Thousand Core Chips—A Technology Perspective

Shekhar Borkar
Intel Corp, Microprocessor Technology Lab, JF2-04, 2111 NE 25th Ave, Hillsboro, OR 97124.
Shekhar.Y.Borkar@intel.com

ABSTRACT

This paper presents the many-core architecture, with hundreds to thousands of small cores, to deliver unprecedented compute performance in an affordable power envelope. We discuss fine grain power management, memory bandwidth, on die networks, and system resiliency for the many-core system.

Categories and Subject Descriptors

C.0 *Computer Systems Organization, General*

General Terms

Design, Performance, Reliability.

Keywords

CMOS, Power, Memory, Variability, Reliability.

1. Introduction

Moore's Law continues with technology scaling, improving transistor performance to increase frequency, increasing transistor integration capacity to realize complex architectures, and reducing energy consumed per logic operation to keep power dissipation within limit. Advances in software technology, such as rich multimedia applications and run time systems, exploited this performance explosion, delivering end users with higher productivity, seamless internet connectivity, and even multimedia & entertainment. The technology treadmill will continue, providing integration capacity of billions of transistors; however, with several fundamental barriers [1]. In this paper, we will examine some of the barriers, ways to get around them, how it changes the landscape, and how the future advances in technology, architecture, and software, all together could help continue this treadmill.

2. Is Multi-Core enough?

Integration capacity of billions of transistors exists today, and will double every two years. This trend is shown in Figure 1, starting from 2001 with 130nm technology generation, with a $300mm^2$ die capable of integrating one billion transistors.

Assuming about half the die area being allocated for logic, and the other half for large memory arrays such as caches, the trend shows that by 2015 you will have 100B transistors on a $300mm^2$ die, with almost 1.5B transistors available for logic. The logic transistors tend to be larger than transistors in the memory, take larger space, and consume more power.

How will you employ these logic transistors to deliver performance? The evolutionary approach is to continue today's trend with a few large processor cores, each employing 20 to 100 million logic transistors, and a large shared cache.

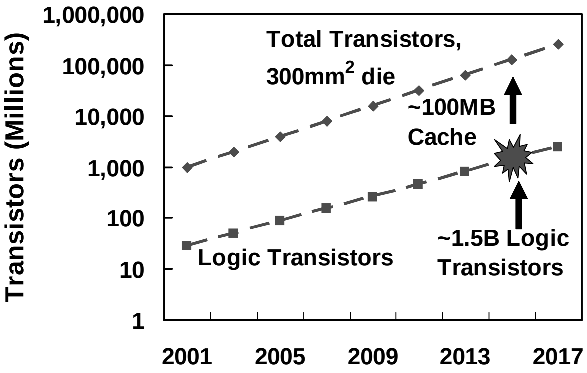

Figure 1: Transistor integration capacity

Performance increase by microarchitecture alone is governed by Pollack's Rule, which states that performance increase is roughly proportional to square root of increase in complexity. In other words, if you double the logic in a processor core, then it delivers only 40% more performance—as evidenced by all the leading processors in the past as shown in Figure 2. It plots integer performance increase of new microarchitectures against area (power) increase from the previous generation microarchitecture, in the same process technology.

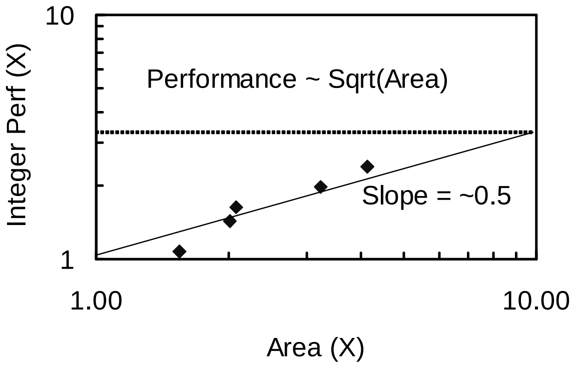

Figure 2: Pollack's Rule

A multi-core microarchitecture, on the other hand, has potential to provide near linear performance improvement with complexity and power. Two smaller processor cores, instead of a large monolithic processor core, can potentially provide 70-80% more performance, as compared to only 40% from a large monolithic core. Multiprocessors have several other benefits as

DAC 2007, June 4–8, 2007, San Diego, California, USA.
Copyright 2007 ACM 978-1-59593-627-1/07/0006 ...$5.00.

well: (1) each processor core can be individually turned on or off, thereby saving power; (2) each processor core can be run at its own optimized supply voltage and frequency; (3) easier to load balance among processor cores to distribute heat across the die; and (4) can potentially produce lower die temperatures improving reliability and leakage.

As technology scales further, transistor performance will not increase at the historic rates, due to excessive sub-threshold leakage current, and supply voltage scaling slowing down [1]. Taking these effects into consideration, Figure 3 estimates power consumption of a 300mm^2 processor die.

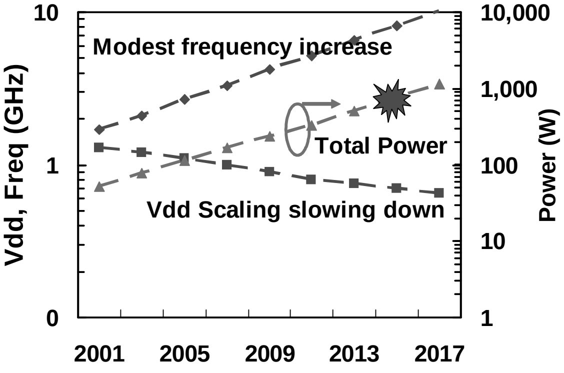

Figure 3: Frequency and Power Consumption

Notice that such a multi-core die will consume almost 1,000 watts of power, which is unreasonable. Therefore, we need to go beyond multi-core, and apply Pollack's rule to the extreme to deliver compute performance in a reasonable power envelope.

3. From Multi—to Many-Cores

Therefore, business as usual is not an option. You cannot simply follow the path of multi-core evolution, integrating multiple complex cores on a die. Instead, we propose that you integrate lots of smaller cores. Each small core delivers lower performance than a large complex core; however, the total compute throughput of the system is much higher as follows.

If you have 1B logic transistor budget, instead of integrating 10 large 100M transistor cores, we propose to integrate 100 medium 10M transistor cores, or even 1,000 small 1M transistor cores. Applying Pollack's rule inversely, performance of a smaller core reduces as square-root of the size, but power reduction is linear, resulting in smaller performance degradation with much larger power reduction. Overall, the compute throughput of the system, on the other hand, increases linearly with the larger number of small cores.

A many-core system on a die does not necessarily have to be symmetric or homogenous. An asymmetric system may have a few large cores to deliver higher single-thread performance, but will predominantly have large number of small cores. A heterogeneous system may even integrate diverse special purpose cores for hardware acceleration, e.g. graphics engines.

Figure 4 illustrates such a heterogeneous many-core system with general purpose cores (GP), and special purpose cores (SP), each core having local cache memory, and all cores connected together with an on-die interconnection network.

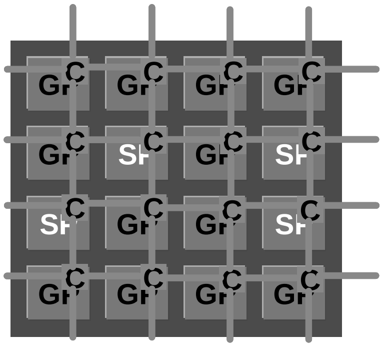

Figure 4: Illustration of a Many Core System

4. Performance of a Many-Core System

Although it is true that a many-core system will deliver higher compute throughput than a multi-core system for the same die size and in the same power envelope, it may be difficult to harvest the performance. The limitation is Amdahl's Law, which states that the parallel speedup is limited by the serial code in a program:

Parallel Speedup = 1/(Serial% +(1-Serial%)/N)

If the serial percentage in a program is large, then parallel speedup saturates with small number of cores. Figure 5 illustrates impact of serial percentage of code on parallel speedup.

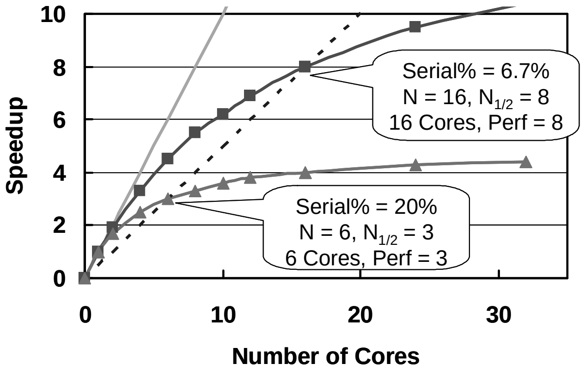

Figure 5: Amdahl's Law limits parallel speedup

Notice that even 7% serial code impacts parallel speedup adversely with diminishing return beyond 16 cores, delivering the performance of only 8 cores, thereby limiting application of inverse Pollack's rule up to only 16 cores. This limitation is true if only one application is running on the system at any given time, and you try to parallelize a single application across all the cores. In practice there are multiple applications running, each with multiple tasks and multiple threads, and thus there exists opportunity to harvest the performance of a many core system.

To explore this concept further, consider three systems on a chip comprising 1: 12 large (60MT) cores, 2: 48 medium (15MT) cores, and 3: 144 small (5MT) cores, all of them with the same amount of total cache, and all of them in the same power envelope. Figure 6 compares their relative performance.

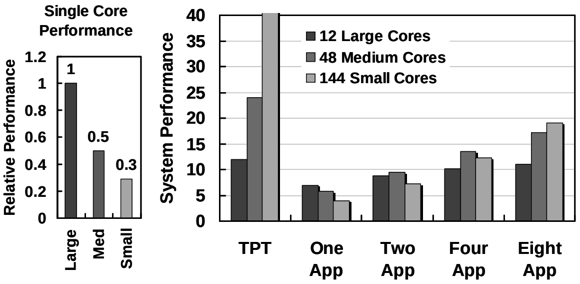

Figure 6: Performance of Large, Medium, and Small Cores

Small core is 12 times smaller than the large core, but its performance is only about a third of the large core. Total throughput of the small core system (TPT), however, is much higher than the large core system. If you parallelize one application for a 12 large core system and a 144 small core system then the small core system performs poorly, as expected from Amdahl's Law. Notice that as the number of applications increase, total system performance of medium and small core systems increase. Therefore, to harvest the performance of a many-core system, we cannot just depend on parallelizing a single application, but must utilize task level and application level parallelism.

5. Power and Energy

So far we have discussed how to maximize compute performance using many-cores, using hundreds or even thousands of small cores. Now we discuss how to fit the system in an affordable power envelope.

The best lever to reduce power with minimal impact on performance is to use voltage scaling. Applying voltage scaling indiscriminately to the entire system would lower power, but may not be optimal. Instead we propose to exploit the fact that there are hundreds to thousands of cores, and each core can be voltage scaled individually, thus employing fine grain power management. Individual cores can be voltage and frequency scaled to any arbitrary voltage and frequency in the possible range, but this could be difficult and cumbersome. Different cores running at different voltages and frequency would create asynchronous interfaces, adding latency, meta-stability, and would require complex power delivery system.

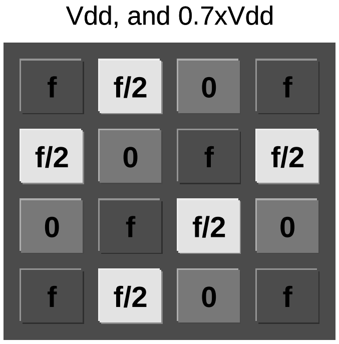

Figure 7: Fine grain power management

We propose a much simpler method of fine grain power management illustrated in Figure 7. A core operates at one of the two frequencies: f and f/2, where f is the maximum frequency of operation. When a core operates at f/2, it uses lower voltage, say 0.7xVdd, thus consuming only 25% of the maximum power. Idle cores could be turned off to save leakage power. Thus you need only two supply voltages, two sleep transistors [2] in each core to select a supply voltage, and a frequency divider in each core to select the frequency of operation. Since the number of cores is large, this coarse grain power management of a core enables fine grain power management of the entire many core system, with only two supply voltages, simplifying design and power delivery.

If all the cores were operating at half the frequency then the power consumption would be approximately 200W. Therefore, we estimate power consumption of about 300W for a 300mm^2 die, or 100W for a 100mm^2 die, fitting in the affordable power envelope.

6. Design Considerations

Since a core should be capable of operating at a much lower voltage, it warrants careful design practices. In the past, designs had to operate at the highest possible frequency to deliver high performance, but not anymore; it's the system performance that matters. Hence, design styles, such as domino logic, which are power hungry and do not scale well with voltage should not be employed. Instead, simple and robust static CMOS logic, which consumes much lower power, must be used. Large memory arrays such as caches and register files tend to be sensitive to lower voltages too, often due to design tradeoffs to increase their density. These arrays will have to be designed for robustness for low voltage operation, sacrificing some density.

7. On die network

The backbone of the many-core system is the network on the chip, connecting all the cores together, and carrying memory and IO traffic. In a shared memory system, this network carries cache coherence traffic to keep caches coherent, which is bandwidth intensive and latency sensitive. The network can be a ring [3] or a higher dimensional network such as a mesh [4] to reduce latency. It is typically implemented as a packet switched network carrying packets buffered at each node.

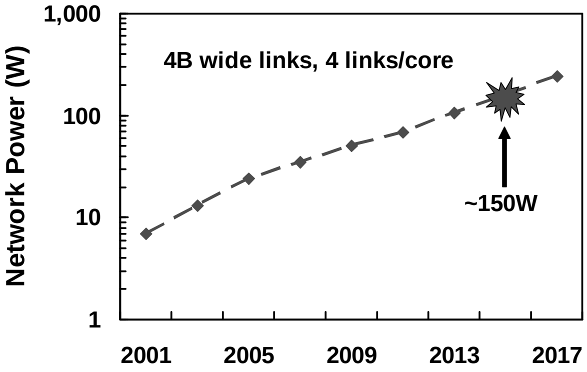

Figure 8: Network power estimate

These packet switched networks tend to be power hungry, consuming almost 500mW of power at each node. Assuming even narrow 4 byte wide links, 4 per core, with 1MT small cores, Figure 8 shows estimated network power for a many core die, confirming that the network power is substantial.

Power management in the network is difficult, if not impractical, because any power management technique, such as clock gating or sleep-states, incurs wake up latency, impacting system performance. On the one hand, smaller cores give you higher throughput performance in the same power envelope as discussed before, but on the other hand, it also increases the number of network nodes, increasing the network power. A careful study of the system power and performance is necessary to balance the size of the core, and the number of cores.

8. Memory Bandwidth

A many core system with thousands of cores will demand 100's of GB of memory bandwidth, and a traditional memory subsystem solution is not sufficient. A memory bus IO circuit consumes ~25mW/Gbps, and consequently a memory bus alone would consume about 25W to deliver 100GB/s memory bandwidth, which is excessive. The IO circuits employ sophisticated signal processing techniques to attain high data rate, consuming much of the power. If the bus lengths are somehow made substantially smaller, say a few millimeters, then the buses behave like lumped capacitor, rather than as transmission lines, and the IO circuits would consume substantially lower power, of the order of 1-2mW/Gbps.

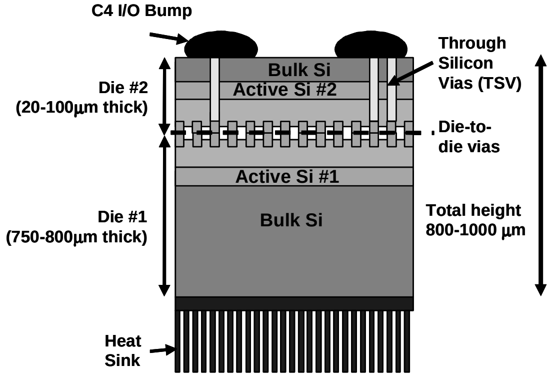

Figure 9: Three dimensional interconnect with stacking

A three dimensional integration (3D) of a memory with processor is a potential solution, where a thinned memory die is placed between the processor and the package [4]. Signals and power to the processor are routed through the memory die using through silicon vias, as shown in Figure 9 & Figure 10.

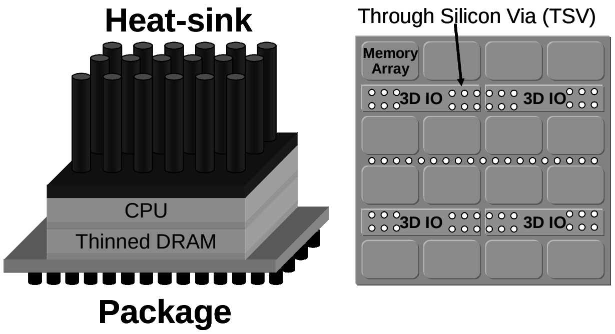

Figure 10: Assembly of 3D memory

This 3D integration of memory will consume only a few watts of IO power, and provide substantial memory bandwidth to the many-core system. Of course, architecturally, this adds another level of memory hierarchy and needs further attention.

9. Resiliency

As technology scales further, variations become prominent [5]. Chips will encounter dynamic variations of supply voltage and temperature; frequent and intermittent soft-errors; and transistors that slowly age and degrade over time, degrading circuit performance [6]. Despite these difficulties, users expect the system to remain reliable and to continue to deliver the rated performance. This challenge will undoubtedly require a major paradigm shift in all aspects of VLSI design—fabrication, design, microarchitecture, testing, software, and applications. Many-cores, with hundreds to thousands of cores provide resiliency to combat this problem.

Many-cores in a system will provide redundancy with spare cores, and functional redundancy checking employed at a coarse-grained level. For example, one core could check results produced by several cores; of course, software and applications will have to support this concept whenever possible. We could distribute test functionality as a part of the hardware to dynamically detect errors, or to isolate and correct aging and faulty cores, by replacing from the spare cores.

This microarchitecture strategy, with many-cores to assist in redundancy, is called resilient microarchitecture. It continually detects errors, isolates faults, confines faults, reconfigures the hardware, and thus adapts. If we can make such a strategy work, there is no need for one-time factory testing or burn-in, since the system is capable of testing and reconfiguring itself to make itself work reliably throughout its lifetime.

10. Conclusion

The many-core architecture with hundreds to thousands of small cores delivers unprecedented compute performance in an affordable power envelope. Fine grain system power management, an optimized on-die-network, and 3D memory technology are vital, and a many-core system also provides resiliency to combat variability and reliability.

11. References

[1] Shekhar Borkar, "Design Challenges of Technology Scaling, IEEE Micro", July-August 1999.

[2] Tschanz J., et al, " Dynamic sleep transistor and body bias for active leakage power control of microprocessors", IEEE Journal of Solid State Circuits, November 2003.

[3] Pham D. et al, "The design and implementation of a first-generation CELL processor", ISSCC 2005.

[4] Vangal et al, "An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS", ISSCC 2007.

[5] Shekhar Borkar et al, "Parameter Variations and Impact on Circuits and Microarchitecture", Proceedings, DAC 2003.

[6] Shekhar Borkar, "Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation", IEEE Micro, November-December 2005.