LogCA: A Performance Model for Hardware Accelerators

Muhammad Shoaib Bin Altaf and David A. Wood
University of Wisconsin-Madison
shoaibbinalt@wisc.edu david@cs.wisc.edu

Abstract—To address the Dark Silicon problem, architects have increasingly turned to special-purpose hardware accelerators to improve the performance and energy efficiency of common computational kernels, such as encryption and compression. Unfortunately, the latency and overhead required to off-load a computation to an accelerator sometimes outweighs the potential benefits, resulting in a net decrease in performance or energy efficiency. To help architects and programmers reason about these trade-offs, we have developed the LogCA model, a simple performance model for hardware accelerators. LogCA provides a simplified abstraction of a hardware accelerator characterized by five key parameters. We have validated the model against a variety of accelerators, ranging from on-chip cryptographic accelerators in Sun's UltraSparc T2 and Intel's Sandy Bridge to both discrete and integrated GPUs.

Index Terms— Accelerators, Performance of Systems, Heterogeneous systems, Modeling techniques,

1 Introduction

To mitigate the effects of Dark Silicon [6], architects have increasingly turned to specialized hardware accelerators [2]. Accelerators have shown performance and energy efficiency improvement for some common functions by one to more than two orders of magnitude over conventional hardware [10].

Unfortunately, accelerators do not always live up to their name or potential. Off-loading a computation from a CPU to an accelerator incurs latency and overhead that depends on the amount of data, where the accelerator sits in the system architecture, and how it interfaces with the system. For some computations and accelerators, these factors can easily outweigh the potential benefits, resulting in a slower performance or lower energy efficiency.

Figure 1 plots unaccelerated and accelerated execution time for AES-192 [1], an encryption algorithm, on UltraSparc T2. For smaller block sizes, the unaccelerated version is faster than the accelerated version while the accelerated version outperforms the unaccelerated version for larger block sizes. The execution time for both versions becomes equal at the *break-even* point.

Understanding which factors affect accelerator performance and energy efficiency is important both for programmers and architects. Programmers need to be able to predict when off-loading a computation will be performance efficient. Similarly, architects need to understand how the accelerator's interface—and the resulting latency and overheads to off-load a computation—will affect the achievable accelerator performance.

To address these issues, this paper presents *LogCA*, a performance model for hardware accelerators. LogCA derives its name from the five parameters listed in Table 1. These parameters characterize the communication latency (L) and overheads (o) of the accelerator interface, the granularity/size (g) of the off-loaded data, the complexity (C) of the computation, and the accelerator's performance improvement (A) as compared to a general purpose core.

Manuscript submitted: 23-Jul-2014. Manuscript accepted: 18-Aug-2014. Final manuscript received: 28-Aug-2014.

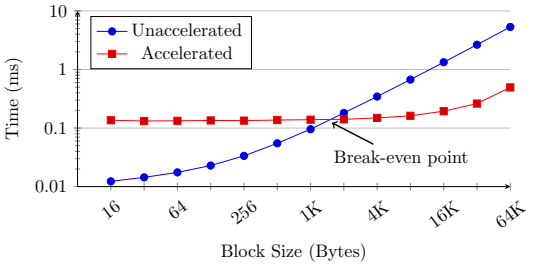


Fig. 1. Variation in execution time for AES-192 on UltraSparc T2

Our model is inspired by *LogP* [3], the well-known parallel computation model. *LogP* sought to find the right balance between overly simple models (e.g., PRAM) and the detailed reality of modern parallel systems. *LogCA* seeks to strike the same balance for hardware accelerators, providing sufficient simplicity such that programmers and architects can easily reason with it, while maintaining sufficient accuracy.

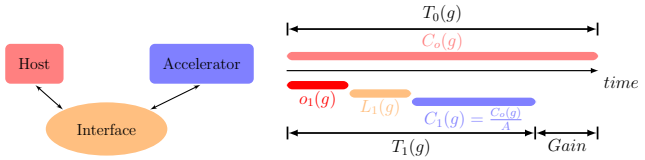
Just as *LogP* was not the first model of parallel computation, LogCA is not the first model for hardware accelerators [4], [8], [9]. With LogCA, our goal is to develop a simple model that supports the important implications (§2) of our analysis and use as few parameters as possible while providing sufficient accuracy. In Einstein's words, we want our model *to be as simple as possible and no simpler*.

Our main contributions are:

- A high-level performance model for hardware accelerators that strikes a useful balance between too simple and too complex (§2).
- Formalization of performance metrics for predicting the "right" amount of off-loaded data (§2).
- Providing an answer to *what-if* questions for both programmers and system designers at an early design stage (§2).
- Demonstrating the validity of the model on four disparate accelerator designs: cryptographic accelerators in Sun UltraSparc T2 and Intel's Sandy Bridge [11], a discrete NVIDIA GPU, and an integrated AMD GPU (§3).

TABLE 1
Description of the LogCA parameters.

Parameter	Symbol	Description	Units
Latency	L	A parameter for the time to move data from the host to the accelerator across the interface. This includes the time data spends in the caches or memory	Seconds
Overhead	o	A parameter for the time the host spends in setting up the algorithm	Seconds
Granularity	g	Size of the off-loaded data	Bytes
Computational Index	C	A parameter for the time the host spends per byte of data	Seconds/Bytes
Acceleration	A	The peak speedup of an accelerator.	N/A



(a) (b)

Fig. 2. Top level description of the LogCA model (a) Shows the various components (b) Time-line for the computation performed on the host system (above) and on an accelerator (below)

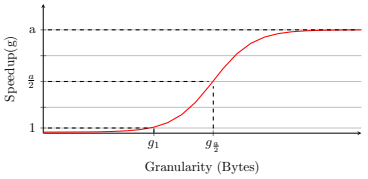


Fig. 3. A graphical description of the performance metrics

2 The Model

The model assumes an abstract system with three components (Figure 2 (a)). *Host* is a general purpose processor. *Accelerator* is a hardware device designed for the efficient implementation of an algorithm. *Interface* connects the host and accelerator, and abstracts away all system details including the memory hierarchy.

Our model uses the abstraction of an interface to provide intuition for the overhead and latency of dispatching work to an accelerator. This helps in hiding the details from the model about how the accelerator is connected – directly connected, system bus or PCIe. This also gives the flexibility to use our model for both on-chip and off-chip accelerators. This abstraction can also be trivially mapped to shared memory systems or other memory hierarchies in heterogeneous architectures.

Figure 2 (b) shows the overhead and latency model for an un-pipelined accelerator where computation 'i' is returned before requesting computation '$i+1$'. Pipelined accelerators can be modeled by representing $o_1(g)$ and $L_1(g)$ as the non-overlapped parts of overhead and latency, respectively.

Figure 2 (b) also shows the breakdown of time for an algorithm on the host and accelerator. We assume that the algorithm's execution time is a function of granularity. With this assumption, the unaccelerated time T_0 (time with zero accelerators) to process data of *granularity* g, will be $T_0(g) = C_0(g)$, where $C_0(g)$ is the computation time on the host.

When the data is offloaded to an accelerator, the new execution time T_1 (time with one accelerator) is $T_1(g) = O_1(g) + L_1(g) + C_1(g)$, where $O_1(g)$ is the host overhead time in off-loading, $L_1(g)$ is the interface latency and $C_1(g)$ is the computation time in the accelerator to process data of granularity g.

An accelerator with peak acceleration 'a' can decrease, in the absence of overheads, the algorithm's computation time by a factor of 'a' compared to the host. Thus the computation time on the accelerator will be $C_1(g) = \frac{C_0(g)}{a}$. This reduction in the computation time results in performance gains, and we quantify these gains with speedup, the ratio of the un-accelerated and accelerated time:

$$Speedup(g) = \frac{T_0(g)}{T_1(g)} = \frac{C_0(g)}{O_1(g) + L_1(g) + C_1(g)} \quad (1)$$

To make our model more concrete, we make several simplifying assumptions. We assume that the computation time is a linear function of the computational index 'c' and granularity i.e., $C_0(g) = c*g$. We have made this assumption as accelerators with linear complexity are prevalent in the field of high performance computing.

For many algorithms and accelerators, the overhead is independent of the granularity, i.e., $O_1(g) = o$. Latency, on the other hand, will often be granularity dependent, i.e., $L_1(g) = l * g$. Latency may be granularity independent if the accelerator can begin operating when the first byte (or block) arrives at the accelerator, i.e., $L_1(g) = l$. We define *computational intensity* as the ratio of computational index to latency, $(\frac{c}{l})$. *Computational intensity* signifies the amount of work done on a host per byte of offloaded data.

For simplicity, we begin with the assumption of granularity independent latency (The *Sandy Bridge* accelerator exhibits this behavior). With these assumptions,

$$Speedup(g) = \frac{c * g}{o + l + \frac{c*g}{a}} \quad (2)$$

Before discussing how granularity affects speedup, we explain how a programmer/designer can vary the parameters in Table 1. The latency may be reduced by a closer integration of accelerators while the overheads can be reduced by decreasing the time of algorithm initialization, data copying between buffers (memories), address translations etc. The computational index can be increased by redesigning the algorithm so that it does more work per unit time and investing more chip resources to an accelerator increases the acceleration.

2.1 Effect of Granularity

A key aspect of LogCA is that it captures the effect of granularity on the accelerator's speedup. Figure 3 shows this behavior i.e., speedup increases with granularity and is bounded by the peak acceleration 'a'. At one extreme, for large granularities, equation (2) simply becomes,

$$\lim_{g \to \infty} Speedup(g) = a \quad (3)$$

While for small granularities, equation (2) reduces to:

$$Speedup(g)_{g=1} = \frac{c}{o + l + \frac{c}{a}} < \frac{c}{o + l} \quad (4)$$

Equation (4) is simply *Amdahl's Law* [5] for accelerators, demonstrating the dominating effect of overheads at small granularities.

2.2 Performance Metrics

In the previous section, we describe bounds on speedup and how granularity affects it. But it does not help a programmer in deciding, *when* and *how much* to offload. To quantify this information, we define performance metrics inspired from the vector machine metrics N_v and $N_{1/2}$ [7], where N_v is the vector length to make vector mode

faster than scalar mode and $N_{1/2}$ is the vector length to achieve half of the peak performance. Since vector length is an important parameter in determining performance gains for vector machines, these metrics characterize the behavior and efficiency of vector machines with reference to scalar machines. Our metrics tend to serve the same purpose in the accelerator domain.

g_1 : The granularity to achieve a speedup of 1 (Figure 3). It is the break-even point where the accelerator's performance becomes equal to the host machine. Thus it is the minimum granularity at which an accelerator starts providing benefits. Solving equation (2) for g_1 gives:

$$g_1 = \left(\frac{a}{a-1}\right) * \left(\frac{o+l}{c}\right) \tag{5}$$

Implication 1. *g_1 is essentially independent of peak acceleration, for large values of 'a'.*

Implication 2. *Doubling computational index reduces the break even point by half.*

$g_{\frac{a}{2}}$: The granularity to achieve a speedup of half of the peak acceleration. This metric provides information about a system's behavior after the break-even point and shows how quickly the speedup can ramp towards peak acceleration. Solving equation (2) for $g_{\frac{a}{2}}$ gives:

$$g_{\frac{a}{2}} = a * \left(\frac{o+l}{c}\right) \tag{6}$$

For a relationship between g_1 and $g_{\frac{a}{2}}$, using (5) and (6)

$$g_{\frac{a}{2}} = (a-1) * g_1 \tag{7}$$

Implication 3. *Doubling peak acceleration 'a', approximately doubles the granularity to attain $\frac{a}{2}$.*

Generally a designer may prefer higher acceleration, and lower values of g_1 and $g_{\frac{a}{2}}$. But equation (7) shows that increasing peak acceleration increases $g_{\frac{a}{2}}$. This presents a dilemma for the system designer to either favor higher acceleration or reduced granularity. LogCA helps by exposing these trade-offs at an early design stage.

2.3 Granularity dependent latency

Earlier we have assumed latency to be granularity independent but we have also observed granularity dependent latencies in UltraSparc T2 and GPUs. So in this section, we discuss the effect of granularity on speedup and derive performance metrics assuming granularity dependent latency.

With this assumption, equation (1) reduces to:

$$Speedup(g) = \frac{c * g}{o + l * g + \frac{c*g}{a}} \tag{8}$$

For large granularities, equation (8) reduces to:

$$\lim_{g\to\infty} Speedup(g) = \left(\frac{a*c}{a*l+c}\right) \tag{9}$$

Unlike equation (3), speedup in the above equation approaches $\frac{c}{l}$ (*computational intensity*) for large values of peak acceleration.

Implication 4. *The achievable speedup is limited by computational intensity for accelerators with granularity dependent latency.*

For very small granularities, equation (8) reduces to:

$$Speedup(g)_{g=1} = a * \frac{c}{a*(o+l)+c} \tag{10}$$

Similar to equation (4), the above equation exposes the increasing effects of overheads at small granularities. Solving equation (8) for g_1 gives:

$$g_1 = a * \frac{o}{c*(a-1) - a*l} \tag{11}$$

For a positive value of g_1, the above equation must satisfy $\frac{c}{l} > \frac{c}{a*l} + 1$.

Implication 5. *Computational intensity must be greater than 1 to achieve any speedup.*

The above implication can be helpful for programmers early in the design process. For a given system, with these two parameters (c and l), they can quickly reach to the conclusion whether to put time and effort in porting a code to an accelerator.

Similarly solving equation (8) for $g_{\frac{a}{2}}$ gives:

$$g_{\frac{a}{2}} = a * \frac{o}{c - a*l} \tag{12}$$

For a positive value of $g_{\frac{a}{2}}$, the above equation must satisfy $\frac{c}{l} > a$.

Implication 6. *Computational intensity must be greater than the peak acceleration to achieve half of the peak acceleration.*

Consider a system with minimum desirable speedup of one half of the peak acceleration but has a communication to computation ratio of less than the peak acceleration. With the above implication, a system designer/programmer can infer early in the design stage that the speedup goals can only be achieved by investing more resources in increasing the computational index or an efficient interface.

We are also interested in quantifying the limits on achievable speedup due to overheads and latencies. To do this, we assume a hypothetical accelerator with infinite peak acceleration, and calculate the granularity (g_a) to achieve a speedup of peak acceleration 'a'. With this assumption, the desired speedup of 'a' is only limited by the overheads and latencies. Surprisingly we find that g_a equals $g_{\frac{a}{2}}$.

Implication 7. *If a speedup of $\frac{a}{2}$ is not achievable on an accelerator with peak acceleration 'a', despite increasing peak acceleration to ã (where ã > a), the speedup is bounded by 'a'.*

The above implication can help a system designer in allocating more resources for an efficient interface and not for increasing peak acceleration.

3 Evaluation

This section describes the experimental setup and the evaluation methodology for validating LogCA on real machines using an encryption benchmark (AES-192 [1]). We have used this benchmark as it's the only algorithm which can be run on all of the available accelerators. We get similar results not only with other encryption benchmarks but also with more complex benchmarks e.g., Blackscholes and Image Compression. We have omitted those results due to space constraints.

The experimental setup includes an on-chip Cryptographic unit on *UltraSparc T2*; *AES-NI (AES New Instruction)* [11] on *Sandy Bridge*; an integrated GPU (ATI Radeon 6750 D) on AMD Fusion also called the *Accelerated Processing unit* APU; and a discrete GPU, *NVIDIA* GTX 580.

We have used Linux utilities to calculate the execution time on *Sandy Bridge* and *UltraSparc T2*. We have used NVIDIA and AMD OpenCL profilers to compute the setup, kernel and data transfer times for the GPUs.

We calculate the computational index (C) by measuring the computation time on the host system. We measure overheads (o) by computing the execution time for very small granularities, while peak Acceleration (A) for each accelerator is the speedup at very large granularities. And we run some simple micro benchmarks to calculate the latency (L). We have to measure these parameters only once for a system and then can be used repeatedly.

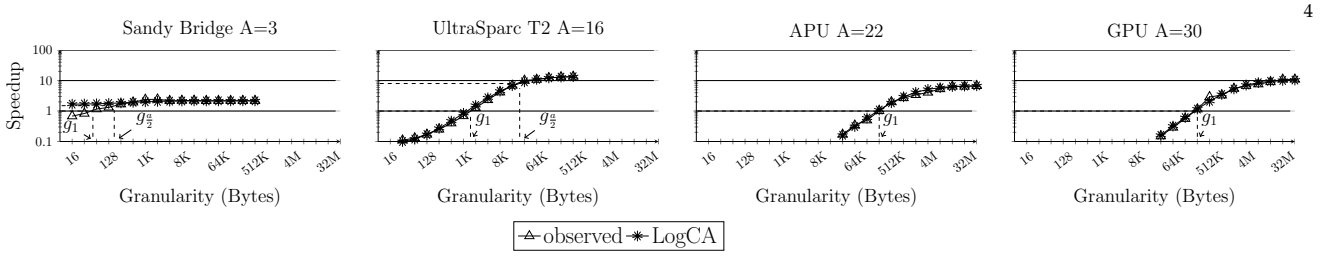


Fig. 4. Speedup curve fittings plots comparing LogCA with the observed values of AES-192 [1] over a range of granularities. The peak acceleration for each accelerator is also listed.

Figure 4 shows our model closely captures the accelerators' behavior. We have also marked g_1 for each accelerator. A large g_1 (on the right side of the plot) implies increases in the complexity of the interface and/or an accelerator further away from the host. For example, g_1 for Sandy Bridge lies on the extreme left while for the discrete GPU, g_1 lies on the right. So a programmer/designer can infer information about the system by just locating g_1 on a plot.

It is worth mentioning that we have marked $g_{\frac{a}{2}}$ for *Sandy Bridge* and *UltraSparc T2* but not for APU and GPU. For APU and GPU designs, the computation to communication ratio is less than the peak acceleration and as we have mentioned earlier in *Implication* 6 that for such designs $g_{\frac{a}{2}}$ doesn't exist.

Figure 4 also shows that *UltraSparc T2* provides higher speedups compared to *Sandy Bridge*, but it breaks-even at a larger granularity. *Sandy Bridge*, on the other hand, breaks-even at very small granularity but provides limited speedup. The discrete GPU with powerful processing cores has the highest peak acceleration among others. But its observed speedup for AES is less than *UltraSparc T2* due to high overheads and latencies involved in communicating through the PCIe bus.

The above information can be helpful in deciding about a target accelerator for a given task. For example, for reasonable speedups with lower break-even point, design similar to *UltraSparc T2* is preferable. But for high speedups with lower g_1 , GPUs are preferable with an improved design of the interface.

4 Related Work

To the best of our knowledge, this is the first attempt in developing a high level performance model for the hardware accelerators. Attempts have been made in the past to develop models for the accelerators but they were too complex for the programmers and designers to easily reason about the trade-offs of using an accelerator.

In terms of simplicity, our study most closely matches the Roofline model [12]. Roofline provides a "bound and bottleneck" analysis for multiprocessors and we envision ours to provide the same for the accelerators. Meswani et al. [9] also tried to answer the question that when is it beneficial to off-load data to an accelerator. They used GPU and FPGA as the target devices to expose the importance of communication cost in accelerators.

Hempstead et al. [8] introduced 'Navigo', an early stage modeling framework for estimating the amount of specialization to maintain required performance in the future. We, on the other hand, propose an early stage modeling framework to reason about accelerators for a given task. Chung et al. [4] did a detailed model study to predict the future landscape of heterogeneous computing. Conceptually, their U-core's parameter is similar to our Acceleration parameter. Both of these studies use ITRS road maps for building their models.

5 Limitations and Future Work

The applicability of our model can be limited by the simplifying assumptions of its parameters. In practice, these parameters can be random variables with arbitrary distribution functions.

In this work, we have also not considered the case for multiple accelerators. So the issues of memory/data sharing and pipelined execution of accelerators are beyond the scope of this work. Also, we have only considered the synchronous execution of accelerators. Accelerators executing in asynchronous mode may have more intricate communications and is not the focus of this work.

We are also exploring LogCA extensions for energy and area analysis as energy efficiency is an important design metric for accelerators.

6 Conclusion

With the recent trend towards heterogeneous computing, we feel that the architecture community lacks a model to reason about the need of accelerators. We have presented a simple, yet accurate, performance model for hardware accelerators. We have demonstrated the validity of our model on real systems for both on-chip and off-chip accelerators.

Acknowledgments

We thank Mark Hill, Michael Swift, Rathijit Sen, and the members of the Wisconsin Multifacet group for their comments on the paper. This work is supported in part with NSF grants CNS-1117280, CCF-1218323, and CNS-1302260. The views expressed herein are not necessarily those of the NSF. Professor Wood has signicant financial interests in AMD, Google and Panasas.

References

[1] "Advanced encryption standard (AES)," http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf
[2] C. Caşcaval *et al.*, "A taxonomy of accelerator architectures and their programming models," *IBM J. Res. Dev*
[3] D. Culler *et al.*, "Logp: Towards a realistic model of parallel computation" in PPOPP '93.
[4] E. Chung *et al.*, "Single-Chip Heterogeneous Computing: Does the Future Include Custom Logic, FPGAs, and GPGPUs?," in MICRO'10
[5] G.M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities" in AFIPS '67.
[6] H. Esmaeilzadeh *et al.*, "Dark silicon and the end of multicore scaling" in ISCA '11.
[7] J.L. Hennessy and D.A. Patterson, "Computer Architecture: A quantitative approach" Morgan kaufmann Publisher Inc., 2002
[8] M. Hempstead *et al.*, "Navigo: An early-stage model to study power-constrained architectures and specialization" in MoBS '09.
[9] M. Meswani *et al.*, "Modeling and predicting application performance of high performance computing applications on hardware accelerators" in IJHPCA '13.
[10] R. Hameed *et al.*, "Understanding sources of inefficiency in general-purpose chips" in ISCA '10.
[11] S. Gueron, "Intel advanced encryption standard (AES) instructions set," Intel, Tech. Rep.,2010.
[12] S. Williams *et al.*, "Roofline: An Insightful Visual Performance Model for Multicore Architectures" in CACM '09.