CHARSTAR:
Clock Hierarchy Aware Resource Scaling in Tiled ARchitectures

Gokul Subramanian Ravi Mikko H. Lipasti
Department of Electrical and Computer Engineering
University of Wisconsin - Madison
gravi@wisc.edu,mikko@engr.wisc.edu

ABSTRACT

High-performance architectures are over-provisioned with resources to extract the maximum achievable performance out of applications. Two sources of avoidable power dissipation are the leakage power from underutilized resources, along with clock power from the clock hierarchy that feeds these resources. Most reconfiguration mechanisms either focus solely on power gating execution resources alone or in addition, simply turn off the immediate clock tree segment which supplied the clock to those resources. These proposals neither attempt to gate further up the clock hierarchy nor do they involve the clock hierarchy in influencing the reconfiguration decisions. The primary contribution of **CHARSTAR** is optimizing reconfiguration mechanisms to become clock hierarchy aware. Resource gating decisions are cognizant of the power consumed by each node in the clock hierarchy and additionally, entire branches of the clock tree are greedily shut down whenever possible.

The **CHARSTAR** design is further optimized for balanced spatio-temporal reconfiguration and also enables efficient joint control of resource and frequency scaling. The proposal is implemented by leveraging the inherent advantages of spatial architectures, utilizing a control mechanism driven by a lightweight offline trained neural predictor. **CHARSTAR**, when deployed on the CRIB tiled microarchitecture, improves processor energy efficiency by 20-25%, with efficiency improvements of roughly 2x in comparison to a naive power gating mechanism. Alternatively, it improves performance by 10-20% under varying power and energy constraints.

CCS CONCEPTS

• **Hardware** → **On-chip resource management**;

KEYWORDS

Dynamic reconfiguration, Spatial architectures, Clock gating, Power gating

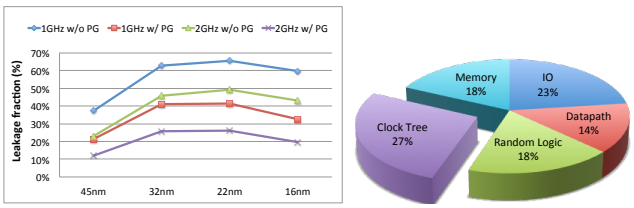
ACM Reference format:
Gokul Subramanian Ravi Mikko H. Lipasti Department of Electrical and Computer Engineering University of Wisconsin - Madison . 2017. CHARSTAR: Clock Hierarchy Aware Resource Scaling in Tiled ARchitectures. In *Proceedings of ISCA '17, Toronto, ON, Canada, June 24-28, 2017,* 14 pages.
https://doi.org/10.1145/3079856.3080212

ISCA '17, June 24-28, 2017, Toronto, ON, Canada

ACM ISBN 978-1-4503-4892-8/17/06…$15.00
https://doi.org/10.1145/3079856.3080212

1 INTRODUCTION

The demand for highest performance from current-day processing architectures has resulted in several challenges to achieving optimum energy efficiency. In order to exploit the maximum available parallelism in applications, these architectures are often significantly over-provisioned with execution resources. But more often than not, a significant portion of such resources remain underutilized. An ideal energy efficient architecture should dynamically avoid incurring any power or energy overhead from such unused resources.



(a) (b)

Figure 1: (a) Leakage fraction over tech. nodes, (b) Mali GPU power distribution

The two main sources of *avoidable* power consumption in a processing core are excess leakage and clock tree power. Figure 1 illustrates the fractions of leakage power and clock power in modern architectures. In Fig.1.a leakage power is measured over multiple technology nodes on McPAT [43, 69] for an IBM POWER7 model [69]. The leakage power forms a formidable fraction of chip power, especially in designs operating at lower frequencies. It is evident that in designs without power gating, the leakage fraction can go even up to 70%. While leakage power is significant, so is the power dissipated by the clock distribution network - even more so if the leakage power fraction declines. In Figure 1.b, clock power is observed to consume more than a quarter of the total power on ARM Mali GPUs [6]. Prior work has shown that the power consumed by the clock tree (CT) can be as high as 50% of the total circuit power, or, at the very least, consumes a major portion of the dynamic power [11, 44, 45, 50]. Such inefficiencies would increase with architectures designed even more aggressively for greater peak performance, and intelligent clock/resource aware mechanisms are necessary to optimize for better energy efficiency.

Prior work on architecture reconfiguration perform intelligent power gating to shut down *inefficient or underutilized* resources during particular periods of an application's execution. When resources are power gated, the leakage power that they would have dissipated if awake, is saved. These implementations can naively be extended to shut down immediate portions of the clock distribution system (clock gating) which solely feed these particular resources, reducing clocking power as well. *But these proposals neither study gating further up the clock hierarchy nor do they involve the clock hierarchy in influencing the reconfiguration decisions, both of which could have a large impact on energy efficiency.*

The primary contribution of this work is optimizing reconfiguration mechanisms to be CT-aware. Shutting down of a particular resource in a CT-unaware mechanism might only expect a linear increase in power savings, but if shutting down that resource allows the clock gating of a large portion of the clock hierarchy, then power savings are more significant. We show that the reconfiguration decisions made by a CT-aware vs. a CT-unaware mechanism are reasonably different, with a large impact on energy efficiency. Our proposed reconfiguration mechanism is aware of the clock hierarchy and the power consumed by each node in the tree. It selects resources to power down in a greedy manner and further, appropriately shuts down entire branches of the clock tree whenever possible.

Apart from proposing clock-aware reconfiguration and analyzing the same for multiple state of the art clock distribution systems, this paper makes some secondary contributions as well. First, we argue that for best benefits from reconfiguration, the mechanism should operate at the appropriate *spatio-temporal* granularity to efficiently capture application characteristics. This balances the ability to adapt to an application's needs quickly (temporal granularity) and accurately, in terms of resources, (spatial granularity) and at the same time, keep overheads to a bare minimum.

Second, we jointly optimize both DVFS and clock-aware power gating (PG-DVFS) to achieve the ideal configuration for each phase of an application, in terms of both ILP as well as frequency. Jointly optimizing reconfiguration and clock rate has been studied for multiprocessors—optimizing Thread Level Parallelism (TLP) vs. clock rate—but there have been no concrete proposals for *intra-core* dynamic control to balance Instruction Level Parallelism (ILP) against core frequency.

Finally, our analysis shows that the combined savings from clock hierarchy aware integrated PG-DVFS is significant, but requires sophisticated control to predict the most efficient resource configuration for each application phase. We show that standard linear control mechanisms perform poorly in comparison to an oracular approach. We advocate the use of a lightweight machine learning-based control mechanisms by showing near ideal accuracies of a multi-layer perceptron in this domain of reconfiguration.

While the benefits of clock-aware reconfiguration extend to all processing frameworks, this work limits complexity of reconfiguration control by focusing on spatial architectures. The inherent advantage of spatial (or tiled) architectures, which cluster resources into regular groups, is to simplify reconfiguration and thus make them an appealing fabric for these mechanisms. Moreover, spatial architectures, with their regular shape and layout, present a well structured hierarchical clock tree. Spatial architectures encompass (but are not limited to) GPUs, tiled microarchitectures [27, 34, 56], general purpose accelerators [24] as well as special functional accelerators [9, 18]. The focus of this work is microprocessors, but key results are broadly applicable to all of the above.

We evaluate *CHARSTAR* on a high performance tiled microarchitecture, in which multiple resource tiles exploit the maximum available ILP from applications. Our implementation power-gates one or more of these tiles and their respective clock hierarchies when such ILP opportunities is limited, improving energy efficiency and transforming it into a *dynamically adaptable heterogeneous core* [29].

The paper is organized as follows. Section 2 discusses clock distribution systems in modern processors. Section 3 qualitatively motivates the need for clock-aware reconfiguration. Section 4 discusses secondary contributions - spatio-temporal adaptivity, PG-DVFS and dynamic control for reconfiguration. Section 5 discusses CHARSTAR implementation atop a spatial architecture. Section 6 and beyond discuss quantitative results and conclude the paper.

2 OVERVIEW OF CLOCK DISTRIBUTION

The clock distribution network consumes a significant portion of the total power in processors and SOCs, prior work claims 30-50% of total chip power and/or up to 70% of the dynamic power [11, 44, 45, 50]. A clock distribution system (CDS) is typically the largest net in the circuit netlist and operates at the highest speed of any signal within the entire synchronous system, hence consuming significant power [12, 26, 42, 61].

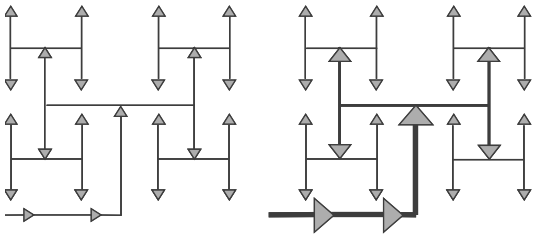
Clock power has increased with technology scaling [15, 42]. First, long global interconnect wires have become significantly more resistive as wires become thinner [30, 42]. Clock signals are affected by this increased wire resistance and require precise control of clock-signal arrival times, as they otherwise severely limit the maximum performance of the entire system. This has boosted the demand for repeaters in clock networks, increasing their power profile and complicating their synthesis. Second, with shrinking cycle times, the impact of process, voltage and temperature (PVT) variation also impacts clock skew and reliable clock networks have become more costly in terms of area and power [42].

2.1 Different Clock Distribution Systems

Clock distribution systems can broadly be divided into 3 styles - trees, grids (meshes) and hybrids. Different tree distributions and mesh/hybrid are shown in Fig.2 and Fig.3 respectively.

2.1.1 ***Trees:*** H-trees and binary trees are commonly used for clock distribution. Trees are attractive because of their low power and low metal usage [68]. If a H-Tree is completely balanced, it exhibits identical nominal delay and identical buffer and interconnect segments from the root of the distribution to all branches and thus would exhibit low skew. But if unbalanced, skew could be very high. In general, idealized buffer placements associated with a balanced H-tree may be difficult to achieve.

In the case of a tapered H-tree, the trunk widths increase geometrically toward the root of the distribution to maintain impedance matching at the T-junctions [22]. This strategy minimizes reflections of the high-speed clock signals at the branching points. Circuit analysis shows that the impedance of the conductor leaving each branch



(a) H - Tree (b) Tapered H - Tree (c) Binary - Tree

Figure 2: Clock Trees

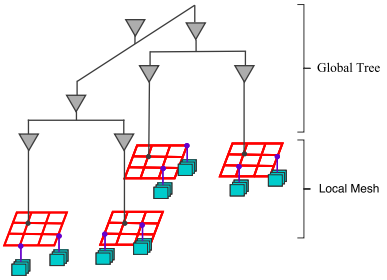


Figure 3: Hybrid Clock Distribution

Processor	Pentium 4	Itanium 2	SPARC IV	AMD K7	Alpha EV7	Xeon	Opteron	POWER6
CDS	Spine/Tree/Grid	Global/Local Tree	Grid	B-Tree	Tree/Grid	Tree/Grid	H-Tree/Grid	Sym. H-Tree/Grid

Table 1: Commercial Clock Distribution Systems

point must be twice the impedance of the conductor providing the signal to the branch point in a tapered H-tree structure [22].

A binary tree is uni-directional and delivers the clock in a balanced manner in either the vertical or horizontal dimension [68]. Similar to H-Trees, all branches exhibit identical buffer-interconnect segments, zero structural skew, and similar PVT tracking. In contrast to the H-tree, the buffers in a binary tree can be placed in close proximity along a centralized stripe. The closer physical proximity of the buffers in a binary tree can result in sensitivity to on-die variation but minimize floor-plan disruptions and are easier to design in a balanced manner.

2.1.2 ***Grids/Meshes:*** The lowest clock skew is achieved with Grid style networks but they require lot of metal resources and dissipate a lot of power [68]. A clock grid resembles a mesh with fully connected clock tracks in both dimensions and grid drivers located on all four sides (shown in red in Fig.3). Local clocks are supplied by directly connecting to the grid. The grid effectively shorts the output of all drivers and helps minimize delay mismatches. The shorted grid node helps balance the load non-uniformities and results in a more gradual delay profile across the region.

2.1.3 ***Hybrids:*** Hybrid clock distribution systems combine one or more of the above techniques as shown in Fig.3 Some hybrid systems have different parts of the chip clocked using different distribution techniques. Most current processors use hybrid techniques as they simplify skew reduction and have relatively low power dissipation.

Table 1 lists clock distribution systems employed by popular commercial processors from the previous decade [42, 55]. The analysis in this paper assumes a standard tree-grid hybrid, with the use of grids to clock local intra-tile resources and a global tree to drive the local grids. Our results analyze clock aware reconfiguration across H-Tree, Tapered H-Tree and Binary Tree hierarchies. Portions of the these global tree topologies can be gated when not in use, while the local grid can only be gated as a whole.

2.2 Estimating Clock Node Power

In this section, we present a model for clock distribution power [68]. Let us consider the switching power of a single unconditional clock at the final distribution stage M with capacitance C (load + portion of interconnect) and voltage V at frequency f.

$$P_{Clk_M} = C * V^2 * f \tag{1}$$

Considering there are N clock nodes in the stage, the total dynamic power in stage M is -

$$P_M = N * C * V^2 * f \tag{2}$$

Assuming a fan-out of k at each stage, the power in stage $M-1$ would be -

$$P_{M-1} = (N/k) * C * V^2 * f \tag{3}$$

Summation over all M stages gives the total clock distribution power -

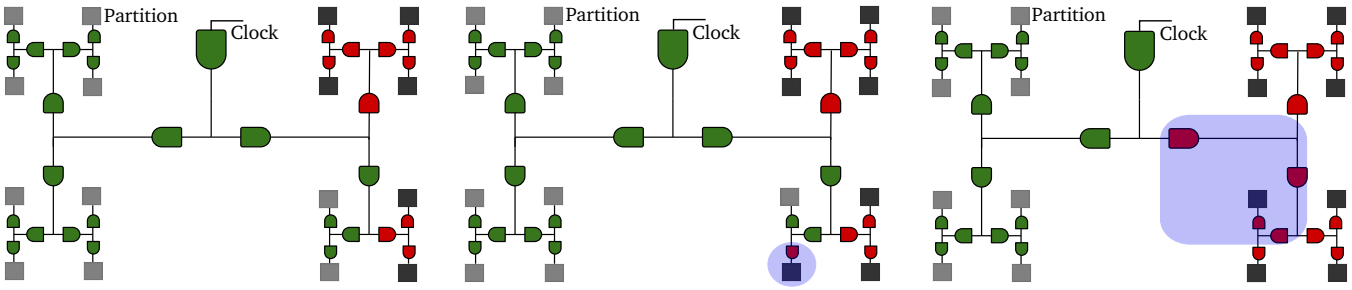
$$\sum P_i = [\frac{1-(1/k)^M}{1-(1/k)}] * N * C * V^2 * f \tag{4}$$

If sub-trees within this distribution network are power gated, that portion of power can be reduced from the above accordingly.

3 CLOCK TREE AWARE POWER GATING

When portions of resources (ALUs, compute nodes, or shader cores) are not in use, power gating saves leakage power, but gating the portion of the clock hierarchy supplying these resources (clock gating) also has huge energy saving potential. Many reconfiguration mechanisms either focus solely on power gating execution resources alone or in addition, simply turn off the clock tree segment (local mesh and global tree's leaf node) which supplied the clock to those resources.

Clock tree aware gating techniques, will be capable of further shutting down upper levels of the global clock tree hierarchy if the corresponding sub-tree is not in use. In addition, *Clock tree aware reconfiguration mechanisms* can make intelligent resource reconfiguration decisions to maximize gating of resources - providing best energy efficiency while guaranteeing required performance. CHARSTAR's reconfiguration mechanism is aware of the clock tree hierarchy and the power consumed by each node in the tree, and selects resources to power down in a greedy manner that shuts off entire branches of the clock tree whenever possible. Our proposal is aware of the portion of the clock tree that can be turned off with every resource configuration. It accounts for the power consumption of both resources *and* the clock tree in the reconfiguration algorithm. We present a simple example highlighting how significantly clock-tree awareness can impact reconfiguration.



(a) 10 Partitions (b) 9 Partitions (c) 8 Partitions

Figure 4: Example motivating Clock-Tree aware reconfiguration

Fig. 4 illustrates 3 scenarios with different number of tiles or partitions (10/9/8) enabled. Light and dark gray squares indicate awake and gated partitions respectively. Green and red gates indicate enabled and disabled channels in the clock tree respectively. If the performance gap between the 3 configurations is within reasonable limits, the reconfiguration mechanism would have to choose between the 3 configurations to maximize energy efficiency.

A naive reconfiguration mechanism would assume that transitioning from 10 -> 9 -> 8 partitions provides linear savings in power (proportional to the partitions being power gated). Thus even a marginal performance improvement with 9 partitions as compared to 8, might result in the 9-partition configuration being chosen as the ideal setting providing best energy efficiency. In contrast, the figure shows that moving from 10 -> 9 partitions, saves just one extra node on the clock tree (apart from the extra partition) but moving from 9 -> 8 partitions enables the shut down of the entire half of the clock tree leading to a larger (15x) clock tree savings. The additional partitions and portions of the clock hierarchy shut down are shown by the blue shaded regions.

This can also means that even if the drop in performance is *not insignificant* from shutting down each partition, choosing the 8-partition configuration can be largely more energy efficient in comparison to the 9 or 10 partition configurations, due to significant power reduction from a large portion of the clock tree being turned off. This illustrates that full knowledge of the clock hierarchy can produce very different results and larger energy savings than prior, naive mechanisms. Higher power savings from the clock hierarchy could instead even be used to run at higher frequencies, possibly providing higher performance than the 9/10 partition scenarios. This is discussed further in Section 4.1.

It should be noted that clock-tree aware reconfiguration is not restricted to tiled architectures alone and can be performed on other compute fabrics as well. The potential benefits as well as complexity involved would depend on the reconfiguration style and how the clock tree feeds the different reconfigurable nodes. Architectures with flat clock trees or non-uniform clock distribution to various reconfigurable resources may be structurally limited in how large a portion of the clock hierarchy can actually be power gated. For instance, an out of order processor power gating some portion of its Register File, ALUs and/or LSQ, will potentially need a complex clock-tree gating mechanism and might still provide only limited gains. On the other hand, a regularly arranged spatial architecture like a GPU might shut down some number of shader cores based on the amount of parallelism available, with comparatively lesser complexity. Finally, clock hierarchies could be designed from the ground up to better suit reconfiguration. We leave exploration of this option to future work.

4 DYNAMIC RECONFIGURATION CONTROL

Having motivated the importance of a reconfiguration mechanism aware of the clock distribution system, we next discuss the implementation of the control mechanism itself. We raise key questions needed to be answered to design an efficient control mechanism for a generic reconfiguration fabric and then discuss them in the following sub-sections.

(1) What global set of resources are made reconfigurable? (Sec 4.1) (2) How often do we want to reconfigure resources? (Sec 4.2) (3) How many different levels of reconfiguration for each resource? (Sec 4.2) (4) How to group resources to control reconfiguration efficiently? (Sec 4.3) (5) How to assess requirements dynamically and perform reconfiguration accurately? (Sec 4.4)

Note that while the following discussion is largely focused on clock-aware reconfiguration of tiled microprocessors, the overall approach is applicable to power-gating on a variety of substrates.

4.1 Integrating resource/frequency scaling

CHARSTAR identifies program phases with limited ILP, and power gates resources that are not necessary for reaching that ILP level. This leads to more energy-efficient execution under a fixed performance constraint. Alternatively, given a fixed power budget, the savings from power gating can be applied to increase voltage and frequency to deliver better performance for the baseline power. Our mechanism integrates the scaling of both resource and frequency *within the core*. As we show, joint optimization for both ILP and clock rate often leads the controller to choose configurations that provide greater benefit than choosing either in isolation. An integrated PG-DVFS mechanism has significant potential within a single core because energy/performance gains from frequency variations are closely dependent on which resources are actually in use.

The following example (Fig. 5) motivates DVFS-PG integration. The example builds on a baseline processor with four tiles running at nominal (100%) frequency. If this processor encounters an application phase wherein ILP is limited, different reconfiguration mechanisms will affect the processor in different ways. These scenarios are depicted in Fig.5, (a) to (d). Darker and lighter gray shades of tiles refer to power gated and awake tiles respectively. Also, the percentage values within awake cores refers to their running frequency in proportion to the baseline.

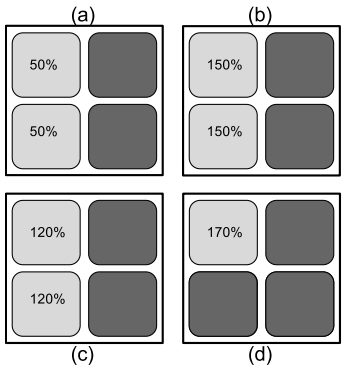


Figure 5: PG-DVFS Example

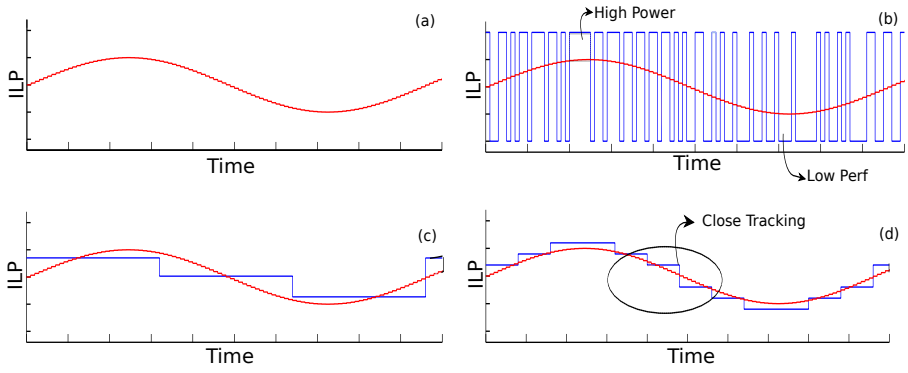


Figure 6: Fine grained adaptability across two dimensions.

• If DVFS and resource sizing are completely decoupled, both mechanisms could kick in independently and shut down multiple tiles and as well as reduce frequency, unaware of the other. This results in scenario A wherein two tiles are shut down *and* the frequency drops to only 0.5x resulting in performance dropping below what can be ideally achieved.

• If the controller performs resource sizing first, followed by naive throttling of frequency (in accordance to the baseline power budget), as shown in scenario B, the increased frequency might be largely beyond the requirement for maximum performance. For instance, shutting down 2 tiles may theoretically allow a 50% increase in frequency at constant power budget, but the 1.5x frequency might be so high that the memory becomes the bottleneck. This results in higher than ideal power dissipation.

• An integrated mechanism, on the other hand, acknowledges both the available ILP and the frequency requirements in conjunction, resulting in scenario C which has enough tiles (two in this example) for maximum ILP and the minimum sufficient frequency (1.2x) to achieve maximum throughput providing better efficiency.

• An integrated mechanism can further increase efficiency if situations exist (scenario D) wherein shutting down more tiles than the ILP bound (only one tile awake though max ILP is achieved with two tiles) and increasing the frequency further (1.7x, in comparison to scenario D with 1.2x), might actually be more advantageous. This is akin to the example illustrated in Fig.4 wherein it is possible that enabling 8 partitions allows large potential power savings in the clock tree which in turn allows increasing the frequency sufficiently to provide speedup above the 9/10 enabled partition scenarios (as described earlier in Section 3).

Prior work in trading off power gating and clock rate (TLP vs. DVFS) has focused on boosting the frequency of one or more cores while power-gating others (e.g. Intel Turbo Boost [7]). In contrast, CHARSTAR targets joint optimization of power gating and DVFS **within a single core**, which has not been explored in prior work. While the parallelism vs. frequency argument is the same in both cases, the architectural trade-offs, when analyzed, are very different. Intra-core PG-DVFS involves higher orders of dependencies between the usage of intra-CPU resources and the core frequency (in comparison to TLP vs. clock rate) and requires a sophisticated control mechanism, as proposed here.

4.2 Design for Spatio-Temporal balance

In order to enable effective reconfiguration of resources, granularities of adaptivity across the different adaptive dimensions need to be chosen appropriately. As clearly elucidated by Lee et al. [41], the adaptive computing paradigm provides flexibility of microarchitecture in two dimensions - temporal and spatial. The *temporal dimension* corresponds to the rate at which resources can be efficiently reconfigured. This could range from domain-level to application-level to phase-level adaptivity. The *spatial dimension* represents the microarchitectural scope of reconfigurations - the number of unique resources which can be resized and the number of configurations they are each capable of attaining. This might range from the level of a cluster or a single core to finer levels like pipeline stages or resources such as register files and execution units.

4.2.1 ***Temporal Granularity:*** To capture fine grained phases of applications, in the order of hundreds or thousands of instructions, recent proposals such as Composite Cores [46, 49] and the MorphCore [37] propose a temporally fine-grained design by building two contrasting engines within a core. Similar works on resizing of resources, even at the granularity of tens of cycles [40], enable doubling or halving of buffer sizes (eg. ROB) to adapt quickly to the application's needs. These cores are able to adapt very quickly largely due to the simplicity of the control mechanism and very limited resource configurations to choose from (such as the big and little engines in the Composite Core). Thus, such architectures that target very fine grained temporal adaptivity are incapable of finer variations in the spatial dimension.

Moreover, whilst these architectures are *theoretically capable* of adapting to the smallest of application changes over time, the ability is somewhat underutilized. At very small switching intervals on the order of hundreds of cycles, the minuscule variations within most applications are not significant enough to warrant shifts between coarsely granular architectural configurations (such as Big and Little engines). Such configurations are far apart on the spectrum of performance and power, thereby frequently rendering such schemes ineffective.

4.2.2 ***Spatial Granularity:*** At the other end of the adaptability spectrum, prior works have aimed at *protean* architectures, capable of simultaneously resizing multiple processor resources. These resources could range from the front end, to functional units and/or to the back end [16, 52], to enable the processor to adapt precisely

(resource-wise) to the needs of the application. By varying multiple microarchitectural parameters, each across a range of values, these processors often encounter an adaptable design space with billions of nodes (configurations) [16]. The large design space means that searching through these nodes to find the ideal configuration can happen only at a coarse grained temporal granularity of millions of cycles. Such schemes lose out on finer temporal variations, again skewing the spatio-temporal adaptivity balance.

It is evident that in designing any reconfiguration control mechanism, it is necessary to be able to efficiently capture and adapt to application characteristics. The ability to achieve extremely fine granularities of adaptivity in both temporal and spatial dimensions is limited by complexity of control and power gating mechanisms. We propose a balanced design over the two dimensions of adaptivity with the following qualitative analysis. While it is intuitive that balanced fine granularities in both dimensions provides benefits from both dimensions of adaptivity, we also show that coarse granularity on one dimension in fact hinders the effectiveness of fine grained adaptability along the other - *the effects of the two dimensions are not independent*.

From Figure 6, we make the following observations:

• Figure 6.a uses a simple sinusoidal curve to illustrate the variation in ILP of an application over time. Ideally, achieving extremely fine grained adaptability in both dimensions would allow resizing of resources, receptive to the smallest application variations, but this is infeasible due to unreasonable overheads.

• Figure 6.b portrays one solution allowing very fine temporal granularities but with coarse grained resource levels (only 2 levels, akin to Big/Little). The core can change its configuration very frequently but is allowed a very limited number of configurations. Thus, there are considerable periods when the core configurations overshoot the ideal requirement, resulting in excess power consumption with no performance gain and other periods when the core configuration undershoots the ideal, causing performance loss.

• Figure 6.c portrays an architecture allowing multiple possible resource configurations, but large switching times. Many of the *possible* resource levels (12 in this example) are never reached due to averaging across coarse temporal granularities and again, the large area between the curves show inefficiencies in performance and energy.

• The optimum solution as present in 6.d is a balance between adaptability along both dimensions - the adaptability is *reasonably* fine grained in both dimensions but not as fine grained as the previous solutions. It can be seen that the region between curves is minimal and the core is able to efficiently adapt to the needs of the application.

4.3 Tiled Architectures

The combined savings from clock hierarchy aware integrated PG-DVFS is significant, but at the same time is complex to implement. The inherent advantages of tiled architectures, via clustering of resources into small groups can limit these complexities and thus make them an ideal fabric for reconfiguration. Tiled architectures provide easy boundaries for powering down clusters of resources with minimal complexity and routing overheads [38]. Resizing resources by dynamically power-gating and waking up these reasonably sized clusters allows fine spatial granularities of adaptability. At the same time, the consolidated structure of these tiles reduces the reconfigurable design space significantly and their overall well defined topology minimizes circuit overhead, allowing for reasonably fine temporal granularities. Moreover, tiled architectures with their regular shape and arrangement, present a well structured hierarchical clock tree. This allows straightforward control of disabling branches of the clock tree as well. In all, a tiled architecture provides inherent advantage for spatio-temporally balanced reconfiguration with minimal overheads and with enough simplicity and regularity to be managed by a lightweight prediction mechanism.

In tiled microprocessor architectures, processor resources (functional units, buffer entries, registers and, in some cases, even caches) are structured in the form of multiple small tiles or partitions. Our specific focus is on the CRIB architecture [27], though our findings are applicable to a broad class of tiled machines [56, 59, 60]. Other tiled processor architectures which have been proposed with a variety of objectives in mind include TRIPS [56], RAW [60], Wavescalar [59], WiDGET [65], Sharing Architecture [70] and Core-Fusion [34]

Tiled or spatial frameworks are not limited to microprocessor architectures and, in fact, are more common among other compute engines. GPUs are a common example. Accelerators are often designed in a spatial framework as well. For example, neural network accelerators such as NPUs [18], Eyeriss [9], the DiaNao family [8, 14] all consists of a sea of compute nodes called processing engines(PEs). General purpose accelerators such as DySER [24] are also organized as a spatial framework.

Through a comprehensive limit study (Section 5.2), we analyze the overhead and benefits of fine- vs. coarse-grained spatial and temporal reconfiguration for tiled architectures, and choose a moderately fine-grained operating point for both dimensions. Our results indicate that, *though CHARSTAR does not exploit the finest possible adaptive granularity in either dimension, it reaches a balance across both and minimizes overheads and design complexity, thereby improving ease of actual implementation and achieving benefits nearly matching the ideal opportunity.* While this result is applicable to all reconfiguration mechanisms, this is especially important in relatively complex mechanisms targeting multiple optimization opportunities such as our own clock-aware PG-DVFS.

4.4 Neural Prediction Mechanism

To adapt to application phases, CHARSTAR must predict the best tile/frequency configuration for each quantum, using as inputs the behavior of the previous quantum. Studies [20, 35] have shown that while a first order model of performance prediction based on microarchitectural events can provide rough estimates (of resource requirements), they are often inaccurate due to overlap between a large portion of these events. Moreover, standard controllers with simple modeling may be sufficient when focused solely on reconfiguration or DVFS, whereas in the case of a more complex clock-tree aware PG-DVFS mechanism, the inaccuracies are further exacerbated.

For instance, the Composite Cores architecture [46] feeds multiple statistics from each quantum into a linear regression model which predicts the execution mode (big engine vs. little engine) for the next quantum. But the prediction accuracy is limited (Section 6.3), as

linear models are unable to understand non-linear relations among architectural events. Our results show that linear regression models or those that track a unique microarchitectural resource (related works are further discussed in Section 8) are inadequate for clock aware PG-DVFS prediction.

Machine learning techniques are capable of adapting to non-linear relations in inputs. Prior work in effectively applying machine learning techniques are described in Section 8. While we believe in the potential for different ML techniques towards resource reconfiguration (eg. decision tree learning etc.), our chosen technique is a Multi-Layer Perceptron (MLP). This is due to a large resource of recent prior work in modular design implementation of neural networks allowing ease of analyzing the trade-off of area/power overheads vs. MLP accuracy. Our results show high predictor accuracy with low effective computation and communication latency along with reasonable area/power overheads (Sections 5.3.4 and 6.3).

Our predictor makes use of a MLP with one hidden layer, capable of fitting any finite input-output mapping problem, enabling prediction of configurations with high accuracy. At runtime, the MLP is dynamically fed with the following statistics for every quantum of instruction: ① Branch mispredictions, ② I-Cache misses, ③ D-Cache misses, ④ L2-Cache misses, ⑤ Average % of the tiles occupied, ⑥ Number of times the tiles were full, ⑦ IPC, ⑧ Clock hierarchy, and ⑨ Topology overheads, to predict the configuration for the next quantum. These statistics are selected based on prior work in performance regression analysis [20, 35, 46].

The MLP is trained using a typical cross-validation approach, by partitioning the data collected from a testing portion (1%) of each benchmark into two sets - a training set with 70% of the data, and a validation set with the remaining 30%. The training is performed on MATLAB with random seeds using the Levenberg-Marquardt optimization based back-propagation algorithm. While our training and prediction is focused on the SPEC CPU2006 benchmark suite, we expect similar prediction accuracy across a wider range of applications. In general, a one time offline training with small portions of the standard applications that an architecture expects to execute, is sufficient for high prediction accuracy; this conclusion is supported by our results. This is due to the iterative nature and presence of hot functions in most applications.

The different metrics used for MLP training are performance, energy efficiency and energy-delay. The *configuration* is the combined selection of the enabled (awake) tiles plus the core frequency. The selection is usually constrained by the power budget - fewer tiles kept awake would mean that frequency could be raised further if it proves beneficial. This power constraint is relaxed in some cases, discussed in Section 6.2. The clock hierarchy power gating is completely tied to the number of tiles awake and is not a separate reconfigurable resource. The clock tree awareness influences the energy efficiency while training - enabling different number of tiles requires considerably different portions of the clock hierarchy branches to be enabled (as discussed in Section 3). Moreover, the different clock hierarchies (H-Trees, TH-Trees, B-Trees) affect the energy metric differently. This would train the MLP differently and the reconfiguration decisions vary accordingly.

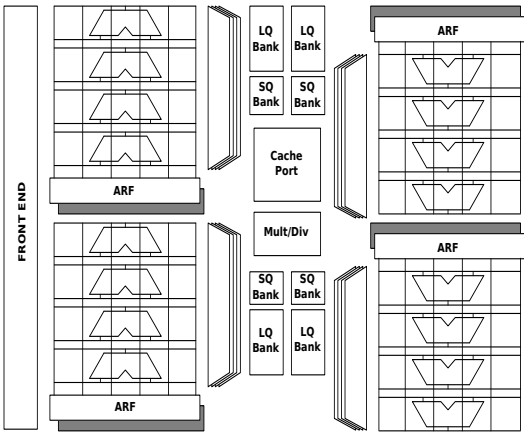


Figure 7: Tiles in the CRIB design [27]

5 CHARSTAR IN A TILED ARCHITECTURE

5.1 The CRIB Architecture

We implement CHARSTAR atop the CRIB tiled architecture [27]. CRIB achieves dramatic power savings by avoiding pipeline latches, register files, complex scheduling logic, and conventional register renaming. In the CRIB core, the RAT (register alias table), the RS (reservation station), and the ROB are consolidated into one structure namely, the consolidated rename/issue/bypass block, or CRIB. The CRIB consists of multiple partitions, as shown in Fig.7. Each partition consists of 4 CRIB entries. Each CRIB entry contains routing logic that connects logical register columns to an ALU as well as the ALU result back to the appropriate register column. Each instruction in the CRIB taps its source operands from the register columns, evaluates the result and then overwrites its destination register column accordingly. When all entries in a partition complete execution, data is written to the Architecture Register File (ARF) and the partition is committed, after which new instructions are inserted into it. The partitions are connected in a circular fashion with instances of ARF between the partitions. Only the ARF at the head of the partition has the committed state of the program and as every partition completes execution, the head or commit pointer is moved to the next ARF instance. Ready instructions across multiple CRIB partitions can evaluate concurrently, exposing OOO-like parallelism. The CRIB partitions form the minimum spatial granularity of reconfiguration in our reconfigurable design.

5.2 Granularities of adaptivity

In this section we perform quantitative analysis to choose optimum temporal and spatial granularities on the CRIB processor, our tiled architecture of choice. We analyze the gains from spatio-temporally balanced tile reconfiguration (without DVFS) across both adaptive dimensions in conjunction. Figure 8 illustrates the energy saving potential across the two dimensions of adaptivity in comparison to the ideal case of highly fine granularities in both dimensions - 500 instructions (temporal) and a single partition (spatial). This analysis is averaged across the SPEC CPU2006 benchmark suite. Note that darker portions of the heat map indicate higher energy efficiency. The range of spatial and temporal granularities are in accordance with prior work.

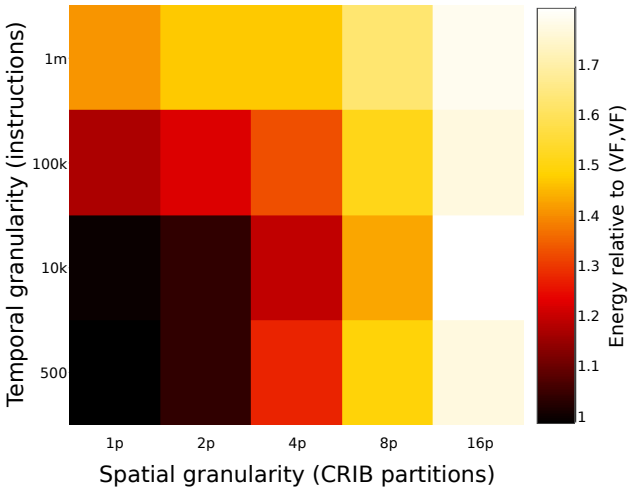


Figure 8: Bi-dimensional granularity impact on Energy

Temporal: Moving from coarse (1m instructions) to fine temporal granularities (500) of resizing improves energy efficiency but the improvements slowly begin to saturate. Despite low break even points for gating resources, circuit overheads as well as architectural steps such as draining/squashing the pipeline start becoming more significant at very fine granularities.

Spatial: The ability to allocate or cut back on resources at finer levels without losing out on performance improves energy efficiency in the case of finer spatial granularities. The benefits slowly saturate due to penalties caused by dependent instructions in different partitions and increased overhead from prediction mechanisms.

As noted earlier, the ideal configurations at the bottom left - (500,1p), (500, 2p), (10k,1p) cannot be achieved due to increased overheads of implementation. Among other options, it is evident that the balanced configuration of (10k, 2p) is more energy efficient than the extreme configurations across either dimension - this is in agreement with our qualitative analysis. Considering the above analyses, we choose a temporal granularity of *ten thousand* instructions and a spatial granularity of *2 partitions*. This provides a spatio-temporal balance across both dimensions and sacrifices benefits only marginally on either dimension.

Fig. 9 illustrates the sweet spot across the two dimensions of adaptivity for each benchmark in the SPEC CPU2006 benchmark suite. It is evident that while a majority of the benchmarks achieve best efficiency at the balance point of (2p,10k), there are a few differing in their sweet spot based on the specific benchmark characteristics. We believe such forms of classification will go a long way in designing ideal reconfigurable architectures suited to their native applications, and are a precursor to architectures which can *dynamically adapt their granularities of reconfiguration* itself.

In contrast to our observation for the need for balanced adaptivity, most of the existing proposals cited in Fig.10 propose reconfiguration techniques that aggressively tackle one dimension of adaptivity, but usually at the cost of the other, thereby achieving less than ideal efficiency. In figure, the axes labels denote V - Very, F - Fine, C - Coarse.

5.3 Quantifying Overheads

5.3.1 ***Clock tree awareness:*** Our implementation models the clock-tree as a binary/H-tree and tracks the per-node power consumption in a manner akin to well established clock-tree power modeling [11, 13, 58] and as discussed in earlier sections. The per-node power (as a fraction of the entire clock hierarchy) varies in proportion to the node's *level* in the clock hierarchy as well as its *fan-out*. The influence of the clock power adds a layer of heterogeneity to a homogeneous set of tiled resources and accordingly increases the training complexity of the control mechanism. The inference/deployment complexity remains unaffected since the design space remains the same (in comparison to clock unaware reconfiguration) - tile configuration and frequency level. For instance, if 8 tiles are required to be enabled, the chosen tiles are as shown in Fig3.c rather than any other distribution of 8 tiles (as that would enable more clock tree branches).

5.3.2 ***Power Gating and Awakening logic:*** The prediction mechanism passes resizing directions to the PG/awakening logic and the specific number of tiles are put to sleep or awoken. Based on the disabled tiles, the corresponding portions of the clock tree are also shut down or powered up. The latency overhead involved in power gating and waking up functional resources is found to be to the order of 10 or, at most, a 100 cycles [33, 62, 63]. Since the granularity of resizing in our implementation is every 10,000 instructions, the impact of draining the CRIB partitions followed by power gating or waking up of partitions are minimal (< 2%). Moreover, the delays from power gating a subset of partitions have no impact on the other awake partitions and they are ready to use without any overheads. Similarly, when partitions need to be awoken, already ready partitions are first filled up with instructions while others are being awoken in parallel. Therefore, the overheads from utilizing the newly awoken ones are reduced. The area overheads are negligible in comparison to the size of the tiles and the widespread clock-tree.

5.3.3 ***Fine-grained voltage-frequency control:*** The hardware resources required to control the voltage and frequency of the core for each quantum is minimal. Relative to standard DVFS techniques, this falls in the realm of fine grained DVFS, achievable using on-chip voltage regulators [19, 39]. Based on the voltage-frequency relation established in [28], we require a voltage range of 1V - 1.2V corresponding to a frequency range of 800 MHz - 1.3 GHz. To enable switching across this entire domain every quantum (an aggressive assumption) we would need fine grained voltage regulation with a switching capability of 0.2V between quanta. A quantum size of 10000 instructions and allowing an overhead of 1% for DVFS would require the ability to switch a maximum of 0.2V over 35 ns (@ 1.3 GHz), allowing a safe assumption of a design towards a switching speed of 6 mV/ns. Eyerman et al. [19] consider voltage switching speeds up to 200 mV/ns while Kim et al. [39] design regulators capable of switching at 50 mV/ns with a 15% energy and 2-3% area overhead.

The voltage switching and energy overheads increase super-linearly at smaller temporal granularities (depicted in Fig.11) due to increased complexity in regulator design. For example, quantum sizes

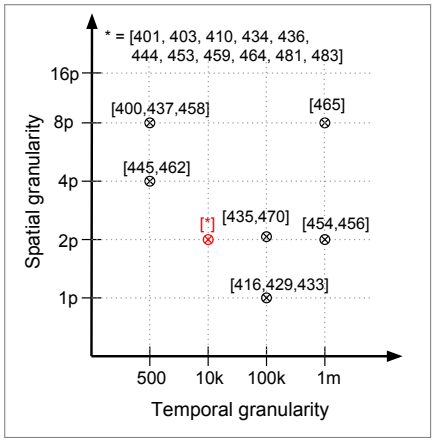


Figure 9: Sweet-spots

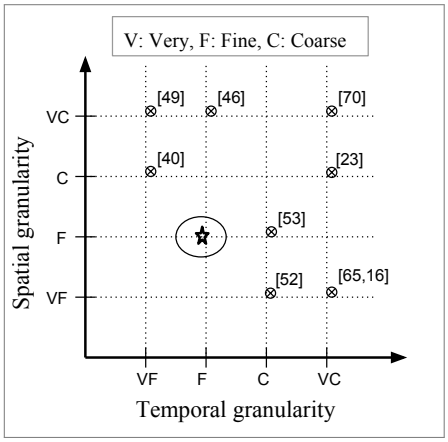


Figure 10: Prior works

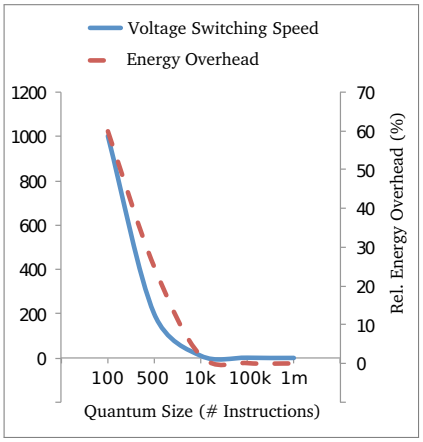


Figure 11: Volt. Switching Overheads

of 500 instructions results in energy overheads of 20-25%, preventing voltage/frequency control at very fine granularities from being a viable option. Overheads can be marginally controlled if the processor operates through voltage transitions by either quickly ramping down frequency before voltage ramps down or quickly switching to higher frequency just as the voltage is settling at the higher levels [10, 39]. Considering this analysis and the prior work [19, 39], we believe that our requirement is within reasonable limits, with insignificant overheads (2%) in energy and even lesser in area.

5.3.4 ***NPU design and training:*** Our mechanism searches within our limited design space representable in 10 bits (8 (1-hot) bits for tiles + 2 bits for 4 frequency levels) and predicts ideal configurations with high accuracy. Considering the design space, a rudimentary predictor managed by a condensed neural network with few neurons is sufficient. Initial design space exploration showed that 10 neurons in the hidden layer provided high levels of accuracy. The lightweight design with relatively few neurons and only a single hidden layer keeps complexity and overheads minimal, *aptly suited for architectures aiming to strike a spatio-temporal balance in adaptivity*. The use of a tiled architecture baseline significantly reduces the design space complexity over prior work since only two parameters (number of tiles and frequency level) need to be predicted.

Different implementations of neural processing units (NPU) have been explored in prior works [17, 18, 25] with varying trade-offs in performance, power, area and complexity. We adopt a simple digital NPU implementation proposed by Esmaeilzadeh et al. [18]. Similar to this, we consider communication latency (tightly-coupled NPU) and computation latency (one hidden layer and 10 neurons) to both be 10s of cycles. Further, latency could be possibly hidden by starting the MLP computation marginally before the completion of the previous quantum. The power overhead is expected to be marginal ($< 1\%$) as it has few neurons and is used only for 10s of cycles every 10k cycles and power gated otherwise. Based on estimates of neural functional unit area [14], processor area [27] and design synthesis, the area overhead is roughly 0.5%.

Resource	Configuration
CRIB	16 x 4-entry Int. CRIB; 16 x 2-entry FP CRIB
Compute	Int. ALU (1 cycle); FP add (4); FP mult. (4)
Core Mem.	32 LQ/SQ; 2-way 64KB L1I (2); 4-way 32KB L1D (2)
Uncore Mem.	L2: 2MB, 8-way (12); Off-chip mem (168)

Table 2: CRIB Specification

6 RESULTS

We extend the Gem5 Simulator [2] to support *CHARSTAR* atop the CRIB architecture. Performance and per-quantum statistics are obtained from Gem5 by running multiple Simpoint [51] slices (of size 100 million instructions) of the entire SPEC CPU2006 benchmark suite compiled for ARM ISA. The CRIB section of the processor is implemented in RTL and analyzed with the Synopsys Design Compiler. Power numbers for other processor resources are obtained from McPAT [43, 69]. All power and area results are obtained at a technology node of 22nm. Architecture specifications of CRIB with 16 partitions are presented in Table 2. Our results provide numerical benefits for the processor portion including tightly coupled L1 and L2 caches but excludes the rest of the memory subsystem.

6.1 Impact of Clock Tree awareness

In this section, we compare results from clock hierarchy aware power gating, normalized to a baseline of savings obtained from power gating only (i.e. no clock gating). We show results for 3 techniques: a) **Leaf**: which naively clock gates only the leaf node of the clock tree along with power gating the under-utilized resource, b) **Unaware**: which gates further up the clock hierarchy but the reconfiguration mechanism is unaware of (and uninfluenced by) the power saved from the clock nodes, and c) **Aware**: which uses our proposed clock-hierarchy aware gating mechanism. The benefits from these techniques are stacked upon one another in each of the figures 12, 13 and 14. These results are analyzed across H-Trees, tapered H-Trees (TH-Tree) and B-Trees for 3 different reconfiguration optimization goals.

Fig.12 shows increase in energy efficiency (over a PG-only baseline) obtained from reconfiguration optimized for lowest energy consumption when constrained by a requirement (or service level agreement i.e. SLA) that the performance falls no lower than 95% of the baseline. Key characteristics are highlighted below -

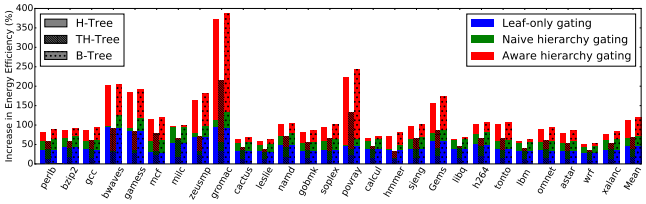


Figure 12: Clock-gating benefits w/ Perf. Constraint

(1) First, it is evident that there are higher energy savings possible in H-Trees and B-Trees in comparison to the tapered H-Tree. This is because nodes higher up the clock hierarchy consume more power in tapered trees, meaning that shutting down an extra *lower* node (i.e. a node that is, or is close to, a leaf) has a lower impact on the overall energy benefits. (2) Results are marginally better for B-Trees in comparison to H-Trees because there are deeper hierarchies in B-Trees in comparison to H-Trees (the number of nodes double at each level in B-Trees, while they quadruple in H-Trees), thereby providing more opportunities to gate unused common nodes in the former. (3) Clock gating leaf nodes alone provides 20-45% higher energy savings in comparison to the PG-only baseline, highlighting the importance of clock gating. (4) The benefits of gating higher up the clock tree, even without clock hierarchy aware reconfiguration provides 20-27% higher energy savings compared to only leaf node clock gating. This quantitatively reiterates the need for designing gating mechanisms to shut down clock nodes further up the clock hierarchies and not just clock gating leaf nodes alone. (5) Finally, clock hierarchy aware reconfiguration provides a 27-39% increased energy savings over clock hierarchy unaware reconfiguration in tune with our qualitative motivation earlier. (6) Overall, optimized CT-aware reconfiguration provides 65%-111% greater relative energy savings in comparison to PG-only reconfiguration.

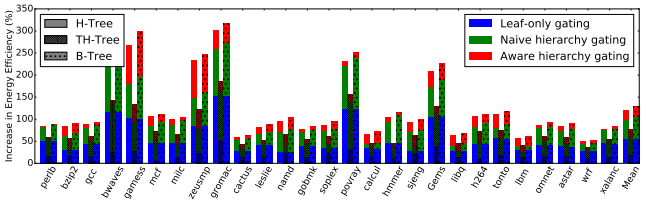


Figure 13: Clock-gating benefits w/o Perf. Constraint

We also show benefits for reconfiguration tuned to optimum energy (Fig. 13) and optimum energy-delay (Fig. 14) when unconstrained by any strict performance requirements. The observed benefits are similar to what was described above.

In the unconstrained optimum energy case (Fig. 13), the energy benefits of gating up the clock hierarchy (**Unaware** vs. **Leaf**) is higher than the constrained performance scenario (improvements are 36-45% here) while benefits of clock aware reconfiguration (**Aware** vs. **Unaware**) is lower (15-20% improvements here). The reason is that since there is no performance constraint in this scenario, the chosen configurations are mostly low resource configurations. This is because the most energy efficient configurations are usually ones with low resources with low performance - huge power savings at a loss in performance still provides higher energy savings. Therefore, there is significant scope to even naively gate higher up the clock hierarchy which increases its benefits in comparison to gating only leaf nodes. Since naive gating of upper hierarchy already provides considerable benefits, the extra benefits from smarter clock awareness correspondingly reduces. Overall benefits are higher: optimized CT-aware reconfiguration provides 72%-118% increased unconstrained energy savings in comparison to PG-only reconfiguration.

Energy-Delay optimization (Fig. 14) improves both energy efficiency and performance. Performance gains are obtained via boosting the frequency when nodes are power gated (as a part of PG-DVFS). More details of PG-DVFS benefits are discussed in the following section. ED improvements range from 21% to 36% in comparison to PG-only baseline.

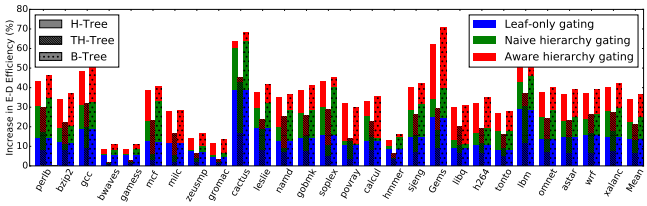


Figure 14: Benefits for optimum Energy-Delay

6.2 Impact of integrated PG-DVFS

Next, we examine the performance, energy and E-D efficiency improvements from integrated resource-frequency scaling in comparison to a baseline without reconfiguration (Fig.15). The results utilize clock-tree aware reconfiguration, averaging the clock power to be 33% of the total power consumption.

(1) Column 1 in figure shows the energy savings from integrated PG+DVFS, when optimizing for best energy efficiency under a minimum performance target of 95% (SLA) in comparison to the baseline. (2) Columns 2 and 3 show performance gains and energy-delay reduction with a control mechanism targeting maximum performance under the condition that each quantum operates within the baseline power budget. (3) Columns 4 and 5 show performance gains and energy-delay reduction when the control mechanism optimizes for lowest energy-delay over every quantum with no strict constraint on the power budget (UnC = unconstrained).

In summary, the figure shows energy savings of 19% under a 95% SLA, performance improvements of 11/18% and ED reduction of 22/26% under power constrained and unconstrained environments. The utility of the integrated mechanism is evident from significant improvements in all above metrics. While not shown in the graph, the energy savings increases by 1.9x when moving from a naive PG+DVFS mechanism to the integrated mechanism proposed here (qualitatively discussed via scenarios A-D in Section 4.1).

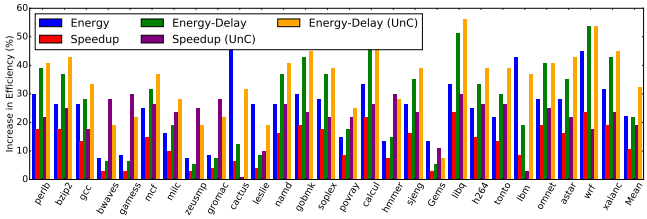


Figure 15: Benefits from integrated PG-DVFS.

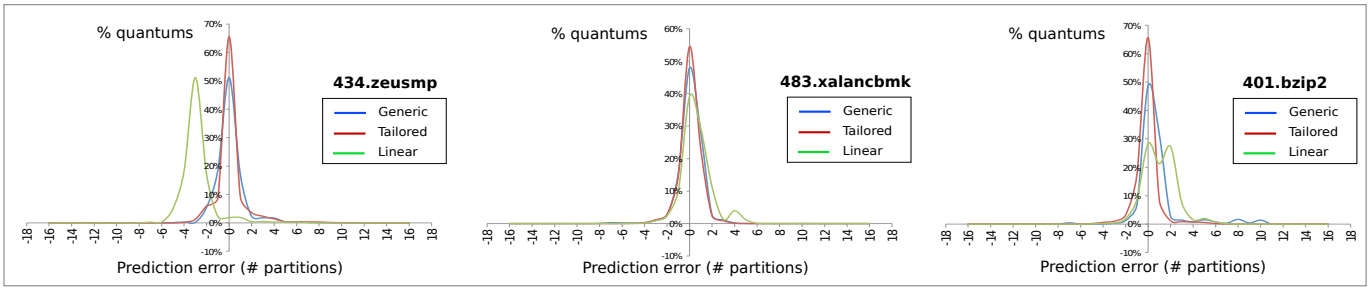


Figure 16: Accuracy of prediction: Tailored vs Generic vs Linear

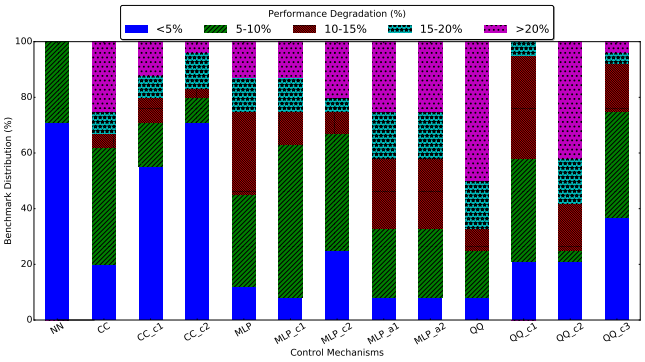


Figure 17: Perf. degradation across mechanisms

6.3 Impact of control mechanism

To motivate the accuracy of a generic one-time offline trained effort we make the following comparison. We compared the accuracy in prediction of a *generic* MLP (trained one-time offline) with a *tailored* MLP trained on a per benchmark basis and a linear regression model, across the SPEC CPU2006 benchmark suite. The *tailored* MLP provides a limiting model as to the best accuracy achievable with an MLP based predictor with reasonable overheads. Figure 16 shows the comparison for 3 benchmarks in terms of their prediction errors. It is evident that the accuracy of *generic* closely follows that of the *tailored* while *linear* is inaccurate. On average (in 16-partition CRIB), *generic* shows error of 0.45 partitions-per-prediction, *tailored* sees 0.28, and *linear* sees 1.8. When the training set for *generic* is reduced to samples from a random half of the SPEC benchmarks, error increases to 0.49. Further reduction of the training set to samples from a random quarter of the SPEC benchmarks increases the error to 0.67 partitions-per-prediction. These are still significantly more accurate than the linear predictor. The high accuracy of the generic MLP predictor suggests that *a generalizable non-linear function, trained offline, captures the behavior of many workloads*.

Figure 17 compares the deployment accuracy of multiple prediction mechanisms. It illustrates the performance degradation suffered due to prediction inaccuracies across the SPEC benchmark suite, for each prediction mechanism. The degradation is in comparison to the performance obtained by an ideal oracular predictor. Our proposed mechanism using a neural network is shown as *NN*. Existing prediction mechanisms shown are - CC [46], MLP [40](memory level parallelism) and QQ [53]. To broaden our comparison, we also implement more aggressive (eg. *MLP_a1*) as well as more conservative (eg. *QQ_c1*) versions of these.

Performance degradation from prediction is important to consider so as to avoid violations of SLAs and issues with tail latency [32]. For instance, if we consider an SLA criterion that requires at least 75% of benchmarks in the suite to fall within 10% performance degradation, the only prediction mechanisms meeting these criteria are our own mechanism (*NN*), along with *CC_c2* and *QQ_c3*. It is important to note that even within this limit, only *NN* meets an almost ideal prediction accuracy of 70% of the benchmarks within 5% degradation and 100% of the benchmarks within 10% degradation. Analysis of ED^2 product [4, 36], also shows that the *NN* predictor closely matches an ideal predictor and provides roughly 20% better ED^2 compared to the competing mechanisms.

7 DISCUSSION

Some key discussion points are put forth below:

Firstly, the ideas of PG-DVFS and using MLP based control for reconfiguration are applicable to any general architecture, and not limited to tiled architectures alone. The ideas of clock tree awareness and spatio-temporal balance are suited to any architecture with well defined clock hierarchy and resource clusters - so suited to all tiled architectures.

Second, the focus of this work is on single-threaded applications. CHARSTAR could be equally useful in a multi-threaded or multi-programmed context, but MLP complexity would increase based on number of threads/programs allowed simultaneously. Each thread's activity (cache misses, branches etc) would ideally need to be uniquely monitored and resources should be uniquely controlled (in terms of DVFS and PG). This could increase the complexity of the MLP super-linearly. But the current size of the MLP is very small (10 neurons) so the increase in its size to control a reasonable number of threads may not cause a huge overhead. Our future work hopes to enable running multi-programmed and multi-threaded workloads on tiled architectures with the proposed optimizations.

Third, it is also important to note that there are also other parameters that can affect tile configuration, such as inter-tile communication latency, thermal hot spots, etc. In designs with distributed shared memory, data reuse across tiles is another influential factor. Clock hierarchy aware configuration selection bodes well with keeping inter-tile communication latency low and spatial data reuse, but may not be the best configuration thermally. Further, if the tiles themselves are heterogeneous, then some tiles could be prioritized over others even if it might detrimental to clock power, communication latency, hot spots and so on. Designing a control mechanism that is cognizant of all these aspects could achieve the best overall reconfiguration efficiency and is potential future work.

Finally, we discuss some topological constraints. In the CRIB architecture, when dependent instructions lie in different partitions,

there is a one cycle overhead per partition to transfer data from older to younger dependent instructions. When a number of the partitions are power gated, the overhead involved in transferring data across multiple power gated partitions becomes significant. Such overheads are applicable to all tiled architectures. In order to minimize this overhead, we make use of *shortcut links* between partitions to bypass the penalty of transferring data across power gated partitions. Exploration of different topologies shows increasing overheads with more shortcut links due to increasing size of routers, input buffers, crossbars and allocation logic etc. Optimizing for the right amount of bypassing for a control mechanism cognizant of the different configuration-influencing parameters (discussed above) is also worthy of future exploration.

8 RELATED WORK

Core folding and DVFS are implemented in CMPs, but usually in a decoupled manner. Works such as [47], in fact, suggest that per-core power gating and DVFS should be implemented in a decoupled fashion due to increased complexity and difference in characteristics between the mechanisms. On the other hand, others [64] have shown the smart combination of both mechanisms can only improve efficiency at the CMP level. The few works exploring these as coupled mechanisms [54, 64] are restricted to the CMP level and only perform heuristic based control. In contrast, CHARSTAR targets joint optimization of power gating and DVFS within a single core using an MLP.

Within a single core, Albonesi et al. [1] have studied complexity-adaptive hardware i.e. the dynamic control of clock-rate/latency vs. resource size for particular resources such as issue queue and caches. Sen et al. [57] explore opportunities in shutting down portions of cache resources and increasing the core clock frequency. Instead, CHARSTAR takes an integrated and more comprehensive approach. At the circuit level, significant prior work analyzes various features of clock trees, their gating and related design optimizations [11, 13, 48, 58, 66].

Prior reconfiguration works include prediction mechanisms that track unique resources, such as the occupation of the instruction queue [53], IPC variation [5], multiple L1 cache misses [31], L2 caches misses [40] or the contribution of the most recently enabled tile [21], but these are less effective in modeling performance. Prior work towards CMPs include improving uncore energy efficiency via DVFS [67], anticipating the system-level performance impact of resource allocation across multiple cores at runtime [3], adapting multiple cores with *lanes* to suit stringent power budgets by sampling on multiple configurations [52] and to uniformly scale multiple cores and their resources [23]. As far as a single core is concerned, Dubach et al. [16] propose a high-end multi-dimensional machine learning based scheme to perform a limits analysis in spatial granularity - scaling a large number of resources within a single core, each with multiple resource sizes.

The underlying constraint in all of these works is that they make use of very complex (and often online-based) ML models as they generally target fine grained architectural (spatial) variations. This requires searching through a complex design space making them unsuited to fine temporal granularities.

9 CONCLUSION

This paper explores novel techniques to improve energy efficiency and/or performance through dynamic reconfiguration by effectively reducing both leakage and clock power. We introduced *CHARSTAR*, integrated power gating and DVFS within the core while also considering the power consumed by each node in the clock tree hierarchy. Our control mechanism is aware of the power consumption of each node in the clock tree, thereby choosing an ideal resource configuration which minimizes the combination of clock tree power *and* leakage power. Varying benefits are analyzed for different types of clock trees. Integrated PG-DVFS scaling is shown to be more effective at saving energy and/or achieving higher performance in compared to naive decoupled mechanisms, achieving optimum configuration per application phase in terms of both ILP and frequency. We explored the inherent advantages of tiled architectures towards dynamic reconfigurability and optimized it for balanced spatio-temporal adaptivity. Finally, we make use of a lightweight MLP predictor, suited to fine temporal granularities, to accurately predict the configurations for each application phase.

CHARSTAR, when deployed on the spatial CRIB architecture, shows improved processor energy efficiency by 20-25% in comparison to an unoptimized baseline, with efficiency improvements of roughly 2x in comparison to naive power gating mechanism. Our optimization can alternatively improve performance by 10-20% under varying power/energy constraints.

ACKNOWLEDGEMENTS

The authors would like to thank anonymous reviewers for their insights and comments, Essan Swain for relevant CRIB RTL design and Michael Mishkin for CRIB topology generation, all of which have improved the contributions of this work. This work was supported in part by NSF grants CCF-1318298 and CCF-1615014. Mikko Lipasti has a financial interest in Thalchemy Corp.

REFERENCES

[1] David H. Albonesi. 1998. Dynamic IPC/Clock Rate Optimization. In *Proceedings of the 25th Annual International Symposium on Computer Architecture (ISCA '98)*. IEEE Computer Society, Washington, DC, USA, 282–292. https://doi.org/10.1145/279358.279397

[2] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. 2011. The Gem5 Simulator. *SIGARCH Comput. Archit. News* 39, 2 (Aug. 2011), 1–7. https://doi.org/10.1145/2024716.2024718

[3] Ramazan Bitirgen, Engin Ipek, and Jose F. Martinez. 2008. Coordinated Management of Multiple Interacting Resources in Chip Multiprocessors: A Machine Learning Approach. In *Proceedings of the 41st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 41)*. IEEE Computer Society, Washington, DC, USA, 318–329. https://doi.org/10.1109/MICRO.2008.4771801

[4] D.M. Brooks, P. Bose, S.E. Schuster, H. Jacobson, P.N. Kudva, A. Buyuktosunoglu, J.-D. Wellman, V. Zyuban, M. Gupta, and P.W. Cook. 2000. Power-aware microarchitecture: design and modeling challenges for next-generation microprocessors. *Micro, IEEE* 20, 6 (Nov 2000), 26–44. https://doi.org/10.1109/40.888701

[5] Alper Buyuktosunoglu, David Albonesi, Stanley Schuster, David Brooks, Pradip Bose, and Peter Cook. 2001. A Circuit Level Implementation of an Adaptive Issue Queue for Power-aware Microprocessors. In *Proceedings of the 11th Great Lakes Symposium on VLSI (GLSVLSI '01)*. ACM, New York, NY, USA, 73–78. https://doi.org/10.1145/368122.368807

[6] Inc. Cadence Design Systems. 2013. Best Practices for Implementing ARM Cortex(R)-A12 Processor and MaliTM-T6XX GPUs for Mid-Range Mobile SoCs. (2013). http://www.armtechforum.com.cn/2013/3_Cadence.pdf

[7] James Charles, Preet Jassi, Narayan S Ananth, Abbas Sadat, and Alexandra Fedorova. 2009. Evaluation of the Intel® Core(TM) i7 Turbo Boost feature. In

Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on. IEEE, 188–197.

[8] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. 2014. DianNao: A Small-footprint High-throughput Accelerator for Ubiquitous Machine-learning. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '14)*. ACM, New York, NY, USA, 269–284. https://doi.org/10.1145/2541940.2541967

[9] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. 2016. Eyeriss: A Spatial Architecture for Energy-efficient Dataflow for Convolutional Neural Networks. In *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA '16)*. IEEE Press, Piscataway, NJ, USA, 367–379. https://doi.org/10.1109/ISCA.2016.40

[10] L.T. Clark, E.J. Hoffman, J. Miller, M. Biyani, Yuyun Liao, Stephen Strazdus, M. Morrow, K.E. Velarde, and M.A. Yarch. 2001. An embedded 32-b microprocessor core for low-power and high-performance applications. *Solid-State Circuits, IEEE Journal of* 36, 11 (Nov 2001), 1599–1608. https://doi.org/10.1109/4.962279

[11] Monica Donno, Alessandro Ivaldi, Luca Benini, and Enrico Macii. 2003. Clock-tree Power Optimization Based on RTL Clock-gating. In *Proceedings of the 40th Annual Design Automation Conference (DAC '03)*. ACM, New York, NY, USA, 622–627. https://doi.org/10.1145/775832.775989

[12] Monica Donno, Enrico Macii, and Luca Mazzoni. 2004. Power-aware Clock Tree Planning. In *Proceedings of the 2004 International Symposium on Physical Design (ISPD '04)*. ACM, New York, NY, USA, 138–147. https://doi.org/10.1145/981066.981097

[13] Monica Donno, Enrico Macii, and Luca Mazzoni. 2004. Power-aware Clock Tree Planning. In *Proceedings of the 2004 International Symposium on Physical Design (ISPD '04)*. ACM, New York, NY, USA, 138–147. https://doi.org/10.1145/981066.981097

[14] Zidong Du, Robert Fasthuber, Tianshi Chen, Paolo Ienne, Ling Li, Tao Luo, Xiaobing Feng, Yunji Chen, and Olivier Temam. 2015. ShiDianNao: Shifting Vision Processing Closer to the Sensor. In *Proceedings of the 42Nd Annual International Symposium on Computer Architecture (ISCA '15)*. ACM, New York, NY, USA, 92–104. https://doi.org/10.1145/2749469.2750389

[15] D. Duarte, V. Narayanan, and M. J. Irwin. 2002. Impact of technology scaling in the clock system power. In *Proceedings IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI 2002*. 52–57. https://doi.org/10.1109/ISVLSI.2002.1016875

[16] Christophe Dubach, Timothy M. Jones, Edwin V. Bonilla, and Michael F. P. O'Boyle. 2010. A Predictive Model for Dynamic Microarchitectural Adaptivity Control. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '43)*. IEEE Computer Society, Washington, DC, USA, 485–496. https://doi.org/10.1109/MICRO.2010.14

[17] Hadi Esmaeilzadeh, Pooya Saeedi, Babak Nadjar Araabi, Caro Lucas, and Sied Mehdi Fakhraie. 2006. Neural network stream processing core (NnSP) for embedded systems. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*. IEEE, 4–pp.

[18] Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. 2012. Neural Acceleration for General-Purpose Approximate Programs. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-45)*. IEEE Computer Society, Washington, DC, USA, 449–460. https://doi.org/10.1109/MICRO.2012.48

[19] Stijn Eyerman and Lieven Eeckhout. 2011. Fine-grained DVFS Using On-chip Regulators. *ACM Trans. Archit. Code Optim.* 8, 1, Article 1 (Feb. 2011), 24 pages. https://doi.org/10.1145/1952998.1952999

[20] Stijn Eyerman, Lieven Eeckhout, Tejas Karkhanis, and James E. Smith. 2006. A Performance Counter Architecture for Computing Accurate CPI Components. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XII)*. ACM, New York, NY, USA, 175–184. https://doi.org/10.1145/1168857.1168880

[21] Daniele Folegnani and Antonio González. 2001. Energy-effective Issue Logic. In *Proceedings of the 28th Annual International Symposium on Computer Architecture (ISCA '01)*. ACM, New York, NY, USA, 230–239. https://doi.org/10.1145/379240.379266

[22] E. G. Friedman. 2001. Clock distribution networks in synchronous digital integrated circuits. *Proc. IEEE* 89, 5 (May 2001), 665–692. https://doi.org/10.1109/5.929649

[23] Hamid Reza Ghasemi and Nam Sung Kim. 2014. RCS: Runtime Resource and Core Scaling for Power-constrained Multi-core Processors. In *Proceedings of the 23rd International Conference on Parallel Architectures and Compilation (PACT '14)*. ACM, New York, NY, USA, 251–262. https://doi.org/10.1145/2628071.2628095

[24] V. Govindaraju, C. H. Ho, T. Nowatzki, J. Chhugani, N. Satish, K. Sankaralingam, and C. Kim. 2012. DySER: Unifying Functionality and Parallelism Specialization for Energy-Efficient Computing. *IEEE Micro* 32, 5 (Sept 2012), 38–51. https://doi.org/10.1109/MM.2012.51

[25] Beayna Grigorian, Nazanin Farahpour, and Glenn Reinman. 2015. BRAINIAC: Bringing reliable accuracy into neurally-implemented approximate computing. In *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*. IEEE, 615–626.

[26] Paul E Gronowski, William J Bowhill, Ronald P Preston, Michael K Gowan, and Randy L Allmon. 1998. High-performance microprocessor design. *IEEE Journal of Solid-State Circuits* 33, 5 (1998), 676–686.

[27] Erika Gunadi and Mikko H. Lipasti. 2011. CRIB: Consolidated Rename, Issue, and Bypass. In *Proceedings of the 38th Annual International Symposium on Computer Architecture (ISCA '11)*. ACM, New York, NY, USA, 23–32. https://doi.org/10.1145/2000064.2000068

[28] H. Hanson, S.W. Keckler, S. Ghiasi, K. Rajamani, F. Rawson, and J. Rubio. 2007. Thermal response to DVFS: analysis with an Intel Pentium M. In *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*. 219–224. https://doi.org/10.1145/1283780.1283827

[29] Mark D. Hill and Michael R. Marty. 2008. Amdahl's Law in the Multicore Era. *Computer* 41, 7 (July 2008), 33–38. https://doi.org/10.1109/MC.2008.209

[30] R. Ho, K. W. Mai, and M. A. Horowitz. 2001. The future of wires. *Proc. IEEE* 89, 4 (Apr 2001), 490–504. https://doi.org/10.1109/5.920580

[31] Houman Homayoun, Avesta Sasan, Jean-Luc Gaudiot, and Alex Veidenbaum. 2011. Reducing Power in All Major CAM and SRAM-Based Processor Units via Centralized, Dynamic Resource Size Management. *IEEE Trans. Very Large Scale Integr. Syst.* 19, 11 (Nov. 2011), 2081–2094. https://doi.org/10.1109/TVLSI.2010.2064185

[32] Chang-Hong Hsu, Yunqi Zhang, Michael A Laurenzano, David Meisner, Thomas Wenisch, Lingjia Tang, Jason Mars, Ron Dreslinski, Vinicius Petrucci, Michael A Laurenzano, and others. 2015. Adrenaline: Pinpointing and reining in tail queries with quick voltage boosting. *Proceedings of the 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), HPCA* 15 (2015).

[33] Zhigang Hu, Alper Buyuktosunoglu, Viji Srinivasan, Victor Zyuban, Hans Jacobson, and Pradip Bose. 2004. Microarchitectural Techniques for Power Gating of Execution Units. In *Proceedings of the 2004 International Symposium on Low Power Electronics and Design (ISLPED '04)*. ACM, New York, NY, USA, 32–37. https://doi.org/10.1145/1013235.1013249

[34] Engin Ipek, Meyrem Kirman, Nevin Kirman, and Jose F. Martinez. 2007. Core Fusion: Accommodating Software Diversity in Chip Multiprocessors. In *Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA '07)*. ACM, New York, NY, USA, 186–197. https://doi.org/10.1145/1250662.1250686

[35] Tejas S. Karkhanis and James E. Smith. 2004. A First-Order Superscalar Processor Model. In *Proceedings of the 31st Annual International Symposium on Computer Architecture (ISCA '04)*. IEEE Computer Society, Washington, DC, USA, 338–. http://dl.acm.org/citation.cfm?id=998680.1006729

[36] Stefanos Kaxiras and Margaret Martonosi. 2008. Computer Architecture techniques for power-efficiency. *Synthesis Lectures on Computer Architecture 3* (2008), 1–207.

[37] Khubaib, M. Aater Suleman, Milad Hashemi, Chris Wilkerson, and Yale N. Patt. 2012. MorphCore: An Energy-Efficient Microarchitecture for High Performance ILP and High Throughput TLP. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-45)*. IEEE Computer Society, Washington, DC, USA, 305–316. https://doi.org/10.1109/MICRO.2012.36

[38] Jason Sungtae Kim, Michael Bedford Taylor, Jason Miller, and David Wentzlaff. 2003. Energy Characterization of a Tiled Architecture Processor with On-chip Networks. In *Proceedings of the 2003 International Symposium on Low Power Electronics and Design (ISLPED '03)*. ACM, New York, NY, USA, 424–427. https://doi.org/10.1145/871506.871610

[39] Wonyoung Kim, M.S. Gupta, Gu-Yeon Wei, and D. Brooks. 2008. System level analysis of fast, per-core DVFS using on-chip switching regulators. In *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*. 123–134. https://doi.org/10.1109/HPCA.2008.4658633

[40] Yuya Kora, Kyohei Yamaguchi, and Hideki Ando. 2013. MLP-aware dynamic instruction window resizing for adaptively exploiting both ILP and MLP. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 37–48.

[41] Benjamin C. Lee and David Brooks. 2008. Efficiency Trends and Limits from Comprehensive Microarchitectural Adaptivity. In *Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XIII)*. ACM, New York, NY, USA, 36–47. https://doi.org/10.1145/1346281.1346288

[42] Dong Jin Lee. 2011. *High-performance and Low-power Clock Network Synthesis in the Presence of Variation*. Ph.D. Dissertation. Citeseer.

[43] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. 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 (MICRO 42)*. ACM, New York, NY, USA, 469–480. https://doi.org/10.1145/1669112.1669172

[44] Hong-Ting Lin, Yi-Lin Chuang, and Tsung-Yi Ho. 2011. Pulsed-latch-based Clock Tree Migration for Dynamic Power Reduction. In *Proceedings of the 17th IEEE/ACM International Symposium on Low-power Electronics and Design (ISLPED '11)*. IEEE Press, Piscataway, NJ, USA, 39–44. http://dl.acm.org/citation.cfm?id=2016802.2016813

[45] Jingwei Lu, Wing-Kai Chow, and Chiu-Wing Sham. 2012. Fast Power- and Slew-aware Gated Clock Tree Synthesis. *IEEE Trans. Very Large Scale Integr. Syst.* 20, 11 (Nov. 2012), 2094–2103. https://doi.org/10.1109/TVLSI.2011.2168834

[46] Andrew Lukefahr, Shruti Padmanabha, Reetuparna Das, Faissal M. Sleiman, Ronald Dreslinski, Thomas F. Wenisch, and Scott Mahlke. 2012. Composite Cores: Pushing Heterogeneity Into a Core. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-45)*. IEEE Computer Society, Washington, DC, USA, 317–328. https://doi.org/10.1109/MICRO.2012.37

[47] Kai Ma and Xiaorui Wang. 2012. PGCapping: Exploiting Power Gating for Power Capping and Core Lifetime Balancing in CMPs. In *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques (PACT '12)*. ACM, New York, NY, USA, 13–22. https://doi.org/10.1145/2370816.2370821

[48] Jaewon Oh and Massoud Pedram. 2001. Gated clock routing for low-power microprocessor design. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 20, 6 (2001), 715–722.

[49] Shruti Padmanabha, Andrew Lukefahr, Reetuparna Das, and Scott Mahlke. 2013. Trace Based Phase Prediction for Tightly-coupled Heterogeneous Cores. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-46)*. ACM, New York, NY, USA, 445–456. https://doi.org/10.1145/2540708.2540746

[50] Jatuchai Pangjun and Sachin S. Sapatnekar. 2002. Low-power Clock Distribution Using Multiple Voltages and Reduced Swings. *IEEE Trans. Very Large Scale Integr. Syst.* 10, 3 (June 2002), 309–318. https://doi.org/10.1109/TVLSI.2002.1043334

[51] Erez Perelman, Greg Hamerly, Michael Van Biesbrouck, Timothy Sherwood, and Brad Calder. 2003. Using SimPoint for Accurate and Efficient Simulation. In *Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '03)*. ACM, New York, NY, USA, 318–319. https://doi.org/10.1145/781027.781076

[52] Paula Petrica, Adam M. Izraelevitz, David H. Albonesi, and Christine A. Shoemaker. 2013. Flicker: A Dynamically Adaptive Architecture for Power Limited Multicore Systems. In *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA '13)*. ACM, New York, NY, USA, 13–23. https://doi.org/10.1145/2485922.2485924

[53] Dmitry Ponomarev, Gurhan Kucuk, and Kanad Ghose. 2006. Dynamic Resizing of Superscalar Datapath Components for Energy Efficiency. *IEEE Trans. Comput.* 55, 2 (2006), 199–213. https://doi.org/10.1109/TC.2006.23

[54] Ramya Raghavendra, Parthasarathy Ranganathan, Vanish Talwar, Zhikui Wang, and Xiaoyun Zhu. 2008. No "Power" Struggles: Coordinated Multi-level Power Management for the Data Center. In *Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XIII)*. ACM, New York, NY, USA, 48–59. https://doi.org/10.1145/1346281.1346289

[55] Nitya Ranganathan and Norman P Jouppi. 2007. Evaluating the potential of future on-chip clock distribution using optical interconnects. *Hewlett-Packard Development Company, Tech. Rep. HPL-2007-163, Oct* (2007).

[56] Karthikeyan Sankaralingam, Ramadass Nagarajan, Haiming Liu, Changkyu Kim, Jaehyuk Huh, Nitya Ranganathan, Doug Burger, Stephen W. Keckler, Robert G. McDonald, and Charles R. Moore. 2004. TRIPS: A Polymorphous Architecture for Exploiting ILP, TLP, and DLP. *ACM Trans. Archit. Code Optim.* 1, 1 (March 2004), 62–93. https://doi.org/10.1145/980152.980156

[57] Rathijit Sen and David A Wood. 2013. *Cache Power Budgeting for Performance*. Technical Report UW-CS-TR-1791. University of Wisconsin - Madison Computer Sciences Department.

[58] Weixiang Shen, Yici Cai, Xianlong Hong, and Jiang Hu. 2010. An effective gated clock tree design based on activity and register aware placement. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 18, 12 (2010), 1639–1648.

[59] Steven Swanson, Ken Michelson, Andrew Schwerin, and Mark Oskin. 2003. WaveScalar. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 36)*. IEEE Computer Society, Washington, DC, USA, 291–. http://dl.acm.org/citation.cfm?id=956417.956546

[60] Michael Bedford Taylor, Walter Lee, Jason Miller, David Wentzlaff, Ian Bratt, Ben Greenwald, Henry Hoffmann, Paul Johnson, Jason Kim, James Psota, Arvind Saraf, Nathan Shnidman, Volker Strumpen, Matt Frank, Saman Amarasinghe, and Anant Agarwal. 2004. Evaluation of the Raw Microprocessor: An Exposed-Wire-Delay Architecture for ILP and Streams. In *Proceedings of the 31st Annual International Symposium on Computer Architecture (ISCA '04)*. IEEE Computer Society, Washington, DC, USA, 2–. http://dl.acm.org/citation.cfm?id=998680.1006733

[61] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Baez. 1998. Reducing power in high-performance microprocessors. In *Design Automation Conference, 1998. Proceedings*. 732–737. https://doi.org/10.1145/277044.277227

[62] James W Tschanz, Siva G Narendra, Yibin Ye, Bradley A Bloechel, Shekhar Borkar, and Vivek De. 2003. Dynamic sleep transistor and body bias for active leakage power control of microprocessors. *Solid-State Circuits, IEEE Journal of* 38, 11 (2003), 1838–1845.

[63] Kimiyoshi Usami, Tatsunori Hashida, Satoshi Koyama, Tatsuya Yamamoto, Daisuke Ikebuchi, Hideharu Amano, Mitaro Namiki, Masaaki Kondo, and Hiroshi Nakamura. 2010. Adaptive power gating for function units in a microprocessor. In *Quality Electronic Design (ISQED), 2010 11th International Symposium on*. IEEE, 29–37.

[64] 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 *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-46)*. ACM, New York, NY, USA, 210–221. https://doi.org/10.1145/2540708.2540727

[65] Yasuko Watanabe, John D. Davis, and David A. Wood. 2010. WiDGET: Wisconsin Decoupled Grid Execution Tiles. In *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA '10)*. ACM, New York, NY, USA, 2–13. https://doi.org/10.1145/1815961.1815965

[66] Shmuel Wimer and Israel Koren. 2012. The Optimal fan-out of clock network for power minimization by adaptive gating. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 20, 10 (2012), 1772–1780.

[67] Jae-Yeon Won, Xi Chen, Paul Gratz, Jiang Hu, and Vassos Soteriou. 2014. Up by their bootstraps: Online learning in artificial neural networks for CMP uncore power management. In *High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on*. IEEE, 308–319.

[68] Thucydides Xanthopoulos. 2009. *Clocking in Modern VLSI Systems* (1st ed.). Springer Publishing Company, Incorporated.

[69] Sam Likun Xi, Hans Jacobson, Pradip Bose, Gu-Yeon Wei, and David Brooks. 2015. Quantifying sources of error in McPAT and potential impacts on architectural studies. In *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*. IEEE, 577–589.

[70] Yanqi Zhou and David Wentzlaff. 2014. The Sharing Architecture: Sub-core Configurability for IaaS Clouds. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '14)*. ACM, New York, NY, USA, 559–574. https://doi.org/10.1145/2541940.2541950