In 1943 Thomas Watson predicted, "I think there is a world market for maybe five computers". Today, that prediction falls short of the market for a single household. Early computers the size of warehouses once served the same purpose that today can be accomplished with a $10 Arduino chip that can fit in the palm of a hand. Many innovations have bridged that gap in the last few decades and one of the most prolific was the advent of the multicore processor. Unfortunately, while the multicore processor has alleviated the power consumption and overheating issues that accompany continuous frequency scaling in single core processors, they present a new set of challenges to surmount.  The largest qualms with multicore design are interference problems that arise when cores try to access shared resources concurrently. In order to perform basic instruction sets, multicores have to share access to DRAM, memory buses, I/O resources and caches. Some potential solutions to these conflicts are: software applications that allocate memory and bandwidth to cores based on demand in real time, replacing traditional bus connections using routers and a hardware communication network between the cores, adjusting the communication hierarchy of routers.  The implications of increased core utilization are significant gains in efficiency of throughput and energy usage.  The incentive is straightforward; if we optimize multicore technology we get faster and more powerful computers.

When computer design was in it’s infancy processors only had a single core.  However, as technology progressed people realized that there were limitations on how fast a single processor could run. The first attempt to increasing computer performance without raising frequency was to place multiple processors onto a single board. This technique proved to be unsuccessful as the system required significantly more power. In addition, communication between the multiple single core units added latency to the system. In order to increase performance without drawing more power, engineers developed the multi-core processor. A multi-core processor is a CPU that contains two or more processing units. This innovative design allows for lower power consumption while increasing CPU bandwidth per processor.

While the multi-core processor did mend most of the issues of the single core processor, it brought upon new challenges. Multi-core CPUs, while having more cores, still use the same resources a single core processor would use. Shared resources include the DRAM, the cache, I/O, memory bus, and the chip network. The issues that stem from multi-core processor occur when cores try to interface the same resources simultaneously. As the number of cores increase, the percentage of resources allocated to a core decreases. Memory related issues occur when cores try to access the same memory bank. The reason being that there is no limit as to how much space each core occupies. If the task executed on one core requires all of the RAM space, it will use up all of the memory. This causes huge latency, as the other cores must wait until the task is done to use the RAM. A solution is to allocate specific memory locations to each core [1]. That way one core’s task would not affect another core’s need to use RAM space. Allocation of core space is also utilized in memory buses. Bottlenecking on the memory bus can take place with simultaneous access. This in turn bogs down CPU execution time. To prevent this a regulator or scheduler is implemented. The regulator for the BUS gives each core an allowance of memory access [1]. Should a core exceed that allowance, the regulator stops the task being done in that core. The task will restart after an allotted amount of time and the allowance resets.

While there are many solutions to the shortcomings of multicore processors a solution that addresses one constraint often comes at the cost of another. In the past resources such as memory, bandwidth, and speed were the limiting factors in computing performance. While these factors are still relevant today, energy efficiency and thermal cooling are emerging as the driving forces behind multicore processor design. Lifetime reliability of a homogeneous processor can be characterized by Amdahl’s law depending on the scaling (parallelization) factor ‘f’ and the processor core composition (number of big vs little cores). A processor made of a complex core and many small cores may use the big core for faster sequential computation and the smaller cores for faster parallel execution. Application specific stresses can be biased to the particular type of core that can best handle the computation. The big (complex) core is the busiest, it is almost always turned on and usually must execute both serial and parallel phases of workload. Failure of big core immediately leads to failure of entire processor. If the processor has more than one big core than any one of them can be used for serial execution while the remainder are power gated until they are needed. Under a given area constraint, increasing the number of big cores on a die reduces the capacity for small cores. For power saving applications, it is best to use the big core on predominantly sequential executions. Energy efficiency can be effectively measured with metrics such as performance per joule or watt. While the heterogeneous processors is efficient in its ability to bias the workload it does require convoluted scheduling methods to manage its resources. There are many implementations of the heterogeneous processor and in order to develop solutions for the multicore processor we must consider each of the different configurations.

In order to evaluate the utility of heterogeneous processors we must compare to the control of both the homogeneous processor, comprised exclusively of simple (small) cores and the homogeneous processor composed of complex (big) cores. For the small core processor, maximum speedup in performance is given by Amdahl’s law (eq1), performance in terms of execution time is reduced by parallelizing the ‘f’ fraction of computations carried out by ‘n’ number of cores. This model is hinged upon the assumptions that power consumption of a small core is normalized to one and the power dissipated in serial executions is equivalent to the power dissipated by a single core. This logic is extended to the power consumption of the entire processor, calculated as n times the number of cores used to execute parallel threads. Under these assumptions energy scaling is normalized to a value of one and power scaling is equivalent to the performance speedup calculated using equation 1. The Processor of complex cores must be compared to the processor of small cores. We must make the assumption that each component big core has ‘s’ times better performance and r times larger area on the die. The area available on the processor is calculated assuming Pollack’s Rule (see vocabulary). Confined to the same area constraints as the small core processor there can be as many as n/r number of complex cores. The improvement in this configuration is achieved in the acceleration of serial executions. The speedup can be calculated using (eq2).

The first heterogeneous processor we will consider is that of multiple big cores. When performing a single thread, one complex core is sufficient to perform the serial executions of the application. In situations where multiple applications are running, multiple complex cores may be required to perform serial tasks concurrently in order to deliver reasonable throughput.  For the simplification of our analysis we will assume maximum processor scheduling, meaning that the processors computational capacity is fully utilized. For this situation we will consider the number of big cores to be ‘b’, the area of the processor not occupied by big cores is predominantly home to n-b\*r small cores. Under the assumption of maximum scheduling, the performance speed up of this configuration is given by (eq4). During serial execution the processor consumes ‘p’  amount of power, equivalent to the power consumption of a single complex core. The other cores that are not used in said serial execution are assumed to be power gated. During parallel phases, total power is calculated as the sum of b\*p +(n-b\*r)\*1 if power used by small core is normalized to 1 and the big core uses ‘p’ times more power, with the first addend of total power accounting for the big cores and the second for the small [4].

The next situation we must consider is a heterogeneous processor that separates the use of distinct types of cores. As previously mentioned, it is far more energy efficient to use complex cores to execute serial threads and run parallel threads on simple cores. The performance speedup of said processor is calculated as (eq6) where (1-f) represents the sequential operation is sped up by a big core that is ‘s’ times faster than a small core and f is the fraction of instructions run in parallel by n-b\*r small cores. The energy usage of such a configuration is given by (eq7) where the big core consumes ‘p’ amount of power on serial instructions and the small cores consume n-b\*r on parallel threads.

The Multi-core Processor alleviated the complications of high power consumption and overheating but these problems still persists. In a paper about Amdahl’s law, researchers evaluated Amdahl's law for performance and energy scaling of multi-core processors. They compared the performance per joule of different kinds of multi-core CPUs and the parallelization factor (fraction of computation with n number of cores). It was discovered that heterogeneous processors with small cores uses up more power than a single complex core [3]. The logical solution is to implement multiple big cores, but that in turn reduces the throughput and therefore reduces performance. However, if the heterogeneous cores were dynamically composed of big and small cores the energy efficiency surpassed that of a single core [3].

Just like all other technological pursuits, engineers strive to optimize the performance of these processors to improve productivity in other scientific endeavors. The multi-core processor was invented to surpass the performance of a single processor while consuming less power. While the single core was subject to improvements of broad scope it appears ,at least for the time being, that solutions implemented for the multicore processor must be application specific to make meaningful progress.

Vocab

·      Heterogeneous Multicore Processor: A single computing component comprised of 2 or more non-identical cores.

·      Object: A particular instance of a class, an object can be a combination of variables, functions and data structures.

·      Die: An integrated circuit produced in large batches on a single wafer of electronics grade silicon.

·      Bias Scheduling: influencing the scheduler to select one type of core that best suits a particular application.

·      Power Gaiting: A technique used on integrated circuits to reduce power consumption by shutting off currents to parts of the circuit not in use.

·      Pollack’s Rule: Microprocessor performance increase due to microarchitecture advances is roughly proportional to to the square root of the increase in complexity.

·      Composed Processor: Splits the message up, routes the sub messages to their appropriate destinations and then re-aggregates the responses back into a single message.

·      Hot Carrier Injection: phenomena in solid state electronics where an electron “hole” gains enough kinetic energy to overcome the potential barrier to break an interface state.

·      Negative Bias Temperature Instability(NBTI): is the key reliability issue in a MOSFET, it is an increase in the threshold voltage and therefore a decrease in drain current and transconductance of the MOSFET.

·      Cartesian Product: The product of two sets of ordered pairs.

·      Instantiation: Creation of a real instance or particular realization of an abstraction or template such as a class of objects or computer process.

·      Feature Vector: is an n-dimensional vector of numeric features that represents some object.

·      Thread: is an independent path of execution within a program.

·      Buffer: a region of physical memory storage used to temporarily store data while it is being moved from one location to another or between processes.

·      Crossbar: A collection of switches arranged in a matrix also known as a switched-medium network.

·      Arbiter: Electronic device that allocates access to shared resources.

·      Flit(Flow control digITs): large network packets are into smaller pieces called flits.

·      Network Packet: formatted unit of data carried by a packet-switched network.

·      Virtual Channel: is a means of transporting data over a packet switched computer network such that appears as though there is a dedicated physical layer link between  the source and destination end systems of this data.

·      Cache: a portion of memory made of high-speed static RAM(SRAM) used to quickly recall data or instructions that must be used repeatedly.

·      Virtual Memory: memory that appears to be stored in main storage although most of it is supported by data in secondary storage.

·      Page Coloring: a process of trying to allocate free pages that are near each other from the CPU cache’s point of view, in order to maximize the total number of pages cached by the processor.

·      DRAM: A memory chip that depends n an applied voltage to keep stored data; stores each bit of data in a capacitor on an integrated chip.

·      Time-Division Multiplex Access: A technique for transmitting two or more signals over the same medium/channel, each signal is sent as a series of pulses or “packets” which are interweaved with those of the other signal and transmitted as a continuous stream.

References: (all accessed via IEEE xplore)

[1] “Real-Time Computing on Multicore Processors”

[2] “Predicting Cross-Core Performance Interference On Multicore Processors With Regression Analysis”

[3] “Amdahl’s Law For Lifetime Reliability Scaling In Heterogeneous Multicore Processors”

[4] “Research On Topology And Policy For Low Power Consumption Of Network-On-Chip With Multicore Processors”