Overcoming Serial Bottleneck

By- Prashant Gupta

Until recently it was believed that improving serial bottleneck is making each single core more and more efficient resulting in improved overall performance. Many sophisticated techniques like Brach Predication, Loop unrolling, removing dependence (like Control Dependence, Output Dependence), VLIW (Very Long Instruction Word), Predicated Execution etc. improved processor performance for a long time but soon researchers realized that the improvements in a single core were limited by clock speed and heat generation. Hence, the focus shifted to multicore processors.

The Hill and Marty's paper [1] points out some key ideas on the application of Amdahl's Law in today's Multicore Era. The focus of the paper is showing how the speedup is getting affected by the serial bottleneck and later comparing the speedup of Symmetric multicore chips with Asymmetric and Dynamic multicore chips. The data shown in the paper clearly vouches for dynamic multicore chips which have enormous speedups when compared to the symmetric chips. The paper points out that, it is only beneficial to increase core resource of a single core when **perf(r)> r** where, r is the number of BCE's (Base core equivalents).

One of the company, that is using this idea is ARM in its **big.LITTLE architecture**[2]. The architecture uses Big and Little cores working on Dynamic Voltage and Frequency scaling (DVFS). The idea behind 2 different types of cores is that, the heavy serial task or high priority task is handled by the Big core and the small core handles the parallel threads or the latency tolerant tasks. The Global Task Scheduling (GTS) schedules the tasks over the cores dynamically, using previous load history time as a bench mark it anticipates the performance needs of the thread next time it runs. Both the cores (Big and little) have a common memory region which makes switching a task between the cores quick and much more power efficient. The cores can shift between different voltage/frequency with the help of DVFS depending on the task assignment of the core. ARM claims to be delivering same or better performance but saving upto 75% of the energy in some cases. This model has been implemented in **Samsung Exynos** processor. Intel came up with its own version of DVFS called the **Turbo boost** [3] which dynamically varies the frequency of the processor cores which promises about 6% reduction in execution time at a cost of 16% increase in power consumption. The processor increases the frequency of the core based on the core temperature, number of active cores and estimated power consumption. When the program enters a sequential phase the processor turns off the power for the idle cores (it also uses power gating to reduce leakage current) and boosts the active core, thus improving the serial performance. When the program turns to parallel all the cores become active to efficiently tackle the parallel thread performance. Asymmetric Multi-core processors (AMP) have shown a lot of performance improvement over the years but it ultimately comes down to how efficient the task scheduling and voltage scaling is? AMP can only be as better as the task scheduler we have.

Another technology that is gaining a lot of interest lately in computing is **FPGA** [Field Programmable Gate Array] because of its ability to program hardware to fulfill a specific user desired task. While earlier FPGA's were just considered as a peripheral but with developments of high speed interconnects like PCI express they have been moved to a category of co-processors. Recently there has been a lot use of FPGA's to make hardware accelerators for

applications like neural networks, image processing, DSP (digital signal processing), encryption – decryption etc. The main reason of this popularity is the promise of application specific hardware giving high performance, which operates at a much lower clock speed as compared to our present computer i.e. using much less power as compared to present computers. Also, these can be stacked together in parallel to get even better performance. The only issue when it comes to FPGA's is programming them is very cumbersome which leads to long development time. Although a lot of tools have been made to make them more user friendly like OpenCL which does not require knowledge of VHDL or Verilog. These improvements have bought the design time down significantly but they still have a long way to go. Intel's recent move to buy Altera shows that no matter how user unfriendly FPGA's are, they are here to stay. After FPGA's if the design looks good and gives you substantial processing improvements one can move to **ASIC** (Application specific integrated circuit) design which can further push the performance bar. As they are designed to accelerate and serve a particular application they are very-very fast and have a small form factor but then you lose the ability to re-configure the system like FPGA's. No doubt that the ASIC machines are the fastest solution to a problem but they become useless once the problem changes, not to forget the high design and fabrication costs involved. Google's TPU (TensorFlow Processing Unit) is a very recent example of ASIC implementation. Google recently announced using its proprietary TPU for machine learning applications like Google Photos, Google Street View, Data Centers etc. Another company that is worth mentioning is Qualcomm with its Snapdragon SOC. The SOC has a dedicated CPU, GPU, DSP unit etc. each of which is present to accelerate a specific set of tasks.

The most interesting research going on in this field is **mixed design** which has processor having some parts as ASIC and rest of it is general purpose. The idea behind this is to give advantages of both worlds. Computationally heavy or frequent tasks (depending on the intended application) can be optimized to be resolved on the ASIC while the seldom occurring or less computationally intensive tasks can be sifted to the general-purpose unit. Not much has been done in this field but this area of research looks most promising. **3D DRAM** [4] is another great idea which can be used to reduce the latency of going to off chip, and thus this will help in reducing the latency of both serial and parallel tasks.

References

1) M. D. Hill and M. R. Marty, "Amdahl's Law in the Multicore Era," in Computer, vol. 41, no. 7, pp. 33-38, July 2008. doi: 10.1109/MC.2008.209
2) K. Yu, D. Han, C. Youn, S. Hwang and J. Lee, "Power-aware task scheduling for big.LITTLE mobile processor," 2013 International SoC Design Conference (ISOCC), Busan, 2013, pp. 208-212. doi: 10.1109/ISOCC.2013.6864009.
3) J. Charles, P. Jassi, N. S. Ananth, A. Sadat and A. Fedorova, "Evaluation of the Intel® Core™ i7 Turbo Boost feature," 2009 IEEE International Symposium on Workload Characterization (IISWC), Austin, TX, 2009, pp. 188-197. doi: 10.1109/IISWC.2009.5306782.
4) Kavi K., Pianelli S., Pisano G., Regina G., Ignatowski M. (2014) 3D DRAM and PCMs in Processor Memory Hierarchy. In: Maehle E., Römer K., Karl W., Tovar E. (eds) Architecture of Computing Systems – ARCS 2014. ARCS 2014. Lecture Notes in Computer Science, vol 8350. Springer, Cham.