MULTIAMDAHL: HOW SHOULD I DIVIDE MY HETEROGENOUS CHIP?

Tsahee Zidenberg, Isaac Keslassy, Uri Weiser
EE Department, *Technion, Israel*
{tsahee@tx, isaac@ee, weiser@ee}.technion.ac.il

Abstract—Future multiprocessor chips will integrate many different units, each tailored to a specific computation. When designing such a system, a chip architect must decide how to distribute the available limited system resources, such as area and power, among all the computational units.
In this paper, we introduce MultiAmdahl, an analytical optimization technique for resource sharing among heterogeneous units. MultiAmdahl takes into account the workload, the performance of each computational unit, and the total available resource. The results obtained by MultiAmdahl allow us, for example, to provide a closed-form solution for an optimal asymmetric-offload chip, and to analyze the impact of different design constraints on an optimal chip architecture.

Index Terms—Chip Multiprocessors, Modeling of computer architecture

1. INTRODUCTION

Emerging heterogeneous multiprocessor chips will integrate a large number of different computational units: e.g., large cores for sequential processing, several smaller cores for parallel processing, Graphics Processing Units (GPUs), media accelerators, helper processors, Digital Signal Processors (DSPs), embedded FPGAs, and application-specific hardware circuits. However, chips have limited available resources, such as real estate or average/peak power. When a single chip contains multiple different units with different roles, it is up to the system architect to distribute the available resources among the different units. To reach an optimal division of resources, the architect should take into account the efficiency of these units as well as the workload.

Figure 1 schematically illustrates the application range and the performance/power ratio over this range for each computational unit of a heterogeneous chip. For instance, the general-purpose Central Processing Unit (CPU) core is designed to execute a wide range of applications, with an adequate average performance/power ratio. On the other hand, a designated accelerator, such as the Fast Fourier Transform (FFT) accelerator, is represented by a spike, providing a good performance/power ratio for a narrow range of applications.

In this paper, we present *MultiAmdahl*, an analytical

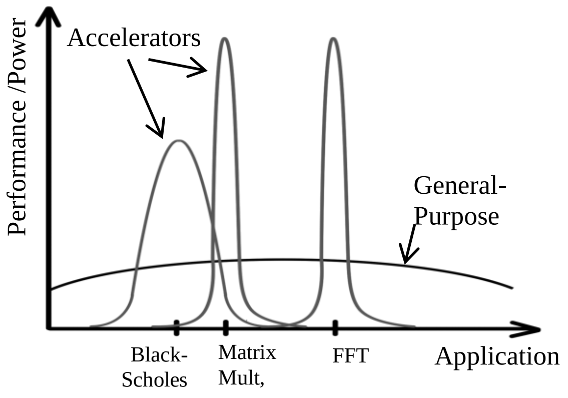

Fig. 1: Application range vs. performance/power: A more specific unit can be more efficient

optimization for resource sharing among heterogeneous units. Amdahl's Law for multiprocessors [3] considers the impact of accelerating one portion of the workload on the overall execution. In MultiAmdahl, we generalize Amdahl's Law from *two* types of executed environments (serial and parallel) to n types. We then define resource distribution as an optimization problem in this light, and provide closed-form solutions for simple cases.

MultiAmdahl differs from previously-published models [1], [2], [3], [4], [7] in that it describes a system with several heterogeneous hardware units. We also differ by directly modeling various design constraints and accounting for their impact.

Manuscript submitted: 16-Jan-2012. Manuscript accepted: 12-Feb-2012. Final manuscript received: 19-Feb-2012

2. MultiAmdahl Framework

The MultiAmdahl framework optimally distributes a limited resource among different units in the heterogeneous architecture. MultiAmdahl may consider different types of resources, such as the total average power, the instantaneous power, and the chip's area. For simplicity, in this section we illustrate a mathematical model and solution using an *area* constraint. Later, in Section 5, we show how the model can be generalized to a *power* constraint.

Workload — We divide the workload into n different execution segments. Each segment of the workload represents the aggregated amount of work that will be executed by a specific accelerator on the heterogeneous chip (Figure 2(a/b)). We define t_i as the execution time of segment i on a Reference CPU. Thus, the total execution time of the workload on the Reference CPU is $\sum_i t_i$.

Area Constraints — The chip area is divided among n hardware computational units (see Figure 2(c)), where each unit i executes segment i. We denote by a_i the chip area that is allocated to unit i. The sum of the areas assigned to all units is bounded by the total chip area A:

$$\sum_i a_i \leq A \tag{1}$$

Accelerators — Accelerator function f_i (see Figure 2(d)) represents the inverted speedup of the i-th accelerator as a function of the area a_i dedicated to the accelerator. Therefore, using the i-th accelerator, the execution time of segment i is $t_i \cdot f_i(a_i)$. An accelerator's performance intuitively increases when adding logic, thus area. For simplicity we assume the accelerator functions are strictly decreasing. We also assume that they are convex and continuously differentiable throughout this paper.

Our goal is to *minimize the total execution time when using accelerators* (see Figure 2(e)). Adding the area constraint, we arrive to the following optimization problem:

$$\begin{aligned} \text{minimize:} \quad & \mathrm{T} = \sum_i t_i\, f_i(a_i) \\ \text{subject to:} \quad & \mathrm{A} = \sum_i a_i \end{aligned} \tag{2}$$

This problem is solved using Lagrange multipliers [5]. First, we create the helper function:

$$\Lambda(\bar{a}, \lambda) = \sum_i t_i\, f_i(a_i) + \lambda\left(\sum_i a_i - A\right) \tag{3}$$

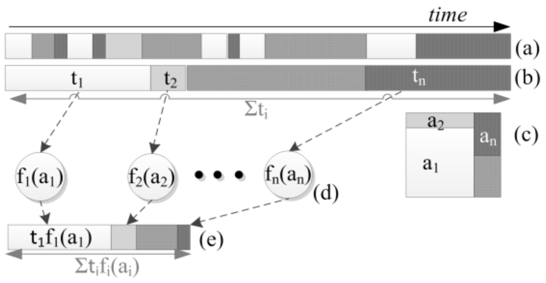

Fig. 2: MultiAmdahl framework: (a) Code segments' execution on a Reference CPU; (b) Aggregated execution time (c) Die area division; (d) Accelerator performance; (e) Execution time using the accelerators

An optimal solution satisfies:

$$\begin{cases} \dfrac{\partial\Lambda}{\partial a_i} = 0 \;\; \forall 0 \leq i \leq n \\ \qquad \dfrac{\partial\Lambda}{\partial\lambda} = 0 \end{cases} \tag{4}$$

Solving for arbitrary index i:

$$\frac{\partial\Lambda}{\partial a_i} = \frac{\partial \sum_i t_i\, f_i(a_i) + \lambda(\sum_i a_i - A)}{\partial a_i} = 0 \tag{5}$$

$$t_i f_i'(a_i) - \lambda = 0 \tag{6}$$

Solving for arbitrary index j, we obtain the dual formula:

$$t_j f_j'(a_j) - \lambda = 0 \tag{7}$$

and so the optimal solution occurs when:

$$\forall \boldsymbol{i}, \boldsymbol{j}: \qquad \boldsymbol{t_i f_i'(a_i) = t_j f_j'(a_j)} \tag{8}$$

where f_i' is the derivative of the i-th accelerator function, a_i is the area assigned to this accelerator, and t_i is the execution time of this segment on the Reference CPU. We observe that *the optimal performance under resource constraints is achieved when the weighted marginal cost/outcome is equal for all accelerators***.**

The intuition behind this rule is that in an optimal solution, any infinitesimal addition to the area creates the same improvement in the total execution time, regardless of the accelerator it is assigned to. If this did not hold, then removing some area from an accelerator that needs it less and giving it to an accelerator that needs it more would improve the solution.

Our optimization technique can be further generalized to deal with numerical optimization scenarios and discrete accelerator functions [5], which are beyond the scope of this paper.

3. Example: MultiAmdahl With Two Execution Segments

Chung *et al.* [2] and Woo *et al.* [7] introduced the *asymmetric-offload* model, based on (and slightly simpler than) the *asymmetric* model suggested by Hill and Marty [3] and Morad *et. al.* [4]. In both models, the architecture includes one large processor of size a_1, and the rest of the area, a_2, is filled with small processors, each of size 1. In the offload model, only the small cores are used in the parallel phase, and only the large core is used in the serial phase. The speedup of the asymmetric processor over a single small processor is given by:

$$Speedup = \frac{t_1 + t_2}{\frac{t_1}{perf(a_1)} + \frac{t_2}{a_2}} \tag{9}$$

where *perf(a₁)* is the performance of the large processor, and t_1 and t_2 are the amounts of time a single small core would spend on the serial and parallel segments, accordingly, if it executes the entire program on its own.

In the terms of MultiAmdahl, we refer to the single large processor as an accelerator for the serial phase, with the following accelerator function:

$$f_1(a_1) = \frac{1}{perf(a_1)} \tag{10}$$

The small cores are considered to be a single accelerator for the parallel function. Since the performance of the parallel section is assumed to be linear with the number of parallel processors, it is also linear with the total area assigned to parallel processors:

$$f_2(a_2) = \frac{1}{a_2} \tag{11}$$

Applying Equation (8) to this system, and using Pollack's Law [6], $perf(a_{serial}) = \sqrt{a_{serial}}$, our MultiAmdahl model provides the following optimal allocation, which is a novel result:

$$a_2 = a_1^{3/4}\sqrt{\frac{2t_2}{t_1}} \tag{12}$$

This formula reveals the optimal relation between the serial large core and the area dedicated to small cores. It shows, for example, that when total area budget increases, the size of the serial core grows faster than that of the serial accelerator. Different results could be similarly obtained for different performance/area functions.

Figure 3(a) reveals the existence of an optimal resource allocation, which changes according to the workload characterized by $t_{parallel} = \frac{t_2}{t_1+t_2}$. For instance, with $t_{parallel} = 0.99$, the optimal allocation consists of $a_1 = 39$ and $a_2 = 217$. Figure 3(b) presents the exact relation between $t_{parallel}$ and the optimal value of a_1 using the MultiAmdahl framework.

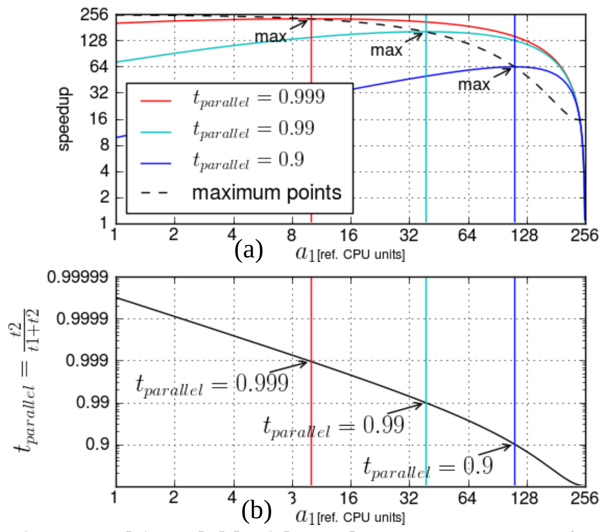

Fig. 3: MultiAmdahl with total area $a_1 + a_2 = 256$; (a) Speedup per serial CPU size per workload; (b) Optimal CPU size for each workload

As the total area budget grows, the area of the serial section grows faster than the area of the parallel section.

4. Modeling Power Constraints

In the previous section, we modeled the area allocation among the different units on a chip. Area is a static resource, i.e. units use their entire assigned resource budget for the full execution time.

Power, however, has both a static and a dynamic component. Distribution of these resources among the units is not independent. We model the dependency between the two parameters as a function. We annotate the dynamic power used by unit *i* as p_i, and the static power consumption of the unit is calculated by the function $s_i(p_i)$.

Power constraints are often set by the heat dissipation, which manifests over long periods of time. Accordingly, we concentrate on the average power. The average power consumption is calculated by dividing the overall energy by the overall execution time. Formally, the constraint is:

$$P_{static} + \frac{E_{dynamic}}{T_{exec}} = \sum_i s_i(p_i) + \frac{\sum_i t_i f_i(p_i) p_i}{\sum_i t_i f_i(p_i)} \le P_{budget}$$

where P_{budget} is the maximal allowed average power consumption. An optimal power allocation is achieved when minimizing a target function $T = \sum_i t_i f_i(p_i)$ under this constraint. This problem is solved in a manner similar to the area-bound problem.

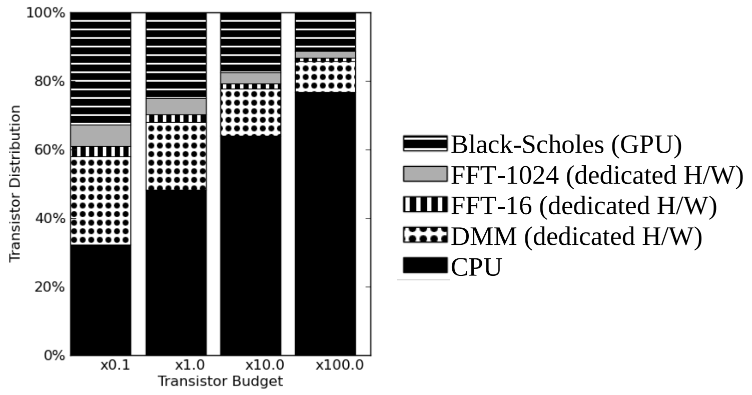

Fig. 4: Transistor count model: The transistor budget increase reduces the relative budget of the accelerators.

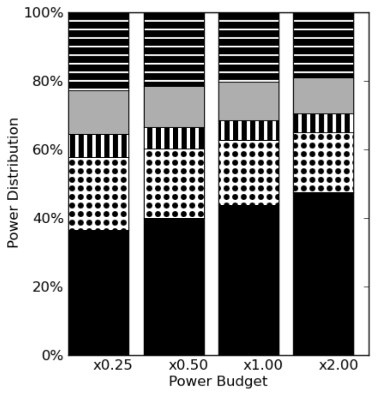

Fig. 5: Power model: The power budget increase reduces the relative budget of the accelerators.

5. Example: Varying System Budget

We modeled an example workload equally distributed between 4 different benchmarks. 10% of each benchmark is executed on the general purpose CPU and 90% on its designated accelerator. We assume accelerated code is essentially parallel and therefore scales linearly with resource. The relation between the area / power efficiencies of the CPU and the various accelerators was retrieved from E.S. Chung *et al.* [2].

We analyze the optimal distribution of resources as we vary (a) the transistor budget in the area model (Figure 4), and (b) the power budget in the power-distribution model (Figure 5).

The equations used to model the power resources are quite different from the equations used to describe the area resources. However, both results show that the least scalable part of the system grows most rapidly when the total budget increases. This fits the current industry trend, which uses accelerators heavily on mobile or power-efficient platforms. On the other hand, our model shows that high-power/ high-performance chips will be dominated by the general-purpose cores.

6. Conclusion And Future Work

MultiAmdahl is an analytical model that takes into account both the characteristics and the application range of each computational unit. Using this model, we present the insight that general-purpose cores are expected to consume a substantial fraction of the resource of high-end chips.

MultiAmdahl is applicable to different resource constraints, efficiency models, and objective functions. Although we focused on computational units, it could be used to find the optimal distribution of resources in various architectural levels from within a CPU core, through the board level and up to large systems. It could be applied to additional resources, such as communication bandwidth and cost.

7. Acknowledgement

The authors wish to thank N. Azuelos, E. Chung, B. Dally, M. Hill, R. Iyer, I. Keidar, Y. Michael, T. Morad, S. Wimer, R. Ronen, O. Rottenstreich, A. Shpiner , K. Singh, G. Srinivasa and I. Tolchinsky for their helpful comments and support.

This work was partly supported by the European Research Council Starting Grant No. 210389 and by the Intel Heterogeneous Computing research grant.

8. References

[1] A.S Cassidy and A.G. Andreou. Beyond Amdahl's law: An objective function that links multiprocessor performance gains to delay and energy. *IEEE Transactions on Computers*, 2011, accepted for publication.

[2] E. S. Chung, P. A. Milder, J. C. Hoe, and K. Mai. Single-chip heterogeneous computing: Does the future include custom logic, FPGAs, and GPGPUs? In *MICRO-43*, 2010.

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

[4] T. Y. Morad, U. C. Weiser, A. Kolodny, M. Valero, E. Ayguadé. Performance, power efficiency, and scalability of asymmetric cluster chip multiprocessors. *Computer Architecture Letters*, vol. 4, 2005.

[5] R. T. Rockafellar. Lagrange multipliers and optimality. *SIAM Review*, vol. 35, pp. 183–283, 1993.

[6] F. J. Pollack. New microarchitecture challenges in the coming generations of CMOS process technologies. *Keynote address*, In *MICRO-32*, 1999.

[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.