MultiAmdahl: How Should I Divide My Heterogeneous Chip?

Tsahee Zidenberg, Isaac Keslassy, Uri Weiser

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, power and energy among all the computational units.

We introduce MultiAmdahl, an analytical optimization technique for resource sharing among heterogeneous units. We identify the optimal resource allocation in the unit using the performance of each computational unit, the workload, the target design goal, and the total available resource. We conclude by analyzing the impact of our optimization framework on chip design.

Background

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. These units are designed specifically for a particular workload, allowing them to be heavily customized, and therefore much more efficient than general-purpose units.

Heterogeneity also has a price; chips have limited available resources, such as die area or average/peak power/energy. All units on the chip compete for the shared resources, and the share that each unit receives is limited. When a single chip contains multiple different units with different roles, it is up to the architect to distribute the resources among the different units. To reach an optimal solution, the architect should take into account the efficiency of these units as well as the workload.

This manuscript is an extended version of [Zidenberg , 2012]. This submission includes several new technical contributions, including an additional section to provide a more intuitive explanation of the resource allocation problem, further analysis and examples of accelerator functions, reworked graphs in Figure 3, and more details on the heterogeneous system test-case, including new results presenting the limits of heterogeneity.

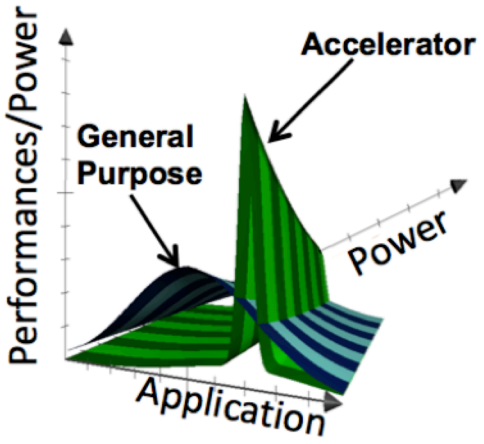


Fig.1: Application range vs. performance/power:
A more specific unit can be more efficient

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 a Fast Fourier Transform (FFT) accelerator, is represented by a spike, providing a good performance/power ratio for a narrow range of applications. The additional axis presents the amount of power allocated for the unit. Investing additional power resource in an existing accelerator can of course increase its performance. However, performance typically does not increase linearly with power, i.e. the performance/power efficiency of the accelerator decreases given more power resources.

In this paper, we present the *MultiAmdahl* framework, an analytical model of the resource distribution problem in heterogeneous systems. We describe the model, present test cases including closed-form solutions for simple cases, and provide intuition regarding the optimal resource distribution.

The Resource Allocation Problem

Consider a heterogeneous system that is composed of several different units, each with a different role. A common resource (for example, the die area) is shared between the different units. The system architect needs to distribute the resource between the available units. Increasing the size of one unit will improve its performance, but also force all other units to share a smaller area, thus reducing their performance. To find the optimal resource allocation, four parameters must be considered.

The first parameter is the overall *resource constraint*. In this paper, we start by considering the die area constraint, and later discuss a power constraint. We can also introduce additional resource constraints, such as cost or energy.

The second parameter we consider is the *accelerator function*, meaning how the additional resource (e.g. area, power, energy) assigned to a given accelerator is translated into a better target performance. For example, a graphic core is a massively parallel unit, therefore, doubling its size is likely to nearly

double its potential performance. On the other hand, a serial general-purpose processor is not likely to provide the same improvement ratio when scaling its size.

To allocate resources wisely, the system architect must also decide how to spend the resource within each unit. In the CPU example – choosing between adding execution pipes, predictors, larger ALUs, cache, etc. Accelerator functions hide these details, always providing the *optimal* output of the unit given the amount of resource it was assigned.

A third parameter of the optimization problem is the *workload* of the system, and especially its distribution amongst the different units. The workload determines how much work is assigned to each of the accelerators. In common scenarios, the accelerator-functions are known *a priori*, while specific details of the workload are only estimated.

The last parameter we discuss is the *design goal*, which the system architect seeks to optimize for. A design constraint must be met, while the design goal presents one design as preferable to another. A design may have several constraints (power, area, budget...), but only a single goal. If, for example, a system requires optimization of both energy and delay, a single criterion must be chosen to compare a low-energy high-delay with a low-delay high-energy system. One such criterion is Energy*delay.

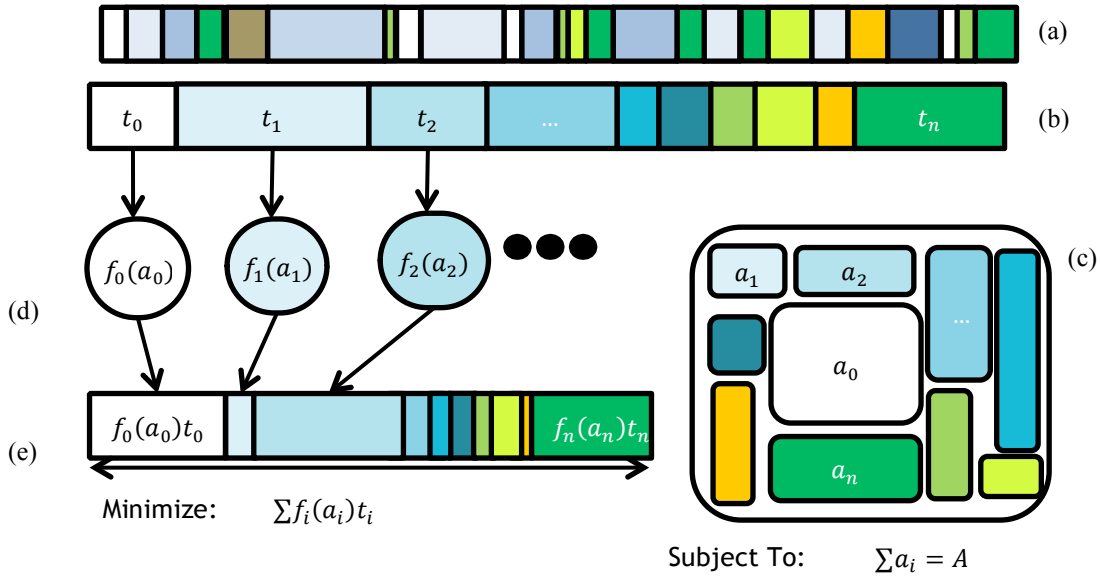


Fig. 2: MultiAmdahl framework:
(a) Code segments' execution on a Reference CPU (b) Aggregated execution time
(c) Die area division (d) Accelerator performance (e) Execution time using the accelerators

A Motivating Example

We present an analytical model for the four entities described in the previous section. For simplicity, the mathematical model and solution is presented for a problem with an *area* 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:

$$\sum_i a_i \leq A \qquad (1)$$

Accelerator functions — 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 $t_i \cdot f_i(a_i)$. When relevant, this function should take into account the migration overhead of the appropriate accelerator [Morad, 2012]. To simplify the analysis, we assume that the accelerator functions are strictly decreasing, convex and continuously differentiable throughout this paper.

Design Goal — 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:

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

Because we assume the accelerator functions are strictly decreasing, we know the solution satisfies $\sum_i a_i = \mathrm{A}$. This allows solving the optimization problem using Lagrange multipliers [Rockafellar, 1993]. First, we create the helper function:

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

An optimal solution satisfies:

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

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:

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

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.

Sidebar: Accelerator function approximation

Analytical models transforming the amount of resource used by a unit to execution time are, by their nature, rough approximations. Transforming added resource to added performance is a process that requires re-implementation and re-design . This process is hard and inconclusive — there is always a chance that better results could be achieved using a better design.

MultiAmdahl optimizes resource allocation for specific design families, assuming some boundaries for the sizes of available accelerators. Within these boundaries, an approximation for the accelerator's performance can be made, using some commonly accepted properties [Pollack, 1999] [Hill, 2008] [Woo, 2008] [Chung, 2010]. Notice that in our model a lower value for the accelerator function means less execution time, which is therefore better:

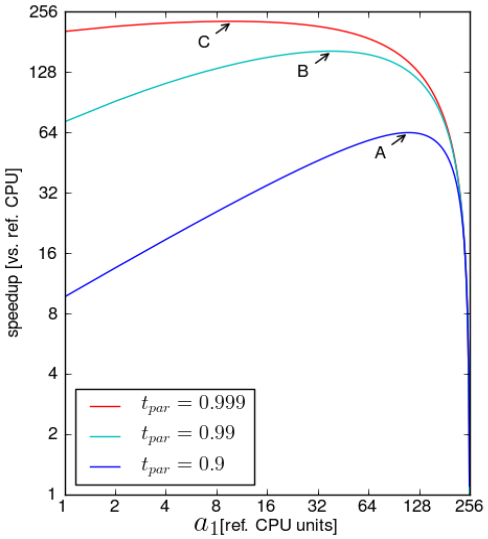
- An accelerator working with zero resource should take infinite time to execute its workload.
- Adding resource to the accelerator should not worsen its execution time. Thus, accelerator functions are non-increasing. An accelerator that is assigned infinite resource should take zero time to execute its workload.
- Accelerators present a "diminishing return". Meaning – spending twice the amount of resource will improve execution time by no more than twice [Kumar, 2003]. Formally: $f(kx) \geq \frac{1}{k} f(x)$.

A simple form of analytical functions that satisfy all of the above properties is a power-law function:

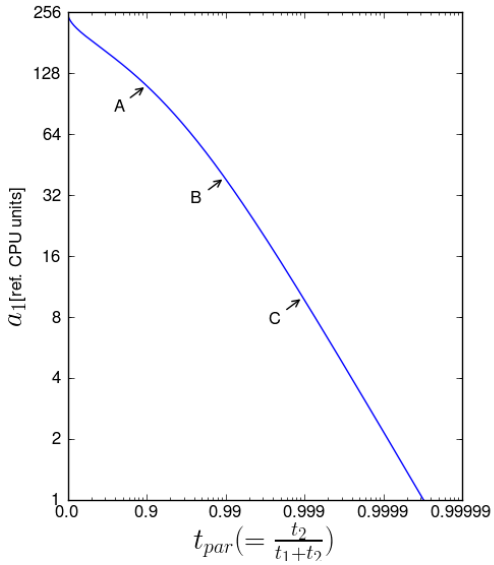
$$f(x) = \alpha x^{-\beta}, \tag{1}$$

where x is the resource assigned to the processor, and α and β are constant coefficients, $\alpha > 0$, $1 \geq \beta > 0$.

Pollack [Pollack, 1999] suggests that the performance of a single-core CPU can be modeled as following the square root of the assigned area. In our notations, $\beta = 0.5$ for single-core CPU performance when considering area resource. In embarrassingly parallel problems, on the other hand, we can model performance as linear in the number of allocated cores (or resource amount). In our notations, these problems are characterized by $\beta = 1$.



(a) Analysis: Speedup per serial CPU size, with a total chip size of 256.



(b) Optimization: Optimal serial CPU size per workload, as found using MultiAmdahl.

Figure 3: Area allocation for asymmetric cores

MultiAmdahl With Two Execution Segments

Chung *et al.* [Chung, 2010] and Woo *et al.* [Woo, 2008] introduced the *asymmetric-offload* model, based on (and slightly simpler than) the *asymmetric* model suggested by Hill and Marty [Hill, 2008] and Morad *et. al.* [Morad, 2005]. 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 with a normalized size of 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-offload processor over a single small processor is given by:

$$Speedup = \frac{t_1+t_2}{\frac{t_1}{perf(a_1)}+\frac{t_2}{a_2}}, \quad (2)$$

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.

MultiAmdahl has a different approach to the problem: Instead of calculating the speedup for each area allocation, we calculate the optimal unit area allocation for each workload. In the terms of

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

$$f_1(a_1) = \frac{1}{perf(a_1)} \tag{3}$$

The small cores are jointly 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:

$$\mathrm{f}_2(a_2) = \frac{1}{a_2} \tag{4}$$

Applying Equation (8) to this system, and using Pollack's Law [Pollack, 1999], $perf(a_1) = \sqrt{a_1}$, our MultiAmdahl model provides the following optimal allocation, which is a novel result:

$$a_2 = a_1^{3/4}\sqrt{\frac{2t_2}{t_1}} \tag{5}$$

This formula reveals the optimal relation between the serial large core and the area dedicated to small cores. It shows that when total area budget increases, the size of the serial core grows faster than that of the parallel 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.

Sidebar: Modeling the Power Resource

When creating an analytical model for the Power Resource, two aspects must be considered. The first is how to describe the dependence between the CPUs power consumption and its performance. One option is assuming a linear relation between area and power consumption [Hill, 2008]. Grochowski and Annavaram [Grochowski, 2006] suggested a model translating power to CPU performance using a power-law that in many ways resembles Pollack's Law. Hampsteed *et al.* [Hampsteed, 2009], on the other hand, created a complicated model that distinguishes wire from transistor power.

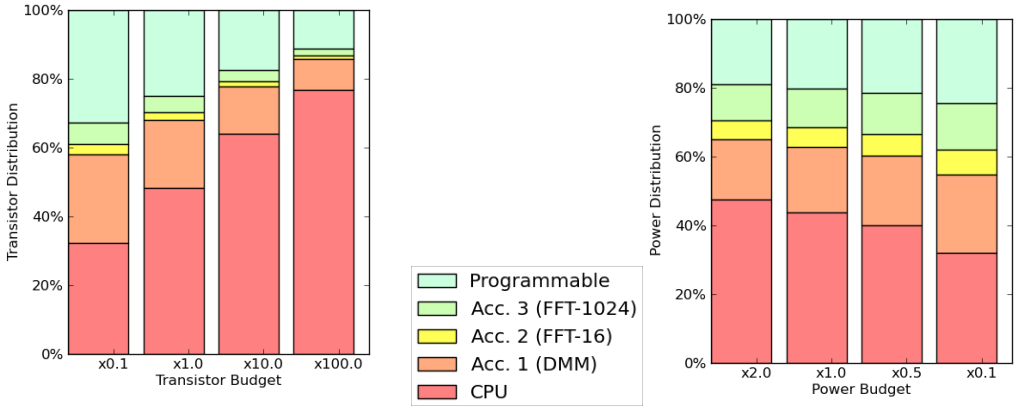
The second aspect with modeling Power is the relation between the dynamic power, which is consumed by the unit only when it is active, and static power, which is always consumed. Many past

models choose to model power as entirely static [Hill, 2008] or entirely dynamic [Chung, 2010]. An acceptable approximation assumes a linear relation between the two [Woo, 2008].

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:

$$P_{static} + \frac{E_{dynamic}}{T_{exec}} = \sum_i k_i p_i + \frac{\sum_i t_i f_i(p_i) p_i}{\sum_i t_i f_i(p_i)} \leq P_{budget} \qquad (6)$$

where P_{budget} is the maximal allowed average power consumption, p_i the dynamic power assigned to unit i, and k_i the coefficient for static power usage (we chose 0.5 for all models). 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.



(a) Process scaling
The process scaling trend increases the relative budget of the CPU.

(b) Different power budgets
The power reduction trend reduces the relative budget of the CPU.

Figure 4: Varying System Budget

Impact of Varying the System Budget

In this section we use MultiAmdahl to test the effect of varying the system budget on the internal resource allocation. To do so, we use data collected by Chung *et. al* [Chung, 2010] and presented in Table 1. We use an arbitrary workload, composed of the various benchmarks measured by Chung in equal part. 10% of each benchmark is assumed to be executed on a serial general-purpose CPU and the rest on the parallel accelerators. Since we assume the benchmarks are embarrassingly parallel, the accelerator functions take the form:

$$f_i(x_i) = \frac{1}{\alpha_i x_i}, \tag{7}$$

where the value for α_i is retrieved from Table 1. The accelerator function for the general-purpose CPU is taken from Grochowski and Annavaram's model [Grochowski, 2006] for power calculations and Pollack's law [Pollack, 1999] for area calculations. As before, only one unit in the chip is active simultaneously.

First we consider the effect of *process scaling*, which increases the available transistor budget in any new technology generation. The calculations use the area efficiency parameters of the accelerators described in the previous section. Figure 4 (a) reveals that the larger the transistor budget is, the larger the CPU's part in it is. From this graph we may understand that in an environment of process scaling, the weight of the CPU increases.

Figure 4 (b) presents the implications of the decreasing power budgets trends in chip design. *As the power budget shrinks, the optimal distribution assigns an increasing part of the available power to accelerators*. We may conclude that mobile or power-efficient chips should rely more on accelerators.

Both results look similar, even though the equations used to model the power resources are quite different from the equations used to describe the area resources. As also illustrated in the previous section, the intuition is that when the total budget increases, *the least scalable part of the system grows most rapidly*. Thus a larger chip, or a chip with a larger power budget, will hold a larger general-purpose unit. This is a good example for insights and trends shown under simplifying assumptions, which still hold for more complex cases.

Table 1: Accelerator Efficiencies

Benchmark	Area efficiency	Power Efficiency
Black-Scholes (programmable accelerator)	x24	x38.7
FFT-1024 (designated accelerator)	x692	x127
FFT-16 (designated accelerator)	x2804	x452
Dense Matrix Multiplication (designated accelerator)	x39	x44

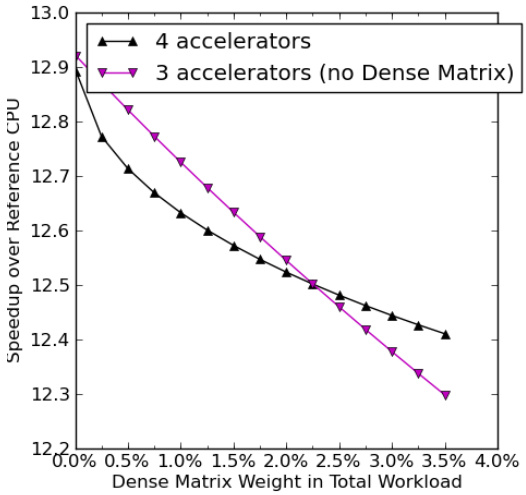


Figure 5: Removing one accelerator
Efficient-enough accelerators are beneficial even if seldom used.

The Limits of Heterogeneity

Designated accelerators might reduce performance even in cases where the exact workload is known *a priori*. If the designated accelerator only executes a small part of the workload, the cost of the resources it consumes might surpass its benefit.

We test the effect of making the design presented in the previous section less heterogeneous by removing the Dense Matrix Multiplier accelerator . The Black-Scholes accelerator presented in Table 1 is in fact a programmable GPU executing Black-Scholes. The same GPU can also execute the DMM benchmark, with a greatly reduced power efficiency (x5.94 instead of x44). Even though a less efficient accelerator is used, the resource is split between less accelerators, and so the remaining accelerators are larger in the three-accelerator system than they were in the four-accelerator system.

The more weight the DMM benchmark has in the overall workload, the better it is to include a DMM accelerator in the design. Figure 5 presents this relation. The x-axis presents the weight of the DMM benchmark in the workload. The remaining workload is equally split between the remaining 3 benchmarks. The two lines present the respective speedups of the optimal three- and four-accelerator systems. Workload is assumed to be known *a priori*, and so the optimal distribution is calculated separately for each point in the graph. In our system, if the DMM benchmark is estimated to be less then 2% of the workload, an optimal design would not include a DMM accelerator.

This presents an inherent limit of heterogeneity. Adding a small specific accelerator will cause a slowdown for its designated benchmark, because it will not match the speedup of the general-purpose accelerator in the less heterogeneous design. On the other hand, a large specific accelerator will create a slowdown for all other benchmarks, because it reduces the resource available to all other accelerators.

Two previously-presented principles mitigate that effect. If the accelerators present a diminishing return, moving a small amount of resource from a large accelerator to a new one will have a small effect on the performance of the large accelerator and a large effect on the performance of the new accelerator. Furthermore, if the resource allocated is partially dynamic, allocating resources to the new accelerator requires a lower reduction of the resources allocated to the rest of the units.

Previous Models

Asymmetric Multicore Models

Most existing analytical models deal with homogeneous multicore processors, either symmetric or asymmetric. The MultiAmdahl framework models heterogeneous multiprocessors.

Gene M. Amdahl presents the inherent limits of multi-processor (or multi-computer) machines [Amdahl, 1967]. The intuition states that part of the workload is inherently serial, and can only be executed on one processor. The serial section, even if small, will become the bottleneck of the system if enough parallelization is added. This intuition was later developed into an analytical formula known as Amdahl's Law.

Morad *et al.* [Morad, 2005], and Hill and Marty [Hill, 2008] create an initial analytical model revealing the relationship between the core sizes inside the chip and the workload executed, under a limiting budget. All units in the chip share a resource. This means that the system architect has a choice weather to use many small cores or few large ones. The serial section of the workload executes faster on a chip with larger, and therefore fewer, cores, while the parallel section executes faster on a chip with more cores, which are therefore smaller. The tradeoff between the two depends on the exact workload, and the relation between its serial and parallel sections. The models present the validity of the *asymmetric* approach – including in the chip both large cores, which will provide single-thread speedup for the serial section, and many small cores, which will provide more speedup for the parallel section.

Woo *et al.* [Woo, 2008] extend this model for *Performance/Watt* and *Performance/Joule* measures. They compare two symmetric systems composed of full-blown or efficient cores with an asymmetric-offload system. Asymmetric-offload is a system with one full-blown that is only used by the serial section, and many efficient cores which are only used by the parallel section. The three systems are compared with one another by limiting them to an equal power budget. Power is considered to have both dynamic and static components – processors use more power when they are active. The asymmetric multicore shows better results for almost any goal or budget.

Heterogeneous Multiprocessor Models

Chung *et al.* [Chung, 2010] also extend Hill and Marty's model. Their model takes into consideration three different constraints: area, power, and memory bandwidth. Power is modeled as a dynamic resource – meaning a unit does not consume power when it is idle. Three chip architectures are tested, which are symmetric multicore, asymmetric-offload multicore, and heterogeneous. The last model is an extension of the asymmetric-offload model, but “unconventional cores” replace the efficient cores. The unconventional cores are modeled as cores with improved area and power efficiency. The model assumes that the entire area dedicated to unconventional cores in the model can execute the entire parallel portion of the workload. The MultiAmdahl framework, in comparison, takes into account both the advantages and disadvantages of heterogeneous chips. Most importantly, we

model the fact that the chip is divided into many different accelerators, and that only some of those accelerators work at any given time.

Hampsteed *et al.* [Hampsteed, 2009] also create a model for systems offloading work to an accelerator. Such a system is defined by the amount of work that is offloaded to the accelerator and the accelerator speedup. No constraints or resource distributions are considered. In MultiAmdahl resource distribution is at the heart of the optimization problem. The actual speedup achieved by the accelerator or CPU depends on the amount of resource allocated to them.

Conclusion

We presented MultiAmdahl, an analytical framework for optimal resource allocation in the heterogeneous environment. Our technique relies on modeling the performance of each unit as a function of the resources it uses, on modeling the workload, and on setting the limited resource as a constraint. We then find the optimal solution using Lagrange multipliers.

In this paper, we presented a few insights from the MultiAmdahl framework:

1. We illustrated how the optimal target performance under resource constraints is achieved when the absolute marginal outcome is equal for all accelerators.

2. We introduced an analytical solution for the previously presented asymmetric-offload multicore chip.

3. We showed how adding resources to a chip (e.g. more transistors or more power) increases the size of the general-purpose accelerators. We deduced that chips with a vast amount of power and/or area will be dominated by the general-purpose cores, whereas mobile and power-efficient chips will be dominated by accelerators

MultiAmdahl can be developed in many future directions. Morad *et al.* [Morad, 2012] have added overhead and saturation points to MultiAmdahl. Other possible directions include modeling multiple resources, or testing the effect that workload changes have on the optimal resource allocation point. MultiAmdahl can be used in various scales, from within the cores to full systems.

In summary, the MultiAmdahl framework provides a wide range of optimization opportunities, and a multi-functional tool for analytic exploration of chip-architecture design options.

Acknowledgments

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

This work was partly supported by the Intel ICRI-CI Center, by an Intel Heterogeneous Computing research grant, and by European Research Council Starting Grant No. 210389.

References

[Amdahl, 1967] G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities, *Proceedings of* AFIPS '67, pp. 483-485.

[Rockafellar, 1993] R. T. Rockafellar. Lagrange multipliers and optimality. *SIAM Review*, vol. 35, 1993, pp. 183-288

[Pollack, 1999] F. J. Pollack. New microarchitecture challenges in the coming generations of CMOS process technologies (keynote address). *Proceedings of MICRO-32*. IEEE, 1999, pp. 2.

[Kumar, 2003] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen. Single-ISA Heterogeneous Multi-core Architectures: The Potential for Processor Power Reduction. *Proceedings of MICRO-36*, IEEE, 2003, pp. 81-92.

[Morad, 2005] 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.

[Grochowski, 2006] E. Grochowski and M. Annavaram. Energy per instruction trends in Intel microprocessors. *Technology@Intel Magazine* 4(3), 2006, pp. 1-8.

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

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

[Hampsteed, 2009] M. Hempstead, G.-Y. Wei, and D. Brooks. Navigo: An early-stage model to study power-constrained architectures and specialization. *ISCA Workshop on Modeling, Benchmarking, and Simulations (MoBS)*, 2009

[Chung, 2010] 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? *Proceedings MICRO-43,* 2010, pp. 225-236.

[Zidenberg, 2012] T. Zidenberg, I. Keslassy, U. Weiser. Multi-Amdahl: How Should I Divide My Heterogeneous Chip? *Computer Architecture Letters,* Mar. 2012

[Morad, 2012] A. Morad, T. Y. Morad, L. Yavits, R. Ginosar and U. Weiser. Generalized MultiAmdahl: Optimization of Heterogeneous Multi-Accelerator SoC. *CCIT report #819,* Technion, Israel.

Tsahee Zidenberg is a B.Sc. graduate and an M.Sc. student in the electrical engineering department of the Technion, Israel. His research interests include CPU and SOC architectures. Contact him at tsahee@annapurnalabs.com

Isaac Keslassy is an associate professor in the electrical engineering department at the Technion, Israel. His research interests include the design and analysis of high-performance routers and on-chip networks. He received a PhD in electrical engineering from Stanford University. He is a senior member of IEEE. Contact him at isaac@ee.technion.ac.il

Uri Weiser is a Professor at the Electrical Engineering department at the Technion, Israel, and acts as an advisor at numerous hi-tech companies. His research interests include memory and heterogeneous systems. He received a PhD in computer science from the University of Utah, Salt Lake City. He is an IEEE and ACM Fellow. Contact him at uri.weiser@ee.technion.ac.il