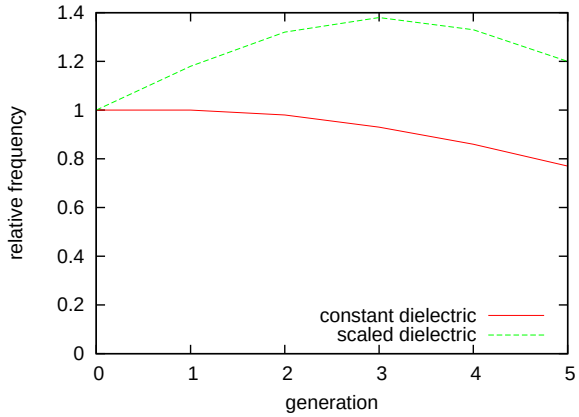


Figure 2: Clock frequency relative to generation 0 when **constant power density** is used. It is assumed that $V_t = 0.3V$ remains constant and $V_{dd} = 1V$ at $g = 0$. The *constant dielectric* scenario assumes that ϵ_{ox}/t_{ox} remains constant and the *scaled dielectric* scenario assumes that ϵ_{ox}/t_{ox} is multiplied by $\sqrt{2}$ on each generation.

the inactive cores are clock-gated and power-gated so that their power consumption is negligible. In this case, we have

$$\text{time-averaged power density} = \frac{\text{core power}}{\text{number of cores} \times \text{core area}}$$

As the core area is divided by 2 on each generation and assuming the power consumption remains constant, we must double the number of core on which we migrate. Hence **the chip area dedicated to sequential execution should remain approximately constant across technology generations**. The power is proportional to

$$\frac{\text{energy}}{\text{gate delay}} \propto V_{dd} \times (V_{dd} - V_t)^2 \times \left(\frac{\epsilon_{ox}}{t_{ox}}\right) \tag{3}$$

If the dielectric is kept constant, it is not necessary to decrease V_{dd} to keep the power constant. On the other hand, if the dielectric is scaled, V_{dd} must be decreased on each generation, but not as much as for constant power density scaling. Figure 3 shows the relative frequency that keeps power constant across generations. For the constant dielectric scenario, V_{dd} is kept equal to $1\,V$ while for the scaled dielectric scenario it decreases by $\sim 8\%$ on each technology generation. Unlike constant power density scaling, constant power scaling allows a significant increase of the clock frequency across generations.

We do not claim that it will be possible to have voltage and frequency as high as what formula (1) says. Formula (1) is simplistic, in particular it ignores short-channel effects. Moreover there

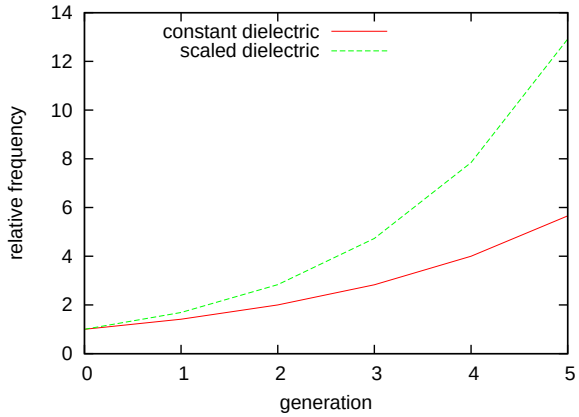


Figure 3: Clock frequency relative to generation 0 when **constant power** scaling is used (activity migration, instantaneous power density doubles on each generation). It is assumed that $V_t = 0.3V$ remains constant and $V_{dd} = 1V$ at $g = 0$. The *constant dielectric* scenario assumes that ϵ_{ox}/t_{ox} remains constant and the *scaled dielectric* scenario assumes that ϵ_{ox}/t_{ox} is multiplied by $\sqrt{2}$ on each generation.

are some other constraints that impact the clock frequency [3] and several problems to solve [20, 18, 6]. Figures 2 and 3 must not be taken as quantitative evidences but as an illustration of the large qualitative difference between constant power density scaling and constant power scaling.

4 Periodic activity migration under constant power scaling

We consider the example of chip layout depicted on Figure 4. In this example, the area dedicated to sequential execution is a square of side $8\,mm$ taking one fourth of the chip area. Half of the sequential area is occupied by fast cores, the other half by the L2 cache. Let us assume that the **time-averaged** power density in fast cores is $q_{avg} = 1\,W/mm^2$ and that the time-averaged power density in the caches is $0.1\,W/mm^2$. We assume that, when not used, the parallel area is clock-gated and power-gated so that its power consumption is negligible. Moreover, let us assume that the silicon die is $0.5\,mm$ (thermal conductivity $110\,W/mK$), that it is attached to a heatsink modeled as a square copper plate $3\,mm$ thick and $5\,cm$ wide ($400\,W/mK$), and that there is some interface material between the die and the plate, $50\,\mu m$ thick and with thermal conductivity $4\,W/mK$. We use the ATMI model to obtain an estimate of the maximum temperature [1]. If we take 40°C as the maximum ambient temperature, and assuming the heatsink thermal resistance is $0.4\,K/W$, the maximum temperature is roughly $T_{max} = 80$ °C. Let N be the number of sequential cores. When

[3] The clock frequency targets on the ITRS 2007 are based on a 17%-per-year increase of the transistor speed [7]. Nevertheless, the ITRS 2007 assumes constant instantaneous power density [6].

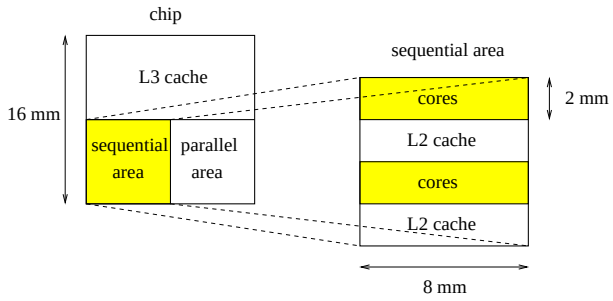


Figure 4: Example of chip layout.

a core is active, its instantaneous power density is q_{on}. When a core is inactive, it is clock-gated and power-gated. We assume that clock gating and power gating are perfect so that only the active core dissipates some power. If the execution migrates frequently enough, the instantaneous power density can be as high as

$$q_{on} = N \times q_{avg}$$

that is, $4\,W/mm^2$ with 4 cores, $8\,W/mm^2$ with 8 cores, and so on. The question is : how frequently must we migrate the execution ? To understand the problem, it is useful to look at how quickly temperature increases when, starting at the ambient temperature, a power density q_{on} is suddenly applied at $t = 0$. In this situation, for small time values t, temperature increases as follows [16, 12] :

$$\theta(t) \propto q_{on}\sqrt{t}$$

When we migrate the execution from a core A to a core B, the temperature in B increases. After we migrate from B to C, temperature in B decreases. So we have a temperature oscillation in each core. The amplitude of this oscillation is roughly proportional to $q_{on}\sqrt{t_{on}}$, where t_{on} is the time spent on each core before we migrate to another core. If we keep the migration frequency constant across technology generations, each time we double the power density q_{on}, we double the amplitude of the temperature oscillation. So even though the time-averaged temperature does not exceed T_{max}, the peak temperature may be much higher than T_{max}. If we want the peak temperature to stay constant, we must divide t_{on} by 4 when we double power density q_{on}.

Figure 5 shows an example layout with 4 cores and with 8 cores, both occupying the same area. Cores are numbered from 0 to $N - 1$. We stay on core j for a fixed time that we call the *migration interval*, and then we migrate the execution to core $(j + 1) \mod N$. Figure 6 shows the peak temperature as a function of the migration interval. The curves were obtained with ATMI [1]. When the migration interval is short, the peak temperature is close to the time-averaged temperature, i.e., 80 °C. As the migration interval becomes longer, the peak temperature increases above 80 °C. With 4 cores and $q_{on} = 4\,W/mm^2$, as long as the migration interval does not exceed a few

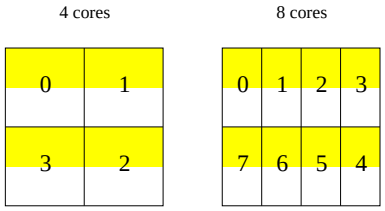


Figure 5: Example layout with 4 and 8 cores in the sequential area.

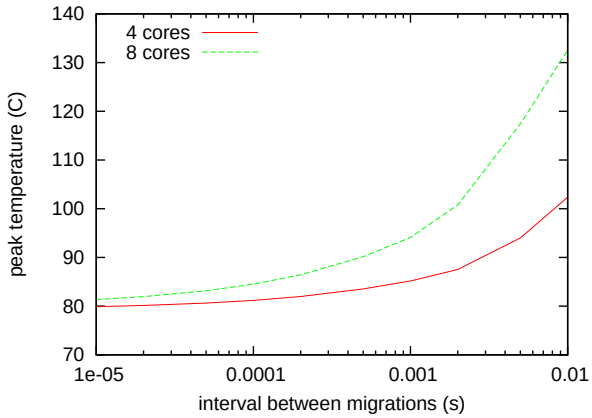


Figure 6: Peak temperature as function of the migration interval, for 4 cores ($q_{on} = 4\,W/mm^2$) and for 8 cores ($q_{on} = 8\,W/mm^2$).

hundreds of microseconds, the peak temperature stays close to 80 °C. If we set the migration interval at 1 millisecond, the peak temperature is about 85 °C. But temperature increases quickly in the millisecond range [4]. For intervals larger than 10 milliseconds, we exceed 100°C. With 8 cores and $q_{on} = 8\,W/mm^2$, we must have a shorter migration interval if we do not want temperature to increase. For example if we want the peak temperature to stay below 85 °C, the migration interval must not exceed 200 microseconds.

It is possible to decrease temperature by using a heatsink with a smaller thermal resistance (i.e., larger, noisier, or more expensive heatsink). This is possible in server and desktop machines, but this is more difficult in notebooks and mobile devices due to size and weight constraints. Anyway, when cores become very small and lead to tiny hot spots, the heatsink contributes little to the overall junction-to-ambient thermal resistance, which is dominated by the interface material.

[4] A rule of thumb is that every 10 °C increase in temperature halves the processor mean-time-to-failure.

5 Cache-induced migration penalty

For activity migration to be as effective as possible, we must minimize the power consumed by inactive cores. This means that clock gating and power gating must be applied in inactive cores. Power gating with sleep transistors may incur a penalty of a few microseconds [21]. In order to hide this penalty, the next core on which the execution will migrate should be woken up ahead of the migration time. Power gating of microarchitected tables means that the information stored in these tables is either lost (full power gating) or not updated (drowsy mode [10]). It was shown by Constantinou et al. that flushing small microarchitected tables, in particular level-1 (L1) caches, incurs little penalty [4]. Flushing the branch predictors, on the other hand, may incur some penalty. Constantinou has shown that maintaining the branch predictors drowsy incurs little performance penalty. However, if the total static power consumed in drowsy mode is still too important, another possibility is to prepare the branch predictor of the next core on which we will migrate by copying the predictor content. This copying does not need to be exact, it may be approximate and/or partial. Actually, the most problematic microarchitected table is the local L2 cache, because the L2 cache is relatively large and because its content must be exact. If the L2 cache uses a write-back policy, a possibility is to allow the active core to access the remote L2 caches of inactive cores. The L2 tags cannot be maintained drowsy because we must be able to invalidate blocks that are written. But maintaining the L2 tags active and the L2 data drowsy will consume significant power, both static (the total L2 capacity increases with the number of cores) and dynamic (on each write, an invalidate request is sent to all the inactive cores). Under constant power scaling, the power overhead spent in inactive cores will decrease the power budget for the active core. To avoid this, a solution could be to flush dirty L2 blocks to the level-3 (L3) cache before migrating. But this flushing may take thousands of cycles. The solution we advocate to keep the migration penalty small is to have a write-back L1 and a write-through L2. A write-back L1 permits decreasing the write traffic when the write locality is high. When a dirty block is evicted from the L1, it is sent to the L2 and L3 caches. Before migrating from core A to core B, the L1 of core A is flushed by sending dirty blocks to the L3, and the L1 and L2 caches on core A are turned off. As the L1 is small compared to the L2, flushing the L1 can be done relatively quickly. However, after the migration, the L1 and L2 caches on core B are empty. The L1 cache warms up quickly [4]. But the L2 may incur a large number of cold-start misses. [5] To solve this problem, a solution is to prepare a certain number of soon-to-be-active cores. Upon a L2 miss, we store the missing block not only in the local L2 but also in the L2 of the next α cores, where α is much smaller than the total number of fast cores. Also, each time a victimized block updates the local L2 (because an older copy of the block is in the L2), it updates the L2 of the next α cores. Figure 7 shows an example with $\alpha = 2$. We propose in this section an analytical model to better understand this situation.

[5] Constantinou has shown that when a drowsy mode is used for the branch predictor, the migration penalty is mainly due to cold-start L2 misses [4].

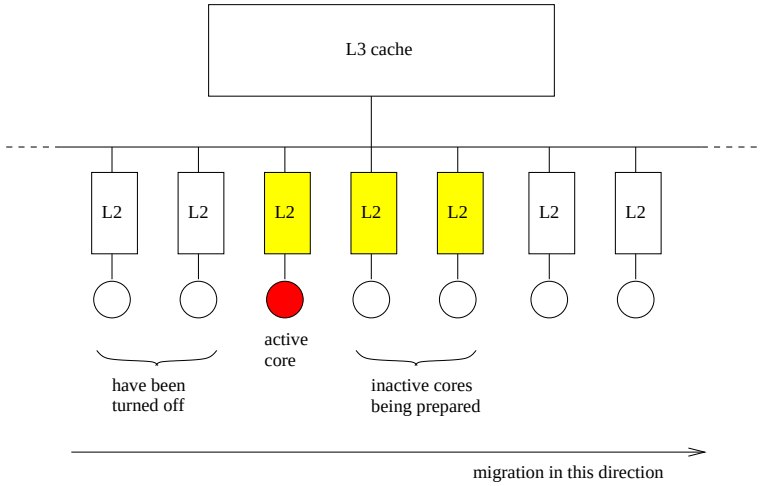


Figure 7: Execution migration : on this example, $\alpha = 2$ cores are being prepared.

5.1 Analytical model based on reuse distances

When an application executes, it accesses the cache blocks $a_1, a_2, a_3, \cdots, a_N$ where a_i is the address of the cache block accessed by the i^{th} reference. We define the *reuse distance* $d_i > 0$ of the i^{th} reference as the number of references that have occurred since the last reference to a_i, that is,

$$\begin{array}{ll} a_i = a_{i-d_i} & \\ a_i \neq a_j & \text{for } j \in]i-d_i, i[\end{array}$$

We define the *reuse distance profile* (RDP) $P(d) \in [0,1]$ as follows

$$P(d) = \frac{\left|\{i \in [1,N]/d_i > d\}\right|}{N}$$

Note that $P(0) = 1$ and $P(d)$ is a decreasing function. The miss ratio of a fully-associative cache with a LRU replacement policy can be obtained from the RDP in a straightforward way. Let us define

$$S(k) = \sum_{d=0}^{k-1} P(d)$$

Note that $S(k)$ is concave increasing. We show in the appendix that when $N \gg k$, $S(k)$ is the average number of unique references in a window of k consecutive references. Consequently, when the number of unique references in a window of k consecutive references is stable and close to its average value, the miss ratio of a cache of size $S(k)$ blocks is approximately equal to $P(k)$.

5.2 Apparent L2 size vs. actual L2 size

Let F be the clock frequency in Hertz, T the migration interval in seconds, μ the average number of L1 misses per instruction, N the number of L1 misses per interval, and X the number of instructions executed per cycle (IPC) when $T = \infty$. Our goal is to have the actual IPC close to X. If this is realized, we have

$$N = \mu FTX$$

Assuming the L2 cache is large enough, when the execution arrives on a core, this core's L2 has been prepared with the last αN references. It means that the average L2 miss ratio m is

$$m(\alpha) = \frac{S((\alpha+1)N) - S(\alpha N)}{N} \leq P(\alpha N)$$

Let W be the actual L2 size in number of blocks. We define the apparent L2 size W_α as the value such that

$$P(S^{-1}(W_\alpha) = m(\alpha)$$

We have

$$W_\alpha \geq S(\alpha N)$$

Ideally, we would like the apparent L2 size to be no less than the actual L2 size W. This way, the contribution of the L2 cache to the migration penalty is null. This can be achieved if we take α such that

$$S(\alpha N) \geq W$$

That is,

$$\alpha \geq \alpha_w$$

with α_w defined as

$$\alpha_w = \frac{n_w}{N} = \frac{n_w}{\mu FTX} \tag{4}$$

where $n_w = S^{-1}(W)$. We measured $S^{-1}(W)$ on some of the SPEC CPU2006 benchmarks and we found $S^{-1}(8192) < 20000$ for 14 out of 20 benchmarks and $S^{-1}(16384) < 41000$ for 13 out of 20 benchmarks.

For example, let us assume $W = 8192$ blocks (512-Kbyte cache with 64-byte blocks) and let us dimension α_w with $n_w = 20000$. Table 1 shows two examples, one where the clock frequency increases according to the constant dielectric scenario, and the other according to the scaled dielectric scenario (see Figure 3). We have assumed $X = 1$ instruction per cycle and $\mu = 0.01$ L1 miss per instruction. The value of α_w increases across technology generations, and it increases faster than the number of cores. As can be seen, scaling the dielectric is better than keeping it constant, because

this allows a higher clock frequency but also because α_w is smaller. With 2 and 4 cores, α is small and it may not be necessary to prepare any core. Notice that when $N > n_w$ we have

$$m(0) = \frac{S(N)}{N} \leq \frac{W + P(n_w)(N - n_w)}{N - n_w} = P(n_w) + \frac{W}{N - n_w}$$

hence the miss ratio overhead compared to no migration is bounded as follows

$$m(0) - P(n_w) \leq \frac{W}{N - n_w} = \frac{\alpha_w}{1 - \alpha_w} \frac{W}{n_w}$$

For example with 4 cores and constant dielectric, $\alpha_w = 0.19$ and we have $m(0) - P(n_w) \leq 0.1$. It means that not preparing any core generates at most 1 extra $L2$ miss every $\frac{1}{0.1 \times \mu} = 1000$ instructions in this case. But with 8 cores and above, it is necessary to prepare some cores.

Applications with different characteristics. One question to consider is whether the value of α should be fixed or should be allowed to vary dynamically. For instance, we may set α at a fixed value based on the assumption $X = 1$ and $\mu = 0.01$, as we did in Table 1. For applications such that $\mu X > 0.01$, the apparent L2 size saturates at the actual L2 size. But applications with $\mu X < 0.01$ may suffer from the migration penalty, as the apparent L2 size is less than the actual L2 size. μX may be small either because X is small or because μ is small. If X is small because the L2 miss ratio is high, migration have little impact, they do not degrade a situation that is already bad. In the case when μ is very small, it means that L1 misses are infrequent, so a high L2 miss ratio is not necessarily penalizing. Instead of fixing the value of α, another possibility is to have some hardware to evaluate μX and to change the value of α dynamically before migrating. Setting α dynamically may be interesting to optimize the tradeoff between performance and power consumption.

What about applications with a very large $S^{-1}(W)$? When $S^{-1}(W)$ is much larger than n_w, the fixed α obtained under the assumption $S^{-1}(W) = n_w$ yields an apparent L2 size smaller than W. But when $S^{-1}(W)$ is large, it means that the first derivative of $P(k)$ is small (in absolute value) and that a smaller L2 is unlikely to increase the miss ratio significantly.

6 Conclusion

As Moore's law continues and allows an exponential increase of the number of cores, it is important to use a part of the silicon area to implement a fast sequential processor, leading to a heterogeneous multicore. A possible way to increase the performance of the fast core is to increase the clock frequency. Provided the power consumption of the fast core does not increase across technology generations, the increase of the instantaneous power density can be managed by doing periodic activity migration on several copies of the fast core. Under constant power scaling and assuming a single fast core is active at any time, the number of fast cores should double on each technology generations. In other words, the chip area dedicated to sequential execution should remain approximately constant. However, to prevent temperature from becoming too high, the migration interval must be divided by

generation	0	1	2	3	4
number of fast cores	2	4	8	16	32
migration interval (s)	10^{-2}	2.5×10^{-3}	6.25×10^{-4}	1.56×10^{-4}	3.91×10^{-5}
constant dielectric scenario					
clock frequency (Hz)	3×10^9	4.2×10^9	6×10^9	8.4×10^9	1.2×10^{10}
α_w	0.07	0.19	0.54	1.52	4.32
scaled dielectric scenario					
clock frequency (Hz)	3×10^9	5×10^9	8.5×10^9	1.4×10^{10}	2.4×10^{10}
α_w	0.07	0.16	0.38	0.90	2.14

Table 1: Value of α_w as defined by formula (4), assuming $n_w = 20000$, $X = 1$ and $\mu = 0.01$. The migration interval is divided by 4 on each technology generation.

four on each technology generation. As the migration interval decreases faster than the clock cycle, the migration penalty may become short enough to hurt the performance. The level-2 cache is one of the biggest potential contributor to the migration penalty, and this problem will become worse as the migration penalty decreases. We propose to solve this problem by preparing a certain number of future cores before they become active.

We believe that the semiconductor industry should take into account core heterogeneity and activity migration in its roadmaps, otherwise some opportunities may be missed. But it is important that the microarchitects bring convincing evidences that activity migration is a viable long-term solution. Our analytical model suggest this is the case, up to a certain point. Detailed simulations are necessary to obtain more certainty.

Appendix

We want to show that the average number of unique references in a window of k consecutive references is $S(k)$, where $S(k)$ is defined as

$$S(k) = \sum_{d=0}^{k-1} P(d)$$

and where $P(d)$ is the frequency of references whose reuse distance is greater than d, that is,

$$P(d) = \frac{\left|\{i \in [1, N] / d_i > d\}\right|}{N} = \frac{1}{N} \sum_{i=1}^{N} H(d_i - d)$$

where $H(x)$ is

$$H(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

Let $u(n,k)$ be the number of unique references in the window of k references $a_n, a_{n+1}, \cdots, a_{n+k-1}$. The number $u(n,k)$ can obtained as

$$u(n,k) = \sum_{j=0}^{k-1} H(d_{n+j} - j)$$

The average number $\overline{u}(k)$ of unique references in a window of k consecutive references is

$$\begin{aligned}
\overline{u}(k) &= \frac{1}{N-k+1} \sum_{n=1}^{N-k+1} u(n,k) \\
&= \frac{1}{N-k+1} \sum_{n=1}^{N-k+1} \Big(\sum_{j=0}^{k-1} H(d_{n+j} - j) \Big) \\
&= \frac{1}{N-k+1} \sum_{j=0}^{k-1} \Big(\sum_{n=1}^{N-k+1} H(d_{n+j} - j) \Big) \\
&= \frac{1}{N-k+1} \sum_{j=0}^{k-1} \Big(\sum_{i=j+1}^{j+N-k+1} H(d_i - j) \Big) \\
&\approx \frac{1}{N} \sum_{j=0}^{k-1} \sum_{i=1}^{N} H(d_i - j) \\
&= \sum_{j=0}^{k-1} P(j) \\
&= S(k)
\end{aligned}$$

where the approximation is based on the assumption that N is much greater than k.

References

[1] ATMI: analytical model of temperature in microprocessors. http://www.irisa.fr/caps/projects/ATMI.

[2] C. Auth, M. Buehler, A. Cappellani, C. h. Choi, G. Ding, W. Han, S. Joshi, B. McIntyre, M. Prince, P. Ranade, J. Sandford, and C. Thomas. 45nm high-k+metal gate strain-enhanced transistors. *Intel Technology Journal*, 12(2), June 2008.

[3] H.B. Bakoglu. *Circuits, interconnections, and packaging for VLSI*. Addison-Wesley, 1990.

[4] T. Constantinou, Y. Sazeides, P. Michaud, D. Fetis, and A. Seznec. Performance implications of single thread migration on a chip multi core. *ACM SIGARCH Computer Architecture News*, 33(4):80–91, November 2005.

[5] R.H. Dennard, F.H. Gaensslen, V.L. Rideout, E. Bassous, and A.R. Leblanc. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, October 1974.

[6] International Technology Roadmap for Semiconductors. Executive summary, 2007. http://www.itrs.net.

[7] International Technology Roadmap for Semiconductors. Process integration, devices, and structures, 2007. http://www.itrs.net.

[8] S. Heo, K. Barr, and K. Asanovic. Reducing power density through activity migration. In *Proceedings of the International Symposium on Low Power Electronics and Design*, 2003.

[9] M.D. Hill and M.R. Marty. Amdahl's law in the multicore era. *IEEE Computer*, 41(7):33–38, July 2008.

[10] N.S. Kim, K. Flautner, D. Blaauw, and T. Mudge. Circuit and microarchitectural techniques for reducing cache leakage power. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(2):167–184, January 2004.

[11] R. Kumar, D.M. Tullsen, N.P. Jouppi, and P. Ranganathan. Heterogeneous chip multiprocessors. *IEEE Computer*, 38(11):32–38, November 2005.

[12] P. Michaud, Y. Sazeides, A. Seznec, T. Constantinou, and D. Fetis. An analytical model of temperature in microprocessors. Research report RR-5744, INRIA, November 2005.

[13] P. Michaud, Y. Sazeides, A. Seznec, T. Constantinou, and D. Fetis. A study of thread migration in temperature-constrained multicores. *ACM Transactions on Architecture and Code Optimization*, 4(2), June 2007.

[14] T.Y. Morad, U.C. Weiser, A. Kolodny, M. Valero, and E. Ayguadé. Performance, power efficiency and scalability of asymmetric cluster chip multiprocessors. *IEEE Computer Architecture Letters*, 5(1), June 2006.

[15] M.D. Powell, M. Gomaa, and T.N. Vijaykumar. Heat-and-run: leveraging SMT and CMP to manage power density through the operating system. In *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2004.

[16] N. Rinaldi. On the modeling of the transient thermal behavior of semiconductor devices. *IEEE Transactions on Electron Devices*, 48(12):2796–2802, December 2001.

[17] K. Skadron, M.R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *Proceedings of the 30th Annual International Symposium on Computer Architecture*, 2003.

[18] T. Skotnicki, J.A. Hutchby, T.-S. King, H.-S. P. Wong, and F. Boeuf. The end of CMOS scaling : toward the introduction of new materials and structural changes to improve MOSFET performance. *IEEE Circuits and Devices Magazine*, 21(1):16–26, January 2005.

[19] M.A. Suleman, O. Mutlu, M. Qureshi, and Y.N. Patt. An asymmetric multi-core architecture for accelerating critical sections. Technical report TR-HPS-2008-003, University of Texas at Austin, September 2008.

[20] S.E. Thompson, R.S. Chau, T. Ghani, K. Mistry, S. Tyagi, and M.T. Bohr. In search of "forever", continued transistor scaling one new material at a time. *IEEE Transactions on semiconductor manufacturing*, 18(1):26–36, February 2005.

[21] J.W. Tschanz, S.G. Narendra, Y. Ye, B.A. Bloechel, S. Borkar, and V. De. Dynamic sleep transistor and body bias for active leakage power control of microprocessors. *IEEE Journal of Solid-State Circuits*, 38(11):1838–1845, November 2003.