AMD

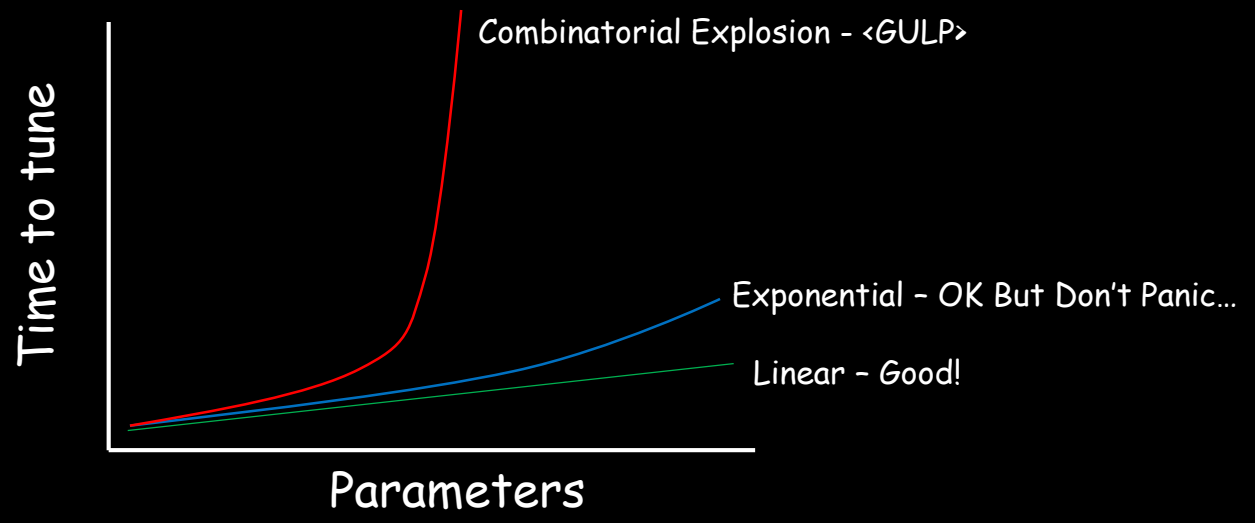
COMPUTER ARCHITECTURE AND FEEDBACK CONTROL

CHALLENGES AND OPPORTUNITIES

KARTHIK RAO

OUTLINE

- “Simple yet effective”
  - Trade-off between software and firmware
  - Code reusability
- Tuning multiple parameters
- Safety controllers
- Machine Learning Solutions
  - Data
  - Security and privacy
  - Time to market
- Designing for heterogeneous systems
  - Controller composability across internal IPs
  - Controller composability between vendors



FEEDBACK CONTROL: STATE-OF-THE-ART

HEURISTICS-BASED DESIGNS

- If (A < Threshold) do B else do C
- Simple and easy to implement
- Lots of parameters. Hence hard to tune
- Verifying correctness is expensive for complex systems
- Examples
  - CoScale (2012) [1]
  - Crank It Up or Dial It Down (2013) [2]
  - Harmonia (2015) [3]

PID AND ADAPTIVE CONTROLLERS

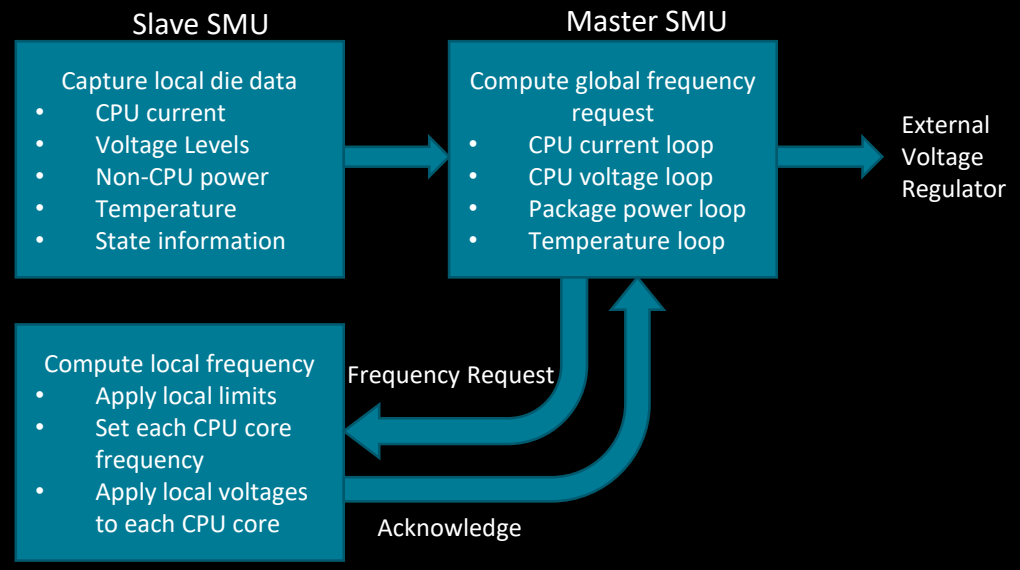
- PID is simple. Hence a popular choice
- Single Input Single Output (SISO)
- PID is great for Newtonian systems
- Computers are inherently non-linear
  - Adaptive gain controllers address some non-linearities
- Examples:
  - Temperature Aware Microarchitecture [4]
  - Adaptive Gain Power Regulator [5]

ADVANCED CONTROL THEORY

- Mathematically rigorous
- Multi – Objective optimization
- Linear Quadratic Gaussian regulator (LQG), Model Predictive Control (MPC)
- Multiple Input Multiple Output (MIMO)
- Examples:
  - DTM with MPC (2009) [6]
  - DTM with Convex Optimization (2008) [7]
  - MIMO and LQG [8]

HEURISTICS-BASED DESIGNS
SIMPLE YET EFFECTIVE

- Heuristics-Based and PID controllers are great candidates for simple designs
- Thresholds in heuristics and Gains in PID require tuning
  - Time consuming process
- Software implementation (e.g., `cpufreq`)
- Suits firmware implementation as well! (e.g., System Management Unit (SMU) [9])
  - Faster response times compared to software
  - Small code size
  - Meets real-time constraints
- Core reusability is very important
  - Must be applicable across products with “minimal” tuning
  - Time to market
- Basically, works like a charm! Until…



MULTI – PARAMETER TUNING

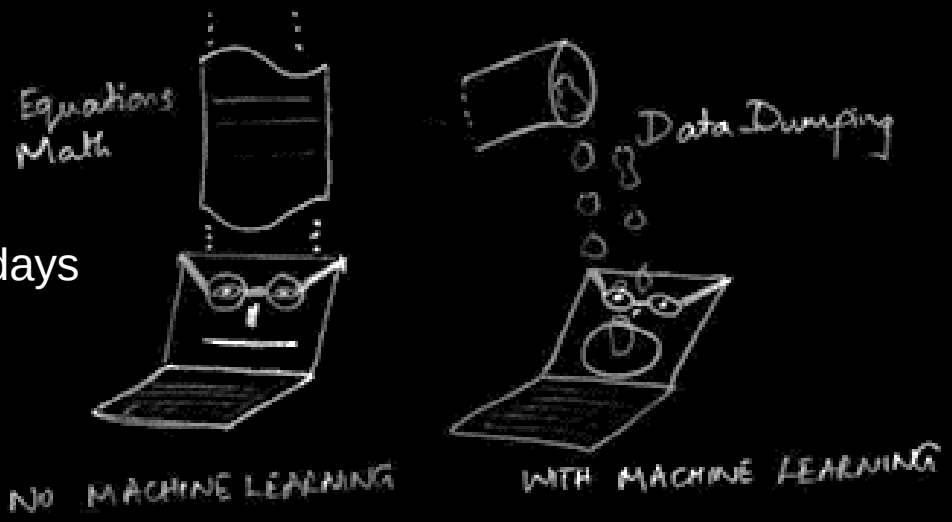
- Performance, Power, Temperature and Energy (PPTE) – high-level outcomes
- Parameters affecting PPTE
  - Cache sizes, associativity, issue width
  - Capacitance, frequency, voltage, leakage current
  - Application behavior: compute vs. memory, bursty vs. steady, CPU vs. GPU
- Ideal situation: PPTE = *f* (All possible parameters) and sensitivity of PPTE w.r.t. each parameter is well understood
- Reality: The model *f* is hard to characterize. *Shmoo*–ing each parameter is practically infeasible
  - Cache size affects performance and power and hence temperature which in turn affects leakage currents
  - Power dissipated in GPU reduces boost clock frequency of CPU (Cooperative Boosting 2013 [10])
- The BIG UNKNOWN: Application behavior
  - Characterize the system and tune for a generic set of applications (3DMark®, PCMark®, Cinebench®, SPEC Power®, etc.)
- Heuristics-based methods and PID controllers begin to show their limitations in such complex environments
- CHALLENGES: What is the best approach to handling multiple parameters?
  - Smaller, simplified piece-wise linearized models? If yes, what determines the metric “simplified”?

SAFETY CONTROLLERS

- Potential Hazards on a typical processor
  - Over heating (> T_j), Thermal Design Power (TDP), Thermal Design Current (TDC), Electrical Design Current (EDC), Current spikes (dI/dt)
- SMU firmware monitors each individual hazard at fine granularity
- Academic proposals do not always consider them
  - Designing within TDP and temperature limits as constraints is only the first step
- A well-rounded, practical power management solution MUST address these critical hazards swiftly and effectively
- Control theory (e.g., Robust control) provides some guarantees on PPTE
- But without explicitly addressing safety hazards can render them practically unsuitable
- CHALLENGES: How should we integrate safety controllers alongside PPTE optimization controllers?
  - Softmax?
  - Use Machine Learning to classify hazard scenarios?
  - Composability (will talk about it in a few slides)

USING MACHINE LEARNING

- "New kid on the block" challenging traditional PID and heuristics-based techniques
- Hank will give an in-depth review later
- Seems to work great when there is lots and lots of training data
  - But how much data is "sufficient"? Does it cover all possible use-cases?
  - Should we simply let ML handle everything?
- User Data Collection: Not the most popular set of three words these days
  - AMD cares about security and privacy
  - Cannot necessarily train on user-specific data (could cause overfitting)
- Model Training Time
  - ML training is time consuming
  - Product shipping deadlines are tight: Time to market is extremely critical
- CHALLENGES: If we decide to take the ML route,
  - Which ML algorithm is best suited for practical implementation? (Recall: Firmware has tight real-time and space constraints)
  - Do we know if applications X, Y and Z represent the full spectrum?
  - What about code reusability?



HETEROGENEOUS SYSTEMS

Cores + Cache

DSP, Network

GPU

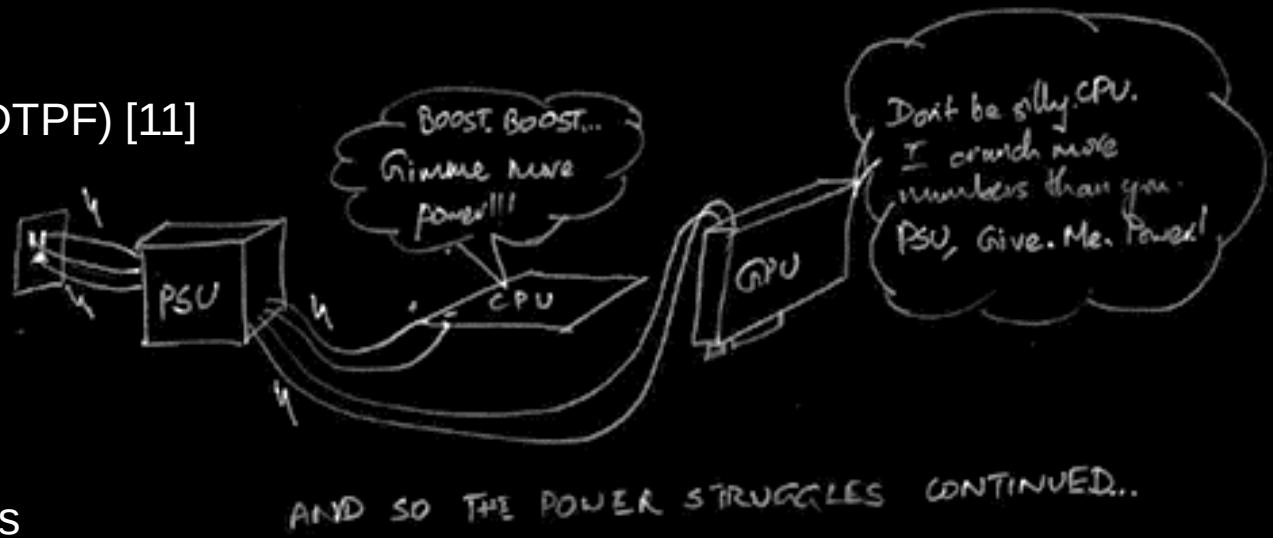
Mem Ctrl + IO

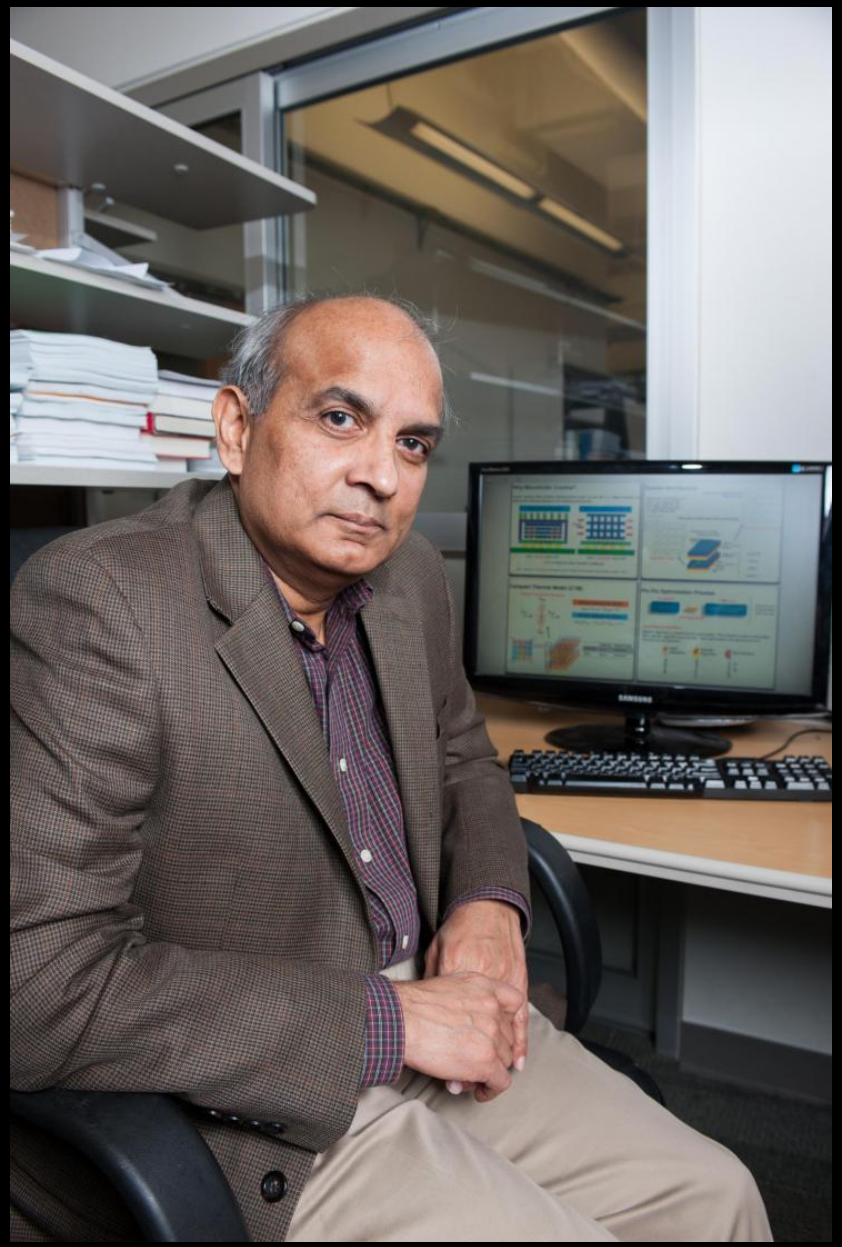
HETEROGENEITY WITHIN A PROCESSOR

- AMD's Accelerated Processing Unit (APU): Heterogeneity in processor types
  - AMD Ryzen™ 7 3750H Mobile Processor with Radeon™ RX Vega 10 Graphics
- Arm® big.LITTLE: Heterogeneity in CPU types
- Power management is a challenge, but all IPs are "in-house"
- Power and thermal models are well understood
- Application characteristics driven power sharing between CPU and GPU (between big and LITTLE cores)
- Chiplet based architecture
  - Great for performance scaling
  - Challenging for power management

SYSTEM–WIDE HETEROGENEITY

- Intel uses Dynamic Power and Thermal Framework (DTPF) [11]
  - Centralized Kernel driver
  - Each chip exposes sensor data
  - DTPF controls the chip inputs
- IBM POWER9 uses On Chip Controller (OCC) [12]
  - Centralized hardware controller
  - Master – Slave strategy for 2 – socket configurations
- Composability of controllers in heterogeneous systems
  - Each vendor optimizes its IP “locally”. Does it imply “global” optimality as well?
  - Will locally stable controllers work harmoniously when composed together in a large system?
  - Centralized controllers do not scale
  - What is the right granularity for de-centralizing individual controllers?
- Computing systems can be assembled with parts from multiple vendors (AMD, Intel , NVIDIA etc.)
- How should we separate power control between OS and FW?
- Control theory could shed some light on these questions: Raghav will talk in detail next





IN MEMORY OF MY ADVISOR

DR. SUDHAKAR YALAMANCHILI

THANK YOU !
QUESTIONS?

Thanks to Joseph Greathouse, Indrani Paul, and Leonardo Piga for their inputs!


AMD

REFERENCES

[1] Qingyuan Deng, David Meisner, Abhishek Bhattacharjee, Thomas F. Wenisch, and Ricardo Bianchini. 2012. CoScale: Coordinating CPU and Memory System DVFS in Server Systems. In International Symposium on Microarchitecture.

[2] Augusto Vega, Alper Buyuktosunoglu, Heather Hanson, Pradip Bose, and Srinivasan Ramani. 2013. Crank It Up or Dial It Down: Coordinated Multiprocessor Frequency and Folding Control. In International Symposium on Microarchitecture.

[3] Indrani Paul, Wei Huang, Manish Arora, and Sudhakar Yalamanchili. 2015. Harmonia: Balancing Compute and Memory Power in High-performance GPUs. In International Symposium on Computer Architecture.

[4] Kevin Skadron, Mircea R. Stan, Wei Huang, Sivakumar Velusamy, Karthik Sankaranarayanan, and David Tarjan. "Temperature-aware microarchitecture." In 30th Annual International Symposium on Computer Architecture, 2003. Proceedings., pp. 2-13. IEEE, 2003.

[5] Nawaf Almoosa, William Song, Yorai Wardi, and Sudhakar Yalamanchili. 2012. A Power Capping Controller for Multicore Processors. In American Control Conference.

[6] Francesco Zanini, David Atienza, Luca Benini, and Giovanni De Micheli. "Multicore thermal management with model predictive control." In 2009 European Conference on Circuit Theory and Design, pp. 711-714. IEEE, 2009.

[7] Srinivasan Murali, Almir Mutapcic, David Atienza, Rajesh Gupta, Stephen Boyd, Luca Benini, and Giovanni De Micheli. "Temperature control of high-performance multi-core platforms using convex optimization." In Proceedings of the conference on Design, automation and test in Europe, pp. 110-115. ACM, 2008.

[8] Raghavendra Pradyumna Pothukuchi, Amin Ansari, Petros Voulgaris, and Josep Torrellas. 2016. Using Multiple Input, Multiple Output Formal Control to Maximize Resource Efficiency in Architectures. In International Symposium on Computer Architecture.

[9] Thomas Burd, Noah Beck, Sean White, Milam Paraschou, Nathan Kalyanasundharam, Gregg Donley, Alan Smith, Larry Hewitt, and Samuel Naffziger. 2019. “Zeppelin”: An SoC for Multichip Architectures. IEEE J. Solid-State Circuits 54, 1 (Jan. 2019), 133–143. https://doi.org/10.1109/JSSC.2018.2873584

[10] Indrani Paul, Srilatha Manne, Manish Arora,W. Lloyd Bircher, and Sudhakar Yalamanchili. 2013. Cooperative Boosting: Needy Versus Greedy Power Management. In International Symposium on Computer Architecture.

[11] Intel Corporation. 2015. Intel Dynamic Platform and Thermal Framework (DPTF) for Chromium OS. https://01.org/intel%C2%AE-dynamic-platformand-thermal-framework-dptf-chromium-os/documentation/implementationdesign-and-source-code-organization. Accessed: 2019.

[12] Todd Rosedahl, Martha Broyles, Charles Lefurgy, Bjorn Christensen, and Wu Feng. 2017. Power/Performance Controlling Techniques in OpenPOWER. In High Performance Computing, Julian M. Kunkel, Rio Yokota, Michela Taufer, and John Shalf (Eds.). Springer International Publishing, 275–289.

Disclaimer

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.