On Accelerators: Motivations, Costs, and Outlook
Simha Sethumadhavan
1/30/11

Futurist Ray Kurzweil has been quoted as saying that in the next 20 years breakthroughs in nanotechnology and consequently solar power generation will create a society in which energy is plentiful and unbounded, and that this bounteous energy will fuel another wave of computing advances that will result in computers that will rival/surpass human cognition. Reading between the lines, Kurzweil implicitly makes a couple of interesting points. First, the next wave of computing innovations rests on finding solutions to the energy problem. In other words, energy consumption ultimately determines the limits of computing. This observation rings true based on our experience with computers ranging from grand challenge supercomputers to ubiquitous mobile devices which are mostly limited in capabilities because of recurring energy costs. Second, there is no solution in sight to the energy generation problem for the next 20 years! If the high-risk (but high-payoff) efforts to generate clean energy do not pan out there is serious danger of stalling societal progress and in some cases even regressing back to the dark ages before the information revolution. (How is that for a "Sputnik" scare! ☺) Certainly, it will be unwise to put on hold efforts to make computing energy efficient in the hope of bounteous future energy.

Thinkers and Doers alike have recognized this fact for some time now and in their latest attempts to build energy-efficient computers have embraced accelerator based computing systems. In this article, we briefly describe the backstory on accelerators, the cost of designing accelerators, and important questions for computer architects on accelerators, and close with limitations of current accelerator based approaches. The goal is to simultaneously simulate research on this topic and also highlight the dangers of lining up all research chips behind accelerators.

**Accelerators: How we got here?**

(1) Actually, accelerators have always been around for a long time.

Casvacal and his collaborators define an accelerator as a "special-purpose hardware device that is optimized to perform a certain function of set of functions as part of the a general-purpose computational system." By this definition accelerators have been around all along and have kept pace with the development of traditional general-purpose architectures. Accelerators have been used to support network traffic, audio, video, graphics, physics, and specific math operations.

One key difference, however, between last generation of accelerators and the current wave of accelerators is that the new accelerators share the same die as the general-purpose computational system which has been possible because of technology scaling.

(2) … there is increased demand for accelerators now

Mobile smartphone users these days want (and have grown to expect) longer usage time between recharges. Simultaneously they also expect their mobile devices to do more compute intensive tasks like watching HD movies, editing/remixing HD videos, and interfacing with speech or actions. It is likely that in the near future they will expect devices to adapt to their emotions and perform various augumented reality tasks. And perhaps in the not so near future, expect these devices to capture and transmit holographic projections! Given that a majority of these tasks are very compute intensive (and because battery technology isn't improving significantly), accelerators are a terrific solution as they can significantly reduce energy usage.

(3) Now it is less risky for companies to pursue accelerator designs

The mass adoption of mobile phones has made it less risky for chip designers to spin off mobile chips with accelerators. Similarly, -- and this is my hypothesis -- in the cloud back-end services, aggregation of workloads from millions of users will create opportunity for deploying special purpose chips. For example, one can easily imagine including a sqlite accelerator that speeds up retrieval/search from a mail database from a personal account. (gmail ☺)

(4) Accelerators are simpler to program.

Despite the widespread availability of multicore chips for half a decade now, the state of parallel program hasn't substantially improved. We haven't witnessed major insights/discoveries that can simplify parallel programming to level of sequential programming. Parallelization is still in part (black) art and *continues to be an application specific endeavor* i.e., parallelization techniques used for one application typically do not directly apply to another application. Further, even after parallelization, performance of parallel applications is highly sensitive to data inputs and the microarchitecture and not generally portable. In contrast, performance improvements from accelerators are very predictable. Programming on accelerator-based hardware is often a simple matter of calling a function call provided by a vendor.

(5) The role of accelerators goes beyond commodity applications mentioned above.

Consider grand-challenge problems that are critical to science and society – these problems are extremely compute intensive, and general-purpose chips are horribly inefficient at solving these problems. If we need to make any headway in solving the grand-challenge problems, my firm belief is that we need hardware acceleration. To solve these problems we should form vertically integrated groups that include not only scientists from different disciplines but also engineers who will build special-purpose computers for these tasks.

**Accelerators: design and deployment costs**

Corporations are driven more by profits than by desires to advance science and society. Unless hardware and software firms believe that a particular technology will bring them continued profits it is unlikely that the technology will gain any traction. In this section, we briefly discuss the costs associated with accelerator designs from hardware and software perspective. We will use this then to derive a research agenda for computer architects.

Microprocessor costs can be accounted as recurring and one-time costs. One time costs include design and verification costs. Recurring costs include cost of manufacture, test and packaging, and the cost for power consumed. Deployment costs include cost of developing/maintaining software and maintaining the hardware.

(1) One-time cost: Design and verification costs

An accelerator may be developed and verified in-house or purchased from a 3rd party IP vendor. The DV cost of procuring 3rd Party IP can work out cheaper and may decrease the time to market for the product. On the other hand, if the accelerator is developed in house it is likely to increase design and verification costs.

(2) Manufacturing costs

We discuss manufacturing costs of the accelerator-based system in relation to a general-purpose system. This baseline system has N (homogeneous) cores, C MB of caches, X percent of the area for network routers and other supporting paraphernalia. For the discussion below, we assume that both the accelerator system and general-purpose system are manufactured in the *same process technology.*

The manufacturing and packaging costs to the first order are dependent on the die area and pin out of the chip. An accelerator-based system can be profitably manufactured by trading off performance, specificity and desired yield.

(A) Accelerator-based system larger than general-purpose die

Accelerator augmented systems i.e., systems that have accelerators added to them without depleting any of the general-purpose functionality or on-chip caches, will naturally be larger and cost more to manufacture. The increased die size may also result in lower yields, as larger dies are typically more prone to faults during initial spins. Thus, this non-profitable approach makes sense only if general-purpose and the accelerator communicate very frequently. If communication is sparse and latency insensitive, the same performance may be achieved using discrete chip on the same board without the risks and costs involved in building a larger die.

(B) Accelerator system has the same die size as general-purpose die

Accelerator systems that have the same die size as a general-purpose system will likely have lower throughput on workloads that run only on the general-purpose cores. If the general-purpose system is depleted of cache capacity or supporting paraphernalia to make way the accelerators then a similar throughput can probably be maintained. The latter case however assumes that the initial system was overprovisioned or had a sub-optimal design for most of the workloads.

The best operational case for the accelerator system occurs when the application mix matches the hardware composition of the chip in terms of its functionality. In this case, the accelerator systems will likely have higher energy-efficiency than the general-purpose system.

(C) Accelerator system is smaller that general-purpose die

In some ways, this is the best case for companies and consumers interested in accelerator-based systems, as this design point will increase the profit margins of companies. Reducing the size of accelerator-based system is possible when one or few applications in the workload mix dominate execution. In such a setting the number of general-purpose cores can be reduced and few small but powerful accelerators can be added in place of these cores.

From the above analysis, we opine that options (b) and (c) are the only ones that will not cost the processor manufacturers more money.

It is easy to extend the analysis to the case to the technology scaling case. Here we assume that the processor manufacturer is moving from either a fully general-purpose system or one of the three accelerator based system on a new smaller technology generation. With these assumptions, options (b) and (c) continue to be favorable to chip manufacturers, and even the augmented model becomes commercially feasible.

(3) Testing costs

Accelerator based systems will necessarily increase testing costs. With homogeneous multi-core systems the test patters are internally circulated from one core to another; this is simply not possible in heterogeneous environments. Increased tester time will directly add the chip cost.

(4) Deployment costs

As discussed before, software development for accelerators is likely to be easier for accelerators than writing traditional parallel programs. Maintaining these programs (e.g., updating them for new releases) is also fairly trivial.

Summary: What all this means?

As the above analysis shows, accelerator-based chips can be a profitable for both hardware and software companies. While software firms stand to gain more (since software development costs are orders of magnitude more than hardware development costs), if hardware design and testing challenges can be mitigated, it will easier to convince hardware vendors to produce accelerator based systems.

**Questions for Computer Architects:**

Computer architecture research should be focused on ameliorating the additional costs incurred by processor companies on accelerator-based designs. Research should be focused on identifying accelerators the give the biggest bang for the buck in terms of type and granularity, failure models, porting legacy software to accelerator based systems, on providing backward compatibility and reducing testing costs.

(1) What type of accelerators?

First and foremost, there needs to be clear methodology for discovering accelerators. Currently the process is completely ad-hoc. Researchers build accelerators for programs that are used in benchmarking or just make up important accelerators based on hunches! Without a clear methodology it will be hard to convince design teams to include special-purpose hardware.

(2) Granularity of accelerators?

Computer architects need a holistic model that can be used to calculate the area of the chip that can be allocated for accelerators. Based on this they can decide how much local storage, compute should be present in the accelerator.

(3) Reducing overheads

It is likely that these accelerators will interface to other units through an on-chip network. Basically, the on-chip network interface becomes an "overhead" structure for the accelerator. To reduce this overhead, multiple accelerators may have to share a network interface through some form of link/router concentration. Similarly, some local storage can be virtualized among multiple on-chip accelerators to improve performance.

(4) Dealing with failures

One of the advantages of multicore chip business is that partially working chips can still be sold in depleted configurations. Accelerators should have some inbuilt redundancy in their design to work around faults (which will become more prominent in extremely scaled technologies) OR should be able to seamlessly transfer computation on the general-purpose processor when an accelerator fails.

(5) Legacy software, portability

We need methods and tools that will enable legacy software to take advantage of new on-chip accelerators. Also we need a plan for running software that is constructed with the assumption that there will be accelerators but run on machines that do not have accelerators (The scheme suggested above for dealing with failures will probably work here as well.)

(6) Improving accelerator testing time:

Accelerators could potentially have some NV storage that holds the test vectors. These vectors could be run in parallel with the main core test without using the external tester bandwidth.

(7) Improving accelerator design and verification time

High level synthesis could help but I really don't think this will happen any time soon.

**Limitations of Accelerators**

(0) What about memory?

We can speed up computation all we want to accelerators but ultimately slow memory will limit the speedups we can get. Researchers should continue investigation of technologies that will mitigate this problem; new technologies like

optics that increase off-chip bandwidth, and intelligent prefetchers are worthwhile as well.

(1) Dark Silicon

Current silicon scaling trends clearly indicate that not all transistors in a chip can be on all at once. Shutting down some of the transistors is often referred to as dark silicon. While dark silicon is likely to acceptable for consumer markets (and likely to gain traction), it is unclear to me if problems that require really oomph – say, grand-challenge problems can be solved with the dark silicon limitation. Having a portion of the chip turned off all the time ultimately limits throughput significantly and can hurt progress in critical fields.

(2) What next?

Say we have identified and built accelerators for common (and some uncommon) programs. What next? Where will future energy-efficiency benefits come from?

I seem to think I know the answers to these limitations. ;-)