Optimization of Asymmetric and Heterogeneous MultiCore

Amir Morad, Tomer Morad, Leonid Yavits and Ran Ginosar

Abstract—An analytical optimization model, based on Lagrange multipliers, allocates constrained area and power among the cores of either asymmetric or heterogeneous multicore so as to optimize execution time. While in the asymmetric multicore each one of the concurrent workload task may execute on any core, concurrent tasks are pre-assigned to cores in the heterogeneous case. The performance of each core is modeled as a function of its area and the power allocated to it. In an optimal solution, all utilized cores must have the same execution time and the same first area- and power-derivatives of the execution time function. The model is applied to balanced and imbalanced asymmetric and heterogeneous multicores, showing several non-intuitive outcomes. When more area and power are made available to an asymmetric multicore, the stronger cores are allocated larger area, up to eliminating the weakest cores, but the weaker cores are allocated more power. In heterogeneous multicores the weaker cores are allocated larger area and more power.

Index Terms—Chip Multiprocessors, Modeling of computer architecture

1 Introduction

LARGE scale multicore may be composed of several asymmetric or heterogeneous cores working in parallel. It is up to the system architect to efficiently allocate the resources among the cores, while complying with physical constraints of the design, such as area and power, as well as the specifics of the workload.

In an asymmetric multicore architecture, all cores share the same instruction set architecture (ISA) but may differ in performance, clock frequency, cache size and other micro-architectural characteristics. Heterogeneous architecture, on the other hand, allows different functionality and different ISA in the cores. In an asymmetric multicore architecture, each task can be executed on any one of the cores, whereas in a heterogeneous architecture, tasks can typically execute only on designated cores.

Several studies have used analytic models to derive optimal resource allocation of multiprocessors. Cassidy *et al.* [2] have optimized processor area, L2 cache area and the number of cores for an area-constrained symmetric multicore using Lagrange multipliers [16]. Elyada *et al.* [3] have considered a multiprocessor with unknown workload and attempted to dynamically set frequency-voltage work-points for each core, with a goal to minimize a defined energy-performance criterion. Yavits *et al.* [12] modeled the serial-to-parallel synchronization impact on multicore performance. Zidenberg *et al.* [20] presented the MultiAmdahl model that considered the implications of accelerating portions of the workload on the overall execution. Wentzlaff *et al.* [5] introduced an analytic model to study the tradeoffs of utilizing increased chip area for larger caches versus more cores. Khan *et al.* [10] presented a method for performance maximization of a 3D cache-stacked multicore system keeping the temperature under a given limit while by assigning the clock frequencies and number of cache banks to each core. Yavits *et al.* [12] presented a closed form analytical solution for optimizing the CMP cache hierarchy and optimally allocating area among hierarchy levels under constrained resources. Blem *et al.* [18] modeled a multicore with cache and memory bandwidth constraints. Morad *et al.* [4] have presented the Generalized MultiAmdahl model (Figure 1) that minimized sequential execution time of a heterogeneous set of accelerators by optimally selecting which accelerators to allocate and what area to assign to the allocated accelerators.

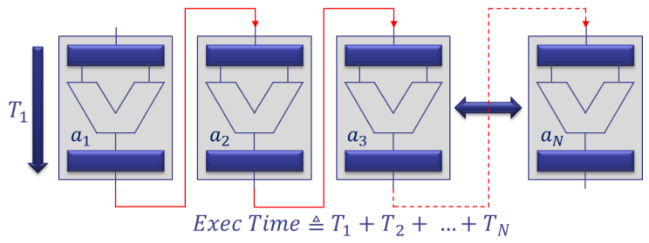

Figure 1. Generalized MultiAmdahl [4]: Optimizing SoC resource allocation executing sequential series of heterogeneous workload segments

While the Generalized MultiAmdahl addresses a series of heterogeneous workload segments that are not concurrent (only one core operates at a time), in this paper we address the following complementary question: Given a multicore and a workload consisting of concurrent tasks and resource constraints (Figure 2), what is the optimal selection of a subset of the available cores and what is the

- *Amir Morad (*), E-mail: amirm@tx.technion.ac.il.*
- *Tomer Morad (*), E-mail: tomerm@tx.technion.ac.il.*
- *Leonid Yavits (*), E-mail: yavits@tx.technion.ac.il.*
- *Ran Ginosar (*), E-mail: ran@ee.technion.ac.il.*

(*) *Authors are with the Department of Electrical Engineering, Technion-Israel Institute of Technology, Haifa 32000, Israel.*

optimal resource allocation among them.

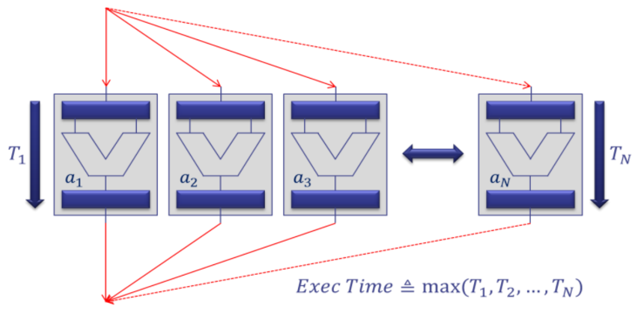

Figure 2. Optimizing multicore resource allocation executing concurrent series of workload segments

At the optimal resource allocation point, all cores reach an equilibrium state in which if some area is taken from one core and given to another, the overall runtime increases. While searching for such optimal point, one must take into account all core' characteristics, total area constraints and workload specifics. Thus, any selection criteria based on performance threshold or otherwise electing a subset of the cores may yield only sub-optimal results. Hence, the key contribution of this paper is to enable the multicore architect to select an optimal subset of the cores and allocate resources among them, avoiding having to explore the design space in an iterative ad-hoc fashion, and thus leading to a comprehensive understanding of the design space.

The rest of this paper is organized as follows. Section 2 proposes and investigates an analytical model of the area constrained asymmetric and heterogeneous multi-core processors. Power-law acceleration functions are utilized to optimize a dual core processor based on the proposed analytical model. The insights derived from the dual core analytical model are extended to a multi-core implementation. The model is demonstrated by an example optimization of an asymmetric and a heterogeneous quad-core processors. Section 3 proposes and investigates an analytical model of area and power constrained asymmetric multi-core processors, and offers an area/power constrained optimization example of a quad-core processor and a comparison of the results versus an only-area constrained optimization. Section 4 summarizes and concludes the paper.

2 Area Optimization

In this section we propose and investigate an analytical model of area-constrained asymmetric and heterogeneous multi-core processors.

2.1 Area constrained multicore model

Consider a workload consisting of M concurrent tasks. Each task i of the workload requires time t_i to execute on a *reference processor* of 1 Instruction Per Second. Further, consider a multicore architecture consisting of N available cores. Each task i can be accelerated by any one of N available cores. The performance of core j as a function of its area a_j is $Perf_j(a_j)$, relative to the performance on the reference processor. The acceleration function $f_j(a_j)$ represents the inverted performance of core j :

$$f_j(a_j) = \frac{1}{Perf_j(a_j)} \tag{1}$$

The runtime of the ith task running on core j having area a_j is thus $f_j(a_j) \cdot t_i$. The performance of a core increases when additional area resources are assigned to it. Therefore, the acceleration functions $f_j(a_j)$ are strictly decreasing. We assume that the acceleration functions $f_j(a_j)$ are convex and are continuously differentiable.

Given the core areas $A = \{a_1, \dots, a_N\}$, core j execution time of the tasks assigned to it is thus:

$$T_j \triangleq \sum\nolimits_{i=1}^{i=M} f_j(a_j)\, b_{i,j} t_i = f_j(a_j) \sum\nolimits_{i=1}^{i=M} b_{i,j} t_i \tag{2}$$

Where, at the optimal point:

$$b_{i,j} = \begin{cases} 1: & i^{th}\ task\ is\ assigned\ to\ j^{th}\ core \\ 0: & otherwise \end{cases} \tag{3}$$

under the constraint that each task is assigned to one core. The area of core j, namely a_j, may be 0 or positive, and the area assigned to all cores is bounded by total SoC area. Let's assume we are provided with N cores such that:

$$\forall j,k, 1 \le j,k \le N \quad \text{and} \quad j \ne k \qquad f_j(a) \ne f_k(a) \tag{4}$$

The assumption implies that even if two different cores are assigned the same area, their performance is different. Since any task can be executed on any core, the research question becomes: *which subset of the cores should be integrated to achieve optimal execution time, and what is the optimal task and resource (area) allocation among them?*

In a similar manner to [4], in this paper we choose to concentrate on multicore resource assignment while ignoring the effects of synchronization and core to core communication [11]. Further, while considering asymmetric architectures, we focus on homogenous tasks and exclude task heterogeneity (for example, certain tasks may benefit from larger cache sizes, others may benefit from larger branch prediction buffer, etc.). Thus, we assume that tasks runtime depends only on the core's speedup function at its designated area. The first step to resolve this optimization problem is to define the function we wish to minimize. In our case it is the total execution time of the multicore, T, determined by the last core to finish executing its allocated tasks:

$$\begin{aligned} T &\triangleq max(T_1, T_2, \dots, T_N) \\ &= max(T_N, max(T_{N-1}, \dots, max(T_2, T_1))) \end{aligned} \tag{5}$$

Next, we define the optimization constraints, as follows:

A. Total area constraint

The combined area of all cores is less than or equal to A_{Total}:

$$\sum\nolimits_{j=1}^{j=N} a_j \le A_{Total} \tag{6}$$

where A_{Total} is the total area assigned to the chip.

B. *Task per core constraint*

Each task is assigned to a single core, thus the following M constraints (collectively implying that the sum of all assigned tasks equals M):

$$\forall i, \quad \sum_{j=1}^{N} b_{i,j} = 1 \quad \rightarrow \quad \sum_{i=1}^{M}\sum_{j=1}^{N} b_{i,j} = M \tag{7}$$

C. *General task assignment constraint*

This constraint assures that, at the optimal point, $b_{i,j}$ is either 1 (assigned) or 0 (unassigned):

$$\forall i,j: \quad b_{i,j}(b_{i,j}-1) = 0 \tag{8}$$

The Lagrange optimization of the cost function (5) subject to constraint (6), is provided in Appendix A. The conclusion is as follows. For all utilized cores, $\forall j,k$ the following conditions hold at the optimal point:

A. First Optimality Condition: *All utilized cores must have the same runtime:*

$$f_j(a_j)\sum_{i=1}^{i=M} t_i b_{i,j} = f_k(a_k)\sum_{i=1}^{i=M} t_i b_{i,k} \tag{9}$$

B. Second Optimality Condition: *The first derivative of the execution time function across all utilized cores must be equal:*

$$f_j'(a_j)\sum_{i=1}^{i=M} t_i b_{i,j} = f_k'(a_k)\sum_{i=1}^{i=M} t_i b_{i,k} \tag{10}$$

Actual solutions that satisfy (9) and (10) may be found, e.g., by numerical Lagrange solvers [1].

2.2 Power-law acceleration function

In this subsection, we show that when the acceleration functions are expressed as power laws, the above optimality conditions hold, and consequently there exists an area allocation that results in optimal execution time.

Following Pollack's rule [8], [19] we express the acceleration function of a processing core as a power law:

$$f_{Core}(a_{Core}) = \frac{1}{{a_{Core}}^{\beta}} \tag{11}$$

where β typically varies from 0.3 to 0.7 [18]. We also assume that coefficients translating from area to performance units have been scaled to unity.

We now prove that a resource allocation of power-law based multicore as provided by (11), is also the solution of (10). To that end, consider a workload comprising multiple tasks, each incurring an execution time t_i on a reference processor. For simplicity, the target architecture may use up to two cores. The generic case power-law acceleration functions are:

$$f_1(a_1) = a_1^{-\beta_1}, \qquad f_2(a_2) = a_2^{-\beta_2} \tag{12}$$

where the exponents β_i's are constants. Following the first rule of optimality:

$$a_1^{-\beta_1}\sum_{i=1}^{i=M} t_i b_{i,1} = a_2^{-\beta_2}\sum_{i=1}^{i=M} t_i b_{i,2} \tag{13}$$

Extracting a_2:

$$a_2 = a_1^{\frac{\beta_1}{\beta_2}}\left(\frac{\sum_{i=1}^{i=M} t_i b_{i,2}}{\sum_{i=1}^{i=M} t_i b_{i,1}}\right)^{\frac{1}{\beta_2}} \tag{14}$$

Testing the equality of the second rule of optimality (developing equation (55) assuming $a_2 = A_{Total} - a_1$):

$$\frac{\partial[max(T_1,\dots,T_N)]}{\partial a_j} + \lambda\frac{\partial[a_1 + (A_{Total}-a_1)]}{\partial a_j} = \frac{\partial[max(T_1,T_2,\dots,T_N)]}{\partial a_j} = 0 \tag{15}$$

Following (67) at $T_1 = T_2$:

$$\frac{\partial[T_1]}{\partial a_1} = -\frac{\partial[T_2]}{\partial a_1} = \frac{\partial[T_2]}{\partial a_2}\frac{\partial a_2}{\partial a_1} \tag{16}$$

Equation (15) proves that for a dual-core processors where each core is described by a power law acceleration function, there exist an optimal area allocation, and at the optimum, equations (9) and (10) hold.

2.3 Optimizing a dual-core processor

We proceed to derive the optimal resource allocation for a dual core processor, employing power law acceleration functions. For convenience, we express a_2 as function of a_1:

$$a_2 = va_1 \;\rightarrow\; f_2(a_2) = (va_1)^{-\beta_2} \tag{17}$$

where following (14), at the optimal point, the coefficient v can be written as:

$$v = \left(\frac{\sum_{i=1}^{i=M} t_i b_{i,2}}{\sum_{i=1}^{i=M} t_i b_{i,1}}\cdot\frac{a_1^{\beta_1}}{a_1^{\beta_2}}\right)^{\frac{1}{\beta_2}} \tag{18}$$

Finally, following (6):

$$a_1 + a_2 = a_1 + va_1 = A_{Total} \tag{19}$$

Thus:

$$a_1 = \frac{A_{Total}}{v+1}, \qquad a_2 = va_1 \tag{20}$$

Note that in particular, for a balanced exponent ($\beta_1 = \beta_2$), v is a constant and the optimization process yields an explicit solution of area allocation, without the need for numerical solvers. We now consider the following scenarios of balanced and imbalanced exponents:

2.3.1 *Asymmetric dual core with imbalanced β*

In this scenario, the acceleration functions have different exponents ($\beta_1 \neq \beta_2$). From this point onward, the core with the larger exponent shall be referred to as the "*stronger*" core, and the other one shall be referred to as the "*weaker*" core. Consider a design process where we progressively increase the total multicore area A_{Total}, and re-compute optimal area allocation in each step. Initially, both cores are utilized and satisfy (9). As the total multicore area is increased, e.g. from A_{Total} to $A_{Total} + \varepsilon$, one of two possibilities takes place:

- The set of tasks previously assigned to each core remains unchanged, and the area increment is allocated mostly to the weaker core and, to a lesser degree, to the stronger core, so as to maintain equal run times.
- One or more of the tasks previously assigned to the weaker core are shifted to the stronger core, the entire area increment is allocated to the stronger core, and in addition some area is taken away from the weaker core and is also added to the stronger core.

Eventually the stronger core is allocated the entire area and the entire workload, and the weaker core is eliminated. To emphasize this point, consider the following inequality. The left hand side represents the execution time of a dual core processor where each core executes a single task. The right hand side shows the execution time of a single core executing the entire workload:

$$a_1^{-\beta_1} t_1 = a_2^{-\beta_2} t_2, \qquad a_2 = A_{Total} - a_1$$
$$a_1^{-\beta_1} t_1 = (A_{Total} - a_1)^{-\beta_2} t_2 \geq A_{Total}{}^{-\beta_2}(t_1 + t_2) \tag{21}$$

We assume that core 2 is the stronger one ($\beta_1 < \beta_2$), and seek the equality point beyond which allocating the entire area (and both tasks) to the stronger core provides better performance than keeping both cores, each executing its respective task. Solving the inequality yields:

$$a_1 \geq ? A_{Total} \frac{C-1}{C} \tag{22}$$

where

$$C = \sqrt[\beta_2]{\frac{t_1 + t_2}{t_2}} \tag{23}$$

If inequality (22) holds, namely for values of a_1 larger than (22), allocating the entire area (and the entire workload) to the (stronger) core 2 produces the optimal execution time, as exemplified in Figure 3.

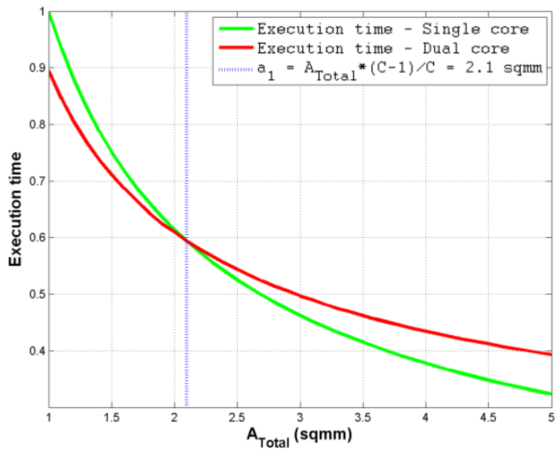

Figure 3. Execution time: Stronger core vs. weaker core

In Figure 3 we compare the execution time of a dual core, each core running a single task, to a single core running both tasks one after the other. The first core is modeled using a power law with exponent $\beta_1 = 0.3$, and the second core with an exponent $\beta_2 = 0.7$. Core 1 is assigned with a task $t_1 = 1$ seconds on a reference processor, and core 2 is assigned with a task $t_2 = 3$ seconds on a reference processor. In the example of Figure 3, a single core is preferred once $A_{Total} \geq 2.1 sqmm$.

Note that our optimization framework allows eliminating a core (that is, not assigning any area to it) by setting $\sum_{i=1}^{M} b_{i,j} = 0$ for core j. This notation is implied in (21).

2.3.2 *Asymmetric dual core with balanced β*

In this scenario, the two acceleration functions share the exact same exponent ($\beta_1 = \beta_2$). Following (18) and (20), any area increment is assigned proportionately to both cores, and thus their relative sizes remain the same. Furthermore, if the sums of t_i allocated to each core are equal to each other, implying v=1, these two cores are symmetric (each core having an area of $\frac{A_{Total}}{2}$).

2.4 Optimizing a multi-core processor

Our findings are extended from dual to a multi-core scenario. Consider a multicore architecture containing N distinct cores, where each core's acceleration function is expressed as a power law. Following (17):

$$f_j(a_j) = (v_j a_1)^{-\beta_j} \tag{24}$$

where:

$$v_j = \left(\frac{\sum_{i=1}^{i=M} t_i b_{i,j}\, a_1^{\beta_1}}{\sum_{i=1}^{i=M} t_i b_{i,1}\, a_1^{\beta_j}} \right)^{\frac{1}{\beta_j}} \tag{25}$$

Following (6):

$$v_1 a_1 + v_2 a_1 + \cdots + v_N a_1 = A_{Total} \tag{26}$$

Note that per (25), $v_1 = 1$. The resource allocation is:

$$a_1 = \frac{A_{Total}}{\sum_{j=1}^{j=N} v_j}, \qquad a_j = v_j a_1 \tag{27}$$

As shown above, for balanced exponents ($\forall l, k, 1 \leq l, k \leq N,\ \beta_l = \beta_k,$), v_j is constant. The implications discussed in the previous section on dual-core processors apply also to multi-cores: the strongest core (or cores having the same and highest β) dominates the area when total multicore area is sufficiently large. If all cores share the same β, the area is divided among them proportionately, as per (25).

2.5 Optimizing a heterogeneous multicore

In contrast to asymmetric architecture, in heterogeneous architecture tasks can execute only on their designated cores. Consider a baseline heterogeneous architecture, where each task is assigned to a distinct core (M=N, and task i is assigned to core j=i, and hence $b_{i,j}$ is the identity matrix). Further, the acceleration functions are characterized by power law (11). Equation (25) reduces to:

$$v_{Hete-MC\,j} = \left(\frac{t_j}{t_1} \frac{a_1^{\beta_1}}{a_1^{\beta_j}} \right)^{\frac{1}{\beta_j}} \tag{28}$$

where the optimal resource allocation follows (27). We now consider the following scenarios of imbalanced and balanced exponents.

2.5.1 *Heterogeneous multicore with imbalanced β*

In this scenario, the acceleration functions have different exponents ($\forall l, k, 1 \le l, k \le N,\ \beta_k \neq \beta_l$). Since all cores are utilized, the area of the weaker cores (i.e., the ones with the smaller exponent) grows faster than the area of the stronger cores as the total area budget grows, so as to uphold the first optimality condition (9). To emphasize this point, consider a dual core architecture in which the first processor's exponent is larger than the second:

$$a_1^{-\beta_1} t_1 = a_2^{-\beta_2} t_2 \tag{29}$$

Assume that A_{Total} grows by small area $\varepsilon = \varepsilon_1 + \varepsilon_2$, and area is initially distributed such that the first core is allocated with ε_1 while the second core is allocated with ε_2. Since $\beta_1 > \beta_2$, in order to keep the equality, $\varepsilon_2 > \varepsilon_1$.

This is in contrast to the asymmetric architecture discussed above, in which the stronger cores are allocated larger area at the expense of the weaker cores. This transpires because it is impossible to port tasks from a weak core to a stronger one due to heterogeneity. In the extreme case of the heterogeneous multicore, the weaker core consumes nearly the entire multicore resources.

2.5.2 *Heterogeneous multicore with balanced β*

In this scenario, the two acceleration functions share the exact same exponent ($\beta_l = \beta_k,\ \forall l, k\ 1 \le l, k \le N$). Following (28), any area increment is assigned proportionately to each core, and thus their relative sizes remain unchanged. Furthermore, if the sum of the reference execution time of all tasks allocated to each core is equal, implying v_j=1, it implies symmetric cores, each core having an area of $\frac{A_{Total}}{N}$.

2.6 Maximum Speedup of area constrained multicore

In this section we derive the maximal speedup at the optimal area allocation point. Let the acceleration functions obey the power-law. Let the first core having the largest exponent have an index of 1, that is: $\forall j, 1 \le j \le N$, $\beta_1 \ge \beta_j$. Further, if two or more cores share the same exponent, let the core having the smaller index execute the smaller portion of the workload. The total speedup is defined as the ratio between the serial execution time of the tasks on the reference processor to the multicore execution time. According to the first optimality condition (9), all processors must have the same total execution time, and we can write the asymmetric speedup as:

$$Speedup_{Asy-MC} = \frac{\sum_{i=1}^{i=M} t_i}{\frac{1}{N}\sum_{j=1}^{j=N}\left(f_j(a_j)\sum_{i=1}^{i=M} t_i b_{i,j}\right)} = \frac{\sum_{i=1}^{i=M} t_i}{\frac{a_1^{-\beta_1}}{N}\sum_{j=1}^{j=N}\left(\sum_{i=1}^{i=M} t_i b_{i,1}\right)} = N a_1^{\beta_1} \tag{30}$$

Note that (30) assumes that all processors are utilized; if this is not the case, the result $Na_1^{\beta_1}$ is replaced by $Ra_1^{\beta_1}$ where R is the number of utilized processors. In a similar manner, the speedup of the heterogeneous architecture is:

$$Speedup_{Hete-MC} = N a_1^{\beta_1} \tag{31}$$

Since the first core has the largest exponent and executes the smallest portion of the workload out of the cores sharing the same largest exponent, to uphold (9), a_1 is thus assigned with the smallest portion of the multicore total area: $\forall j, 1 \le j \le N, a_1 \le a_j$. Thus, following (6), $a_1 \le 1/N$. The larger the asymmetry of the core's allocated area, the smaller a_1 is. Further, a_1 peaks when all cores are symmetric, at which point $a_1 = 1/N$.

Now that we have derived an expression for the speedup of both asymmetric and heterogeneous architectures, let us proceed with maximizing them. According to (27), a_1 is a function of A_{Total}, of the task assignment of each core and of the exponents β_j. Assuming that the architect may select the acceleration function of each core, what criteria should he utilize in order to achieve maximum speedup? To find the answer, we maximize (30):

$$max\, N a_1^{\beta_1} = max\, N \left(\frac{A_{Total}}{\sum_{j=1}^{j=N} v_j}\right)^{\beta_1} \tag{32}$$

This maximum speedup point may also be obtained by finding the minimum of the inverse of (32), as follows. Substituting (25):

$$min \frac{1}{NA_{Total}^{\beta_1}} \sum_{j=1}^{j=N} \left(\frac{\sum_{i=1}^{i=M} t_i b_{i,j}}{\sum_{i=1}^{i=M} t_i b_{i,1}} \frac{a_1^{\beta_1}}{a_1^{\beta_j}}\right)^{\frac{1}{\beta_j}} \tag{33}$$

To reach the minimum, several factors are considered:

- N (the number of utilized cores) should be maximized. Note that the maximal concurrency is attained when all available cores are employed, such that each core is assigned a single task.
- Minimize $\frac{\sum_{i=1}^{i=M} t_i b_{i,j}}{\sum_{i=1}^{i=M} t_i b_{i,1}}$. Since $\forall j, 1 \le j \le N,\ \beta_1 \ge \beta_j$, the sum of the reference run times of tasks assigned to core 1 is the largest (stronger cores are assigned the larger portion of the workload).
- We seek the relationship among the exponents β that leads to the minimum value.

To that end, let us differentiate the expression with respect to a_1, and equate the derivative to zero, as follows:

$$\frac{\partial}{\partial a_1}\left[\frac{1}{NA_{Total}^{\beta_1}} \sum_{j=1}^{j=N} \left(\frac{\sum_{i=1}^{i=M} t_i b_{i,j}}{\sum_{i=1}^{i=M} t_i b_{i,1}} \frac{a_1^{\beta_1}}{a_1^{\beta_j}}\right)^{\frac{1}{\beta_j}}\right] = \frac{1}{NA_{Total}^{\beta_1}} \sum_{j=1}^{j=N} \left(\frac{\sum_{i=1}^{i=M} t_i b_{i,j}}{\sum_{i=1}^{i=M} t_i b_{i,1}}\right)^{\frac{1}{\beta_j}} \left(\frac{\beta_1 - \beta_j}{\beta_j}\right) a_1^{\frac{\beta_1 - \beta_j}{\beta_j} - 1} = 0 \tag{34}$$

The only solution for the above expression is:

$$\forall j, 1 \le j \le N, \qquad \beta_1 = \beta_j \tag{35}$$

Equation (35) implies that cores should have the same and highest exponent. The rationale is that weaker cores are eventually removed (in the asymmetric architecture), or strengthened at the expense of the stronger cores (in the heterogeneous architecture). In particular, if all tasks have the same reference runtime, the optimal architecture is symmetric. This conclusion is consistent with the conclusions of [19] and [12], that is, when the entire workload is parallelizable and there is no serial fraction, the best multicore is the symmetric one. However, if the tasks have different lengths, the optimal architecture is asymmetric despite the fact that the workload could be fully parallelizable (all tasks are done in parallel). This conclusion complements [19] and [12] that assumed that the parallelizable portion of the workload could be equally scaled by any number of cores.

2.7 Example: Optimal quad-core area allocation

Consider a workload consisting of four concurrent tasks, incurring execution times $t_i = \{100, 90, 80, 70\}$ on a reference processor. The multicore architect wishes to optimize the execution time under an area constraint, and may utilize up to four distinct cores. Consider Figure 4–Figure 6, where the horizontal axis depicts four multicore area scenarios, characterized by multiples of total area, 1×, 2×, 4× and 8× respectively, and the vertical axis shows the distribution of that total area among the four cores. Note that in a sufficiently large multicore chip, stronger cores are allocated larger area in asymmetric multi-cores, while in heterogeneous architectures the weaker cores are allocated a larger area.

2.7.1 Asymmetric architecture with imbalanced β

In this scenario, we consider an asymmetric quad-core having imbalanced exponents $\beta_j = \{0.7, 0.6, 0.5, 0.4\}$, respectively. In Figure 4, as the total multicore area increases, tasks shift from the weaker cores (core #2, #3 and #4 having lower β) to the stronger core, and the weaker cores are progressively eliminated. See analysis in 2.3.1.

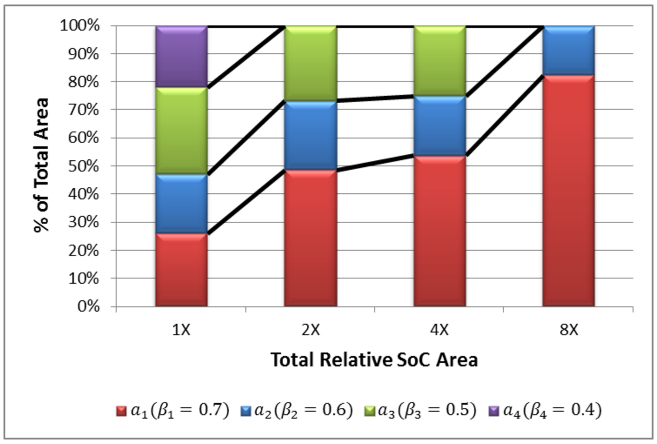

Figure 4. Optimal resource allocation of asymmetric imbalanced β quad-core multi-processor, as a function of multicore area

2.7.2 Asymmetric architecture with balanced β

In this scenario, we consider an asymmetric quad-core architecture where all cores share the same exponent, $\beta_j = 0.5$. In contrast with the first scenario, Figure 5 shows that as the total multicore area increases, tasks and areas do not shift. Rather, partitioning among the cores remains constant. See analysis in section 2.3.1.

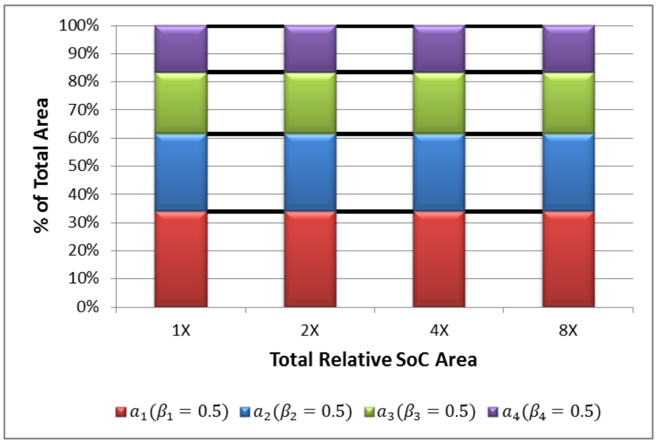

Figure 5. Optimal resource allocation of asymmetric balanced β quad-core multi-processor, as a function of multicore area

2.7.3 Heterogeneous architecture with imbalanced β

As in the first scenario, the cores have different exponents, $\beta_j = \{0.7, 0.6, 0.5, 0.4\}$, respectively. Consider Figure 6. As the total multicore area gets larger, area resources shift from the stronger core to the weaker ones (core #2, #3 and #4 having lower β). See analysis in section 2.5.1.

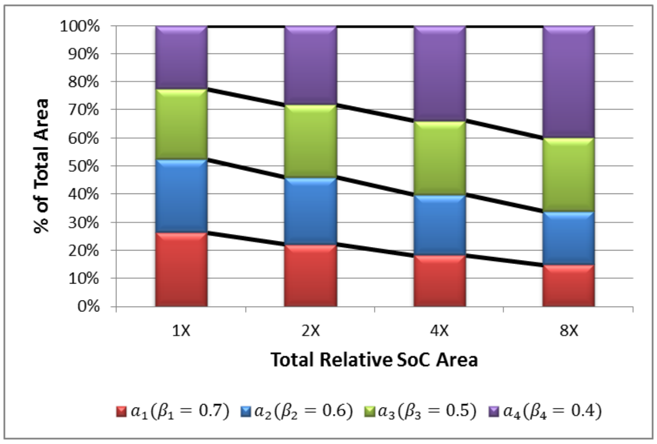

Figure 6. Optimal resource allocation of heterogeneous imbalanced β quad-core multi-processor, as a function of multicore area

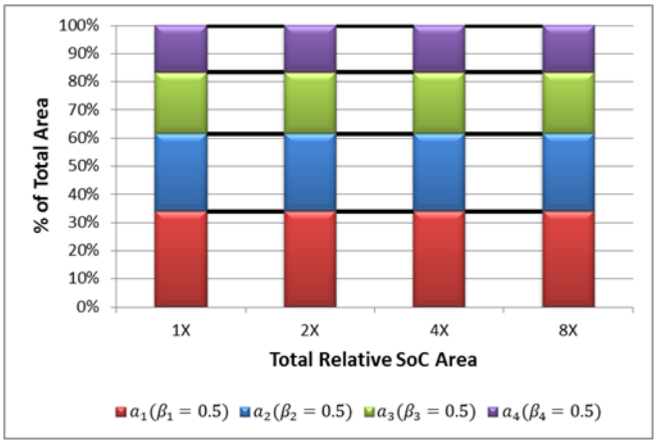

Figure 7. Optimal resource allocation of heterogeneous balanced β quad-core multi-processor, as a function of multicore area

2.7.4 Heterogeneous architecture with balanced β

In this scenario, we consider a heterogeneous quad-core architecture where all cores share the same exponent, $\beta_j = 0.5$. In contrast with the third scenario, Figure 7 shows that as the total multicore area increases, tasks and areas do not shift from one core to another. Rather, partitioning among the cores remains constant. See analysis in section 2.3.12.5.2.

3 Combined Area and Power Optimization

In the previous section, we have established and analyzed the analytic optimization framework that distributes area resources in an area constrained multicore. It has been suggested (e.g., [15]) that power consumption is a more severe constraint than chip area, making the per-core power allocation a key design constraint. In this section, we extend the framework discussed in section 2 to handle both area and power constraints.

3.1 Area and power constrained multicore model

We modify the acceleration function (1) of core j to depend on both its area a_j and its dynamic power p_j, as follows:

$$f_j(a_j,p_j) = \frac{1}{Perf_j(a_j,p_j)} \tag{36}$$

Note that p_j represents actual power dissipated in core j rather than the maximum power that can be consumed by that core, as determined by its area, maximum voltage, maximum frequency and other physical constraints; for instance, it is possible that at some optimum point for the entire chip, a particular core is operated at $p_j < P_{\max j}$.

We wish to minimize the execution time, defined by T:

$$T_j \triangleq \sum_{i=1}^{i=M} f_j(a_j,p_j)\, b_{i,j} t_i = f_j(a_j,p_j) \sum_{i=1}^{i=M} b_{i,j} t_i \qquad T \triangleq max(T_1, T_2, \ldots, T_N) \tag{37}$$

under the constraints detailed in (6)-(8), as well as a power constraint. Area is a static resource, i.e., it does not change during execution. Power, however, has both static and dynamic components, and the distribution of these resources among the cores is not identical. Static power depends on temperature [9], and temperature, in turn, depends on power density (related to dynamic power). We shall separate the static power consumption of the core into two components, idle static power, and temperature induced static power. Assume that the multicore employs Dynamic Voltage and Frequency Scaling (DVFS). The idle static power is annotated as a manufacturing technology related constant s_1 multiplied by the core's allocated area a_j representing the static power when: (a) gate temperature is at the low end of the operating conditions range; (b) core voltage is at the low end of the DVFS voltage range; and, (c) core frequency is at the low end of the DVFS frequency range. Within the normal operating temperature range (say, 55°C–85°C), leakage power consumption may be estimated using a linear function of temperature [21]. Thus in normal operating conditions, we can express the temperature induced static power as a manufacturing technology related constant s_2 times its dynamic power p_j.

$$P_{Static} = \sum_{j=1}^{j=N} s_1 a_j + s_2 p_j \tag{38}$$

Note that in our analysis we choose to ignore floorplan-induced leakage power, that is, temperature increase due to the heat generated by adjacent cores. Given that the cores have large enough radial shape, floorplan induced leakage is limited to the boundary and hence is a second order effect on total leakage. Note however that this assumption may not always hold.

Power constraints are often set by heat dissipation, which manifests itself over long period. Accordingly, we focus on average power. The average dynamic power consumption is calculated by dividing overall energy by overall execution time:

$$P_{Dynamic} = \frac{E_{Dynamic}}{T} = \sum_{i=1}^{i=N} \frac{\sum_{j=1}^{j=N} T_j\, p_j}{max(T_1, T_2, \ldots, T_N)} \tag{39}$$

Since in the optimum $\forall j, T_j = T$ (as shown in section 2), the power constraint becomes:

$$P_{Static} + P_{Dynamic} = \sum_{j=1}^{j=N} \left(s_1 a_j + s_2 p_j + p_j\right) \le P_{Total} \tag{40}$$

where P_{Total} is the maximum allowed average power consumption. Optimal power allocation is achieved when minimizing the cost function T (37) under this constraint. This problem is solved in a manner similar to the area-bound problem of section 2. The Lagrange optimization of cost function (37) subject to constraints detailed in (6)-(8) and (40) is provided in Appendix B. The conclusion is that for all utilized cores, $\forall j,k$ the following conditions hold at the optimal point:

A. First Optimality Condition: *All utilized cores must have the same runtime.*

$$f_j(a_j,p_j) \sum_{i=1}^{i=M} t_i b_{i,j} = f_k(a_k,p_k) \sum_{i=1}^{i=M} t_i b_{i,k} \tag{41}$$

B. Second Optimality Condition: *The first derivatives of the execution time functions across all utilized cores with respect to area and power must be equal.*

$$\frac{\partial f_j(a_j,p_j)}{\partial a_j} \sum_{i=1}^{i=M} t_i b_{i,j} = \frac{\partial f_k(a_k,p_k)}{\partial a_k} \sum_{i=1}^{i=M} t_i b_{i,k} \tag{42}$$

and

$$\frac{\partial f_j(a_j,p_j)}{\partial p_j} \sum_{i=1}^{i=M} t_i b_{i,j} = \frac{\partial f_k(a_k,p_k)}{\partial p_k} \sum_{i=1}^{i=M} t_i b_{i,k} \tag{43}$$

Note that dividing (42) by (43) yields the following:

$$\frac{\frac{\partial f_j(a_j,p_j)}{\partial a_j}}{\frac{\partial f_j(a_j,p_j)}{\partial p_j}} = \frac{\frac{\partial f_k(a_k,p_k)}{\partial a_k}}{\frac{\partial f_k(a_k,p_k)}{\partial p_k}} \tag{44}$$

Equation (44) shows that at the optimal area and power allocation, the area gradient $\frac{\partial f_j(a_j,p_j)}{\partial a_j}$ is proportional to the power gradient $\frac{\partial f_j(a_j,p_j)}{\partial p_j}$.

3.2 Power-law acceleration functions

Consider a workload comprising multiple tasks, each incurring execution time t_i on a reference processor. The target architecture may utilize up to N cores. Following Pollack's rule [8] and [19], each task i can be accelerated by any one of N available cores, where each core's performance $Perf_j(a_j)$, relative to performance on the reference processor, follows a power-law function of its area a_j (see (11) in section 2).

Assume further that the multicore employs DVFS. Each core's frequency corresponds to its power budget, p_j, enabling clocking at a range of frequencies, from zero (when the core is idle) to F_{max} (maximal frequency possible by the operating conditions and the physical constraints of the design). The maximal dynamic power of core j can be written as follows:

$$P_{max\,j} = \alpha_1 C_j F_{\max j} V_{max}^2 \tag{45}$$

where α_1 is the activity factor. Voltage is inversely proportional to gate delay, and thus it is proportional to frequency $V_{\max j} = \alpha_2 F_{\max j}$, where α_2 is a constant translating Hz to Volts. Capacitance C_j is proportional to area a_j, $C_j = \alpha_3 a_j$. Assume that all cores are subject to the same activity factor α_1. Assume further that all cores are driven by the same voltage range ($\forall j, V_{min} \le V_j \le V_{max}$). We can write (45) as:

$$P_{max\,j} = \alpha_1 C_j F_{\max j}\left(\alpha_2 F_{\max j}\right)^2 = \alpha a_j F_{\max j}^3 \tag{46}$$

where α is a constant absorbing α_1, α_2 and α_3. We can also model $P_{max\,j} = c_j a_j$, where $c_j = \alpha F_{\max j}^3$ is a constant translating units of area to power, in agreement with [5] and [14]. Next, assume that each core is driven at some operating frequency, $F_{\text{oper } j}$, dissipating dynamic power p_j. Following (46):

$$p_j = \alpha a_j F_{\text{oper } j}^3 \tag{47}$$

Equation (47) complements [17] who noted through an empirical study that a typical CPU power is a polynomial function of frequency $p_{core} \propto f^\eta$ where η typically ranges from 1.5 (in a low power manufacturing technology) to 2.4 (in a high performance manufacturing technology). The present theoretical analysis leads to $\eta = 3$ instead, but any other number will do and will not significantly change our results. We annotate the normalized frequency of the core as $F_{norm\,j}$, ranging from 0 (when the core is idle) to 1 (when the core is driven at maximal frequency corresponding to the operating conditions and the physical constraints of the design). Thus:

$$F_{norm\,j} = \frac{F_{oper\,j}}{F_{max\,j}} = \left(\frac{p_j}{P_{max\,j}}\right)^{\frac{1}{3}} = \left(\frac{p_j}{c_j a_j}\right)^{\frac{1}{3}} \tag{48}$$

Even when assigned with infinite power budget, the core's operating frequency cannot exceed its maximum frequency determined by the physical constraints of the design, thus (48) is revised as follows:

$$F_{norm\,j} = \min\left(\left(\frac{p_j}{c_j a_j}\right)^{\frac{1}{3}}, 1\right) \tag{49}$$

Since the core's inverted performance is modeled as power law with respect to its allocated area, we can write the acceleration function of a core as follows:

$$f_j(a_j,p_j) = \frac{1}{{a_j}^{\beta_j}} \frac{1}{F_{norm\,j}} = \frac{1}{{a_j}^{\beta_j}} max\left(\left(\frac{c_j a_j}{p_j}\right)^{\frac{1}{3}}, 1\right) \tag{50}$$

Note that if core j is assigned with null power ($p_j = 0$), $f_j \to \infty$. Conversely, if core j is assigned with infinite power ($p_j \to \infty$), $f_j \to {a_j}^{-\beta_j}$.

We can further break (50) into its analytic representation (eliminating the max function) as follows:

$$f_j(a_j,p_j) = \frac{1}{{a_j}^{\beta}}\left[H\left(1 - \left(\frac{c_j a_j}{p_j}\right)^{\frac{1}{3}}\right) + \left(\frac{c_j a_j}{p_j}\right)^{\frac{1}{3}} H\left(\left(\frac{c_j a_j}{p_j}\right)^{\frac{1}{3}} - 1\right)\right] \tag{51}$$

Figure 8 plots the acceleration function (51) vs. area budget, at a fixed power budget $p_j = 1W$. The red, green and blue graphs represent exponents β_j of 0.2, 1/3 and 0.5 respectively. Note that for the purpose of generating Figure 8 and Figure 9, the parameter c_j is assumed $c_j = 0.2W/mm^2$ for all cores. As shown in the figure, at area below $5mm^2$, where $p_j > c_j a_j$, for all charts, the frequency maxes out and the acceleration follows ${a_j}^{-\beta_j}$. However once the area surpasses $5mm^2$, that is ($p_j < c_j a_j$), the acceleration either (a) decreases in case $\beta_j < 1/3$ (red chart); (b) flattens out in case $\beta_j = 1/3$ (green chart); or (c) increases, but at a lower rate than when $\beta_j > 1/3$ (blue chart). The rationale is that although further area allocation has a positive impact on acceleration, the core's frequency scales back such that the power budget is maintained. Scaling back power has negative impact on performance. In particular, for cores having $\beta_j < 1/3$, the negative impact from power-scaling outpaces the positive impact provided by the increased area and thus the acceleration decreases. The reason that $\beta_j = 1/3$ constitues the crossover point in this case lies in the 3rd power in (46),(47).

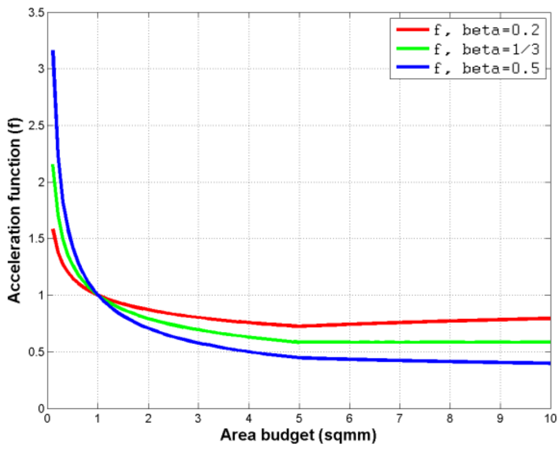

Figure 8. Acceleration function with various exponents, fixed power budget (1W) and fixed c ($0.2W/mm^2$).

Figure 9 plots the acceleration function (51) vs. power budget, at a fixed area budget $a_j = 5mm^2$. As above, the red, green and blue charts represent exponents β_j of 0.2, 1/3 and 0.5 respectively. As shown in the figure, for allocated core frequency, corresponding to power below 1W, that is $p_j < c_j a_j$, the acceleration follows $a_j^{-\beta_j}$. However once the allocated power budget exceeds 1W, that is $(p_j > c_j a_j)$, acceleration is clamped. This happens because excess power allocation beyond $P_{max\ j}$ cannot be utilized.

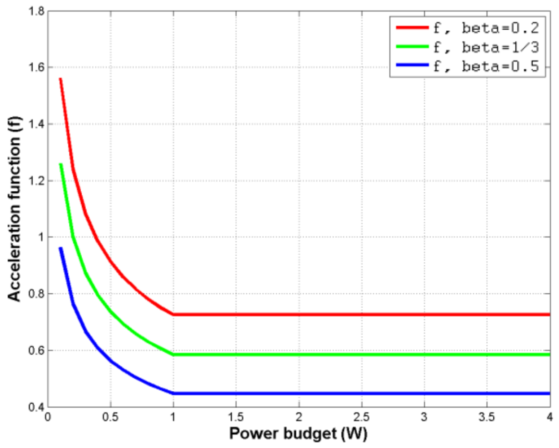

Figure 9. Acceleration function with various exponents, fixed area budget ($5mm^2$) and fixed c ($0.2W/mm^2$)

3.3 Example: Optimal quad-core architecture area and power allocation

Consider a workload consisting of four concurrent tasks, incurring execution times $t_i = \{100, 100, 100, 100\}$ on a reference processor. The multicore architect wishes to optimize execution time under both area and power constraints, and may utilize up to four distinct cores. The example is organized in four cases, related to imbalanced and balanced β and to asymmetric and heterogeneous quad cores. Figure 10, Figure 12, Figure 14 and Figure 16 depict two sets of bars, each set corresponding to two extreme multicore area scenarios, one being eight times larger than the other. Within each set, the first bar corresponds to area optimization, while the second bar corresponds to both area and power optimization. The vertical axis shows the distribution of total area among the four cores. In a similar manner, Figure 11, Figure 13, Figure 15 and Figure 17 share the same horizontal axis as the previous four figures, while the vertical axis depicts the distribution of total average power among the four cores. Consider the following two scenarios.

- Total chip power budget is *larger* than the maximal power that the multicore may dissipate, $P_{Max} \le P_{Total}$. None of the cores is power limited, and the optimal area allocation among the cores is as depicted in the first bar of each set.
- Total chip power budget is *lower* than the maximal power, $P_{Max} > P_{Total}$. Some of the cores are power-limited and hence the optimal area allocation among the cores is different from the first bar of each set.

To examine the second scenario, we set P_{Total} to 80% of the maximal power dissipation of the multicore, that is:

$$\sum_{j=1}^{j=N} p_j \le P_{Total} = 0.8 * P_{Max} \quad \textbf{(52)}$$

and extract the optimal core area, as exemplified below.

3.3.1 *Asymmetric architecture with imbalanced β*

Consider an asymmetric quad-core having imbalanced exponents $\beta_j = \{0.7, 0.6, 0.5, 0.4\}$, respectively. As shown in Figure 10, as the total multicore area increases (from 1× to 8×), tasks shift from the weaker cores (core #2, #3 and #4 having lower β) to the stronger core, and the weaker cores are progressively eliminated. When power is constrained (right hand column in each set), at the same A_{Total}, the stronger cores are provided with even larger area portion, at the expense of the weaker cores. In contrast, Figure 11 shows that in a power limited asymmetric multicore, the weaker cores are provided with even larger portion of the total average power. This is because in a power limited architecture, to uphold (41) the equilibrium is found when area is transferred from the weaker cores to the stronger ones, while power is transferred in the opposite direction, from the stronger cores to the weaker ones.

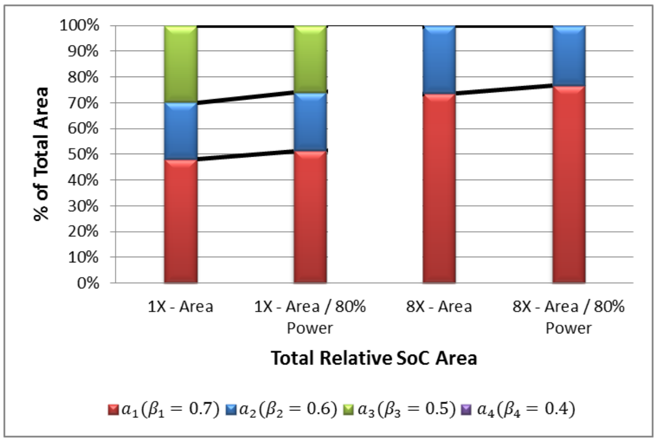

Figure 10. Optimal area allocation of asymmetric imbalanced β quad-core multi-processor, as a function of multicore area

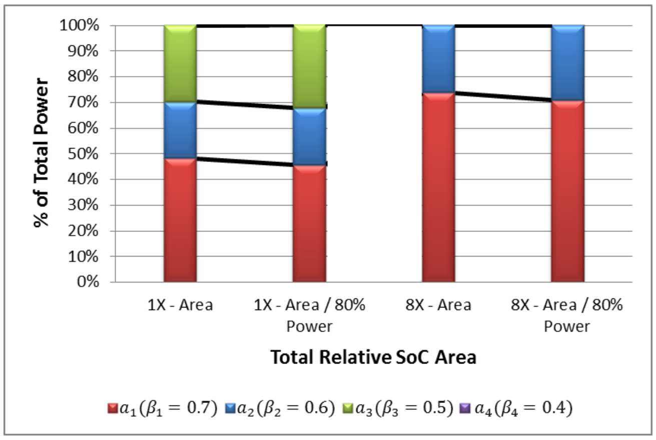

Figure 11. Optimal average power allocation of asymmetric imbalanced β quad-core multi-processor, as a function of multicore area

3.3.2 Asymmetric architecture with balanced β

In this scenario, we consider an asymmetric quad-core architecture where all core share the same exponent, $\beta_j = 0.5$. In contrast with the previous scenario, Figure 12-Figure 13 shows that as the total multicore area increases, task, area and average power allocation do not shift from one core to another; rather, partitioning among the cores remains constant. Similarly, power allocations also do not shift.

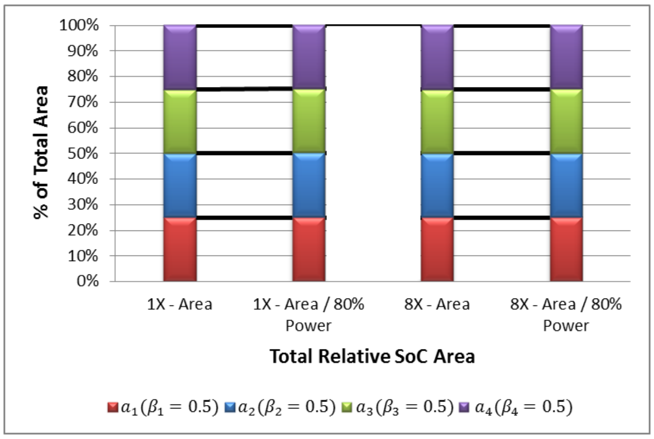

Figure 12. Optimal area allocation of asymmetric balanced β quad-core multi-processor, as a function of multicore area

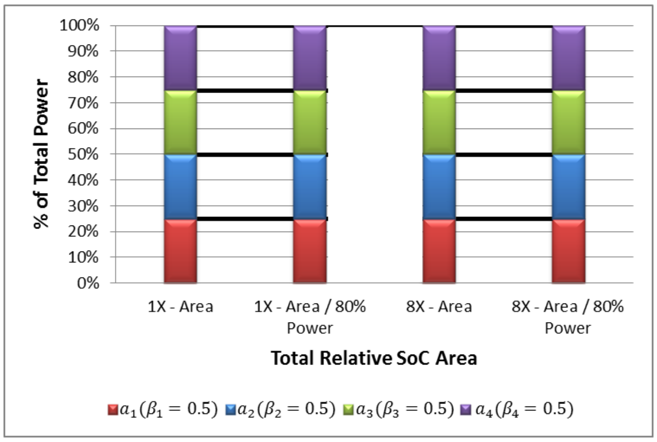

Figure 13. Optimal average power allocation of asymmetric balanced β quad-core multi-processor, as a function of multicore area

3.3.3 Heterogeneous architecture with imbalanced β

Next, consider a heterogeneous quad-core having imbalanced exponents $\beta_j = \{0.7, 0.6, 0.5, 0.4\}$, respectively. As shown in Figure 14, as the total multicore area gets larger, area resources shift from the stronger core to the weaker cores (core #2, #3 and #4 having lower β). Note that while this trend is similar to Figure 10, in the heterogeneous case all cores must be retained while in the asymmetric multicore some cores can be eliminated.

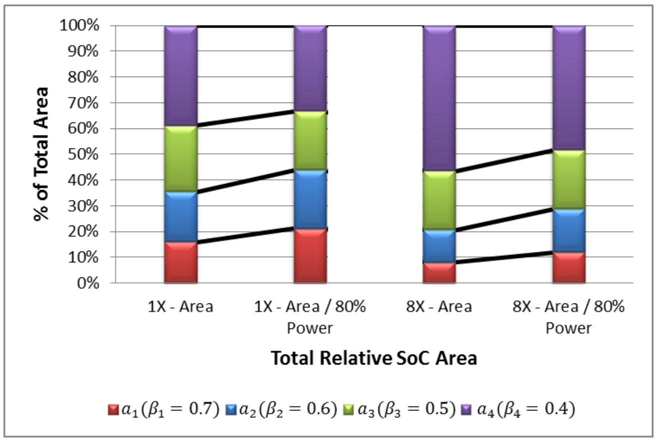

Figure 14. Optimal area allocation of heterogeneous imbalanced β quad-core multi-processor, as a function of multicore area

When power is constrained, the stronger cores are provided with even larger portion of the total area, at the expense of the weaker cores. In contrast, Figure 15 shows that in a power limited heterogeneous multicore, the weaker cores are provided with even larger portion of the total average power. In a power limited architecture, to uphold (41) the equilibrium is found when area is transferred from the weaker cores to the stronger ones, while power is transferred in the opposite direction. Here, too, the trend is similar to Figure 11, but in the heterogeneous multicore all cores must be retained.

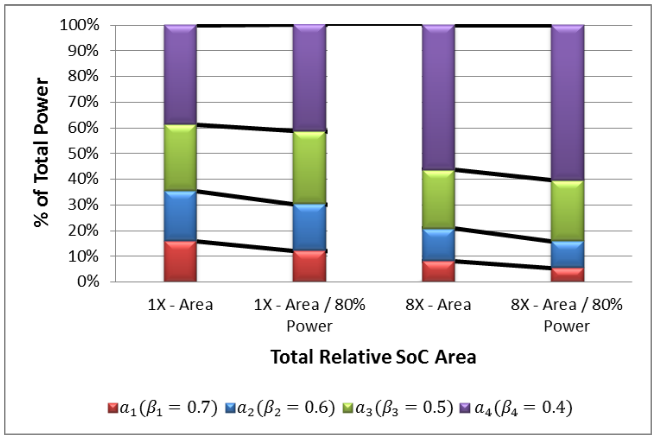

Figure 15. Optimal average power allocation of heterogeneous imbalanced β quad-core multi-processor, as a function of multicore area

3.3.4 Heterogeneous architecture with balanced β

Last, consider the heterogeneous quad-core architecture where all core share the same exponent, $\beta_j = 0.5$. In contrast with the imbalanced heterogeneous multicore, Figure 16-Figure 17 show that area and power allocations do not shift, similarly to the balanced asymmetric multicore of Figure 12-Figure 13.

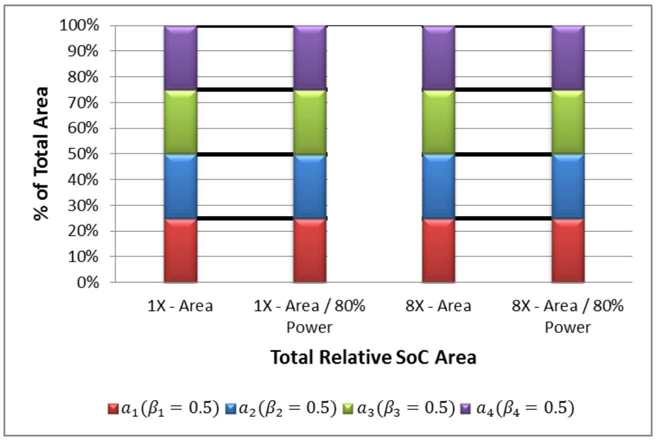

Figure 16. Optimal area allocation of asymmetric balanced β quad-core multi-processor, as a function of multicore area

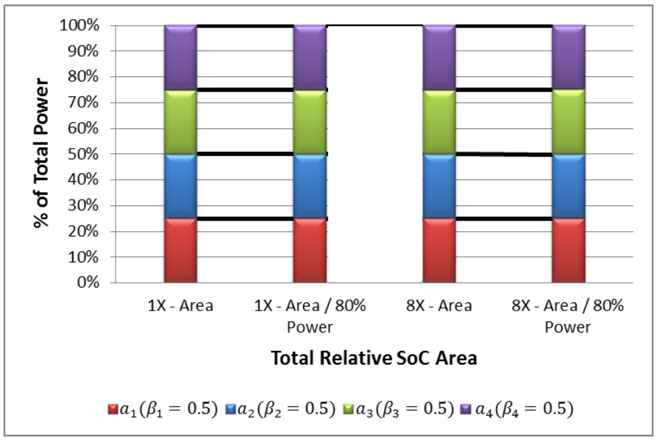

Figure 17. Optimal resource allocation of asymmetric balanced β quad-core multi-processor, as a function of multicore area

4 Conclusions

This paper describes a multi-core optimization framework that, given a concurrent workload and a set of potential cores, selects an optimal subset of the cores and allocates area and power resources among them. The algorithm relies on modeling the performance of each core as a function of the area and power resources it uses.

For area-only optimization, only at the optimal resource allocation point (a) the execution times of all utilized cores are equal across all utilized cores; and (b) the first derivatives with respect to area of the execution time function of each core are equal. The first optimality condition is backed by conventional wisdom: if one of the cores runs faster than the others, it has been allocated with too much area, and it is beneficial to reduce its resources and allocate them elsewhere. The second optimality condition means that at the optimal resource allocation, each additional infinitesimal area would create the same improvement in the total execution time on any of the utilized cores. Otherwise, area allocation is imbalanced and some area should be removed from one utilized core and be allocated to another.

In an imbalanced asymmetric architecture composed of a combination of weak and strong cores, further increase of total multicore area is allocated to the stronger cores. In that regard, the stronger cores are assigned a larger portion of the workload and consequently are allocated additional area resources at the expense of the weaker cores, until the weaker cores have no tasks to run and are eliminated. Further, in cases where the remaining cores share the same strength, these cores will not get eliminated even if the total multicore area increases indefinitely, but will grow proportionally in area. On the other hand, in an imbalanced heterogeneous architecture composed of a combination of weak and strong cores, since tasks cannot transition from one core to another due to the different ISA and hence all heterogeneous cores must be utilized, as the total multicore resources increase, the weaker cores are augmented with additional resources at the expense of the stronger cores. If all cores share the same strength, each core grows proportionally to the growth of the multicore total area.

For optimization that considers both area and power constraints, the second optimality condition is augmented as follows: only at the optimal resource allocation point the first derivatives of the execution time with respect to both area and to power, are equal across all utilized cores. Thus, at the optimal area and power allocation, each additional infinitesimal area and power would create the same improvement in the total execution time on any of the utilized cores. Otherwise, area and power allocation is imbalanced and some area or power should be transferred from one utilized core to another

The insights in the combined area and power optimization are also somewhat different from area-only optimization: In an imbalanced asymmetric architecture composed of a combination of weak and strong cores, an increase of total multicore area is allocated to the stronger cores, to a larger extent than a non-power constrained multicore, up to eventual elimination of some weak cores. On the other hand, the weaker cores will receive a larger portion of the available average power, but to a larger extent than a non-power constrained multicore. In a heterogeneous architecture composed of a combination of weak and strong cores, as the total multicore resources increase, the stronger cores are augmented with additional area resources at the expense of weaker cores, when compared with a non-power constrained multicore (but no core may be eliminated). At the same time, the weaker cores receive a larger portion of the available average power, when compared with a non-power constrained multicore. In both asymmetric and heterogeneous multicores where all cores share the same strength, area and power allocation of the cores grow proportionally to the growth of the multicore total area.

To conclude, we have presented an analytical tool for multicore partitioning leading to optimal execution time, given an area, or area and power, constraints. We have also described the architectural insights obtained thanks to this modeling. Furthermore, our model offers an efficient alternative to iterative ad-hoc processes for exploring the design space. Our framework can be extended in a number of ways, such as bandwidth, energy and other constrained resources.

5 Acknowledgment

This research was funded in part by the *Intel Collaborative Research Institute for Computational Intelligence* (ICRI-CI) and by Hasso-Plattner Institute (HPI).

6 References

[1] “Find a minimum of constrained nonlinear multivariable function”, R2012a Documentation, Optimization Toolbox, www.mathworks.com/help/toolbox/optim/ug/fmincon.html

[2] A. Cassidy and A. Andreou, “Beyond Amdahl Law - An objective function that links performance gains to delay and energy”, *IEEE Transactions on Computers*, vol. 61, no. 8, pp. 1110-1126, Aug 2012.

[3] A. Elyada, R. Ginosar, U. Weiser, “Low-Complexity Policies for Energy-Performance Tradeoff in Chip-Multi-Processors,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Volume: 16, Issue: 9, 2008, Page(s): 1243 - 1248.

[4] A. Morad, T. Morad, L. Yavits, R. Ginosar, U. C. Weiser. "Generalized MultiAmdahl: Optimization of Heterogeneous Multi-Accelerator SoC," *IEEE Computer Architecture Letters*, 2012.

[5] Agarwal, Anant, et al. "Core Count vs Cache Size for Manycore Architectures in the Cloud." (2010).

[6] Blem, E.; Esmaeilzadeh, H.; St. Amant, R.; Sankaralingam, K.; Burger, D., "Multicore Model from Abstract Single Core Inputs," *Computer Architecture Letters*, vol.PP, no.99, pp.1,1, 0. doi: 10.1109/L-CA.2012.27.

[7] D. H. Woo and H. H. S. Lee. “*Extending Amdahl’s law for energy-efficient computing in the many-core era.*” Computer, 41(12):24–31, 2008.

[8] F. Pollack. "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies," *Keynote, Micro 32*, 1999, www.intel.com/research/mrl/Library/micro32Keynote.pdf

[9] K. Banerjee et al., “A self-consistent junction temperature estimation methodology for nanometer scale ICs with implications for performance and thermal management,” IEEE IEDM, 2003, pp. 887-890.

[10] Khan, A.; Kyungsu Kang; Chong-Min Kyung, "Squeezing maximizing performance out of 3D cache-stacked multicore architectures," *Circuits and Systems (MWSCAS), 2011 IEEE 54th International Midwest Symposium on*, vol., no., pp.1,4, 7-10 Aug. 2011. doi: 10.1109/MWSCAS.2011.6026350

[11] L. Yavits, A. Morad, R. Ginosar, “The effect of communication and synchronization on Amdahl’s law in multicore systems”, Technion TR, http://webee.technion.ac.il/publication-link/index/id/611.

[12] Yavits, L.; Morad, A.; Ginosar, R., "Cache Hierarchy Optimization," *Computer Architecture Letters*, vol.PP, no.99, pp.1,1, 0. doi: 10.1109/L-CA.2013.18

[13] M. D. Hill and M. R. Marty. “Amdahl's Law in the Multicore Era,” *IEEE Computer*, July 2008.

[14] Mark Hempstead, Gu-Yeon Wei, and David Brooks. “Navigo: An early-stage model to study power-constrained architectures and specialization,” ISCA Workshop on Modeling, Benchmarking, and Simulations (MoBS). Austin TX., June 2009

[15] N. Hardavellas *et al.*, "Toward dark silicon in servers." Micro, IEEE 31.4 (2011): 6-15, July-Aug. 2011. doi: 10.1109/MM.2011.77

[16] R. T. Rockefellar. “Lagrange multipliers and optimality”, *SIAM Review*, vol. 35, pp. 183–283, 1993.

[17] Rotem, E.; Ginosar, R.; Weiser, U.; Mendelson, A., "Energy Aware Race to Halt: A Down to EARtH Approach for Platform Energy Management," *Computer Architecture Letters*, vol.PP, no.99, pp.1,1, 0. doi: 10.1109/L-CA.2012.32

[18] S. Borkar. “Thousand Core Chips: A Technology Perspective,” *Proc. ACM/IEEE 44th Design Automation Conf. (DAC)*, 2007, pp. 746-749.

[19] 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*, vol. 4, 2005.

[20] T. Zidenberg, I. Keslassy and U. C. Weiser. "MultiAmdahl: How Should I Divide My Heterogeneous Chip?," *IEEE Computer Architecture Letters*, 2012.

[21] Yongpan Liu *et al.*, "Accurate Temperature-Dependent Integrated Circuit Leakage Power Estimation is Easy," *Design, Automation & Test in Europe Conference & Exhibition*, 2007. DATE '07 , vol., no., pp.1,6, 16-20 April 2007, doi: 10.1109/DATE.2007.364517.

Appendix A: Area constraint optimization via Lagrange multipliers

In this appendix, we detail the Lagrange optimization of the cost function (5) subject to constraint (6). To that end, we introduce a new variable (λ), and the area allocation $\bar{a} = \{a_1, a_2, \dots, a_N\}$, and study the Lagrange function defined by:

$$\Lambda(\bar{a},\lambda):\ \ \max(T_1,T_2,\dots,T_N) + \lambda\left[\sum_{j=1}^{j=N} a_j - A_{Total}\right] \tag{53}$$

The optimum is found where all partial derivatives equal zero:

$$\forall j, 1 \le j \le N \ \ \frac{\partial\Lambda}{\partial a_j} = 0 \ \ and \ \ \frac{\partial\Lambda}{\partial\lambda} = 0 \tag{54}$$

The first derivative with respect to $a_j, \forall j, 1 \le j \le N$ follows:

$$\frac{\partial[max(T_1,T_2,\dots,T_N)]}{\partial a_j} + \lambda\frac{\partial\left[\sum_{j=1}^{j=N} a_j - A_{Total}\right]}{\partial a_j} = 0 \tag{55}$$

As a_j are mutually independent variables, and A_{Total} is a constant, differentiating with respect to a_j and a_k where $\forall j, 1 \le j, k \le N$:

$$\begin{gathered}\frac{\partial[max(T_1,T_2,\dots,T_N)]}{\partial a_j} + \lambda = 0\\ \frac{\partial[max(T_1,T_2,\dots,T_N)]}{\partial a_k} + \lambda = 0\end{gathered} \tag{56}$$

Consequently, the Lagrange solution for $\forall j, 1 \le j, k \le N$ satisfies,

$$\frac{\partial[max(T_1,T_2,\dots,T_N)]}{\partial a_j} = \frac{\partial[max(T_1,T_2,\dots,T_N)]}{\partial a_k} \tag{57}$$

Due to the recursive nature of the max function, we can examine a sample dual core scenario, and expand the result to a multicore. The reduced cost function is:

$$D = max(T_1, T_2) \tag{58}$$

Rewriting (57) for a dual core scenario:

$$\frac{\partial[max(T_1,T_2)]}{\partial a_1} = \frac{\partial[max(T_1,T_2)]}{\partial a_2} \tag{59}$$

In a similar manner to [4], the two variable max function is written using a step function $Q(x)$:

$$max(T_1,T_2) = Q(T_1 - T_2)T_1 + Q(T_2 - T_1)T_2 \tag{60}$$

where:

$$Q(x) = \begin{cases} 0, & (x < 0) \\ 1, & (x \ge 0) \end{cases} \tag{61}$$

We use the sigmoid function as a differentiable approximation of the step function:

$$H(x) = \frac{1}{1+e^{-kx}} \cong Q(x) \tag{62}$$

The larger the k, the sharper the transition at $x = 0$. The sigmoid has the following properties:

$$H(x) = 1 - H(-x) \tag{63}$$

$$H'(x) = kH(x)\big(1 - H(x)\big) \tag{64}$$

$$\begin{gathered} H'(x=0) = k/4 \\ H'(x \neq 0) = \lim_{k\to\infty} \frac{ke^{-kx}}{(1+e^{-kx})^2} \cong 0, \end{gathered} \tag{65}$$

Thus, the max function in (60) can be expressed in a differentiable form as follows:

$$max(T_1, T_2) = H(T_1 - T_2)T_1 + H(T_2 - T_1)T_2 \tag{66}$$

Figure 18 depicts a dual core execution time, the area of which is bounded by A_{Total}. When area is taken from the second core and applied to the first, the performance of the first core increases and thus its execution time T_1 decreases. The sigmoid based implementation of the *max* function is drawn for three k values. The higher the k, the sharper the transition at $T_1 = T_2$.

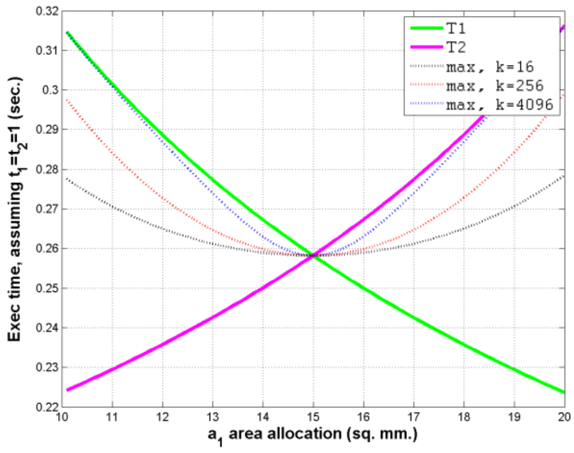

Figure 18. Sigmoid based max function

Resolving $\frac{\partial[\max(T_1,T_2)]}{\partial a_j}$:

$$\begin{gathered} \frac{\partial[H(T_1-T_2)T_1 + H(T_2-T_1)T_2]}{\partial a_j} = \\ kH(T_1-T_2)\big(1-H(T_1-T_2)\big)\frac{\partial[T_1-T_2]}{\partial a_j}T_1 + \\ kH(T_2-T_1)\big(1-H(T_2-T_1)\big)\frac{\partial[T_2-T_1]}{\partial a_j}T_2 + \\ H(T_1-T_2)\frac{\partial[T_1]}{\partial a_j} + H(T_2-T_1)\frac{\partial[T_2]}{\partial a_j} \end{gathered} \tag{67}$$

Substituting $j = 1$, and since $\frac{\partial[T_2]}{\partial a_1} = 0$,

$$\begin{gathered} \frac{\partial[\max[T_1,T_2]]}{\partial a_1} = \\ kH(T_1-T_2)\big(1-H(T_1-T_2)\big)\frac{\partial[T_1]}{\partial a_1}T_1 \\ + H(T_1-T_2)\frac{\partial[T_1]}{\partial a_1} \\ -kH(T_2-T_1)\big(1-H(T_2-T_1)\big)\frac{\partial[T_1]}{\partial a_1}T_2 \end{gathered} \tag{68}$$

Following (63), since $H(T_2 - T_1) = \big(1 - H(T_1 - T_2)\big)$, and $\big(1 - H(T_2 - T_1)\big) = H(T_1 - T_2)$, we can conclude:

$$\begin{gathered} \frac{\partial[\max[T_1,T_2]]}{\partial a_1} = k\frac{\partial[T_1]}{\partial a_1}H(T_1-T_2) * \\ \left[\big(1-H(T_1-T_2)\big)(T_1-T_2) + \frac{1}{k}\right] \end{gathered} \tag{69}$$

In a similar manner,

$$\begin{gathered} \frac{\partial[\max(T_1,T_2)]}{\partial a_2} = k\frac{\partial[T_2]}{\partial a_2}H(T_2-T_1) * \\ \left[\big(1-H(T_2-T_1)\big)(T_2-T_1) + \frac{1}{k}\right] \end{gathered} \tag{70}$$

We can now write (59) as:

$$\begin{gathered} k\frac{\partial[T_1]}{\partial a_1}H(T_1-T_2) * \\ \left[\big(1-H(T_1-T_2)\big)(T_1-T_2) + \frac{1}{k}\right] = \\ k\frac{\partial[T_2]}{\partial a_2}H(T_2-T_1) * \\ \left[\big(1-H(T_2-T_1)\big)(T_2-T_1) + \frac{1}{k}\right] \end{gathered} \tag{71}$$

The above equation has two possible solutions:

A. Case $T_1 = T_2$:

$$T_1 = T_2 \rightarrow f_1(a_1)\sum_{i=1}^{i=M} t_i b_{i,1} = f_2(a_2)\sum_{i=1}^{i=M} t_i b_{i,2} \tag{72}$$

Since $H(0) = 0.5$, the optimal solution becomes:

$$\frac{\partial[T_1]}{\partial a_1} = \frac{\partial[T_2]}{\partial a_2} \tag{73}$$

Thus:

$$f_1'(a_1)\sum_{i=1}^{i=M} t_i b_{i,1} = f_2'(a_2)\sum_{i=1}^{i=M} t_i b_{i,2} \tag{74}$$

Hence, at the optimal resource allocation point, adding an infinitesimal area would create the same improvement in the total execution time on any one of the utilized cores.

B. Case $T_1 \neq T_2$:

Assuming a very large k, even for a very small constant ε in case $T_2 \geq T_1 + \varepsilon$, (71) reduces to:

$$\frac{\partial[T_2]}{\partial a_2} \cong 0 \tag{75}$$

As established above, f_2 monotonically decreases with area, and thus has a negative derivative. Therefore, since t_i is positive and given that $b_{i,j} \geq 0$, the only solution for (75) is:

$$\sum_{i=1}^{M} b_{i,2} = 0 \quad \rightarrow \quad \forall i,\ \ b_{i,2} = 0 \tag{76}$$

Thus no task is assigned to core *2* and hence no area should be allocated to it. At the optimal point, $a_2 = 0$. Similarly, if $T_1 \geq T_2 + \varepsilon$, core *1* is discarded.

Due to the recursive nature of the max function (see (5)), the ensuing result can be extended to multi-core with any number of cores. For all utilized cores, $\forall j, k$ the following conditions hold at the optimal point:

A. First Optimality Condition: *All utilized cores must have the same runtime:*

$$f_j(a_j)\sum_{i=1}^{i=M} t_i b_{i,j} = f_k(a_k)\sum_{i=1}^{i=M} t_i b_{i,k} \tag{77}$$

B. Second Optimality Condition: *The first derivative of the execution time function across all utilized cores must be equal:*

$$f_j'(a_j)\sum_{i=1}^{i=M} t_i b_{i,j} = f_k'(a_k)\sum_{i=1}^{i=M} t_i b_{i,k} \tag{78}$$

Appendix B: Area and Power Constraint Optimization via Lagrange Multipliers

In this appendix, we detail the Lagrange optimization of the cost function (37) subject to constraints detailed in (6)-(8) and (40). To that end, we introduce two new variables (λ_1 and λ_2), area allocation $\bar{a} = \{a_1, a_2, \dots, a_N\}$, and power allocation $\bar{p} = \{p_1, p_2, \dots, p_N\}$, and define the Lagrange cost function as follows:

$$\begin{aligned} \Lambda(\bar{a},\bar{p},\lambda_1,\lambda_2):\ & \max(T_1, T_2, \dots, T_N) \\ & +\lambda_1\left[\sum_{j=1}^{j=N} a_j - A_{Total}\right] \\ +\lambda_2 & \left[\sum_{j=1}^{j=N}\left(s_1 a_j + s_2 p_j + p_j\right) - P_{Total}\right] \end{aligned} \tag{79}$$

The optimum is found when all partial derivatives equal zero:

$$\begin{aligned} \forall j, 1 \leq j \leq N \quad & \frac{\partial\Lambda}{\partial a_j} = 0,\ \ \frac{\partial\Lambda}{\partial p_j} = 0 \\ and \quad & \frac{\partial\Lambda}{\partial\lambda_1} = 0,\ \ \frac{\partial\Lambda}{\partial\lambda_2} = 0 \end{aligned} \tag{80}$$

The first derivative with respect to $a_j, \forall j, 1 \leq j \leq N$ follows:

$$\begin{aligned} & \frac{\partial[\max(T_1, T_2, \dots, T_N)]}{\partial a_j} + \lambda_1 \frac{\partial\left[\sum_{j=1}^{j=N} a_j - A_{Total}\right]}{\partial a_j} \\ & +\lambda_2 \frac{\partial\left[\sum_{j=1}^{j=N}\left(s_1 a_j + s_2 p_j + p_j\right) - P_{Total}\right]}{\partial a_j} = 0 \end{aligned} \tag{81}$$

As a_j, p_j are independent variables and A_{Total}, P_{Total} are constants, differentiating with respect to a_j and a_k where $\forall j,k, 1 \leq j,k \leq N$:

$$\begin{aligned} \frac{\partial[\max(T_1, T_2, \dots, T_N)]}{\partial a_j} + \lambda_1 + s_1\lambda_2 = 0 \\ \frac{\partial[\max(T_1, T_2, \dots, T_N)]}{\partial a_k} + \lambda_1 + s_1\lambda_2 = 0 \end{aligned} \tag{82}$$

Consequently, the Lagrange solution for $\forall j,k, 1 \leq j,k \leq N$ satisfies:

$$\frac{\partial[max(T_1, T_2, \dots, T_N)]}{\partial a_j} = \frac{\partial[max(T_1, T_2, \dots, T_N)]}{\partial a_k} \tag{83}$$

In a similar manner, when differentiating with respect to the power variables, the Lagrange solution for $\forall j,k, 1 \leq j,k \leq N$ satisfies:

$$\begin{aligned} \frac{\partial[max(T_1, T_2, \dots, T_N)]}{\partial p_j} + \lambda_2(s_2 + 1) = 0 \\ \frac{\partial[max(T_1, T_2, \dots, T_N)]}{\partial p_k} + \lambda_2(s_2 + 1) = 0 \end{aligned} \tag{84}$$

Thus,

$$\frac{\partial[max(T_1, T_2, \dots, T_N)]}{\partial p_j} = \frac{\partial[max(T_1, T_2, \dots, T_N)]}{\partial p_k} \tag{85}$$

Following the procedure detailed in section 0 (writing the *max* function in analytical manner using sigmoid, taking the derivative and analyzing the results), the following is derived:

A. First Optimality Condition: *All utilized cores must have the same runtime.*

$$f_j(a_j, p_j)\sum_{i=1}^{i=M} t_i b_{i,j} = f_k(a_k, p_k)\sum_{i=1}^{i=M} t_i b_{i,k} \tag{86}$$

B. Second Optimality Condition: *The first derivatives of the execution time functions across all utilized cores with respect to area and power must be equal.*

$$\frac{\partial f_j(a_j, p_j)}{\partial a_j}\sum_{i=1}^{i=M} t_i b_{i,j} = \frac{\partial f_k(a_k, p_k)}{\partial a_k}\sum_{i=1}^{i=M} t_i b_{i,k} \tag{87}$$

and

$$\frac{\partial f_j(a_j, p_j)}{\partial p_j}\sum_{i=1}^{i=M} t_i b_{i,j} = \frac{\partial f_k(a_k, p_k)}{\partial p_k}\sum_{i=1}^{i=M} t_i b_{i,k} \tag{88}$$