GPU NTC Process Variation Compensation with Voltage Stacking

Rafael Trapani Possignolo, Ehsan Ardestani, Alamelu Sankaranarayanan
Dept. of Computer Engineering
University of California, Santa Cruz

Jose Luis Briz
Dept. of Computer Engineering
Universidad Zaragoza

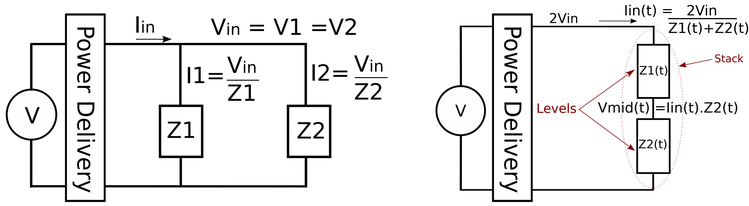
Jose Renau
Dept. of Computer Engineering
University of California, Santa Cruz

Background:

➔ Near Threshold Computing (NTC) [1] improves energy efficiency by reducing voltage.
➔ Performance impact from NTC can be mitigated through parallelism.
➔ GPUs are ideal for such scenario.
➔ NTC requires more current, thus makes the system more sensitive to process variation (PV) [2].
➔ Increased current also reduces power delivery efficiency.

Voltage Stacking:

➔ Instead of the parallel power delivery, use a serial approach.
➔ In a N-stack level, supply voltage is multiplied by N, and current drops by around N.
➔ Challenge: keep the load balance of the stack levels.



(a) Conventional (b) Stacked

Fig 1. Voltage Stacking reduces the overall system current, and thus reduces the pressure on the power delivery.

Voltage Stacking for Process Variation Compensation:

➔ Channel length (Leff) or an threshold voltage (Vth) increase will result in higher impedance and slow device.

➔ **Conventional:** PV results in a lowered current through the core. This results in a higher delay and a slower core. To compensate, higher voltage can be applied.

➔ **Stacked:** the same current passes through the stack, therefore, higher impedance results in a higher voltage across that core, which naturally compensates the PV effects.

Case study: Inverter chain

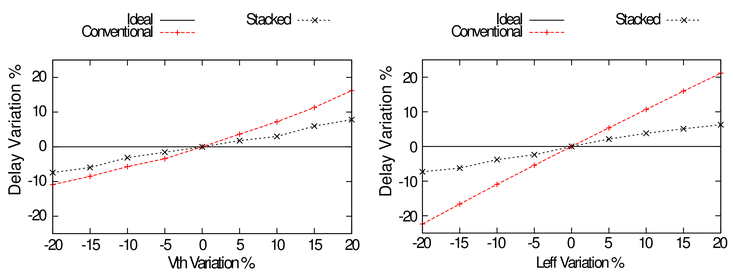


Fig 2. Voltage Stacking naturally compensates PV effects.

Shared Nets Configuration:

➔ PV only known post-silicon.
➔ To allow post fabrication changes we propose Shared Nets (SNETs).
➔ Switches or fuses are used to regulate stacking.
➔ Number of SNETs decided on design time through simulation.

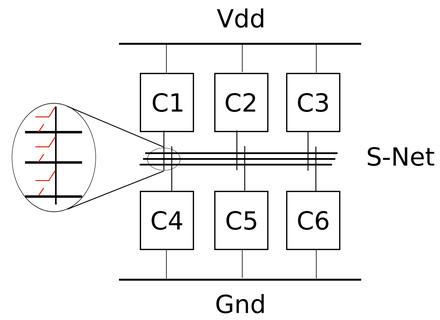


Fig 3. Shared Nets allow for post-silicon configuraility.

What to stack?

➔ Stacking needs balanced load ➔ Stack equal structure.
➔ Stack GPU PEs (lanes) within a SM.
➔ Within a SM, lanes work in lock-step, running the same program.
➔ Foot/Head position is fixed during design time for simplicity.
➔ Compensantion works well for cores with reverse variation.
➔ Stack most positive with most negative variation.
➔ Cluster multiple lanes per SNET.

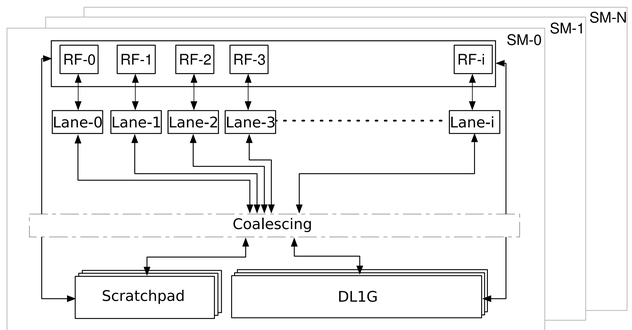


Fig 4. GPU Architecture is composed of uniform PEs, ideal for stacking.

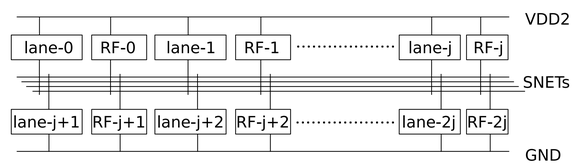


Fig 5. For simplicity, PE position is fixed during design time. After fabrication cores with opposed variation are stacked.

Evaluation Setup:

1) Expected compensation over 10k GPU dies:
   ➔ Varius-NTC [3] to generate 10k variation maps.
   ➔ Calculate expected voltage (from impedances).
   ➔ Varius-NTC [3] to calculate the new delay/power.

2) Check power delivery quality:
   ➔ ESESC [4] to generate power traces.
   ➔ Generate a time varying impedance model for each core.
   ➔ IBM PowerGrid Benchmarks to model power grid.
   ➔ SPICE simulation of the power grid with the core models.

Overal Results:

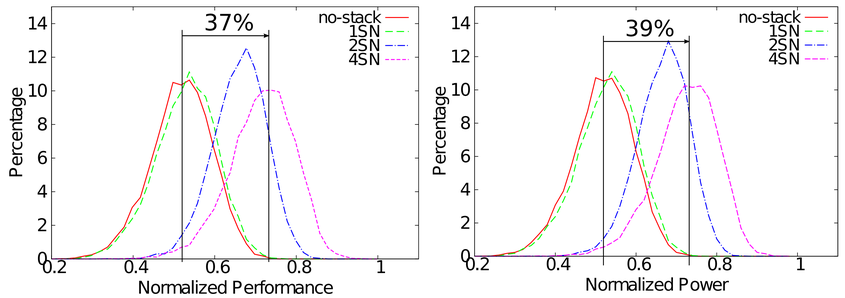


Fig 6. GPU Stacking shifts power and performace towards the ideal (ie no PV) scaling conditions.

Power Grid Quality:

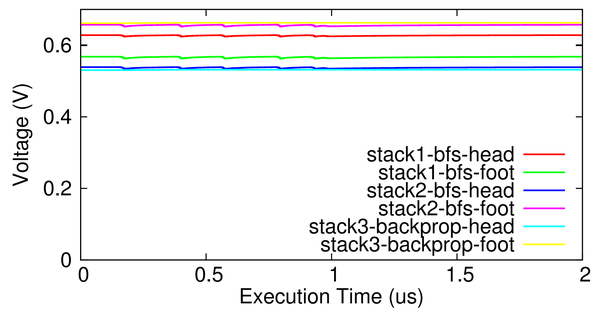


Fig 7. GPU Stacking keeps Vdd and Vmid within 10% of the expected value 99% of the time, and within 15% of the expected value 100% of the time for all the benchmarks tested.

Conclusions:

➔ GPU Stacking manages PV.
➔ Stacking can increase performance under PV at NTC on average by 37%.
➔ Stacking delivers 80% of the performance compared to the no variation conditions.
➔ GPU Stacking did not hurt power delivery quality.
➔ GPU Stacking reduces IR drop.
➔ Reduces the pressure in power grid design.
➔ GPU Stacking is also able to compensate PV effects.

[1] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudgi, “Near-threshold computing: Reclaiming moores law through energy efficient integrated circuits,” February 2010.
[2] S. Lee, D. Brooks, and G. Wei, “Evaluation of Voltage Stacking for Near-threshold Multicore Computing,” in Low Power Electronics and Design (ISLPED), 2012 IEEE International Symposium on, pp. 373–378, ACM, 2012.
[3] U. Karpuzcu, K. Kolluru, N. Kim, and J. Torrellas, “Varius-ntv: A microarchitectural model to capture the increased sensitivity of manycores to process variations at near-threshold voltages,” in Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on, pp. 1–11, IEEE, 2012.
[4] E. K. Ardestani and J. Renau, “ESESC: A Fast Multicore Simulator Using Time-Based Sampling,” in International Symposium on High Performance Computer Architecture, HPCA’19, 2013.