HCAPP: Scalable Power Control for Heterogeneous 2.5D Integrated Systems

Kramer Straube
kkstraube@ucdavis.edu
University of California, Davis
Davis, California

Jason Lowe-Power
jlowepower@ucdavis.edu
University of California, Davis
Davis, California

Christopher Nitta
cjnitta@ucdavis.edu
University of California, Davis
Davis, California

Matthew Farrens
mkfarrens@ucdavis.edu
University of California, Davis
Davis, California

Venkatesh Akella
akella@ucdavis.edu
University of California, Davis
Davis, California

ABSTRACT

Package pin allocation is becoming a key bottleneck in the capabilities of designs due to the increased bandwidth requirements. 2.5D integration compounds these package-level requirements while introducing an increased number of compute units within the package. We propose a decentralized power control implementation called Heterogeneous Constant Average Power Processing (HCAPP) to maintain the power limit while maximizing the efficiency of the package pins allocated for power. HCAPP uses a hardware-based decentralized design to handle fast power limits, maintain scalability and enable simplified control for heterogeneous systems while maximizing performance. As extensions, we evaluate a software interface and the impact of different accelerator designs. Overall, HCAPP achieves 7% speedup over a RAPL-like implementation. The power utilization improves from 79.7% (RAPL-like) to 93.9% (HCAPP) with this design. A priority-based static software control methodology alongside HCAPP provides average speedups of 8.3% (CPU), 5.4% (GPU), and 12% (Accelerator) for the prioritized component compared to the unprioritized version.

ACM Reference Format:
Kramer Straube, Jason Lowe-Power, Christopher Nitta, Matthew Farrens, and Venkatesh Akella. 2020. HCAPP: Scalable Power Control for Heterogeneous 2.5D Integrated Systems. In *49th International Conference on Parallel Processing - ICPP (ICPP '20), August 17–20, 2020, Edmonton, AB, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3404397.3404448

1 INTRODUCTION

With the end of Moore's Law and Dennard scaling [8], the focus in the computer design community has shifted to application-specific architectures [4]. These architectures provide promise for technology node agnostic speedups at the cost of program flexibility. 2.5D [7] and 3D integration [19] designs have arisen to combine multipurpose designs, such as CPUs and GPUs, with targeted accelerators. Intel [2] and AMD [1] have already announced 2.5D integration designs. Under 2.5D integration, multiple chiplets exist within a single package. The package connects the silicon die to the motherboard socket with small wires connected to larger pins. These chiplets share the resources of the package, such as the package pins.

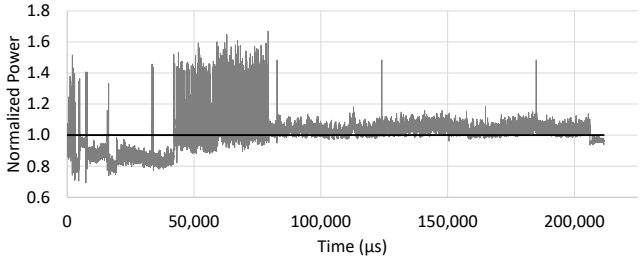

Figure 1: Power usage of heterogeneous system running workloads on all subcomponents in a static configuration normalized to the average power.

The relative scarcity of package pins causes sacrifices in IO and memory interfaces to allocate enough pins for power. The number of pins on a package does not scale with the number of chiplets added to a 2.5D system. Already, in larger designs, the fixed number of pins is a budgeted resource [20]. The number of power pins allocated must be selected based on the worst case package power consumption. The extra pins provisioned for the worst case power come at the cost of memory bandwidth and IO channels. Reducing the provisioned power limit results in lower performance despite freeing additional pins. The provisioned power limit is set by the required performance of the design in the worst case scenario.

Maximizing the use of the provisioned power pins provides additional performance without any additional package pins. The goal in maximizing the utilization of the package power pins lies in keeping the average power as close to the provisioned power as possible without exceeding it. We introduce Provisioned Power Efficiency (PPE) which determines how efficient a power control design is at utilizing the package power pins.

The current problems limiting the design of a fast efficient power control mechanism for 2.5D designs is three-fold:

(1) It is difficult to universally determine whether giving a component additional power will result in a speedup. CPUs,

ICPP '20, August 17–20, 2020, Edmonton, AB, Canada

ACM ISBN 978-1-4503-8816-0/20/08…$15.00
https://doi.org/10.1145/3404397.3404448

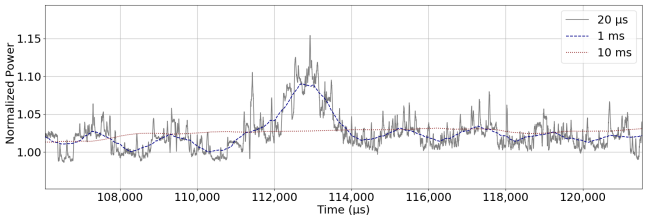

Figure 2: Zoomed in section of the power draw from Figure 1 adjusted for different power limit time windows. Note that the power peaks seen at the $20\mu s$ time window are not visible at the other time windows. This represents power behavior that firmware-based or software-based controllers could not account for without guardbanding.

GPUs and accelerators can all have different relevant metrics for determining their ability to use additional power effectively

(2) The time windows for power limit specifications for 2.5D integrated designs are small. ($10\mu s$-$10ms$)

(3) Current power control methods cannot handle the increased scale as 2.5D integrated systems continue to grow with higher numbers of chiplets on a single interposer.

Difficult universal power-performance efficiency prediction

Many power control methods, such as Intel's Running Average Power Limit (RAPL) or software-based power control methods, determine the future settings of all components in a single controller. The issue of how to determine power profiles and priority between heterogeneous components is difficult. This issue is compounded by the variety of 2.5D designs as different types of accelerators are added or replaced. It is difficult to create a single algorithm that properly controls the variety of component types. This algorithm needs continual adjustment as new systems are designed with different components.

Small time window for power limits

Power limits dictate a maximum power and a time window over which that maximum power is evaluated. If this window did not exist in the specification, then the current draw from capacitors or other components could spike for an extremely short period of time (such as 1 picosecond) to charge up but then drop to zero. The specification focuses on specific time windows to prevent potential issues such as the inability of a voltage regulator to source sufficient current over a larger time window after internal capacitors handle any exceedingly fast spikes. A dynamic scheme is necessary to maximize the utilization of the provisioned power. Figure 1 shows an example of the power draw of a chiplet-based design with a CPU, GPU and SHA accelerator during a test with no power management in a simulated environment. In these multi-chiplet systems, the power becomes volatile due to the behavior of multiple different architectures active simultaneously. The package power limits are defined with time intervals as short as tens of microseconds. Figure 2 shows a section of the previous power draw over different time windows. The grey curve shows the approximate time window required for the package pin power limit ($20\mu s$) while the blue curve and red curve show slower power limit time windows (1 ms and 10 ms). The grey curve remains the constraint even if the measurement and control design cannot accurately control it, often leading to large guardbands.

Software-based power control schemes operate on the order of milliseconds. Thus, software-based dynamic power control schemes cannot handle the fast time intervals in the power limit specifications enforced by the package pin constraint with the behavior of the multi-chiplet system. The fast changing nature of the power in the system requires a high-speed control mechanism. This requires a hardware-based approach. Any centralized approach, such as RAPL, requires enough time for the metrics to be aggregated at the central controller.

Scaling with 2.5D integration

2.5D integration provides the opportunity to radically scale the size of the system in a single package by using many chiplets. As the number of chiplets increases, the number of components in the system increases. With increasingly large systems, in terms of number of compute components, centralized designs such as Intel's RAPL cannot scale easily. These centralized designs need to aggregate metrics from each component to the controller. As the designs increase in scale, the resources in the form of global wires or buses cannot scale adaquately due to routing and congestion issues. These are similar to the issues seen in on-chip networking where crossbars and fully connected networks became inviable.

To solve these three issues, we propose a decentralized power control scheme called Heterogeneous Constant Average Power Processing (HCAPP). HCAPP leverages two main innovations to improve the provisioned power efficiency and moderate the issues enumerated above. First, each component has its own local controller to intepret when and how to use any extra available power. Trying to determine a centralized definition of the amount of work that can be done with increased power consumption across the many different possible component types (CPU core, GPU Cuda core, Media accelerator, etc.) is difficult. Instead, those definitions remain locally defined using specialized logic for that component based on local metrics. Second, the use of the power supply network to communicate using the universal language of voltage and current. This overcomes the issues with global communication without needing additional global bandwidth and maintaining a fast reaction time through the use of a global power controller. The lack of any additional global components allows HCAPP to scale with larger 2.5D systems. As the scale increases, new components will have local controllers and the overall control time of HCAPP is dictated by the physical behaviors of the power supply and distribution characteristics.

Figure 1 shows that for a fixed frequency the peak power is 60% higher than the average power giving a provisioned power efficiency (PPE, defined formally in Section 5) of 62.5%. In this case, the SoC designer must provision (pay) for 60% more pins for power delivery than are used on average. HCAPP's goal is to increase the PPE bringing it closer to 1.0 which results in either saving money by decreasing the number of pins or increasing performance for the same number of pins. We show HCAPP achieves a higher PPE (93.9% on average) than both the firmware-based RAPL scheme (79.9% on average) and software-based capping scheme (69.2% on average).

Throughout this paper, we present HCAPP as a scalable power control implementation for heterogeneous 2.5D integrated systems. HCAPP achieves 7% faster performance than a RAPL-like implementation while improving the provisioned power efficiency by 14.2%. HCAPP also provides an interface for software-based control. A naive static software control scheme achieves speedups of 5-12% with HCAPP over a software-less implementation.

In this paper, we explain the design of HCAPP in detail beginning in Section 3. We define the experimental setup used for evaluating HCAPP including defining the target system in Section 4. We evaluate three power capping techniques: HCAPP, a RAPL-like implementation and a software-like implementation. For each of these control implementations, we evaluate the maximum power, speedup and provisioned power efficiency across a defined test suite in Section 5. We also evaluate the impact of the software interface within HCAPP and the impact of differing component designs within the target system. Finally, we detail future areas to investigate.

2 RELATED WORK

Power capping is defined as maintaining a power limit for a system that can exceed. The idea behind this approach is to underprovisioned the system power relative to the worst case power and use various methods to reduce the worst case power draw. Another related approach is power shifting. This approach shifts power within a power constrained system to different components to improve performance. HCAPP uses elements from both power capping and power shifting to maximize performance under a power limit.

Turbo Boost and RAPL by Intel attempt to implement power capping [9]. Turbo Boost by design exceeds the power cap and instead maintains a thermal cap to improve performance. This invalidates Turbo Boost for a power limited system. RAPL implements power cap behavior by controlling the system power to a certain value. RAPL uses a centralized controller to accumulate all of the useful metrics from the various nodes (CPU cores, GPU streaming multiprocessor, etc.) in the package. Using these metrics and power information, the centralized controller assigns new configuration information (often voltage and frequency assignments) to each node in the system. RAPL's design causes several issues in the chiplet-based computing implementation. First, RAPL must use slow control times due to the necessity of accumulating the metrics from all areas of the package. This limits the ability of RAPL to control the power for certain power limit time windows, such as the power limit of the on-chip voltage regulator. Second, getting the information from each node to the centralized controller requires either separate global wires or shared resources, such as a bus or a network. Both of these solutions cause issues of either wire routing or congestion as the system continues to scale. Lastly, designing a centralized controller with logic for how all of the system metrics and power information can control the various nodes in a system becomes increasingly difficult. Since the non-CPU, non-GPU components will often change to enable speedups on a wide variety of implementations, the controller must find ways to understand the information from each different accelerator and how the configuration for the entire system should change.

Other approaches outside DVFS exist using the ability to scale voltage as a method to improve performance. Adrenaline [13] and Rubik [14] scale the voltage to reduce work variability in datacenter workloads. Harmonia [22] and DynaCo [23] change system configuration parameters dynamically to improve performance. Harmonia and DynaCo are software-based approaches and cannot handle fast power issues.

Several approaches exist targeting the heterogeneous design space with a focus on limiting the power. Co-Cap [21] limits the frequency of either the CPU or GPU to guide the power to the dominant component for that workload. Tsuzuku and Endo [29] maintain a power cap through software-based control. They combine preprofiled frequency models with realtime behavior analysis to reduce energy consumption without impacting performance. Bhattacharjee and Martonosi [6] measure thread criticality to use task stealing and DVFS control to reduce energy consumption without reducing performance.

Tangram [24] uses a decentralized controller architecture to manage heterogeneous systems to improve execution time but manages the system on the millisecond scale, too slow for certain power violations.

HCAPP builds upon the more specifically focused approaches CAPP [27] for CPUs and GPU-CAPP [26] for GPUs. These approaches provide a similar form of decentralized control with a local controllers at each component. CAPP focused on the capabilities of this approach for only CPUs and GPU-CAPP focused on only GPUs. Using the specific behavior and available information for GPUs, the local controller design was explored. These works demonstrate that HCAPP can function for each component and provide local controller designs. However, HCAPP extends these works by integrating CPUs, GPUs and accelerators into a single controlled system.

3 HCAPP FRAMEWORK DETAILS

Heterogeneous Constant Average Power Processing (HCAPP) uses a multi-level modular decentralized power control architecture to enforce an overall power limit while maximizing performance for the various subcomponents. HCAPP is an extension of CAPP and GPU-CAPP to go beyond just the CPU or GPU to the entire chip [26, 27]. HCAPP uses a 3 level architecture with each level optimizing the power usage for a particular goal. Figure 3 shows a conceptual implementation of HCAPP for an example system. The top level global controller maintains the overall power limit and tries to use up the excess available power. The second level domain controller normalizes the voltage to the subcomponent's allowable voltage range and provides a software interface to enable software to inform the hardware power control. The third level (if applicable based on the subcomponent) uses local metrics to perform slight voltage modifications to increase or decrease power usage for optimal efficient performance of that component. The combination of these behaviors enables a dynamic power management scheme that seeks to efficiently use all of the available power without extra limitations on scalability of the system. Since each controller has a simplified task, the design of these controllers can be far simpler than trying to design a monolithic controller for all sub-components in a heterogeneous system.

3.1 Global Controller

The global controller modifies the global voltage to enforce the specified power limit over a preset amount of time. This controller

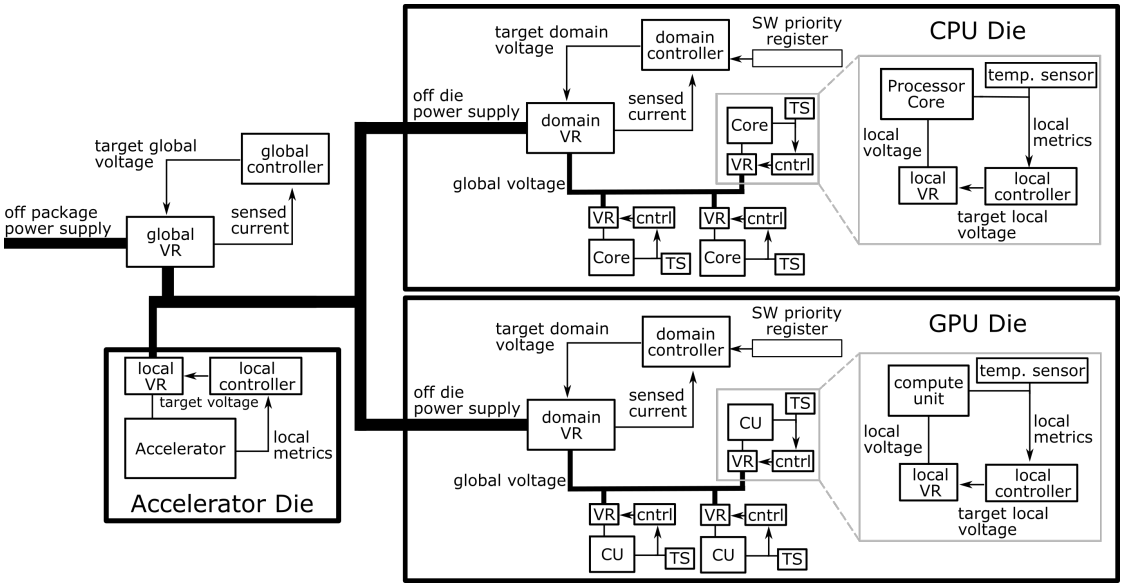

Figure 3: HCAPP: High Level Architecture

uses sensing circuitry built into the voltage regulator (VR) to measure the current and voltage, as seen in commercially available VRs [25]. From these values, the global controller determines the current total chip power. The global controller converts the total chip power to a new voltage output for the VR through two steps. First, the controller calculates the voltage error with Equation 1. The voltage error uses a cubic root of the power error due to the approximate cubic relationship between power and voltage.

$$V_{Err} = \sqrt[3]{P_{SPEC} - P_{NOW}} \quad (1)$$

$$V_{NEXT} = V_{Offset} + K_P * V_{Err} + K_I * \int V_{Err} dt + K_D * \frac{dV_{Err}}{dt} \quad (2)$$

Next, the controller uses PID control shown in equation to convert the voltage error to the new global voltage. The constants ($V_{FeedForward}$,K_P,K_I,K_D) in the PID equation must be tuned to the system to properly maintain the power limit. PID controllers are an implementation of a closed loop control methodology that is widely used to control a variety of systems [3]. We use a common variant of the PID controller which adds an open loop component, the feed forward value. The feed forward value can provide additional responsiveness and stability to a PID implementation. The speed of the control logic impacts the tuning values of the PID function and the capabilities of the controller. The global controller operates at a control cycle time that is long enough to allow the new voltage to propagate to each subcomponent and the new current draw to propagate back to the VR sensing hardware.

PID Tuning

The PID controller of the global voltage controller requires tuning to set the various parameters. The parameters required are the offset voltage (V_{Offset}) and the PID constants (K_P,K_I,K_D). The offset voltage is set to approximately the average voltage expected throughout execution.

To tune the controller constants, we run a single workload combination over a range of proportional gain values until the behavior became unstable. Then, we increase the integral gain value until the steady state output reached the desired behavior. The derivative portion of the PID design is generally unneeded. This results in a PI controller. The tuning for a single benchmark must be verified against the entire experiment workload set to ensure proper behaviors.

3.2 Domain Controller

The second level controller or domain controller normalizes the global voltage to a usable range for the specific subcomponent type using a VR. The specific VR is normally a smaller version of the global VR. In 2.5D integration, a VR is needed for each chiplet due to the differing voltage requirements across the chiplets. For example, a processor may need a voltage in the range of 1V while a specific accelerator needs the input voltage to be between 0.6V to 0.8V. These different domains require the global voltage to be normalized to the local requirements. Certain subcomponents, such as memory, need a constant voltage. In 2.5D integration the various chiplets can be produced on different technology nodes which require different voltage ranges. The domain controller chooses the appropriate scaled value or fixed value based on the needs of the chiplet it controls.

The domain controller also provides an interface for the operating system to interact with the power management hardware based on expected program behavior. The software interface could implement a number of different behaviors such as changing the ratio of the global voltage being used for the domain voltage of each component. The domain controller uses the priority value as a scaling factor for the domain voltage calculation. When a domain de-prioritized by 10%, the domain voltage controller multiplies the global voltage by 0.9x before doing any domain-specific scaling. This provides a sample implementation for how the domain controller can take software inputs to guide power control decisions.

This interface would use a register for each domain controller that contains the relative priority value for that domain. The incoming global voltage would be scaled by the priority value for that domain. The operating system can change the priority value dynamically by modifying the register value. Other implementations for the domain controller software interface are possible.

3.3 Local Controller

The local controller leverages local metrics to change voltage and power behavior locally to improve efficiency but cannot apply to components lacking the ability to change voltage. For example, these local controllers exist for each streaming multiprocessor in a graphics processing unit and for each core in a compute multiprocessor. The local controller uses local metrics such as IPC to change the voltage (and potentially frequency) used by the logic block via a local VR. This enables idle or underutilized components to reduce their power usage and busy components to increase their power usage. The goal at this level is to use power efficiently. This concept of the local controller is proposed in both CAPP [27] for CPUs and GPU-CAPP [26] for GPUs.

The local controller also monitors the component for any thermal effects using local thermal sensors. The thermal effects do not affect the results of this paper because we assume that the system is operating below the thermal limit at all times through careful selection of the power limit. If thermal effects did exist throughout the workload, the local controller would reduce the local voltage at the affected component to prevent failure. Additionally, software-based control can be used due to the slower timescale of thermal effects.

3.3.1 CPU Local Controller. The specific CPU local controller design was implemented in CAPP [27]. We reuse the local controller design from CAPP. This local controller changes the ratio of the domain voltage to use locally based on static thresholds. The behavior is controller by the instructions per clock (IPC) of each core. In the CPU-only evaluation, this local controller design outperforms the alternative designs including a CAPP design lacking a local controller [27]. Using a local controller, allows the CPU cores to dynamically balance relative to the domain voltage. IPC is a relevant metric for the work efficiency of processor cores.

3.3.2 GPU Local Controller. The GPU local controller design determines how the GPU compute units (CUs) measure their efficiency and control the local voltage. GPU-CAPP performed an examination of several possible GPU local controller designs [26]. Ultimately, the most effective GPU local controllers were the dynamic warp design and the dynamic IPC design. We use the dynamic IPC design in this paper. The dynamic IPC local controller changes the local voltage ratio of the domain voltage to use based on the IPC metric.

The specific thresholds used to determine the metric behavior change based on the behavior of the domain voltage. The local controller increases the thresholds when the domain voltage is below a preset target domain voltage value. Similarly, when the domain voltage is above the target value, the local controller decreases the thresholds. This behavior seeks to stabilize the domain voltage at a middle value by forcing the various CUs/cores to separate into a balanced distribution of higher and lower local ratios. This prevents the issue of static thresholds becoming inaccurate for many workloads because they will update to an accurate value. For example, as the thresholds increase, more of the CUs/cores will fail to reach the thresholds and begin to reduce their local voltage. As the local voltage ratios decrease on some of the CUs/cores, the power consumption of the entire domain decreases. This power reduction continues to the global controller calculates a higher global voltage due to the lower current power draw. As a result, the domain voltage gets closer to the target domain voltage and the CUs/cores are more evenly distributed in local voltage ratios. The CUs/cores that were able to meet the higher threshold requirements are more efficient than the ones that did not.

3.3.3 Accelerator Local Controller. The accelerator local controller design varies signficantly based on the accelerator implementation. For this design, we use a simple pass-through local controller which provides overvoltage and undervoltage protection but does not apply a local voltage ratio. Future implementations could add a more intelligent local controller based on the specific metrics and behavior of the accelerator.

The local controller design can change between accelerators or other components based on their specific needs. This flexibility opens up possible adversarial local controller designs that always use all of the available voltage possible, ignoring any local metric information. In this case, HCAPP would set the global voltage to an acceptable value on each control cycle to maintain the power limit. The performance of the other components would be impacted but only based on the actual consumed power. When the uncontrolled component is using less power, the other components would still benefit due to an increase in the global voltage.

3.4 Control Cycle Time of HCAPP

The control cycle time of HCAPP was determined through detailed modeling of the varied components in the system. The key components are outlined in Table 1. The delays of the components dictate the maximum possible roundtrip time of the power behavior in a system. The Raven voltage regulator design [16] specified the voltage regulator delays. This voltage regulator provides fast voltage changes which aids this design. The global voltage regulator and domain voltage regulator both add the delay of the Raven voltage regulator to the roundtrip latency. The power supply network behavior was based on Cadence Spectre simulations using the model by Gupta et. al [11] for on-chip power supply networks. We increased the delay of this model by 5x to account for the likely new delays in a 2.5D integrated system. These delays will vary with each implementation but provide a good approximate value. The sensing circuitry delay and controller delays were measured in Cadence Spectre simulations. With these delays accounted for, HCAPP uses a conservative control cycle time of $1\mu s$.

3.5 Assumptions and Implementation Issues

The design of HCAPP makes several assumptions about the design of the underlying system. The global controller can change the voltage at any time without any communication with the underlying system nodes. This behavior can lead to timing violations if this behavior is not properly accounted for. First, a voltage guardband can be used to ensure that each node has sufficient time to change the local ratio to reach an acceptable voltage setting. Second, the use of *adaptive clocking* at each node can ensure that as the voltage drops, the frequency also drops to a functional value. This technique is described by Keller [15] and applies only to clocked nodes.

Table 1: Breakdown of delays for HCAPP transitions. These values were obtained from literature, technical specifications and Cadence Spectre simulations

Component	Simulated Transition time (ns)	Scaled Transition time (ns)
Voltage Regulator (global and domain)	36-226 (x2)	72-452
Sensing Circuitry	50-60	50-60
Controller	10-30	10-30
Power Supply Network	3-15 (x5)	15-75
Total	**99-331**	**147-617**
HCAPP Control Period		**1,000**

Other nodes such as analog accelerators or asynchronous accelerators do not need this additional control as the mechanisms to handle the global controller behavior are built in. Adaptive clocking also handles any temporary voltage-related issues such as voltage glitches in the power distribution system. For all clocked nodes in this implementation, we assume that adaptive clocking is used but guardbands are a viable alternative.

If guardbanding was used instead of adaptive clocking, the local voltage would be slightly reduced to ensure stable operation of all clocked components. The local controllers could detect the reduced input voltage and reduce the local frequency to prevent any undervoltage scenarios. Since these actions are all local, they can occur very quickly, minimizing the required guardband.

In this paper, we assume that the power constraint is lower than the TDP so temperature effects are not modeled. Temperature-based control exists within the local controllers to handle thermal issues and software-based control could also aid with thermal effect mitigation through the HCAPP software interface.

4 EXPERIMENTAL SETUP

To evaluate the speedup and power management capabilities of HCAPP, we created a series of experiments. These experiments are conducted on a design with a CPU, a GPU, and an SHA accelerator. This system was operated at two power constraints: 100 Watts package pin limit and 100 Watts off-package VR limit. These two power limits test different possible power supply bottlenecks in the system by changing the time window for evaluating the maximum power. These time windows are based on the delay of a current change to arrive at either the package pins or the off-package voltage regulators. By testing at both of these points, we can also examine the sensitivity of HCAPP to the power limit definition. As a baseline, we simulated a fixed global voltage system with no local controllers. This provides the closest equivalent to a fixed frequency system in a heterogeneous design. The fixed global voltage for this system was 0.95V This voltage was selected because it achieved the highest performance without violating the power target. Each component of the system required different workloads and simulators. We designed and use a central simulation controller to implement the global power control behavior and manage the separate component simulators. All of the seperate component workloads were initialized simultaneously. Since the workloads had various lengths in their original implementation, we modified the short workloads to loop them to reach the same approximate timescale as the workloads for the other components. In practice, this only effected the GPU workloads.

Table 2: Details of CPU and GPU Configuration

Component	CPU	GPU
Units	8 Cores	15 SMs
Cores per SM	N/A	1
L1 Cache Size	32 kB	16 kB
Shared Memory Size	N/A	48 kB
L2 Cache Size	256 kB	768 kB
Maximum Frequency	2 GHz	700 MHz
Minimum Frequency	800 MHz	100 MHz

4.1 Central Simulation Controller

The central simulation controller serves two purposes: modeling the global controller and managing the overall simulation state between the various connected simulators.

4.2 CPU Simulator

We used Sniper as the CPU simulator for the CPU portion of our system [12]. Sniper is an x86 processor interval-based simulator from Intel. For power modeling of the CPU, we used McPAT [18].

We used the Nehalem model provided within Sniper. The details for this model are shown in Table 2. The specific CPU local controller was an IPC static controller.If the core IPC exceeds 60% of the maximum possible IPC, the local voltage ratio is increased by 0.05. If the IPC falls below 30% of the maximum possible IPC, the local voltage ratio is decreased by 0.05. The domain voltage controller was only used for the software interface capabilities as the default global voltage scale matched the CPU voltage scale.

For our simulations, we used a specific subset of the PARSEC benchmark suite that showed a variety of power behavior. These benchmarks selected are blackscholes, fluidanimate, ferret and swaptions. This subset captures a wide variety of power behavior on the CPU.

4.3 GPU Simulator

For the GPU portion of the experiments, we used a combination of GPGPUSim and GPUWattch. GPGPUSim is a cycle-level simulator that uses split modeling for the functional and timing portions of the GPU operation [5]. GPUWattch is a detailed energy model that is integrated with GPGPUSim [17]. To ensure that our models were as accurate as possible, we used the GTX480 GPU as the GPU component in our system. More details on the GPU model specifics are in Table 2.

The GTX480 is the most recent *validated* power model within GPUWattch. Using a validated power model is critical for accurate simulation results. The GPU model shows the effect of GPU-like behavior on HCAPP. Other GPUs could be used instead of the GTX480 but the limitations of the simulation environment require

the use of the GTX480 for maximum accuracy. Similarly, we use GPGPU-Sim version 3.2.2 because it is the newest validated release of GPGPU-Sim.

The local controller within the GPU model is a dynamic IPC-based local controller as previously described. In this implementation, the local controller changes the thresholds by multiplying the previous thresholds by ±5% at a time. There is an allowable deadzone of 5% between the target domain voltage and actual domain voltage. The target domain voltage value is 1.05V based on initial profiling.

The domain controller scales the global voltage by 75% to match the approximate voltage range of the GPU relative to the global voltage range.

The benchmarks for the GPU are a subset of the Rodinia benchmark suite [10]. The subset was selected based on power characteristics to provide a range of power behaviors. The specific benchmarks selected are backprop, bfs, myocyte and sradv2. These benchmarks capture a range of power characteristics and provide interesting combinations with other workloads.

4.4 SHA Accelerator Simulator

The SHA Accelerator was modeled in Python based on the power-throughput-voltage relationships from the design by Suresh et al [28]. The points from the relevant figures in the paper were put into lookup tables and, based on the provided voltage, throughput and power for a given time period were calculated. The local controller for the SHA accelerator scales the voltage by 75% to match the accelerator voltage range relative to the global voltage range.

The total work that the accelerator has to complete is modeled as a fixed number. The work completed on each cycle is linearly proportional to the maximum usable voltage setting for the accelerator. Each control cycle, we subtract the work done during that cycle (determined by the throughput over that time) from the total work. When the total work is less than or equal to zero, the accelerator can enter an idle state.

4.5 Why We Selected the GTX480 and PARSEC Benchmark Suite

We selected the GTX480 GPU model and PARSEC benchmark suite despite their relative age to ensure the accuracy of the simulation results. The GTX480 GPU model in GPUWattch is the newest *validated* power model. Using a newer GPU model would lead to inaccurate results since HCAPP uses the power draw of the system as an input. HCAPP will control the power despite the GPU design so the GTX480 design shows how GPU-like behavior will impact HCAPP's effectiveness. Other GPUs could be used instead of the GTX480 but the limitations of the simulation environment require the use of the GTX480 for maximum accuracy. We would expect similar results to the ones shown in Section 5 using newer GPUs. We used the PARSEC benchmark suite for its functionality and verified correctness within the Sniper simulator. Using another newer benchmark suite would reduce the number of usable benchmarks and accuracy of the benchmark behavior within the simulator. The PARSEC benchmark suite includes important program behaviors the are exercised within the simulator. Other benchmarks may include additional program behaviors but these behaviors do not invalidate the behaviors exercised by the PARSEC suite.

Table 3: Benchmark Combinations Used for Validation

Name	CPU	GPU	SHA
Low-Low	Blackscholes	Myocyte	Modeled
Low-Hi	Blackscholes	Backprop	Modeled
Hi-Low	Fluidanimate	Myocyte	Modeled
Hi-Hi	Fluidanimate	Backprop	Modeled
Mid-Mid	Swaptions	Sradv2	Modeled
Const-Burst	Swaptions	BFS	Modeled
Burst-Const	Ferret	Myocyte	Modeled
Burst-Burst	Ferret	BFS	Modeled

4.6 Relevant Comparison Systems

To provide comparisons for HCAPP, we evaluate HCAPP running at two slower control frequencies. These slower control frequencies provide approximate models for RAPL and software-based controllers. These two alternate versions: RAPL-like (100 microsecond control period) and Software-like (10 millisecond control period). We consider both of these control speeds to be aggressive for the systems they represent. Through the comparison of these designs with HCAPP (1 microsecond control period), we show the importance of fast adaptation time. This comparison answers the question of why a fully hardware-based design is necessary, especially as the number of components in a system increases.

5 PERFORMANCE ANALYSIS OF HCAPP

To evaluate the performance and power characteristics of HCAPP, we created a heterogeneous test suite for our system using the benchmark combinations shown in Table 3. The resulting test suite provides interesting power behavior combinations for the various sub-components. The names indicate the power behaviors. This first four tests cover the standard power corner cases. The last four tests exercise the system to demonstrate how bursty power behavior affects the system. The speedup enabled by HCAPP for any of these tests can be translated into extra performance without changing the package pins allocations or power limits.

5.1 Package Pin Power Limit Results

To evaluate the capabilities of HCAPP under a power limit dictated by package power pin allocation, we implemented a power limit of 100 Watts over 20 microseconds. 20 microseconds is an estimate of the amount of time for the power draw from the components in the system to reach the package pins. Failing this specification could result in additional allocated package pins to handle the burst of power. Under this power limit, we tested the all three HCAPP implementations relative to the fixed voltage design for maximum power draw, execution speedup and provisioned power efficiency (PPE).

Figure 4 shows the maximum power relative to the power limit drawn by the various HCAPP implementations and the fixed voltage system. This experiment determines whether an approach violates the power limit or not. Ideally, an approach would exactly reach 1.0 for all tests but that may not be possible due to the differences between the tests. For an approach to be viable, all of the maximum powers across the entire test suite must be below the 1.0 mark. Both the fixed voltage and HCAPP implementations stay below the allowable limit. RAPL-like HCAPP and Software-like HCAPP greatly exceed the 1.0 mark causing a power failure. Thus, these slower approaches cannot be used under the power limit. We omit

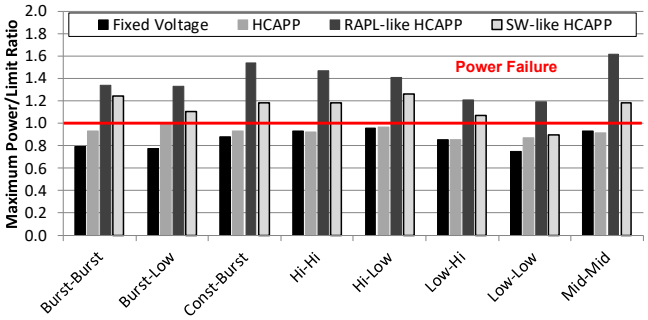

Figure 4: Maximum power relative to 100 Watt, 20 μs power limit. RAPL-like and SW-like HCAPP exceed the 1.0 line and violate the power limit.

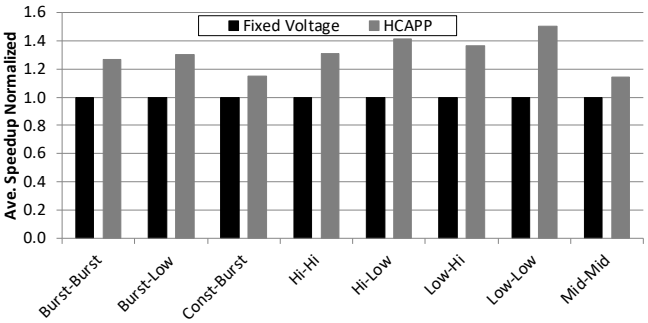

Figure 5: Speedup HCAPP relative to fixed voltage (0.95 V) system. HCAPP speeds up execution time by an average of 21%.

these approaches from the speedup and PPE results because these controller designs are determined to be invalid. In practice, these approaches could be used with additional guardbanding or more conservative schemes. To consider this case, we evaluate these approaches for performance and provisioned power efficiency under the slow power limit shown in Section 5.2.

Figure 5 shows the speedup achieved by HCAPP relative to the fixed voltage system system since these are the valid implementations based on the maximum power results. The available speedup varies based on the frequency sensitivity of the test and the extra available power throughout the execution of the test. The speedup is calculated by geometrically averaging the speedup of the three subcomponents: CPU, GPU and SHA. Equation 3 shows the exact calculation used for the total speedup results where $S_{component}$ denotes the speedup of that component. Overall, HCAPP achieves a 21% speedup over the fixed voltage (0.95 V) system. This speedup is attained by using additional available power throughout each test. By using HCAPP, the system achieves 21% speedup without any additional package pin requirements by using the package pins more efficiently throughout the tests.

$$S_{Total} = \sqrt[3]{S_{CPU} * S_{GPU} * S_{Accel}} \tag{3}$$

$$PPE = \frac{AveragePower}{SystemProvisionedPower} \tag{4}$$

To measure the package resource utilization efficiency of the designs, we define Provisioned Power Efficiency (PPE) in Equation 4. This metric provides insight into the average utilization of the

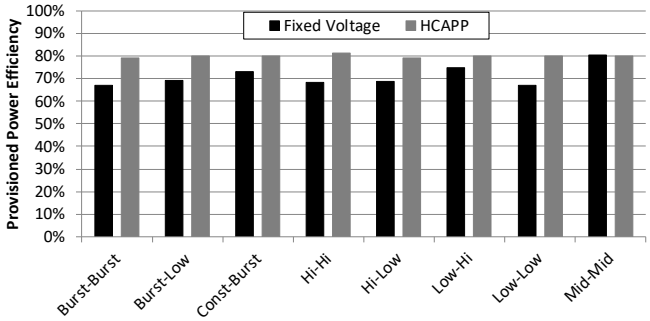

Figure 6: Provisioned power efficiency of HCAPP and fixed voltage system. HCAPP improves the PPE by an average of 10.2%.

package pins throughout the test. Figure 6 shows the provisioned power efficiency of HCAPP and the fixed voltage system relative to the power limit. Ideally, the provisioned power efficiency would always use up all of the available power and provide a result of 100%. However, this is very difficult due to the behavior of the tests. Programs have phases of different power behaviors result in less than ideal results (< 100%). Dynamic schemes must protect against behavior changes to ensure that the maximum power limit is not violated by including a guardband and similar conservative features. HCAPP improves the PPE compared to the fixed voltage system by 10.2%, raising it from 69.1% to 79.3%. This means that the voltage regulators or package pins used for the power distribution are 10% more utilized. This additional power utilization provides the speedup shown previously. The PPE across the entire suite shows very little variance due to the properties of HCAPP's power control. HCAPP has many control cycles within each run and targets a single power value. This repeated control with many control cycles ensures that the average power across the entire run is close to the power target. The power target is not the power limit because HCAPP will have maximum values above the power target and those cannot exceed the power limit.

5.2 Off-Package Voltage Regulator Power Limit Results

Not all designs will be limited by the package pin allocation. To evaluate the range of possible power limits, we evaluated HCAPP under a slower power limit based on an off-chip voltage regulator power supply limitation. This power limit specifies that the approaches cannot exceed 100 Watts over 1 millisecond based on the relative time specification for max off-chip voltage regulator power draw. Failure to meet this specification would result in higher costs due to larger voltage regulator implementations. Similarly to the previous results, we analyze the various HCAPP approaches for the maximum power draw, execution speedup and provisioned power efficiency. The PID control constants remained the same between the two power limits to show the flexibility of the design.

Figure 7 shows the maximum power relative to the power limit for the three HCAPP implementations under the slow power limit. The RAPL-like HCAPP implementation narrowly exceeds the power limit for the Const-Burst test. For the sake of analysis, we will analyze all of the HCAPP implementations in this section to see how the cycle time affects speedup and provisioned power efficiency. We do not reanalyze the fixed voltage configuration as the altered

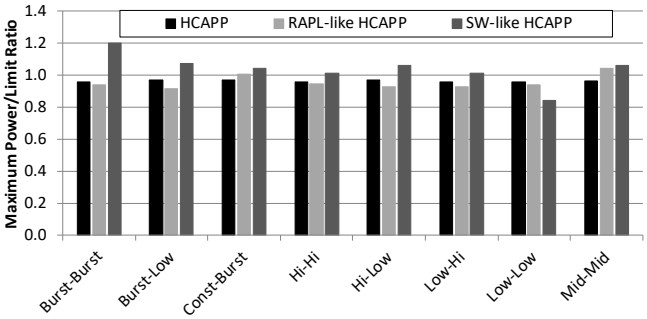

Figure 7: Maximum power relative to 100 Watt, 1 ms power limit. RAPL-like and SW-like exceed the 1.0 value again but we ignore this to allow further analysis.

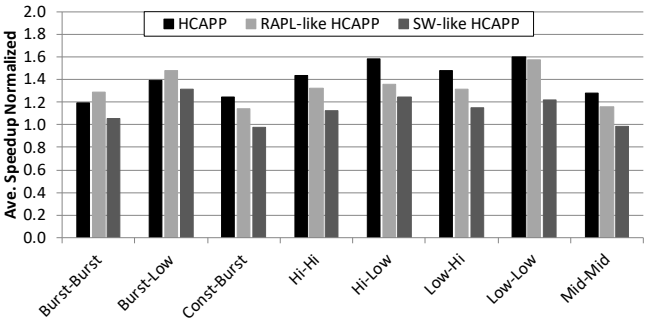

Figure 8: Speedup of multiple HCAPP versions relative to fixed voltage system under slow power limit. HCAPP achieves an average speedup of 43% while RAPL-like HCAPP achieves an average speedup of 36%.

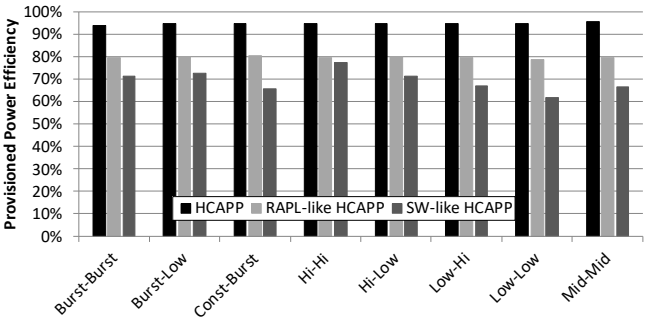

Figure 9: Provisioned power efficiency of multiple HCAPP versions and fixed voltage system under slow power limit. HCAPP averages a PPE of 93.9% while RAPL-like HCAPP averages 79.7%.

power limit does not change the behavior of a static design. Since the fixed voltage configuration was below the power limit at the faster power limit, it is also under this power limit and used for comparisons on speedup and PPE. The HCAPP is the only dynamic design that stays under the power limit. For the sake of analysis, the next experiments ignore the maximum power failures of RAPL-like (100 microseconds) and Software-like (10 milliseconds) control. These approaches are evaluated without any modifications to measure their performance and provisioned power efficiency. This is an overly optimistic view of how these approaches would be implemented to account for additional intelligence in the controllers.

Figure 8 shows the speedup of the HCAPP approaches across the test suite. The speedup is normalized to the performance of the fixed voltage (0.95 V) system. Overall, HCAPP provides the most speedup. However, certain tests show increased speedup with RAPL-like HCAPP. In particular, any test involving the ferret (Burst-*) benchmark benefits from the RAPL-like HCAPP behavior. This occurs because ferret's power behavior is characterized by long times of low activity with short bursts of high power activity mixed in. HCAPP tries to adjust the power for these short bursts resulting in excess power throttling and thus slower execution. Future implementations of HCAPP could work with software control designs to improve the speedup through the HCAPP software interface in cases like this. RAPL-like HCAPP does not react within the short bursts and instead reacts to larger time samples where the effect of these bursts is mitigated. For many of the tests, Software-like HCAPP does not show significant speedup because it cannot react quickly enough to take advantage of the changes in power throughout the program execution. Overall, HCAPP achieves an average speedup of 43% while RAPL-like HCAPP (RAPL-like) reaches 36% average speedup.

Figure 9 shows the provisioned power efficiency of the HCAPP approaches. The provisioned power efficiency is normalized to the power limit. 100% is the ideal value for this metric. HCAPP achieves an average provisioned power efficiency of 93.9% under the 100 Watt over 1 ms power limit. RAPL-like HCAPP and SW-like HCAPP achieve average provisioned power efficiencies of 79.7% and 69.2% respectively. Software-like HCAPP reduces the PPE relative to the fixed voltage system due to slower changes that lag behind the actual program phases. HCAPP and RAPL-like HCAPP are better than the fixed voltage system but the shorter control period of the HCAPP allows for more opportunities to change the power consumption of the system. HCAPP and RAPL-like HCAPP both show little variance in PPE across the test suite due to the length of their control periods relative to the length of each test. Both approaches are able to apply many control inputs throughout the run. This behavior is also seen under the package pin power limit with HCAPP in Section 5.1.

Overall, HCAPP provides the best performance and package resource utilization but RAPL-like HCAPP also provides significant advantages over static designs. A faster dynamic power controller provides better tracking of the power behavior at all time scales as shown by the maximum power results under the package pin power limit. The benefits of a fast reaction time also manifest in the speedup and PPE results at the off-package VR power limit. A fully hardware-based decentralized controller design uses the extra speed to enable potential savings in the system at the on-chip VR, package pin and off-chip VR levels.

HCAPP does not require any PID re-tuning efforts with the changing power limit definitions. This indicates that the power limit could be changed dynamically during a run without needing costly PID analysis. Improved PID constants for each power limit setting may help improve the results even further but are unneeded to achieve satisfactory power and performance behavior. HCAPP provides speedup over a static power design and provides speedup and PPE improvements over systems operating at slower timescales such as centralized controller-based control designs or software-based control designs. It is important to note that software-based

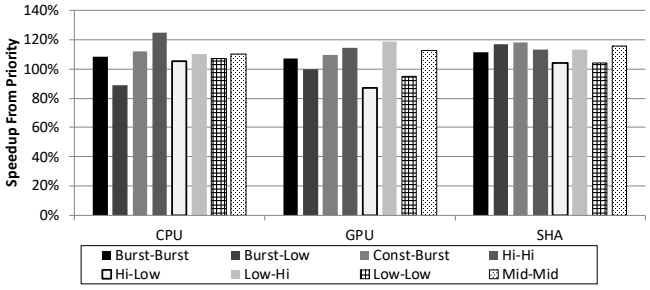

Figure 10: Additional speedup of HCAPP by prioritized component compared to unprioritized. Prioritization provides an average component speedup of 8.3%, 5.4% and 12% for the CPU, GPU and SHA accelerator respectively.

approaches can be used in conjunction with HCAPP to provide further PPE and speedup benefits.

5.3 Software Interface Impact Results

The domain controller allows for interactions with software power control logic. Software-based control can add global and program infomation to the hardware power control behavior to improve relevant speedup and desired power characteristics. As a proof of concept, we implemented a static priority software controller that prioritized a single component in the system. We ran the heterogeneous test suite three times, once with each component prioritized using the 100 Watts over 20 microsecond power limit.

Figure 10 shows the prioritized component speedup across the test suite compared to an unprioritized run. Each bar shows the speedup of a single, unqiue run with that component being prioritized. Across the entire suite, the CPU prioritization accelerates the CPU execution by 8.3%. For the GPU, the average speedup from software priority is 5.4%. For the SHA accelerator, the average speedup is 12%. The maximum power and PPE for these implementations are similar to the previous results for HCAPP. The maximum power and PPE do not significantly change because those behaviors are primarily handled by the global controller which did not change. The new HCAPP software priority-based speedups are promising as they confirm that the software interface functions properly. With better intelligence in the software control, further speedups would be possible. This proves that HCAPP supports a usable mechanism to interface with other software control schemes such as CoCap [21] or DynaCo [23].

Some notable outliers in this data are the three GPU runs that ran a Low GPU workload. The Low workload is myocyte which uses very low power. The CPU and SHA accelerators drive the power management in these scenarios. The CPU has its high power phases spread out since it spends less time waiting for memory. This leaves less power for the GPU work despite being prioritized. A more intelligent scheme could manage the domains independently throughout the workload to accelerate these scenarios as needed.

The other outlier is the Burst-Low CPU result. The Burst CPU workload is ferret which caused the unexpected behavior in the previous section with the RAPL-like HCAPP implementation. This is a bursty workload that can have odd behavior relative to the power provided throughout its operation. This shows that a more aware software algorithm could provide additional logic to better accelerate this kind of workload.

Overall, the validation of the software interface proves that HCAPP can be used in conjunction with software approaches. The presented software approach does not include the needed intelligence to benefit all of the test cases. Other researched software control designs could be integrated to provide additional speedup and power utilization.

6 CONCLUSIONS AND FUTURE WORK

This paper presents HCAPP to implement power capping behavior while maximizing provisioned power efficiency. To accomplish this, HCAPP uses a decentralized hardware-based approach. We design and validate a software interface into HCAPP to ensure that software-based control designs can work with HCAPP. Future explorations into improving HCAPP could focus on adding intelligence to the software controller used alongside HCAPP. In this paper, the software interface was validated with a a static priority-based software controller. Software-based control can allow proactive or predictive control beyond the reactive control that HCAPP implements. The software controllers provide a way to use centralized information to proactively adjust HCAPP parameters to prepare the system for certain power or test behaviors before it happens. This allows better management of the power and higher performance through better power allocation between components in advance. For example, the CPU begins to send work to the GPU and the software detects this. Then, the software controller reduces the HCAPP CPU domain voltage ratio (priority) and increases the GPU domain voltage ratio. By performing behaviors such as this throughout varying workloads, the combined HCAPP and software control scheme would outperform HCAPP alone.

REFERENCES

[1] [n.d.]. AMD Unveils 'Chiplet' Design Approach: 7nm Zen 2 Cores Meet 14 nm I/O Die. https://www.anandtech.com/show/13560/amd-unveils-chiplet-design-approach-7nm-zen-2-cores-meets-14-nm-io-die. Accessed: 2019-01-17.
[2] [n.d.]. Intel's Architecture Day 2018: The Future of Core, Intel GPUs, 10nm, and Hybrid x86. https://www.anandtech.com/show/13699/intel-architecture-day-2018-core-future-hybrid-x86/6. Accessed: 2019-01-17.
[3] [n.d.]. PID controller - Wikipedia. https://en.wikipedia.org/wiki/PID_controller. Accessed: 2019-01-25.
[4] K. Atasu, L. Pozzi, and P. Ienne. 2003. Automatic application-specific instruction-set extensions under microarchitectural constraints. *International Journal of Parallel Programming* 31, 6 (2003), 411–428.
[5] A. Bakhoda, G. Yuan, W. Fung, H. Wong, and T. Aamodt. 2009. Analyzing CUDA workloads using a detailed GPU simulator. In *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*. IEEE, 163–174.
[6] A. Bhattacharjee and M. Martonosi. 2009. Thread Criticality Predictors for Dynamic Performance, Power, and Resource Management in Chip Multiprocessors. *SIGARCH Comput. Archit. News* 37, 3 (June 2009), 290–301. https://doi.org/10.1145/1555815.1555792
[7] Bryan Black. [n.d.]. Die Stacking Is Happening! https://www.microarch.org/micro46/files/keynote1.pdf. Accessed: 2019-02-11.
[8] M. Bohr. 2007. A 30 year retrospective on Dennard's MOSFET scaling paper. *IEEE Solid-State Circuits Society Newsletter* 12, 1 (2007), 11–13.
[9] J. Charles, P. Jassi, N. Ananth, A. Sadat, and A. Fedorova. 2009. Evaluation of the Intel® CoreâĎć i7 Turbo Boost feature. In *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*. IEEE, 188–197.
[10] S. Che, M. Boyer, J. Meng, D. Tarjan, J. Sheaffer, S. Lee, and K. Skadron. 2009. Rodinia: A benchmark suite for heterogeneous computing. In *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*. Ieee, 44–54.
[11] M. Gupta, J. Oatley, R. Joseph, G. Wei, and D. Brooks. 2007. Understanding voltage variations in chip multiprocessors using a distributed power-delivery network. In *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE'07*. IEEE, 1–6.
[12] W. Heirman, T. Carlson, and L. Eeckhout. 2012. Sniper: Scalable and accurate parallel multi-core simulation. In *8th International Summer School on Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems (ACACES-2012)*. High-Performance and Embedded Architecture and Compilation Network of Excellence (HiPEAC), 91–94.
[13] C. Hsu, Y. Zhang, M. Laurenzano, D. Meisner, T. Wenisch, J. Mars, L. Tang, and R. Dreslinski. 2015. Adrenaline: Pinpointing and reining in tail queries with quick

voltage boosting. In *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*. IEEE, 271–282.

[14] H. Kasture, D. Bartolini, N. Beckmann, and D. Sanchez. 2015. Rubik: Fast analytical power management for latency-critical systems. In *Proceedings of the 48th International Symposium on Microarchitecture*. ACM, 598–610.

[15] Ben Keller. 2015. *Opportunities for Fine-Grained Adaptive Voltage Scaling to Improve System-Level Energy Efficiency*. Master's thesis. EECS Department, University of California, Berkeley.

[16] Y. Lee, B. Zimmer, A. Waterman, A. Puggelli, J. Kwak, R. Jevtic, B. Keller, S. Bailey, M. Blagojevic, P. Chiu, H. Cook, R. Avizienis, B. Richards, E. Alon, B. Nikolic, and K. Asanovic. [n.d.]. Raven: A 28nm RISC-V Vector Processor with Integrated Switched-Capacitor DC-DC Converters and Adaptive Clocking.

[17] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. Kim, T. Aamodt, and V. Reddi. 2013. GPUWattch: enabling energy optimizations in GPGPUs. In *ACM SIGARCH Computer Architecture News*, Vol. 41. ACM, 487–498.

[18] S. Li, J. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi. 2009. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 469–480.

[19] G. Loh. 2008. 3D-stacked memory architectures for multi-core processors. In *ACM SIGARCH computer architecture news*, Vol. 36. IEEE Computer Society, 453–464.

[20] S. Pal, D. Petrisko, A. Bajwa, P. Gupta, S. Iyer, and R. Kumar. 2018. A Case for Packageless Processors. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 466–479.

[21] J. Park, C. Hsieh, N. Dutt, and S. Lim. 2016. Co-Cap: energy-efficient cooperative CPU-GPU frequency capping for mobile games. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. ACM, 1717–1723.

[22] I. Paul, W. Huang, M. Arora, and S. Yalamanchili. 2015. Harmonia: Balancing Compute and Memory Power in High-performance GPUs. In *Proceedings of the 42Nd Annual International Symposium on Computer Architecture* (Portland, Oregon) *(ISCA '15)*. ACM, New York, NY, USA, 54–65. https://doi.org/10.1145/2749469.2750404

[23] I. Paul, V. Ravi, S. Manne, M. Arora, and S. Yalamanchili. 2013. Coordinated Energy Management in Heterogeneous Processors. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (Denver, Colorado) *(SC '13)*. ACM, New York, NY, USA, Article 59, 12 pages. https://doi.org/10.1145/2503210.2503227

[24] R. Pothukuchi, J. Greathouse, K. Rao, C. Erb, L. Piga, P. Voulgaris, and T. 2019. Tangram: Integrated Control of Heterogeneous Computers. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture* (Columbus, OH, USA) *(MICRO âĂŹ52)*. Association for Computing Machinery, New York, NY, USA, 384âĂŞ398. https://doi.org/10.1145/3352460.3358285

[25] Richtek Technology Corporation. 2015. Dual Channel PWM Controller with Integrated Driver for IMVP8 CPU Core Power Supply. http://www.richtek.com/assets/product_file/RT3606BC/DS3606BC-00.pdf Accessed: 2016-09-30.

[26] K. Straube, J. Lowe-Power, C. Nitta, M. Farrens, and V. Akella. 2018. Improving Provisioned Power Efficiency in HPC Systems with GPU-CAPP. In *2018 IEEE 25th International Conference on High Performance Computing (HiPC)*. 112–122. https://doi.org/10.1109/HiPC.2018.00021

[27] K. Straube, C. Nitta, R. Amirtharajah, M. Farrens, and V. Akella. 2017. Improving Execution Time of Parallel Programs on Large Scale Chip Multiprocessors with Constant Average Power Processing. In *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 649–652.

[28] V. Suresh, S. Satpathy, S. Mathew, M. Anders, H. Kaul, A. Agarwal, S. Hsu, and R. Krishnamurthy. 2018. A 230mV-950mV 2.8 Tbps/W Unified SHA256/SM3 Secure Hashing Hardware Accelerator in 14nm Tri-Gate CMOS. In *ESSCIRC 2018-IEEE 44th European Solid State Circuits Conference (ESSCIRC)*. IEEE, 98–101.

[29] K. Tsuzuku and T. Endo. 2015. Power capping of CPU-GPU heterogeneous systems using power and performance models. In *Smart Cities and Green ICT Systems (SMARTGREENS), 2015 International Conference on*. IEEE, 1–8.