Generalized MultiAmdahl: Optimization of Heterogeneous Multi-Accelerator SoC

Amir Morad, Tomer Y. Morad, Leonid Yavits, Ran Ginosar and Uri Weiser

Abstract— Consider a workload comprising a consecutive sequence of program execution segments, where each segment can either be executed on general purpose processor or offloaded to a hardware accelerator. An analytical optimization framework based on MultiAmdhal framework and Lagrange multipliers, for selecting the optimal set of accelerators and for allocating resources among them under constrained area is proposed. Due to the practical implementation of accelerators, the optimal architecture under area constraints may exclude some of the accelerators. As the fraction of the workload that can be accelerated decreases, resources (e.g. area) may shift from accelerators into the general purpose processor. The framework can be extended in a number of ways, spanning from SoC partitioning, bandwidth to power distribution, energy and other constrained resources.

Index Terms— MultiAmdahl, Chip Multiprocessors, Modeling of computer architecture.

1 Introduction

Large scale multiprocessor SoCs may be composed of several heterogeneous processing elements. These elements, spanning from DCT and motion estimation to GPUs, are customized hardware accelerators that are specifically designed to speed up certain tasks. Today's SoC architects have at their disposal a wide range of such heterogeneous accelerators to utilize. It is up to the system architect to efficiently allocate the resources among the accelerators, while considering physical limitations (area, power) of the design. To reach an optimal solution, the architect should take into account the efficiency of these units under resource constraints as well as the specifics of the workload.

Several previous studies use analytic models to derive optimal resource allocation of multiprocessors. Zidenberg et al. [9] presented the MultiAmdahl model that considers the implications of accelerating portions of the workload on the overall execution. Cassidy et al. [2] optimized processor area vs. L2 cache area vs. number of cores for a parallel-execution area-constrained symmetric multicore using Lagrange multipliers. MultiAmdahl differs from previously-published models [2], [3], [5], [8] by describing a system with several heterogeneous hardware units operating in sequence, directly modeling various design constraints and accounting for their impact.

The MultiAmdahl model provides the framework to resolve the following question: given an architecture composed of a predefined set of accelerators, what is the optimal resource allocation among the accelerators. The architect therefore must first select, prior to the optimization stage, a set of accelerators to use, and then allocate the resources among the selected accelerators.

At the optimal resource allocation point, all accelerators reach an equilibrium state in which if some area is taken from one accelerator and given to another, the overall runtime increases. While searching for such optimal point, one must take into account all accelerators' characteristics, total area constraints and the workload specifics. Thus, any a-priori, ad-hoc selection criteria based on performance threshold or otherwise (let alone universal criteria) electing a subset of the accelerators may yield only sub-optimal results. Hence, the key contribution of this paper is to enable the SoC architect to select an optimal subset of accelerators and allocate the area among them, without having to iterate in an ad-hoc fashion. In this paper we thus generalize the original MultiAmdahl framework to allow the optimal selection of a subset of available accelerators given the workload and the resource budget.

2 Optimization of Multi-Accelerator SoC

The MultiAmdahl [9] model optimally distributes a limited resource, such as chip area or power, among the units of a heterogeneous chip. The model assumes a workload consisting of M+1 execution segments. Each segment i of the workload requires t_i seconds to run on a reference processor. Assume that each segment i can be accelerated by using a special purpose accelerator, where each accelerator's performance speedup $Perf_j(a_j)$, relative to performance on the reference processor, is a function of the area assigned to it. The acceleration function $f_i(a_i)$ represents the inverted performance speedup of the i^{th} accelerator having area a_i:

$$f_i(a_i) = \frac{1}{Perf_i(a_i)} \quad (1)$$

- *Amir Morad (*), E-mail: amirm@tx.technion.ac.il.*
- *Tomer Y. 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.*
- *Uri C. Weiser (*), E-mail: uri.weiser@ee.technion.ac.il.*

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

Manuscript received (July 5th, 2012).

Posted to the IEEE & CSDL on 10/5/2012
DOI 10.1109/L-CA.2012.34 1556-6056/12/$31.00 © 2012 Published by the IEEE Computer Society

Assume that the acceleration functions $f_i(a_i)$ are monotonic and continuously differentiable. The runtime of the i^{th} segment on the i^{th} accelerator is thus $f_i(a_i) \cdot t_i$. It is further assumed that the first accelerator, namely i=0, is a General Purpose Processor (GPP) responsible for the execution of the portion of the workload that may not be offloaded to any of the hardware accelerators, whereas the rest of the accelerators are application specific. Given the area allocation $A = \{a_0, a_1, \dots, a_M\}$ among the cores, the total execution time of the workload is thus:

$$T_{MA} = \sum_{i=0}^{i=M} f_i(a_i)\, t_i \tag{2}$$

where MA stands for MultiAmdahl [2], and t_i represents the aggregated time devoted to part of the workload that can be accelerated by the i^{th} accelerator. Note further the segments are not concurrent, that is, only a single accelerator operates at a time. The sum of the areas assigned to all units is bounded by the total chip area A_{Total}:

$$\sum_{i=0}^{i=M} a_i \le A_{Total} \tag{3}$$

As the accelerator functions $f_i(a_i)$ are continuously differentiable, the optimal solution is then found using Lagrange multipliers [6], and satisfies:

$$t_i f_i'(a_i) = t_j f_j'(a_j) \tag{4}$$

where $f_i'(a_i)$ is the derivative of the i^{th} acceleration function. MultiAmdahl provides the optimal execution time when the accelerators deliver higher performance than the GPP. Due to practical implementation of accelerators, this precondition is not always met.

Figure 1 depicts a practical model for accelerators, where the accelerator is useful within a limited range, $A_{min_i} \le a_i \le A_{max_i}$.The area A_{min_i} represents the minimal area that must be assigned to the i^{th} accelerator, below which it is ineffective, and A_{max_i} is the diminishing return point beyond which any further area assignment has negligible effect on performance.

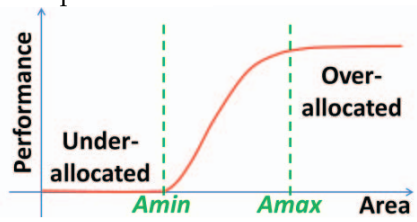

Figure 1. Practical accelerator model (Performance vs. area)

A_{min} is an inherent property of any accelerator as minimal area is required for the accelerator's proper functioning (e.g. basic functionality logic, storage etc.). Moreover, A_{min} incorporates overhead reserved for SoC interconnectivity (e.g. area reserved for bus-interface, arbitration logic and the like). A_{min} accounts for a significant portion of the total area of smaller accelerators. Furthermore, any area assignment below the accelerator's A_{min}will effectively render such accelerator inoperable. For instance, consider an FFT accelerator. The minimum useful area may be a single hardware butterfly unit coupled with the SoC inter-connectivity logic, and the maximum useful area could comprise the largest number of butterfly units that can be active in parallel for a given problem size. There is thus an effective range in which the accelerator outperforms the GPP and therefore should be utilized. Let's assume we are provided with a GPP and M accelerators such that:

$$\sum_{i=0}^{i=M} A_{min_i} > A_{Total} \tag{5}$$

Only a subset of the accelerators can therefore be integrated. The research question is thus generalized as follows: *which subset of the accelerators should be integrated and what is the optimal resource allocation among them?* Since the optimal resource allocation may exclude several accelerators from the potential set, we therefore revise the execution time (2) to select the faster option for each task i:

$$T_{GMA} = f_0(a_0)t_0 + \sum_{i=1}^{i=M} F_i(a_0, a_i)\, t_i \tag{6}$$

where:

$$F_i(a_0, a_i) = \min\left(f_0(a_0), f_i(a_i)\right) \tag{7}$$

Note that GMA stands for the "Generalized MultiAmdahl" model (in which GPP takes over tasks which it executes faster than an accelerator). The *min* function is represented using a *step* function $Q(x)$, obtaining:

$$\begin{aligned} F_i(a_0, a_i) = {} & f_0(a_0)Q\big(f_i(a_i) - f_0(a_0)\big) \\ & + f_i(a_i)Q\big(f_0(a_0) - f_i(a_i)\big) \end{aligned} \tag{8}$$

A *sigmoid* function is used as a differentiable approximation of the *step* function $Q(x)$:

$$H(x) = \frac{1}{1 + e^{-kx}} \tag{9}$$

A larger k corresponds to a sharper transition at $x = 0$. We will thus redefine F_i replacing $Q(x)$ by $H(x)$. Now that the equations are all differentiable, we can proceed to solving them with Lagrange multipliers. We will thus minimize (6), subject to (3). Using Lagrange:

$$\frac{\partial[T_{GMA}]}{\partial a_j} + \lambda \frac{\partial\left[\sum_{j=0}^{j=N} a_j - A_{Total}\right]}{\partial a_j} = 0 \tag{10}$$

For every $0 \le (\forall j, k) \le M$ the following equality holds:

$$\begin{aligned} & \frac{\partial\left[f_0(a_0)t_0 + \sum_{i=1}^{i=M} F_i(a_0, a_i)\, t_i\right]}{\partial a_j} \\ & \qquad = \frac{\partial\left[f_0(a_0)t_0 + \sum_{i=1}^{i=M} F_i(a_0, a_i)\, t_i\right]}{\partial a_k} \end{aligned} \tag{11}$$

To solve (11) we define a helper function:

$$z_j = f_j(a_j) - f_0(a_0) \tag{12}$$

The optimal solution does not contain any accelerators in which $z_j = 0$ (if the GPP and the accelerator provide the very same performance, the allocation in which the accelerator is eliminated and its resource (e.g. area) is assigned elsewhere results in overall better performance). Note that $H'(z_i) \cong 0$ for $z_i \neq 0$ for a very large k. Solving $\frac{\partial F_i(a_0, a_i)}{\partial a_j}$:

$$\begin{aligned} \frac{\partial F_i(a_0, a_i)}{\partial a_j} = \frac{\partial[f_0(a_0)H(z_i) + f_i(a_i)H(-z_i)]}{\partial a_j} = \\ f_0'(a_0)H(z_i)\frac{\partial a_0}{\partial a_j} + f_0(a_0)H'(z_i)\frac{\partial z_i}{\partial a_j} \\ + f_i'(a_i)\frac{\partial a_i}{\partial a_j}H(-z_i) + f_i(a_i)H'(-z_i)\frac{\partial(-z_i)}{\partial a_j} \cong \\ f_0'(a_0)H(z_i)\frac{\partial a_0}{\partial a_j} + f_i'(a_i)H(-z_i)\frac{\partial a_i}{\partial a_j} \end{aligned} \tag{13}$$

Equation (13) reduces to:

$$j = 0 \rightarrow \frac{\partial F_i(a_0, a_i)}{\partial a_0} = f_0'(a_0)H(z_i)$$
$$\forall i = j, j \neq 0 \rightarrow \frac{\partial F_i(a_0, a_i)}{\partial a_j} = f_i'(a_i)H(-z_i) \quad (14)$$

Substituting (14) to (11), $\forall k \neq 0$, all utilized accelerators:

$$f_0'(a_0)\left[t_0 + \sum_{i=1}^{i=M} H(z_i)\, t_i\right] = f_k'(a_k)H(-z_k)t_k \quad (15)$$

Similarly, substituting (14) into (11), $\forall j, k \neq 0$, for all utilized accelerators:

$$f_j'(a_j)H(-z_j)t_j = f_k'(a_k)H(-z_k)t_k \quad (16)$$

Finally, combining (15) and (16), for all utilized accelerators $\forall j, k \neq 0$:

$$f_0'(a_0)\left[t_0 + \sum_{i=1}^{i=M} H(z_i)\, t_i\right] = f_j'(a_j)H(-z_j)t_j = f_k'(a_k)H(-z_k)t_k \quad (17)$$

The optimal resource allocation satisfies (17), and comprises of two sets: accelerators that are utilized and thus included in the SoC ($a_j \neq 0, z_j < 0$) and accelerators that are excluded ($a_j = 0, z_j > 0$). Actual solutions that satisfy (17) may be found, e.g., by numerical Lagrange solvers [1]. Equation (17) reverts back to MultiAmdahl (4) when all accelerators are utilized. Note further that z_j may be augmented with an area bias $z_j = f_j(a_j) - f_0(a_0) + \varepsilon$ representing accelerator's integration overhead.

3 Performance Optimization – Dual Accelerator Architecture

We first demonstrate our model by maximizing performance of a system that integrates a GPP with a multicore accelerator, similar to [3], [5] and [9]. The parallel fraction f of the workload may be executed on either the multicore accelerator or the GPP, whereas the sequential fraction $1-f$ of the workload is executed only on the GPP. Following Pollack's rule [4] and [8], the GPP's inverted performance may be written as follows (coefficients translating from area to performance units are scaled to unity):

$$f_{GPP}(a_{GPP}) = \frac{1}{a_{GPP}{}^{\beta}} \quad (18)$$

The exponent β typically varies from 0.3 to 0.7 [2], [5], [7]. We shall use $\beta = 0.5$ [4] in our plots. Other values of β do not significantly affect our results. The multicore accelerator's inverted performance function may be written as:

$$f_{MC}(a_{MC}) = \frac{1}{a_{MC}H(a_{MC} - a_{MCmin})} \quad (19)$$

where $H(a_{MC} - a_{MCmin})$ nullifies the speedup function when $a_{MC} < a_{MCmin}$. Note that multicore accelerator performance grows linearly with the number of integrated processing cores. a_{MCmin} represents the minimal infrastructure area that must be allocated to the multicore accelerator before it becomes operational (for example, minimal area allocated for shared bus, memories, caches and the like). Figure 2 depicts the GPP performance function vs. several multicore accelerator performance functions measured in Instructions Per Second (IPS), each having a different a_{MCmin}.

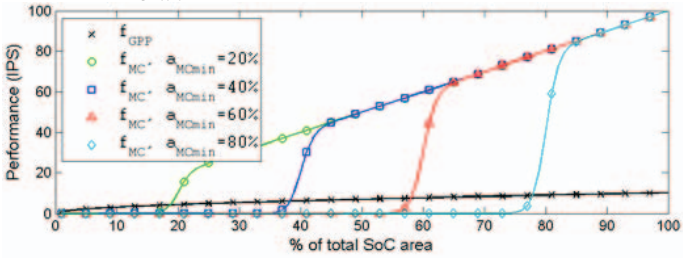

Figure 2: GPP and various Multicore performance functions as a function of their respective area

In Figure 3 and Figure 4, MA represents MultiAmdahl (dotted lines) and GMA represents the Generalized MultiAmdahl model (solid lines). Figure 3 depicts the optimal area of the GPP as a function of a_{MCmin}, the minimal effective area of a multicore accelerator, drawn for $f = 0.6$. Note the following zones:

Zone-A: a_{MCmin} is small enough and the multicore accelerator is allocated area larger than a_{MCmin}. Both MultiAmdahl and GMA yield the same solution; *Zone-B:* for larger values of a_{MCmin}, both MA and GMA select a multicore accelerator of the minimal area, as it is still worthwhile to keep the accelerator despite the larger a_{MCmin}; *Zone-C:* While MA allocates a multicore accelerator, GMA eliminates the multicore and allocates all area to the GPP.

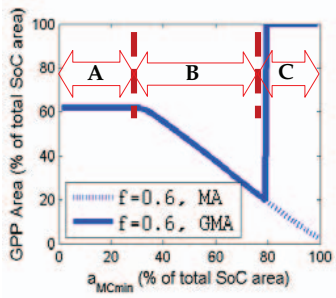

Figure 3: GMA (solid) and MultiAmdahl (dotted) optimal GPP area as a function of the a_{MCmin}

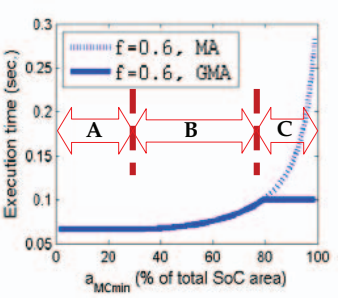

Figure 4. GMA (solid) and MultiAmdahl (dotted) optimal execution time as a function of a_{MCmin}

Figure 4 shows the optimal execution time as a function of the multicore accelerator a_{MCmin}, drawn for $f = 0.6$. In *Zone-C*, GMA's execution time is determined solely by the GPP. The MA execution time is longer because the GPP's segment is slowed down when large area is allocated to the multicore accelerator.

In general, for smaller f, the GMA model shifts area from the multicore accelerator to the GPP, until the execution time is determined solely by the GPP.

4 Performance Optimization – Quad Accelerators Architecture

In this section we consider the impact of A_{Total} constraint on quad-accelerator architecture. Consider a fixed workload consisting of four tasks, with execution times $t_i = \{70, 80, 90, 100\}$ seconds on a reference CPU, respectively. The SoC architect wishes to optimize execution time under an area constraint, and may utilize up to four distinct accelerators, out of which accelerator #0 is the GPP, and the rest are application specific. In Figure 5, the horizontal axis depicts eight total area scenarios, characterized by multiples of a baseline SoC area AT, 1×, 2×, … ,

128×. The vertical axis shows the optimal distribution of total SoC area among the four accelerators. Each hypothetical accelerator speedup function corresponds to:

$$f_i(a_i) = \frac{1}{H(a_i - a_{MINi}){a_i}^{\beta_i}} \quad (20)$$

and is characterized by exponent $\beta_i = \{0.4, 0.5, 0.6, 0.7\}$, minimum area $a_{MINi} = AT \times \{0.99, 0.65, 0.8, 0.95\}$, and maximum area $a_{MAXi} = AT \times \{1000, 2, 2.5, 3\}$ respectively. As evident from Figure 5, when the total SoC area increases from 1× to 2×, the GMA model shifts area resources from the GPP (a_0) to the stronger accelerators with the lowest a_{MINi} (i.e., to accelerator a_3). Under further area growth to 4× all four accelerators are utilized. When area increases to 8× the GPP consumes most of the added resources. This follows Equation (17): the slower GPP needs much more area if it is to match the execution time improvement of the other accelerators.

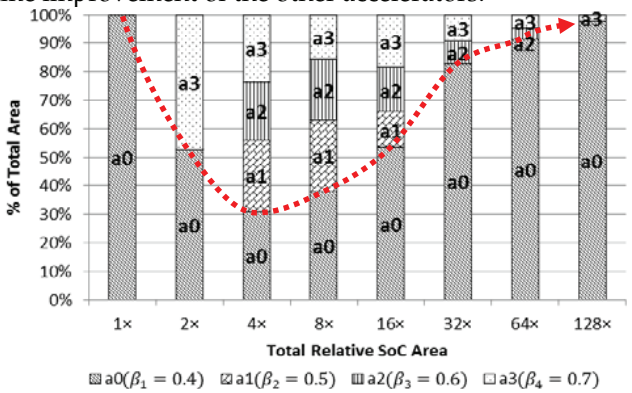

Figure 5: Optimal resource allocation of quad-accelerator multi-processor, as a function of baseline SOC area, and the trend-line (dotted)

Additional area growth to 16× increases the GPP area and thus its performance, while no additional resources are allocated to the a_1 accelerator due to performance saturation, a_{MAX1}. Area increase to 32× makes the a_1 accelerator weaker than the GPP, at which point it is eliminated and its task reverts back to the GPP. In a similar manner, from 64× to 128×, the a_2 accelerator is eliminated, leaving only the GPP and the a_3 accelerators. In general, as the total SoC area grows, the GMA eventually eliminates all lower-performing / saturated accelerators, assigning their tasks to the GPP.

5 Conclusions

This paper describes a multi-accelerator optimization algorithm that, given a sequential workload and a set of potential accelerators, selects an optimal subset of the accelerators and allocates hardware resources among the general purpose processor and the accelerators. The algorithm relies on modeling the performance of each accelerator as a function of the resources it uses. Note that in our case only a single accelerator operates at a time.

Practical implementation of accelerators is subject to certain performance limitations, i.e., A_{max} (e.g. due to overhead and A_{max} (e.g. due to performance saturation) and in a constraint environment, the architect may not need to utilize all potential accelerators. At times, given that the GPP is allocated with sufficient area, a task may be executed faster on the GPP than on an accelerator. This is especially true when the required area for an accelerator lies outside its effective area range, and thus limits the performance of other accelerators. At highly constrained area, the model allocates the entire area to the GPP. As the total SoC area is increased, the model shifts area from the GPP first to the stronger accelerators with the lowest A_{min} having the largest reference runtime, followed by accelerators with larger A_{min} that outperform the GPP under the current GPP area allocation. Further increase in total SoC area increases the GPP area and thus its performance. Thus, accelerators with performance saturation at A_{max} are eventually eliminated (that is, not assigned with any area) and their tasks are reassigned to the GPP.

We have proved that the optimal resource allocation takes into account the time spent on tasks, the accelerator's speedup functions, accelerator's speedup functions first derivative, as well as the GPP's speedup function and its first derivative. Only at the optimal resource allocation point, each addition of infinitesimal area would create the same improvement in the total execution time on any utilized accelerator. Otherwise, area allocation is unbalanced and some area should be removed from one accelerator and reallocated to another.

We have provided the architect a practical analytical tool for SoC partitioning which leads to the optimal execution time, given an area constraint. Furthermore, our framework offers an efficient alternative to iterative processes for exploring the design space. The Generalized MultiAmdahl framework can be extended in a number of ways, spanning from SoC partitioning, bandwidth to power distribution, energy and other constrained resources. In addition the Generalized MultiAmdahl framework may be applied to software resource-sharing and scheduling in heterogeneous systems, where the optimal workload partitioning given a set of tasks, under timing, power, energy and other constrains, is pursued.

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. 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.

[3] 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?," *MICRO-43*, 2010.

[4] 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

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

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

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

[8] 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.

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