ENERGY EFFICIENT CORE DESIGNS FOR UPCOMING PROCESS TECHNOLOGIES

BY

BHARGAVA REDDY GOPIREDDY

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Doctoral Committee:

Professor Josep Torrellas, Chair
Professor Wen-Mei Hwu
Professor Nam Sung Kim
Professor Christopher Fletcher
Dr. Asit Mishra, Nvidia

ABSTRACT

Energy efficiency has been a first order constraint in the design of micro processors for the last decade. As Moore's law sunsets, new technologies are being actively explored to extend the march in increasing the computational power and efficiency. It is essential for computer architects to understand the opportunities and challenges in utilizing the upcoming process technology trends in order to design the most efficient processors. In this work, we consider three process technology trends and propose core designs that are best suited for each of the technologies. The process technologies are expected to be viable over a span of timelines.

We first consider the most popular method currently available to improve the energy efficiency, i.e. by lowering the operating voltage. We make key observations regarding the limiting factors in scaling down the operating voltage for general purpose high performance processors. Later, we propose our novel core design, ScalCore, one that can work in high performance mode at nominal V_{dd}, and in a very energy-efficient mode at low V_{dd}. The resulting core design can operate at much lower voltages providing higher parallel performance while consuming lower energy.

While lowering V_{dd} improves the energy efficiency, CMOS devices are fundamentally limited in their low voltage operation. Therefore, we next consider an upcoming device technology – Tunneling Field-Effect Transistors (TFETs), that is expected to supplement CMOS device technology in the near future. TFETs can attain much higher energy efficiency than CMOS at low voltages. However, their performance saturates at high voltages and, therefore, cannot entirely replace CMOS when high performance is needed. Ideally, we desire a core that is as energy-efficient as TFET and provides as much performance as CMOS. To reach this goal, we characterize the TFET device behavior for core design and judiciously integrate TFET units, CMOS units in a single core. The resulting core, called HetCore, can provide very high energy efficiency while limiting the slowdown when compared to a CMOS core.

Finally, we analyze Monolithic 3D (M3D) integration technology that is widely considered to be the only way to integrate more transistors on a chip. We present the first analysis of the architectural implications of using M3D for core design and show how to partition the core across different layers. We also address one of the key challenges in realizing the technology, namely, the top layer performance degradation. We propose a critical path based partitioning for logic stages and asymmetric bit/port partitioning for storage stages. The result is a core that performs nearly as well as a core without any top layer slowdown. When

compared to a 2D baseline design, an M3D core not only provides much higher performance, it also reduces the energy consumption at the same time.

In summary, this thesis addresses one of the fundamental challenges in computer architecture – overcoming the fact that CMOS is not scaling anymore. As we increase the computing power on a single chip, our ability to power the entire chip keeps decreasing. This thesis proposes three solutions aimed at solving this problem over different timelines. Across all our solutions, we improve energy efficiency without compromising the performance of the core. As a result, we are able to operate twice as many cores with in the same power budget as regular cores, significantly alleviating the problem of dark silicon.

To my parents, for their love, support and inspiration.

ACKNOWLEDGMENTS

I would like express my deep gratitude to a number of people who have helped me in reaching this milestone. Firstly, I would like to express my sincere gratitude to my advisor Professor Josep Torrellas, for his help along the way. It was a wonderful experience working with him and I cannot imagine having a better advisor and mentor for my Ph.D. His hard work, motivation and dedication are inspirational to every graduate student. I hope to emulate his drive, patience and unwavering commitment going forward.

Besides my advisor, I would like thank the rest of my committee: Professor Christopher Fletcher, Professor Wen-Mei Hwu, Professor Nam Sung Kim, and Dr. Asit Mishra for their helpful comments and encouragement. In particular, I wish to thank Professor Christopher Fletcher, whom I had a chance to work with towards the later part of my Ph.D. The stimulating discussions with him on various projects were particularly enjoyable. Next, I would also like thank Dr. Asit Mishra for his help and guidance on my research work as well as during my internship.

Our iacoma research group is full of wonderful, talented and kind students. I would like thank all my fellow lab mates for the stimulating discussions, for the constant encouragement, for enduring the stress together before the deadlines, and for all the fun we have had in the years here. I would like to thank Dimitris and Mengjia in particular for giving me a chance to work with them on exciting projects such as Cloudburst, Sharp and Microscope etc., and for being a constant source of motivation. I would also like to mention my lab mates Raghavendra, Tom and Jiho for sharing the Ph.D. journey with me and for being partners in mutual guidance, learning and discovery. Next, I would like to thank Aditya Agrawal and Amin Ansari, for their advice in the early phase of my research. In addition, I am thankful to all my other collaborators and/or lab mates Yasser, Tanmay, Azin, Antonio, Apostolos, and Serif for their helpful feedback over the years.

I wish to express my special thanks to the various staff in the Computer Science department who helped me along the way. I am grateful to Sherry Unkraut, Madeleine Garvey for all their help with the administrative tasks and for sharing the tips that made my life easier. Similarly, I wish to thank Kara McGregor, Kathy Ann Runck, Mary Beth Kelley, Viveka Kudaligama and Maggie Metzger Chappell for cheerfully guiding me through the processes.

I am lucky to meet a great group of people in Champaign-Urbana who enriched my life significantly and made my stay memorable. In particular, I would like mention our lunch group that consisted of Prasanna, Aditya, Shaileshh and Kaushik. The lunch meet up was

a source of joy and relief that I genuinely looked forward to every day. A special thanks to Sridhar, Nandu and Virajith whose friendship I will always cherish. Finally, I have to mention my friends Ashutosh, Kalyan, Kartik, Kavya, Ketan, Khetan, Madan, Praveen, Sangeetha, Sashi, Shreya and Swetha who made my stay here fun and enjoyable.

Most importantly, I convey my heartful thanks to my wife Pavithra. I am glad to have met you here in Urbana-Champaign during my doctoral study. You have been a great source of motivation and inspiration. You were always there for me; in the times of joy as well as stress. Thank you Pavi. Finally, I would like to express my gratitude and thanks to my parents and my sister for inspiring me, for providing me a great platform to succeed, and for constantly encouraging me in all my endeavors. I proudly dedicate this thesis to you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

In pursuit of higher performance, computer architects have embraced multi-core designs to work around the power wall that halted the rapid progress in increasing single core frequency. Additional transistors available on a chip at smaller process technology nodes are now employed to increase the number of cores on a single chip. This proved to be a good alternative that could provide higher performance at the same operating frequency, for applications that are inherently parallel. However, the number of cores that can run at their highest operating frequency in a large many-core chip is limited by the amount of heat that can be dissipated by the chip. Therefore, only a limited subset of cores can be operated at a time, resulting in dark silicon [7]. Researchers have been exploring several approaches to utilize the additional transistors available on newer process technologies without running into the problem of dark silicon.

One of the widely considered solutions to this problem involves operating the processor cores at a lower voltage and frequency. As a result, the power consumption of each core is reduced and more cores can be turned on within the same power budget. Such a design is also advantageous, as the energy efficiency is higher at lower voltages, i.e. energy consumed per operation is lower at lower voltages. In particular, it is well known that a near-threshold voltage operation provides the highest energy efficiency [8]. However, at these low voltages, the single threaded performance is poor. As applications tend to have alternating serial and parallel code sections, ideally, we need a design that not only operates all the cores available during a parallel section by running them in the most energy efficient manner, but also provides high performance during a critical serial section when only few cores are active. In the first part of my thesis, I present the design of such a core called *ScalCore*. ScalCore operates in two modes – a high-performance mode and an energy-efficient mode, based on the requirements of application.

While lowering the operating voltage of silicon CMOS devices provides higher energy efficiency, it is still limited by the fact that CMOS is an intrinsically-poor switch [9]. CMOS devices need a relatively large change in voltage to go from on-to-off state and vice-versa. As the supply voltage and the corresponding threshold voltage is lowered beyond a point, the leakage power soars, negating the energy savings. Therefore, researchers are actively developing new devices that can turn-off transistor sharply with a small decrease in voltage and vice-versa. Among the various new devices being explored, *Tunneling Field-Effect Transistors (TFETs)* are one of the most promising, thanks to manufacturing feasibility and ability to integrate with the current FinFET CMOS devices [1, 10, 11, 12]. While TFETs

operate efficiently at very low voltage, they do not scale well with increasing voltage. Their performance saturates beyond a certain voltage. Hence, they cannot replace CMOS transistors when high performance is needed. Ideally, we desire a core that is as energy-efficient as a TFET core and provides as much performance as a CMOS core. In the next part of my thesis, I present the design of such a core that judiciously integrates CMOS units and TFET units with in the same core effectively creating a *hetero-device* core, called *HetCore*.

Finally, as the transistor scaling inevitably slows down, it is increasingly apparent that the only way to put more transistors on a single chip is by building vertical transistors and eventually a Monolithic3D (M3D) chip [13, 14]. Traditionally, 3D integration consisted of stacking two or more pre-fabricated dies on top of each other using Through Silicon Vias (TSV) [15, 16]. A Monolithic3D chip, however, is manufactured by sequentially fabricating different layers of devices on top of one another [17, 18]. M3D technology can provide ultra high-density 3D integration and very high-bandwidth communication across layers. It has a significant advantage over TSV 3D stacking in several key metrics such as the area overheads, wire length reductions and thermal characteristics. As Monolithic3D becomes mature, there is a need to understand processor design with this technology. In the final part of my thesis, we analyze the architecture implications of using Monolithic3D, and present the microarchitecture design for a core that uses this technology. We then present architecture solutions to an important limitation of Monolithic3D – degraded transistor performance in the top layer due to manufacturing challenges. Together, we present a core design that overcomes the challenges posed by the Monolithic3D manufacturing technology and exploits the opportunities provided by the Monolithic3D integration.

1.1 SUMMARY OF THE WORK

In this thesis, we present the design of energy efficient cores that are optimized for the upcoming process technology trends. We begin by considering process technologies that are currently available and then proceed to the the ones that are expected to be viable in the medium to long term. The first work involves the design of a core that can operate efficiently at low voltage as well as at high voltage. It focuses on the short-term trends and is feasible with the current process technologies. In the next part, we present a core that judiciously integrates the conventional CMOS and the futuristic TFET units. This work takes advantage of the TFET devices which are expected to be commercially available in a few years. Finally, we present the design of a core in Monolithic3D technology that is expected to succeed the current FinFET based 2D design over a slightly longer timeline. We also provide a detailed analysis of the challenges, opportunities and tradeoffs involved in the

design of such a core. Overall, our work improves the energy efficiency of cores by taking advantage of the upcoming process technology trends.

1.2 CHALLENGES AND OUR CONTRIBUTIONS

The design of a processor inherently involves trade offs between several elements such as performance and energy efficiency. Often, the most energy efficient designs don't provide the best performance and vice-versa. This holds true even with newer device technologies such as TFETs. We present below, a brief of summary of such challenges and the novel techniques we proposed to overcome them.

1.2.1 ScalCore: Designing a Core for Voltage Scalability

Ideally, we want a core that can flexibly work in high-performance mode (*HPMode*) at nominal V_{dd}, and in a very energy-efficient mode (*EEMode*) at low V_{dd} — a voltage lower than that can be attained with DVFS alone. This is tricky because there is a fundamental design trade-off for a core: A core designed for nominal V_{dd} cannot operate at very low voltages as storage cells become failure-prone. Alternatively, if it is designed to operate at very low V_{dd}, the resulting circuit overheads will cause it to consume higher power than needed at nominal V_{dd} — which in turn may trigger performance throttling at nominal V_{dd}.

We make two observations to address this problem. First, we note that the logic and the storage structures in a pipeline scale differently when we lower V_{dd}. Second, at these low voltages a small increase in V_{dd} results in a large increase in frequency. Based on these two observations, we first propose to supply two voltages to the core pipeline, one to the logic stages and a higher one to storage-intensive stages. Next, ScalCore further increases the low V_{dd} of the storage-intensive stages, so that they are substantially faster than the logic ones. Then, it exploits the speed differential by either fusing storage-intensive pipeline stages or increasing the size of storage structures in the pipeline. The result is a design which, in EEMode, is much more energy-efficient as well as faster than the one with conventional cores (using DVFS) while retaining the same performance in HPMode. In the EEMode, we can operate twice as many cores as a conventional design and therefore mitigate the problem of dark silicon.

1.2.2 Hetero-Device Architecture for CPUs and GPUs

As we mentioned earlier, while TFETs operate very efficiently at low voltages, they do not scale well with voltage and their performance saturates quickly. Therefore, they cannot replace CMOS devices completely. Ideally, we desire a core that is as energy-efficient as a TFET core and provides as much performance as a CMOS core.

To attain this goal, we first perform a thorough analysis of the architectural implications of using TFETs in a core. Based on our analysis, we formulate a set of guidelines to identify units that are suitable for replacement by TFETs in a CMOS core. Specifically, an ideal unit to replace with TFETs has the following traits: (i) consumes high power, (ii) amenable to pipelining and/or latency insensitive and (iii) use sizeable area. The CMOS and TFET units in our design are powered at different voltages that are optimal for the respective device technology. We further improve the design by adapting a few micro-architecture optimizations, that are made possible by the presence of slower TFET units. The resulting design, called HetCore, is much more energy efficient than only a CMOS-only core with a small impact on performance. We also show that, in an environment with a fixed power budget, our design can employ twice as many cores as CMOS-only design and provide better performance as well as energy efficiency.

1.2.3 Designing Vertical Processors in Monolithic3D

Monolithic 3D integration offers several new capabilities at a circuit level over a conventional 2D chip or a TSV-based 3D stacked chip. In addition to increasing the transistor density of a chip, M3D has a few key advantages. First, the two layers are extremely close (less than $1\mu m$) and have lower wire length/delay/power in comparison to a 2D or a TSV3D design. Next, the density of interconnection across the layers is very high. Finally, it has good vertical thermal conduction. However, the current M3D manufacturing technology has an important limitation, the layers in the stack beyond the bottom most one have a relatively lower performance.

In this work, we show how to partition a processor for M3D taking into account that the top layer has lower-performance transistors. We design a vertical processor by taking logic, storage, and mixed logic-storage pipeline stages, and partition each of them into two layers. For logic structures, we place the critical paths in the bottom layer and the non-critical ones in the top one. For multiported storage structures, we asymmetrically partition the ports, assigning to the top layer fewer ports with larger access transistors. For single-ported storage structures, we asymmetrically partition the bitcell array, assigning to the top layer a shorter

subarray with larger bitcells. Even with very conservative assumptions on M3D technology, we show that the M3D core executes applications faster than a 2D core, while consuming less energy. Further, under a similar power budget, an M3D multicore can operate twice as many cores as one with 2D cores.

1.3 LONG TERM IMPACT

One of the fundamental challenges in computer architecture is overcoming the fact that CMOS is not scaling anymore. As the computing power on a single die keeps increasing, the ability to power the entire chip keeps decreasing due to the problem of dark silicon. The computing community needs to find a solution to this problem.

One of the ways to address this is by utilizing near-threshold voltage computing. However, the performance of such designs is low. Our work, *ScalCore* [19], addresses this issue directly by providing an energy-efficient mode and a high-performance mode. In the energy-efficient mode, we can enlist as many cores as required by the application. When the application does not need many cores, we can then switch to the high-performance mode. Thus we provide the best of the both worlds: we avoid the problem of dark silicon and still provide very high performance as needed.

ScalCore also provides a fast transition between modes, requiring only a pipeline flush and a voltage change (similar to DVFS). It avoids the numerous overheads associated with thread migration between heterogeneous cores (like in big–little). As a result, the decision of switching between the energy-efficient and high-performance modes can be made at a fine granularity and whenever it is necessary. The overall ScalCore approach is general and applicable to a wide range of core designs. It provides higher energy efficiency at low voltage without impacting performance at nominal voltage. In addition, it enables fast transition between energy-efficient and high-performance modes based on the application needs.

A second way to address the problem of dark silicon is by utilizing Beyond-CMOS devices with better characteristics than the CMOS. While there is no consensus on the type of devices that will be used, TFET is one of the leading candidates. TFET consumes about 8X less power than CMOS, is around 2X slower than CMOS, and can be integrated in the same chip as CMOS. Our work, HetCore [20], provides for the first time, an architectural solution that combines TFET and CMOS in the same core.

The HetCore approach is general and can be applied to any of the Beyond-CMOS technologies with similar properties. The approach that we propose is not specific to TFET devices. If a different device type is found to be more practical, we can apply our techniques—as long as the new device can be integrated with CMOS at pipeline-stage granularity. Our analysis

of the architectural implications provides comprehensive guidelines on how to treat such new devices, and come up with the most efficient micro-architecture.

The HetCore approach is also widely applicable to different types of compute engines — with the main focus on power constrained and throughput-oriented environments. In our work we show it applies very well to GPUs. Similarly, it applies well to accelerators. For example, TFETs can be used in systolic arrays with matrix multiplication units to accelerate deep learning applications. The systolic arrays can be implemented in TFETs without much impact on the overall throughput, while significantly reducing the energy consumption.

Finally, we consider Monolithic3D technology that is broadly considered to be the only way to continue increasing the transistor integration beyond the traditional 2D scaling. As the technology becomes feasible, it is essential for architects to understand the opportunities and challenges of building processors vertically with this technology. While M3D technology has numerous advantages such as short wire lengths, good thermal properties and high density integration, it has a key limitation in the form of degraded performance of top layers.

Our work is the first one to partition a core in M3D technology and present the corresponding tradeoffs in different structures. This is also the first work to address the imbalance in the performance of different layers of silicon, by proposing critical path aware partitioning schemes for both logic and storage structures.

Finally, in all our proposed designs, it is possible to operate twice as many cores within the same power budget as a conventional design, mitigating the problem of dark silicon. In addition, our work uses off-the-shelf software. The program, compiler, or runtime does not need to change at all. This is in contrast to most solutions based on accelerators or heterogeneous architectures. This makes the work highly portable across different platforms. Overall, our work improves the energy efficiency significantly without compromising the performance.

1.4 THESIS ORGANIZATION

This thesis is organized as follows. Chapter 2 describes the design of a core that operates very energy efficiently at low voltage, without compromising the performance at nominal voltage. Based on the insight that the storage and logic structures scale differently with voltage, our design supplies two separate voltages within the core and further reconfigures the pipeline to exploit the speed differential for better performance and energy efficiency. Chapter 3 presents the design of a core that mixes CMOS and TFET units within the core pipeline. We describe the principles to decide whether a unit should be placed in CMOS or TFET. Later, we discuss a few microarchitectural techniques to offset any imbalance

created in the pipeline. In Chapter 4, we analyze the Monolithic3D integration technology and its implications on core design. We address an important constraint in the current M3D manufacturing technology and present the partition of the core into different layers that overcomes the limitation. In Chapter 5, we qualitatively compare our work to the prior work. Finally, Chapter 6 presents the summary of our work in conclusion.

CHAPTER 2: SCALCORE: DESIGNING A CORE FOR VOLTAGE SCALABILITY

2.1 INTRODUCTION

Upcoming trends call for flexible processors. Users will continue to demand higher energy efficiency from computing devices in all domains, even as workloads become more dynamic. For example, sometimes all the cores in a large manycore can contribute to the application, but we can avoid dark silicon only if they all run in a most energy-efficient manner. At other times, only a few cores are active, executing a critical or serial section, and we want them to deliver high performance at any energy cost.

For homogeneous chips, some of this flexibility is currently attained with DVFS [21, 22]. A core operates at high voltage (V_{dd}) and frequency (f) when it needs to deliver high performance, and at low V_{dd} and f when it needs to run energy-efficiently. However, the effectiveness of DVFS is limited. Typically, its lowest V_{dd} is still "high" compared to the most energy-efficient regime.

It is well-known that the most energy-efficient operating point occurs at ultralow V_{dd}s [8]. However, this is a challenging environment, where circuits are affected by process variations. In particular, storage cells become failure-prone, since the V_{dd} is close to their minimum voltage for correct operation (V_{min}) [23, 24]. Intel has recently prototyped the experimental Claremont core [25, 26], which aggressively works all the way down to 0.28V.

The goal of this work is to design a core for *Voltage Scalability*, meaning that it can flexibly work in high-performance mode (*HPMode*) at nominal V_{dd}, and in a very energy-efficient mode (*EEMode*) at low V_{dd} — a V_{dd} lower than can be attained with DVFS but not as low as Claremont's challenging levels. This is tricky because there is a fundamental design trade-off for a core: if it is designed to operate at very low V_{dd}, the resulting circuit overheads will cause it to consume higher power than needed at nominal V_{dd} — which in turn may trigger performance throttling at nominal V_{dd}. This is unacceptable, since we want our core to be competitive with the state of the art at nominal V_{dd}.

To address this problem, we make two observations. First, we note that the logic and the storage structures in a pipeline scale differently with lower V_{dd} [27]. If both share V_{dd} and f then, at low V_{dd}, the storage structures force the logic to work less energy-efficiently than it could. If, instead, storage cells are designed for low V_{min}, they have to use larger or more transistors, which consume additional power when operating at nominal V_{dd}. Hence, we propose to supply two different low V_{dd}s to the pipeline in EEMode: logic-intensive pipeline stages are powered at a lower V_{dd} (V_{logic}), while storage-intensive stages like those accessing

registers and load/store queues are powered at a higher V_{dd}.

Secondly, at these low V_{dd} levels, tiny V_{dd} increases enable big f gains. Therefore, given that storage structures consume little dynamic energy, we propose to set the low V_{dd} of the storage-intensive stages in EEMode (V_{op}) to a value that is *higher* than we would need if we only took into account V_{min}. The result is storage structures that are substantially faster than logic ones. We propose to exploit this speed differential by either fusing storage-intensive pipeline stages or increasing the size of storage structures in the pipeline. Both changes increase IPC and reduce total energy. The result of our proposals is a *Voltage-Scalable* core, or *ScalCore*.

We describe the pipeline modifications to fuse stages and to increase storage structure sizes. The resulting pipeline is kept relatively simple as it has a single f, and operates very energy-efficiently in EEMode. In HPMode, we disable the dual V_{dd}s, fused stages, and larger structures, recreating a plain out-of-order core optimized for high performance.

We use simulations of 16 cores to show that a design with ScalCores in EEMode is much more energy-efficient than one with conventional cores and aggressive DVFS: for approximately the same power, ScalCores reduce the average execution time of programs by 31%, the energy (E) consumed by 48%, and the ED product by 60%. In addition, dynamically switching between EEMode and HPMode based on program phases is very effective: it reduces the average execution time and ED product by a further 28% and 15%, respectively, over running in EEMode all the time.

The main contributions of this work are:

- The design of a voltage-scalable core based on our two observations
- Pipeline changes to exploit the faster storage
- Evaluation of ScalCore and comparison to aggressive DVFS.

2.2 MOTIVATION

2.2.1 The Need for Voltage Scalability

In this work, we consider a high-performance, power constrained large manycore of the future, e.g. targeting cloud servers and high-performance computing. In such environments, users will continue to demand increasing energy efficiency while, at the same time, requiring different execution modes. Sometimes, a program will be highly parallel. In this case, we will attain highest performance by enlisting all the cores in the manycore — as long as they

run very energy-efficiently to avoid dark silicon. At other times, only a few cores will be able to run, as they execute mostly-serial sections. In this case, they will run with the highest performance, taking all the power budget of the chip.

Currently, there are two main approaches to address this conundrum: heterogeneity and DVFS. With heterogeneity, the chip contains some cores designed for energy efficiency and some for high performance. In the example above, the former would be used in the parallel section, while the latter in the mostly-serial one. The unused cores are power gated. An example of this approach is ARM's big.LITTLE [28].

While this approach is useful, it is suboptimal. First, the partition between the two types of cores in the chip is fixed, and may not be the best one for a given application. Second, there is always a fraction of the chip area that is unused — in a big-little pair, either the area of the big core or that of the little one (which can be ≈30% of the big core's area [29]). Finally, changing regimes involves migrating state, which has a performance overhead — e.g., ≈20μs [30].

DVFS [21, 22] uses cores of a single type and changes their V_{dd} and f values (and active core count) depending of the regime. However, this approach is also suboptimal. First, logic and storage structures scale differently with V_{dd} [27]. Hence, either at the high-V_{dd} or low-V_{dd} end, either the logic or the storage structures function suboptimally. Note that this is not fully solved by providing one V_{dd} domain for the core and one for the caches: the core pipeline still has both logic and storage structures. The second reason for suboptimality is that the lowest V_{dd} with DVFS is still "high" compared to the most energy-efficient regime.

Such regime is at significantly-lower V_{dd}s [31, 23, 32]. In this regime, Intel has prototyped the Claremont processor, which supports V_{dd} scaling all the way to 0.28V [25, 26]. Core and caches are in two different V_{dd} domains. However, a core designed to operate at such ultralow V_{dd} needs to employ various circuit-level techniques that increase the area and, when the core operates at nominal V_{dd}, induce higher energy consumption.

This work goes deeper into the general DVFS approach. Our goal is to design a core that scales V_{dd} to values lower than conventional DVFS — but not as low as Claremont, to minimize design complexity, area cost, and power overhead at nominal V_{dd}. Such power overhead is unacceptable because it may trigger performance throttling, and makes the core non-competitive in HPMode. Note that our goal is not to compare the heterogeneity and DVFS approaches. Comparing our design to a heterogeneous one is outside our scope.

2.2.2 Logic and Storage Structures in the Pipeline

A pipeline contains multiple storage structures, such as the register file, load/store queue, or ROB. In general, these structures are built with static cells, which become failure-prone as the V_{dd} goes below a value called V_{min} [23, 24]. There is a fundamental tradeoff between V_{min} and cell size: if we want a lower V_{min}, we need a cell with more transistors, with larger transistors, or FinFETs with more fins [33]. Hence, memory cells designed for lower V_{min} consume higher power and energy when operating at nominal V_{dd}.

This problem worsens for the storage structures used in the pipeline because they are heavily multi-ported. In this case, more transistors are connected to the cross-coupled inverters that form the core of the storage cell. The resulting higher loading effect on the cross-coupled inverters makes the cell more sensitive to process variations [34, 8, 35]. Consequently, we need to increase the cell size, which increases its consumption at nominal V_{dd}.

Since our goal is to keep the processor competitive at high V_{dd} operation, this is an unacceptable tradeoff. For example, Zhao et al. [35] show that going from a 1-fin 8T cell to a 2-fin 8T cell increases the leakage current by ≈20%. Moreover, our Spice simulations show that increasing the number of fins from 1 to 2 causes the 8T cell to consume 21% more power at nominal V_{dd}. This is shown in Figure 2.1, which plots the energy of 1-fin 8T and 2-fin 8T cells for different V_{dd}s normalized to 1-fin 8T at the nominal V_{dd} of 0.9V. The figure corresponds to 22nm and an activity factor of 1. Overall, since we want the storage cells in the pipeline storage structures to be competitive at high V_{dd}, we propose to use finFET-based cells with a single fin.

As we lower the V_{dd}, both logic and storage structures in the pipeline become slower. However, logic and storage structures scale differently [27]. This is shown in Figure 2.2, which we generate with Spice simulations of 22nm technology. The figure shows the increase in delay for a chain of FO4 inverters (*LogicDelay*) and for an 8T register-file bank (*SRAMDelay*) as V_{dd} decreases. The delay is the same and normalized to 1 at nominal V_{dd}. This plot includes the effect of random dopant fluctuation based on ITRS [14]. We see that storage structures become relatively slower. This is in line with the observation made by [36].

In related work, Dreslinski et al. [37] characterized the energy consumption of cores and caches at near-threshold voltage, and found that the energy-optimal V_{dd} for caches makes them 2-4x faster than the cores. We go beyond and exploit the different behavior of logic and storage structures *within the pipeline*.

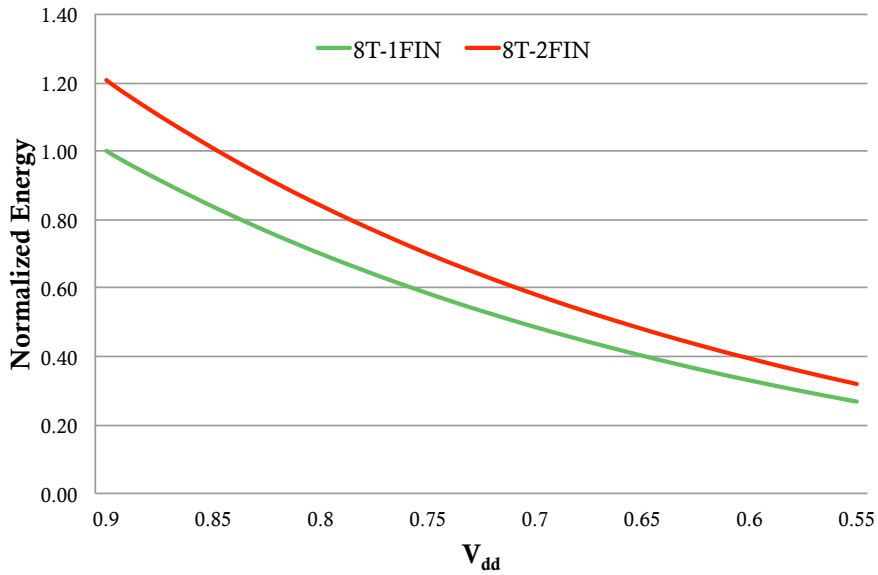

Figure 2.1: Effect of the number of fins in the FinFETs of the 8T cell on energy consumption for different V_{dd}s.

2.3 SCALCORE CONCEPT

2.3.1 Main Ideas

Our goal is to design a core for *Voltage Scalability*, which means that it can flexibly work in a high-performance mode (*HPMode*) at nominal V_{dd}, and in a very energy-efficient mode (*EEMode*) at low V_{dd}. The low V_{dd} is the lowest that can be sustained by the logic structures in the pipeline (but not the storage ones) before their performance becomes substantially degraded — and without requiring changes to basic circuit structures, or changes to transistor size or doping that can hurt the operation at nominal V_{dd}. To attain our goal, we rely on three ideas: (i) provide separate low V_{dd}s in the pipeline for logic and storage structures, (ii) further increase the low V_{dd} for the storage structures, and (iii) leverage the higher speed of the storage structures in the pipeline.

Two Low V_{dd}s in Pipeline: Logic & Storage Designing storage cells to work at very low V_{dd} results in power inefficiency at nominal V_{dd}. Hence, we propose to modify the core to feed two V_{dd}s to the pipeline in EEMode: (1) logic structures are powered at a very low V_{dd} that still enables them to perform acceptably (V_{logic}); and (2) storage structures such as the register file and load/store queue are connected to a V_{dd} that is higher than V_{logic} (and at least as high as their V_{min}). With this design, the core operates with high energy-efficiency.

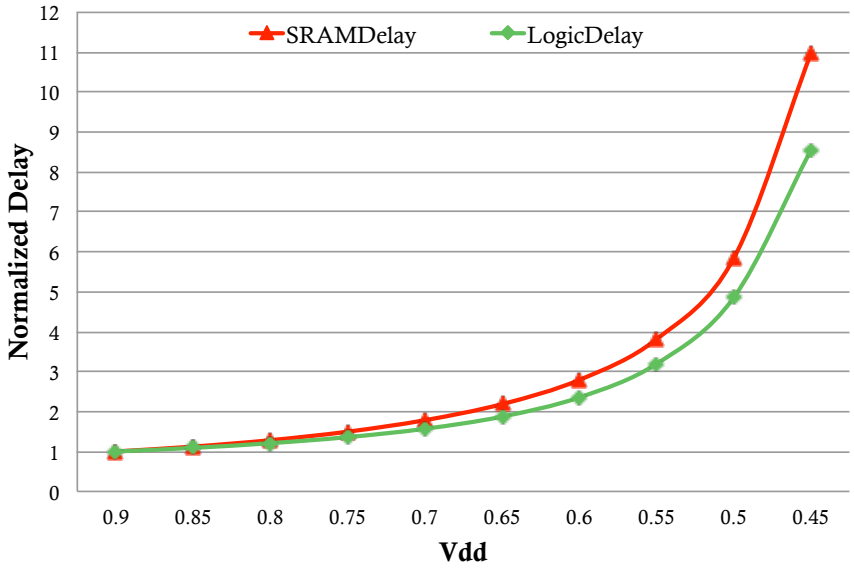

Figure 2.2: Increase in delay for logic and storage structures in the pipeline as we decrease V_{dd}.

Moreover, when we do not want to operate in EEMode, we apply a higher, equal V_{dd} level to both logic and storage structures.

To determine the V_{dd}s, we proceed as follows. We take a four-issue out-of-order core (more details in Section 2.7) and use McPAT's [38] high-performance process at 22nm to determine its f_{nom} at the nominal V_{dd} of 0.9V. Such f_{nom} is ≈3.5GHz. Starting from this point, we use our Spice simulations of Figure 2.2 to generate the V_{dd}-f scalability curves for logic and storage structures. We then adjust these curves with the effects of systematic process variations, using VARIUS [39] with the systematic variation values of EnergySmart [40]. The resulting V_{dd}-f curves are shown in Figure 2.3.

To pick V_{logic}, we observe that the logic delay curve in Figure 2.2 has a knee at V_{dd} ≈0.5V. Going below such value results in increasingly slower structures, which would cause the program to run substantially slower and consume substantially more leakage energy. This observation is consistent with the data in Kaul et al. [8]. Hence, in Figure 2.3, we set V_{logic}=0.5V, which corresponds to f_{logic} ≈600MHz.

To find the V_{min} of the storage structures, we argue that upcoming storage cells are likely to be aggressively designed for energy efficiency, and hence for low V_{min}. For example, Intel has attained SRAMs with V_{min}=0.6V in 22nm [41] and 14nm [42]. Hence, we set V_{min}=0.60V, which in Figure 2.3 corresponds to f_{min} ≈900MHz. This operating point {V_{dd}=0.60V, f=900MHz} is the *lowest point* that we assume can be reached with conventional

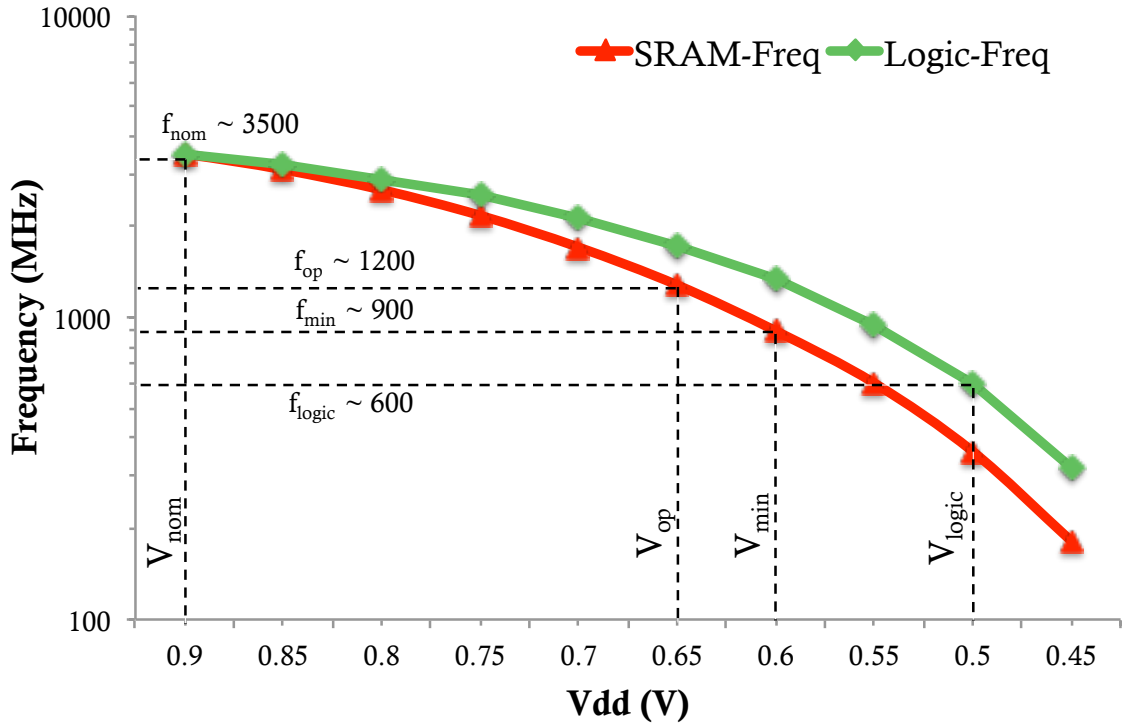

Figure 2.3: V_{dd}-f curves for logic and storage structures.

DVFS with a single V_{dd} domain for the whole pipeline. While the V_{dd} looks aggressively low by today's standards, we think it is plausible, given the need for energy efficiency in upcoming designs.

Further Increase V_{dd} for the Storage Structures We can improve the energy efficiency of the EEMode if we consider the following traits of the EEMode regime:

- While the storage structures need a higher V_{dd} than the logic ones for safe operation ($V_{min} > V_{logic}$), the storage structures at V_{min} can in fact operate faster than the logic structures at V_{logic} [27, 43]. This is seen in Figure 2.3.
- At this range of V_{dd}s, a small increase in V_{dd} provides a significant boost to the operating frequency f.
- For storage structures at these low V_{dd}s, the dominant component of power consumption is leakage.

These observations suggest that, in EEMode, a small further increase in the V_{dd} of the storage structures can make them significantly faster, while consuming only a little more

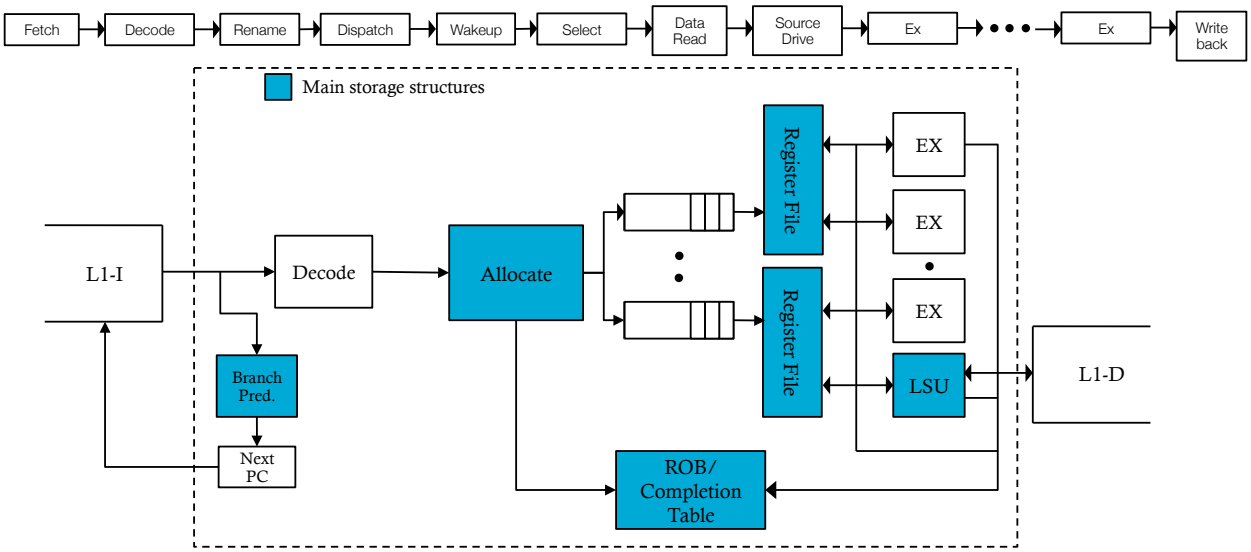

Figure 2.4: Pipeline of an out-of-order processor with the main storage structures.

power. A higher V_{dd} increases the dynamic power comparatively more than the static power. However, since the dynamic power of storage structures in EEMode is small, the overall power will increase little.

Hence, we propose *ScalCore* as a core where, in EEMode, the V_{dd} of the storage structures is set to a voltage V_{op} that is higher than V_{min}. Specifically, we set $V_{op} > V_{min}$ such that storage structures can operate at 2x the f of the logic structures (which use V_{logic}). This is shown in Figure 2.3, where V_{op} ≈0.65V and f_{op} ≈1200MHz.

By setting the storage structures to V_{op} rather than V_{min}, we will improve the IPC of the cores. The resulting lower execution time of the applications in turn reduces the leakage energy — not only of the cores, but also of the caches. This reduction more than compensates the small increase in dynamic energy in the storage structures induced by going from V_{min} to V_{op}.

Leverage the Higher Speed of Storage Finally, we exploit that storage structures are faster than logic ones in ScalCore's EEMode in one of two ways:

- Without changing the core's f, we fuse two consecutive pipeline stages that are dominated by storage structures into one. This reduces pipeline depth and improves the IPC.

- Without changing the core's f, we enable more entries in critical storage structures in the pipeline, consuming some of the available time slack. This increases the exploitable

instruction-level parallelism (ILP) and memory-level parallelism (MLP), and improves the IPC.

In either case, in EEMode, all the stages of the ScalCore pipeline cycle at the same f. This keeps the pipeline relatively simple. Outside EEMode, we disable the fusing of pipeline stages and the larger storage structures, and use a single V_{dd}. The core becomes a plain out-of-order core optimized for high performance. It can use conventional DVFS (with a single V_{dd} in the pipeline) to vary its operating point.

Overall, ScalCore can flexibly deliver high performance in HPMode and high energy efficiency in EEMode. However, we need to carefully select which pipeline stages to fuse and which structures to resize , so that energy-efficiency is maximized. In addition, supporting this fusing and reconfiguration in EEMode should not hurt the performance in HPMode. The remainder of this section and the next addresses these two issues.

2.3.2 Analysis of Pipeline Stages

We analyze the pipeline stages that have storage structures to identify opportunities to improve EEMode operation. At low V_{dd}, Dreslinski et al. [37] showed that operating the L1 caches at a higher V_{dd} than the core delivers energy efficiency. This same approach was used in Claremont [25, 26]. Hence, all of our low-V_{dd} designs (including the baseline) operate the L1 caches at V_{op} by default. With ScalCore, we go beyond and use two V_{dd} domains in the pipeline.

Figure 2.4 shows the pipeline of an out-of-order processor and identifies the main storage structures. They are the register file, allocation structures, load/store unit (LSU), ROB, and branch predictor. We now analyze each of these structures for possible enhancements when we operate them at V_{op}. The enhancements can consist of either fusing stages or increasing the size of structures. The implementation details are discussed later in Section 2.4.

Register File The critical path of a register file access in a pipelined design consists of two steps. In the first step, the operands are read from the data array into a buffer; in the second step, they are delivered to the execution units. This process takes at least two cycles in a typical high-frequency design [44]. By operating at V_{op} in EEMode, we can enhance the register file in one of two ways: either reducing its access latency in cycles or increasing its size.

In the first case, we operate the two steps of a register file access at twice the speed, and fuse them into a single cycle. In the second case, we increase the size of the physical register file without changing the two cycles of access time.

Allocation Structures The allocation step primarily involves register renaming and instruction dispatch. A detailed analysis of the critical path in renaming is performed by Palacharla et al. [45]. The authors found that the critical path in rename consists of reading the current mappings of the source registers in the Register Alias Table (RAT), followed by updating the mappings of the destination registers in the RAT. The dependence check among the registers currently being renamed is not in the critical path, and is implemented using low-power transistors, which reduces the dynamic power. Also, the design of the RAT is similar to the register file. Hence, we operate the rename unit at V_{op} in EEMode.

The rename delay in an out-of-order core is proportional to the core's width. In a very wide core, the rename operation may take more than one cycle. However, for a core width of four like ours, renaming in a baseline design can be completed in a single cycle. Hence, in ScalCore, we can combine the rename and dispatch stages and fuse them into one cycle operating at V_{op}. Since the dynamic power consumption of the dispatch unit is low [46], the increase in V_{dd} to V_{op} has only a small effect. We do not change the size of the rename unit because we find that it does not have a critical resource whose size can be increased to improve performance.

Load/Store Unit (LSU) At a high level, the LSU consists of two stages, each taking one cycle. The first stage performs address generation, and the second one memory disambiguation. Also, in parallel to the disambiguation, store instructions write values to the store buffer. The load/store queue in the LSU contains in-flight memory instructions and maintains the ordering between instructions.

Since the LSU takes two cycles and also has a resource (load/store queue) whose size can be increased, as we operate the LSU at V_{op}, we can do one of two things: either we fuse the two stages to reduce the latency to one cycle, or we increase the load/store queue size. When we fuse the two stages, stores write to the store buffer in half of a cycle. When we increase the load/store queue size, we also increase the store buffer size. Note that increasing the size of the load/store queue requires extra care, since it uses CAM structures to perform memory disambiguation (Section 2.4.1).

ROB We consider a merged register file-based renaming scheme as described in [44]. The ROB only acts as a completion table, keeping track of the in-flight instructions. The entries in the ROB are reserved on instruction dispatch and are freed on commit. The commit process in HPMode takes only one cycle. Hence, there is no obvious opportunity to reduce the latency in EEMode. However, we modify the size of the ROB in EEmode to enable more in-flight instructions. This allows us to exploit more ILP or MLP, based on the application.

The dynamic power of the ROB itself is low [46]. Hence the increase of the V_{dd} to V_{op} causes a modest increase in dynamic energy.

Branch Predictor Branch prediction in modern processors consists of a fast BTB to provide prediction in a single cycle, which is backed-up with a more complex and accurate branch predictor with a longer latency. In our baseline design, we have a tournament predictor with a total size of 48K entries and a BTB of 2K entries. Although the BTB is structurally similar to a register file or cache, there is no obvious opportunity to further reduce its latency. While it is possible to take advantage of the faster operation at V_{op} by using a fancier branch prediction, it is out of our scope. Also, by simply increasing structure sizes, the expected improvement in IPC is small, considering the high baseline size [47, 48]. Hence, we do not optimize the branch predictor.

2.3.3 Summary

Based on the above analysis, Table 2.1 lists the pipeline structures that can be enhanced by ScalCore in EEMode, and the enhancement options. Note that a given structure cannot both run faster and be bigger at the same time. Outside EEMode, all structures are operated at the same V_{dd}, and at their baseline speed and size.

Table 2.1: Enhancements considered for different structures.

Structure	Faster	Bigger
Register File	✓	✓
Allocation Struc.	✓	✗
Load Store Unit	✓	✓
ROB	✗	✓
Branch Pred.	✗	✗

2.4 SCALCORE DESIGN

To support the enhancements in Table 2.1, ScalCore requires some changes over conventional processors. We classify them into microarchitectural changes in the datapath and controlpath, and circuit changes. We consider each in turn.

2.4.1 Datapath Microarchitectural Changes

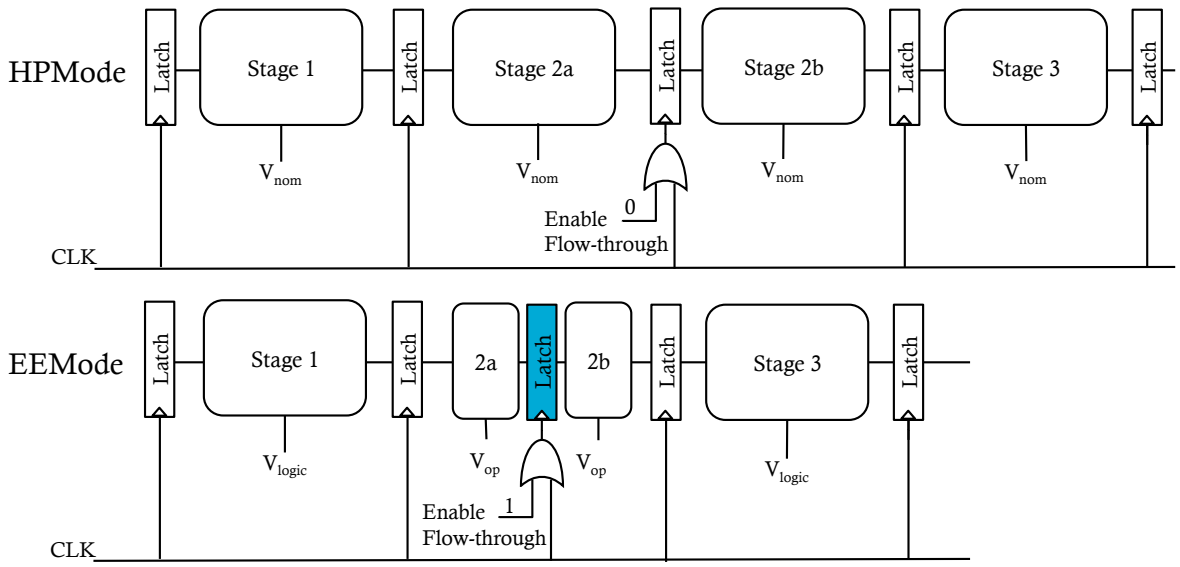

Figure 2.5: Datapath changes in ScalCore: Fusing pipeline stages using a flow-through latch. The shaded components are disabled.

Reduction in Latency Fusing two consecutive pipeline stages in EEMode is accomplished by transforming the latch that sits in between them into a transparent (or flow-through) latch [49, 50]. This is done by ORing the clock to the latch with an Enable Flow-through signal. When the signal is set to logic one, the latch is transparent.

The logic design is shown in Figure 2.5. In HPMode (upper chart), Stages 2a and 2b take one cycle each, and operate at nominal voltage (V_{nom}) like the other stages. The Enable Flow-through bit in the OR gate is set to logic zero. This causes the latch to be controlled by the clock.

In EEMode (lower chart), Stages 2a and 2b are fused together to execute in a single cycle. The Enable Flow-through bit is set to logic one, which makes the latch transparent. In addition, Stages 2a and 2b operate at V_{op}, which is higher than the V_{logic} used in Stages 1 and 3.

This OR gate is added to the latches connecting the pairs of stages to be fused: (i) the two stages of a register file access, (ii) rename and dispatch, and (iii) the two stages in the LSU. Designs based on flip-flops can be modified in a similar manner, by providing a bypass path that is enabled in EEMode.

Bigger Structures The size of a storage structure in ScalCore is increased by enabling an additional array with transmission gates. This general approach was proposed by Buyuktosunoglu et al. [51] for issue queues. By toggling the input to the transmission gates, we can

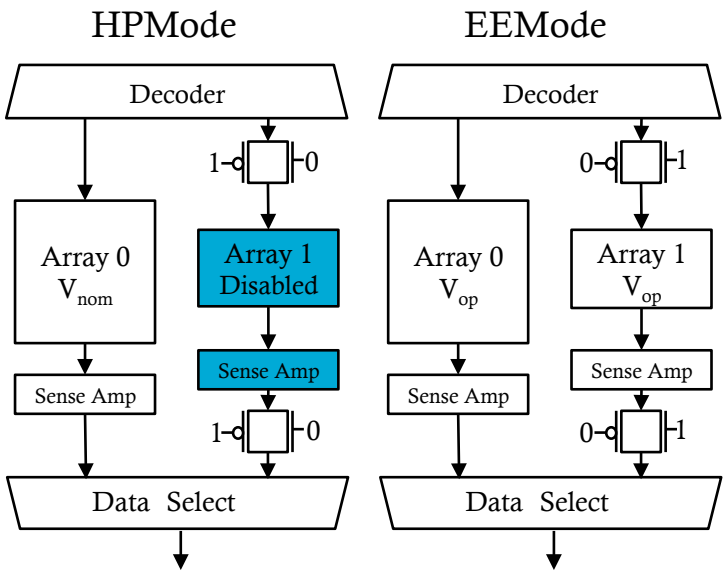

Figure 2.6: Datapath changes in ScalCore: Increasing the size with transmission gates. The shaded components are disabled.

easily enable or disable the new array. Figure 2.6 shows an example of a structure with an original array (Array 0), and an additional one (Array 1) connected with transmission gates. In HPMode (left chart), the transmission gates disable Array 1, while Array 0 operates at V_{nom}. In EEMode (right chart), the transmission gates enable Array 1. Both arrays are active and run at V_{op}.

With this design, the register file, load/store queue, store buffer, and ROB can use more entries in EEMode than in HPMode. CACTI analysis [52] shows that, at the higher V_{op}, we could increase the structure sizes substantially and still meet the cycle time. However, very large structures incur area and leakage overheads, and are hard to use cost-effectively. As we will see, a possible design is to increase the size of these structures by 50% in EEMode.

We make special arrangements for the load/store queue, since it uses CAM structures to perform memory disambiguation. Specifically, to reduce complexity, ScalCore uses a segmented load/store queue [53] with two segments. In HPMode, only one segment is active and the other is power gated. Hence, the delay and power in HPMode are not impacted. In EEMode, both segments are active and are sequentially searched on a request. Since the segments operate at V_{op}, both segments are searched in a single cycle.

In EEMode, the dynamic power of the load/store queue increases only for the searches that overflow into the second segment. Also, techniques like low-swing, selective precharge of search line/matchline reduce the dynamic power of CAM by more than 50%, with no

delay impact over a conventional design [54]. By using such techniques, the dynamic power component becomes small even in the baseline. Hence, the increase in load/store queue size causes only a small increase in the dynamic energy.

2.4.2 Controlpath Microarchitectural Changes

Reduction in Latency Reducing the latency of some operations has an impact on the scheduling of various tasks in the pipeline. So, we need to modify the control logic responsible for those tasks. Specifically, in EEMode, by reducing the register file read latency (which includes the source drive of operands) from two to one cycle, the execution units receive operands in one cycle, and hence can begin execution one cycle earlier. In addition, as the execution finishes a cycle earlier, the dependent instructions can be woken up and scheduled earlier. Hence, the execution schedule time and the generation of the wakeup signal need to be updated based on the mode of operation. Similarly, to enable the speculative issue of load-dependent instructions, the issue logic should be updated with the new latency of the LSU.

Reducing the latency of the allocation operation does not have an obvious direct impact on the scheduling of tasks. The process of renaming and dispatching instructions in order can proceed in the same manner in both modes.

Therefore, the only controlpath modifications required are to ensure that the part of the issue-queue state machine responsible for generating ready signals accurately reflects the mode of operation. The issue queue already contains functionality to generate the wakeup signal at different times based on the latency of the functional unit (e.g., ADD vs. DIV) [44]. Hence, by modifying the counters used for this process, we ensure correct scheduling in both HPMode and EEMode.

Bigger Structures The availability of a bigger resource should be conveyed to its corresponding resource manager to enable its usage. In EEMode, the sizes of the register file, load/store queue, store buffer, and ROB are higher. To benefit from the bigger physical register file, the list of free registers in the rename unit must be updated. The list of free registers is typically maintained as a circular buffer with head and tail pointers, which can be resized based on the mode of operation.

The dispatch unit is responsible for checking various execution resources and stall in case of unavailability. It maintains counters for the availability of each resource. To indicate the sizes of the current ROB, load/store queue, and store buffer, we just need to update these counters based on the mode of operation. Overall, therefore, the changes required in

the controlpath for bigger structures involve reconfiguring the counters in the rename and dispatch stages.

Table 2.2 summarizes the microarchitectural changes in the datapath and controlpath to enable our enhancements.

Table 2.2: Microarchitectural changes in ScalCore.

Component	Reduced Latency	Bigger Size
Datapath	Gated clocks for transparent latches	Transmission gates to resize structures
Controlpath	Programmable counters for latency	Programmable counters for size

2.4.3 Circuit Changes

The circuit changes involve supporting dual voltage rails and level converters.

Dual Voltage Rails In ScalCore, each pipeline stage is connected to one of two V_{dd} rails, as shown in Figure 2.7. All the storage-intensive stages are connected to one rail. They are the two stages of register file access, the rename stage, the dispatch stage, the two stages of LSU access, and the commit stage. The other stages are connected to the other rail. Each rail is set to the required V_{dd} level, based on the mode of operation. In EEMode, the storage-intensive stages receive V_{op}, while the rest receive V_{logic}; outside EEMode, both rails supply the same voltage.

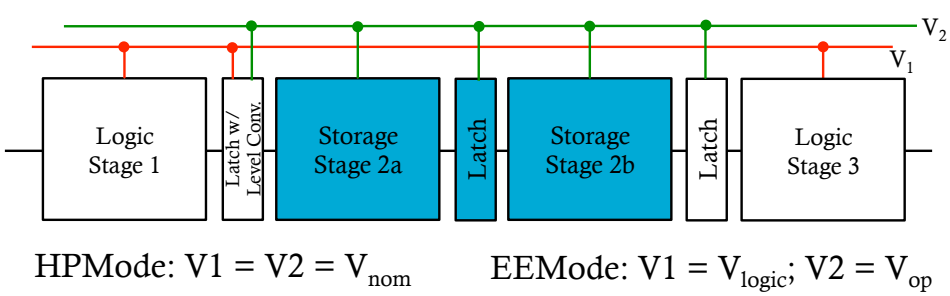

Figure 2.7: ScalCore pipeline with dual voltage rails.

These storage-intensive stages share the V_{dd} domain with the L1 caches which, as indicated in Section 2.3.2, also operate at V_{op} in EEMode. As a result, there is no need to add any

additional voltage regulator in ScalCore over the baseline design. Note that the L1 caches are laid out together with the pipeline in current designs.

Level Converters All the stages in ScalCore always operate at a common *f*. This simplifies the design by avoiding multiple clock trees and synchronization overhead across stages. However, consecutive pipeline stages may operate at different V_{dd} levels in EEMode. Hence, a level converter is required on the boundary between a stage at V_{logic} and one at V_{op}, to provide full swing input to the higher V_{op} domain. Note that the difference in V_{dd} levels is only 150mV.

Level converters can be designed as part of the latches or flip-flops that separate the stages (Figure 2.7). The design of a level converter is shown in Figure 2.8. It is based on [55], where pulsed half-latch level converting flip-flops are shown to be efficient, both in area and in energy-delay, compared to asynchronous level conversion in a dual-V_{dd} system.

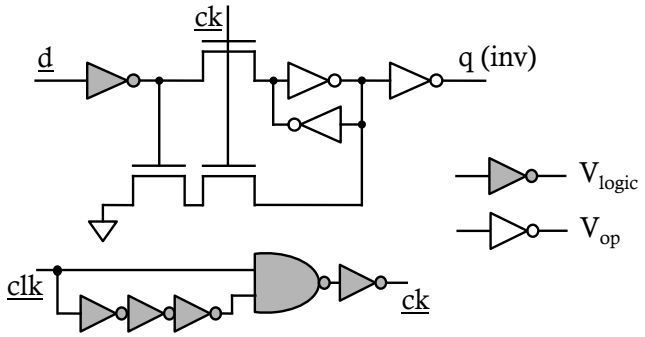

Figure 2.8: Level converter for up-conversion.

2.5 IMPLEMENTATION CONSIDERATIONS

This section considers three implementation issues: the transition between modes, a summary of ScalCore overheads, and a comparison to Intel Claremont.

2.5.1 Transition Between Modes

In highly-parallel sections, all the cores in a large manycore are powered-on and run in EEMode. In sections with little parallelism, only a few cores are powered-on and run in HPMode, using all the power budget of the chip. While in HPMode, ScalCore can also use DVFS.

ScalCore provides a simple way to reconfigure the pipeline between the two modes. First, pipeline reconfiguration can only occur when the pipeline is empty. Consequently, reconfiguration requires a pipeline flush. Such a flush can take a variable number of cycles — e.g., depending on whether there are pending writes in the store buffer (pending reads are squashed). However, the average flush is not likely to take more than several tens of cycles. Second, we need to change the f and V_{dd} of the core. Specifically, from a common V_{dd} in all stages in HPMode, we transition to V_{op} for the storage-intensive stages and V_{logic} for the others in EEMode (or vice-versa). This transition uses conventional DVFS mechanisms. Its overhead is likely to be modest in the future, as increasing DVFS speed is an active area of research and development. In our evaluation, we set this overhead of changing f and V_{dd} between modes to 1 μs. Note that pipeline stages *do not switch* V_{dd} rails; ScalCore merely changes the rails' voltage.

Overall, the overhead of transitioning from EEMode to HPMode and vice-versa is small. The reconfiguration can be triggered either in hardware by the power management unit, or in software by the operating system or program.

2.5.2 Summary of ScalCore Overheads

Table 2.3 summarizes the ScalCore overheads and their impact on HPMode. The first issue is dual V_{dd} rails. Their main overheads are the additional area they take and the need to customize their layout/routing, since automatic tools may not be able to handle them. One implementation of dual rails [56] estimates the area cost to be ≈5% of the core.

Table 2.3: Summary of ScalCore overheads.

Issue	Type of Overhead	Impact on HPMode
Dual V_{dd} rails	1) Custom layout and routing. 2) Area increase	≈5% area [56]
Level converters	1) Carefully manage clock skew/timing. 2) Add gates in critical path	≈5% delay [55]
Fusing Pipeline Stages	1) OR gate added to the clock signal for each latch connecting fused stages. 2) Control logic for counters for latency	Tiny area and power
Increasing Array Sizes	1) Additional array. 2) Transmission gates to enable/disable additional array. 3) Segmented ld/st queue. 4) Control logic for counters for size	Area of power-gated array. Tiny power and delay [51]

The second issue is level converters. They require carefully managing the clock skew and timing across domains. Moreover, they add a few gates to the pipeline stage. Based on

Ishihara et al.'s [55] work, we estimate a delay impact in HPMode of ≈5%. The additional area and power of the level converters is negligible [55].

The third issue is fusing pipeline stages. It requires an OR gate added to the clock signal for each latch connecting fused stages, and control logic for counters for latency (Section 2.4.2). In HPMode, it adds a tiny area and power overhead but no delay in the critical path.

A fourth issue is increasing array sizes. It requires an additional array, transmission gates to enable/disable the additional array, a segmented load/store queue, and control logic for counters for size (Section 2.4.2). In HPMode, the additional array is power gated, but takes up area. The additional logic introduces only tiny power and delay overheads. Buyuktosunoglu et al. [51] show that the transmission gate delays are negligible. Their design is more involved than ours in that they get multiple, dynamically variable latencies, which require careful synchronization with the rest of the pipeline. In our case, we only have two configurations, and the large one has ample timing slack. Similarly, Park et al.'s [53] segmented load/store queue is more involved because disambiguation can take a variable number of cycles, while ours always takes one cycle.

Finally, ScalCore introduces design complexity and verification costs, which are hard to quantify.

2.5.3 Comparison to Intel Claremont

Table 2.4 compares Intel's Claremont prototype [25, 26] to ScalCore. A main difference is that Claremont targets a very wide V_{dd} operating range (1.2V to 280mV) while ScalCore targets a more modest range (0.9V to 0.5V). As a result, Claremont uses more aggressive techniques, including manually prioritizing the placement of logic paths that have a high

Table 2.4: Comparing Intel Claremont to ScalCore.

Trait	Claremont	ScalCore
V_{dd} range	Very wide: 1.2V to 280mV	Wide: 0.9V to 0.5V
Focus	Circuit techniques: 1) Variation aware pruning & beefing-up of cells. 2) Circuits optimized for reliability at ultralow V_{dd}	Architectural techniques: 1) Separate V_{dd} for memory intensive pipe stages. 2) Fuse pipeline stages. 3) Increase array sizes
V_{dd} domains	1 for the pipeline + 1 for the L1	1 for the memory intensive pipe stages and L1 + 1 for the other pipe stages

percentage of interconnect, or using two level-converters in series to bring the voltage to high V_{dd}.

Another difference is that Claremont's focus in on circuit techniques, while ScalCore's is on architectural techniques. Claremont includes variation-aware selective pruning and beefing-up of the standard cell library, and circuits optimized for reliability at ultralow V_{dd} — e.g., avoiding wide transmission-gate multiplexers and circuits with high transistor stacks. The result of needing to reach an ultralow V_{dd} is device area bloat.

ScalCore focuses on architectural techniques such as a separate V_{dd} for memory-intensive pipeline stages, fusing pipeline stages, and increasing array sizes.

Although the number of V_{dd} domains in both designs is the same, they are organized differently. Claremont has one domain for the pipeline and one for the L1; ScalCore has one for the memory-intensive pipeline stages plus the L1, and one for the rest of the pipeline stages.

2.6 IMPLICATIONS ON APPLICATIONS

The performance of a core is dependent on various factors, including the frequency, cache/memory latency, pipeline depth, and pipeline structures. In ScaleCore, we modify the pipeline depth and sizes of structures in EEMode. We now qualitatively discuss the impact of these modifications on application performance. We will discuss the detailed evaluation results later in Section 2.8.

2.6.1 Pipeline Stage Fusion

In EEMode, ScalCore can perform register access, rename-dispatch, and LSU handling in one cycle each, rather than in two cycles each. In total, the pipeline depth reduces by three cycles for load/store instructions and by two cycles for others. This reduction has a few implications, as noted by Tullsen et al. [57]. First, it reduces the branch misprediction penalty by 2 cycles. Second, it results in fewer instructions from the mispredicted path in the pipeline, which saves energy and resources. Third, registers are now held for a shorter duration, reducing the contention for physical registers during renaming. Finally, reducing the LSU latency also helps the optimistic issue of load-dependent instructions that assume a cache hit.

Overall, these effects result in an increase in the average IPC for a broad range of application types — especially for integer applications, which have more branches.

2.6.2 Bigger Structures

Increasing the sizes of the register file, load/store queue (LSQ), store buffer, and ROB enables an out-of-order core to extract more ILP/MLP and, therefore, boost performance. Traditionally, increasing LSQ size to exploit higher MLP results in additional pipeline stages, hurting the IPC. In ScalCore, by operating the LSQ at V_{op}, we are able to increase the size and still keep the number of stages constant.

These changes improve the IPC for most applications, especially those that stress the current structures — e.g., memory intensive codes constrained by LSQ size, or FP applications with sizable ILP.

2.7 EVALUATION METHODOLOGY

To evaluate ScalCore, we use the SESC architectural simulator [58], modeling up to 16 4-issue out-of-order cores. We use McPAT for detailed energy calculation [38]. For a more accurate evaluation of dynamic and leakage energies, we model the L1 and L2 caches with CACTI [52]. Table 2.5 shows the parameters of the simulated architecture.

Table 2.5: Parameters of the simulated architecture for the evaluation of ScalCore.

Parameter	**Value**
Architecture	Up to 16 4-issue out-of-order cores at 22nm
Register file; ROB	128 regs; 128 entries
Issue queue	48 entries
Ld queue; St queue	48 entries; 32 entries
Branch prediction	Tournament predictor: bimodal (16K entry), gshare (16K entry) and selector (16K); 32-entry RAS; 4-way 2K-entry BTB
Functional units:	
2 integer ALU	4-cycle Mult/Div, 1-cycle for rest
2 LSU	2 cycles
2 FPU	1-cycle Add, 4-cycle Mult, 12-cycle Div
Private I-Cache	64KB; 2way; 64B line; Round-trip (RT): 2cycles (HPMode) or 1cycle (EEMode or any low-V_{dd} design)
Private D-Cache	64KB; 4way; writeback (WB); 64B line; RT: 2cycles (HPMode) or 1cycle (EEMode or any low-V_{dd} design)
Shared L2	Per core: 1MB; 8way; WB; 64B line; RT to local bank: 8cycles (HPMode) or 6cycles (EEmode or low-V_{dd} design)
DRAM latency	RT: 50ns
Network	2D mesh with MESI directory-based protocol
EEMode-HPMode change overhead	Pipeline flush + 1μs (for V_{dd} and f change)

Table 2.6 shows the f and V_{dd} for HPMode and EEMode. They were justified in Section 2.3.1. We have penalized the f of our HPMode by 5%, due to the delay overhead of ScalCore in Table 2.3. Note that the cache latencies in cycles in Table 2.5 are higher in HPMode than in EEMode (and all of the low-V_{dd} designs that we will analyze). This is because the core's f is different. Finally, in EEMode and all of the low-V_{dd} designs, caches operate at the higher V_{op}=0.65V to improve efficiency, as suggested in [37].

Table 2.6: HPMode and EEMode design points.

HPMode	V_{dd}=V_{nom}=0.9V; f=3.3GHz /* 5% penalty as per Table 2.3 */
EEMode	V_{logic}=0.50V for logic and V_{op}=0.65V for storage; f=0.6GHz

2.7.1 Design Configurations

We evaluate several configurations operating at low V_{dd}, as shown in Table 2.7. First, *DVFS+* is the most energy-efficient voltage-frequency setting that we assume an aggressive DVFS can reach. It operates the whole pipeline at V_{dd}=V_{min}=0.6V, and f=0.9GHz. In addition, the caches operate at the more energy efficient V_{op}=0.65V (like all the other designs in the table) — hence, the suffix "+". This is the V_{dd} domain arrangement of Dreslinski et al. [37] and Claremont [25, 26], although the actual V_{dd} levels are different.

Table 2.7: Configurations explored in low V_{dd}.

Config.	**Parameters**
DVFS+	Whole pipeline at V_{dd}=V_{min}=0.60V; f=0.9GHz. /* Uses the V_{dd} domain arrangement of Dreslinski et al. [37] and Claremont [25, 26] */
Pipe2Vdd	Pipe logic at V_{logic}=0.50V and pipe storage at V_{dd}=V_{min}=0.60V; f=0.6GHz. /* Storage not fast enough to exploit the speed difference */
SC:	Pipe logic at V_{logic}=0.50V and pipe storage at V_{dd}= V_{op}=0.65V; f=0.6GHz. /*ScalCore EEMode variations*/
SCsp1	Reg = 1 cycle latency
SCsp2	Reg, alloc, LSU = 1 cycle latency
SCsz	Reg, LSQ, store buff, ROB = 1.5x size
SCmx	Reg, alloc = 1 cycle lat; LSQ, store buff, ROB = 1.5x size

Pipe2Vdd adds a dual V_{dd} domain in the pipeline: it sets V_{logic}=0.50V for the logic stages, and V_{dd}=V_{min}=0.60V for the storage stages. However, storage is not fast enough to exploit the speed difference with logic. f is 0.6GHz.

Next, *SC* creates ScalCore by keeping V_{logic}=0.50V for the logic stages but increasing V_{dd}=V_{op} =0.65V for the storage stages. *f* remains at 0.6GHz. We have four variations with different hardware support. First, *SCsp1* and *SCsp2* (for speed) reduce latencies by fusing pipeline stages. Specifically, SCsp1 fuses the two stages in the register file access into one. SCsp2 augments SCsp1 by also fusing the two stages in the allocation and the two in the LSU into one each.

SCsz (for size) keeps the pipeline unchanged but increases the sizes of the register file, LSQ, store buffer, and ROB. We have evaluated different amounts of size increases. For brevity, we present data for only one size, which delivers one of the best tradeoffs between energy and performance. The design is for 1.5x structure sizes. It can be shown that other sizes are less cost-effective or only marginally more cost-effective.

SCmx (for mixed) combines fusing stages and increasing structure sizes. Specifically, it fuses the two stages in the register file access into one, and the two in the allocation into one. It sets the sizes of the LSQ, store buffer, and ROB to 1.5x their original sizes. Note we can only increase either the speed or the size of a given unit at a time.

All of these low-V_{dd} designs have the same low cache latencies in cycles as ScalCore's EEMode. The values were shown in Table 2.5.

At V_{nom}, we evaluate *HPMode* (our ScalCore) and *HPRef* (state-of-the art, like *HPMode* but without the 5% *f* penalty).

2.7.2 Applications & Metrics

We evaluate the architectures with a variety of parallel applications. From SPLASH-2, we use Barnes (16K particles), Cholesky(tk29.O), FFT(2^{20}), FMM(16K), LU(512x512), Radiosity(batch), and Radix(2M keys). From PARSEC, we use Blackscholes (sim medium), Fluidanimate (sim small), and Streamcluster (sim small). From NAS, we use BT, LU, and SP (all W size). We run each application to completion.

Our metrics are execution time, energy consumption (E), and energy-delay product (ED). In our evaluation, we start by comparing the different ScalCore EEMode configurations to *HPRef*, *DVFS+*, and *Pipe2Vdd*. We do it in an environment with a fixed power, and one with no power constraints. In both cases, we run the SPLASH-2 and PARSEC codes and do not change configurations dynamically. Then, we consider the environment with the fixed power again, and allow changing the configurations dynamically. In this case, we run the NAS codes. In each NAS code, we prevent the parallelization of some parts so that the serial section takes ≈30% of the total execution time.

2.8 EVALUATION

2.8.1 Environment with a Fixed Power

We compare the different ScalCore EEMode configurations, *HPRef*, *DVFS+*, and *Pipe2Vdd* for a fixed amount of power. To do this, we note that the power consumption of a single *HPRef* core and its caches is ≈12W. Then, each of the other configurations can have as many cores as needed to get to ≈12W. Figure 2.9 shows, for each configuration, the power consumed and the number of cores that can execute. From the figure we see that, for our applications, for the power of one *HPRef* core (*1-HPRef*), we can run one *HPMode* core (*1-HPMode*), or eight *DVFS+* cores (*8-DVFS+*), or 16 *Pipe2Vdd* cores (*16-Pipe2Vdd*), or 16 cores under the various ScalCore EEMode configurations (*16-SC**).

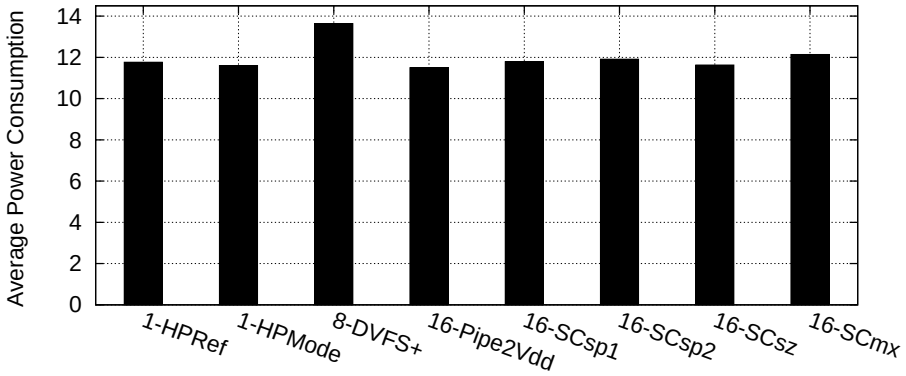

Figure 2.9: Configurations for a fixed power.

We now take these designs and compare their execution time, total E, and ED product running the applications. Figure 2.10 shows, for each design, the execution time of each application and the average. In each application, the bars are normalized to *16-Pipe2Vdd*.

We consider first the different ScalCore designs. We see that all of its variants reduce the execution time over *16-Pipe2Vdd*. The gains come from the increase in the V_{dd} of the storage structures in the pipeline, and the resulting pipeline reconfiguration. Most of the applications show higher gains due to a reduction in the latency (*16-SCsp1* and *16-SCsp2*) than due to an increase in the size of the structures (*16-SCsz*). This is because applications typically exhibit sensitivity to branch misprediction penalty and load-to-use delay, both of which are reduced in the *16-SCsp** designs. However, for these applications, the additional changes going from *16-SCsp1* to *16-SCsp2* have minimal impact, as they are hidden by other effects such as pipelining and inter-thread synchronization. In addition, applications that

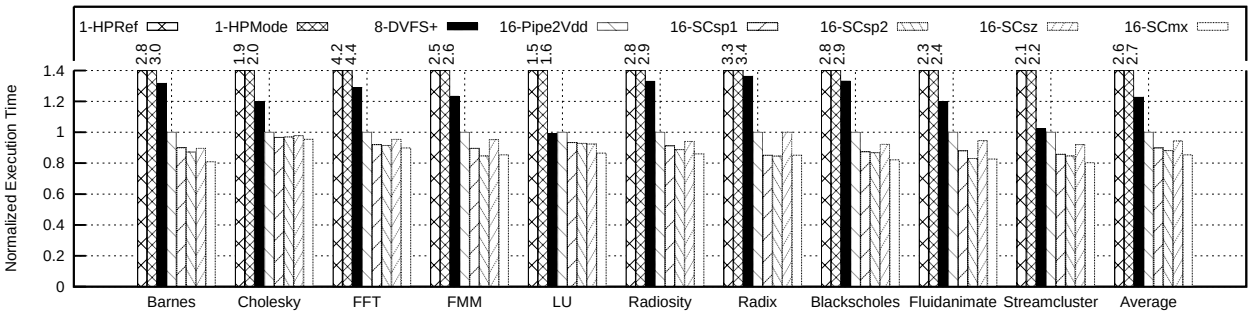

Figure 2.10: Execution time of different designs with the same power, normalized to *16-Pipe2Vdd*.

exhibit higher levels of MLP/ILP can also take advantage of bigger structures (*16-SCsz*). Overall, *16-SCmx* performs the best, as it provides a latency reduction of 2 cycles for all instructions, and also bigger structures for applications that can benefit from them.

Looking at the aggressive DVFS (*8-DVFS+*), we see that, despite having a frequency advantage, it performs much worse than *16-SCmx*. This is due to its lower number of cores. The high performance designs (*1-HPRef* and *1-HPMode*) perform even worse, and they are practically identical.

Overall, for our programs, which try to represent the load in large manycores, *16-SCmx* reduces the execution time by an average of 31% relative to *8-DVFS+*, and by 15% relative to *16-Pipe2Vdd*. These are substantial reductions.

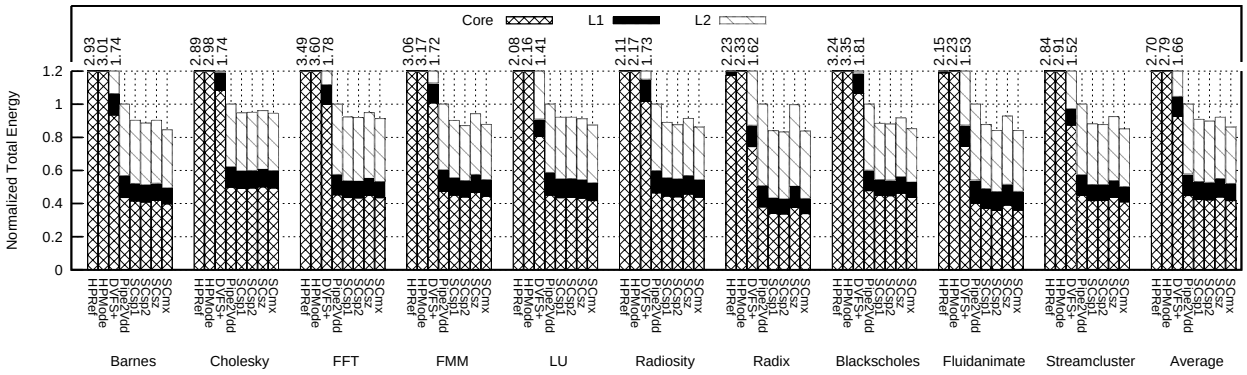

Figure 2.11: Energy consumption of different designs with the same power, normalized to *16-Pipe2Vdd*.

Figure 2.11 shows the energy consumed by all the designs. The figure is organized as the previous one, except that the bars are broken down into the contributions of core, L1, and L2. Since this experiment is done at approximately constant power, the bars in Figure 2.11

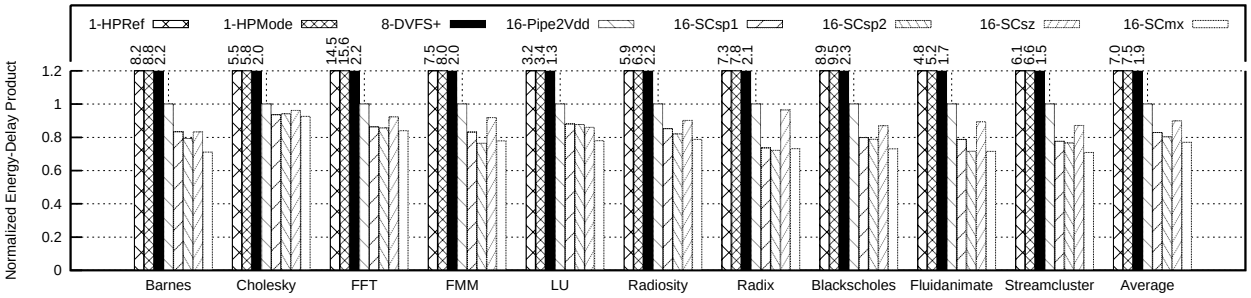

Figure 2.12: Energy-delay product of different designs with the same power normalized to *16-Pipe2Vdd*.

roughly follow those of Figure 2.10. From the figure, we also see that, going from *Pipe2Vdd* to *SC**, there are good savings in L2 energy. This is mostly leakage eliminated by finishing the application earlier — even though the storage structures in the pipeline use a higher V_{dd} in *SC**.

Overall, the best design (*16-SCmx*) reduces the energy consumption by an average of 48% relative to *8-DVFS+*, and by 13% relative to *16-Pipe2Vdd*.

Finally, we consider the ED product. Figure 2.12 shows the ED product for all the designs. We see that most of the ScalCore EEMode designs provide substantial ED product reductions. *16-SCmx* reduces the ED product by an average of 60% relative to *8-DVFS+*. We can see that not even this aggressive DVFS level can get close to the efficiencies delivered by ScalCore. Moreover, *16-SCmx* reduces the ED product by an average of 23% relative to *16-Pipe2Vdd*. Note that *16-Pipe2Vdd* is already very energy efficient, with its separation of voltage rails for pipeline logic and storage structures. These results provide a strong motivation for ScalCore.

2.8.2 Environment without Power Constraints

We now repeat the previous experiments without imposing any power constraint. All the configurations run with 16 cores and, as a result, the power consumption is very different. Figure 2.13 shows the execution time organized like in Figure 2.10. The names of the designs do not include the core count anymore. Note that, in the figure, *DVFS+* and, especially, *HPRef* and *HPMode* are faster than *SC**. Also, Figure 2.14 shows the resulting ED products of the different designs. Thanks to the faster execution of *HPRef* and *HPMode*, their ED is now substantially lower than *SCmx*. However, the ED of *DVFS+* is still higher than *SCmx*.

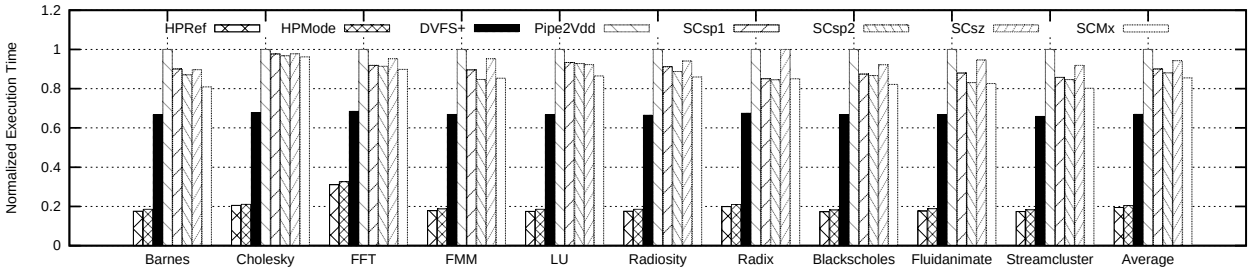

Figure 2.13: Execution time of different designs with 16 cores normalized to *Pipe2Vdd*.

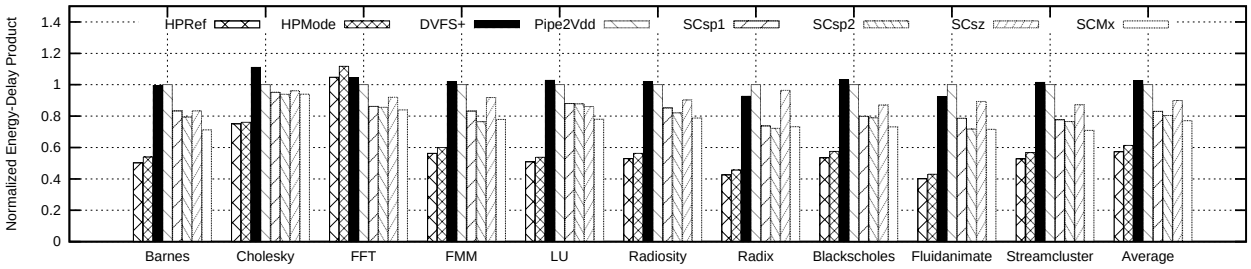

Figure 2.14: Energy-delay product of different designs with 16 cores normalized to *Pipe2Vdd*.

Unfortunately, these execution time improvements of *HPRef*, *HPMode*, and *DVFS+* come at a *staggering power cost*. Indeed, extrapolating from Figure 2.9, we can see that each *HPRef* and *HPMode* core consumes ≈16x the power of a *SCmx* core. In addition, a *DVFS+* core consumes over 2x the power of a *SCmx* core. Consequently, in practice, this much power may not be available in the chip, and some of the cores under *HPRef*, *HPMode*, and *DVFS+* may be powered down, becoming dark silicon. In fact, *HPRef* and *HPMode* cores are most effective when executing mostly-serial codes, where a few cores can use all the chip power budget. This motivates the next section.

2.8.3 Dynamically Changing Configurations

To deliver the most energy-efficient execution, ScalCore can dynamically reconfigure — e.g., between its energy efficient *SCmx* EEMode and its HPMode. Hence, in this section, we impose the same power limit as in Section 2.8.1, but allow ScalCore to dynamically reconfigure between the *SCmx* EEMode and HPMode. Recall that HPMode is penalized with a 5% f reduction relative to *HPRef*. At any point, any unused cores are power gated. We call the design *Dyn-SC*.

We run the NAS applications, where an application executes a series of parallel loops and

serial sections. We instrument the applications, so that ScalCore transitions to 16 *SCmx* cores before entering a loop, and transitions to one *HPMode* core after finishing the loop. This setup models support for program hints that directly invoke the power management unit. The analysis of more advanced interfaces to trigger the reconfiguration is beyond our scope.

We compare *Dyn-SC* to the best reconfigurable design that uses conventional processors. Such design dynamically switches between one *HPRef* core outside loops (a very high performance core without any f penalty) and 8 *DVFS+* cores in the loops (a very aggressive DVFS that sets V_{dd} to 0.6V). We call this design *Dyn-HPRef/8-DVFS+*. Such design is free of any ScalCore modifications.

As a reference, we also compare to three other designs. One is a single HPRef core all the time (*1-HPRef*). A second one is 16 SCmx cores all the time (*16-SCmx*). The third one is a reconfigurable environment that transitions between one HPMode core outside loops and 16 Pipe2Vdd cores in the loops (*Dyn-HPMode/16-Pipe2Vdd*). This is a design that includes some of the ideas of ScalCore, but not all. Specifically, it includes two V_{dd} domains in the pipeline and, hence, level converters. However, it does not include ScalCore's pipeline stage fusing or storage structure resizing.

Figure 2.15 shows the execution time, energy consumed, and ED product for the five designs. For each metric, the bars correspond to the average of the three NAS applications and are normalized to *16-SCmx*.

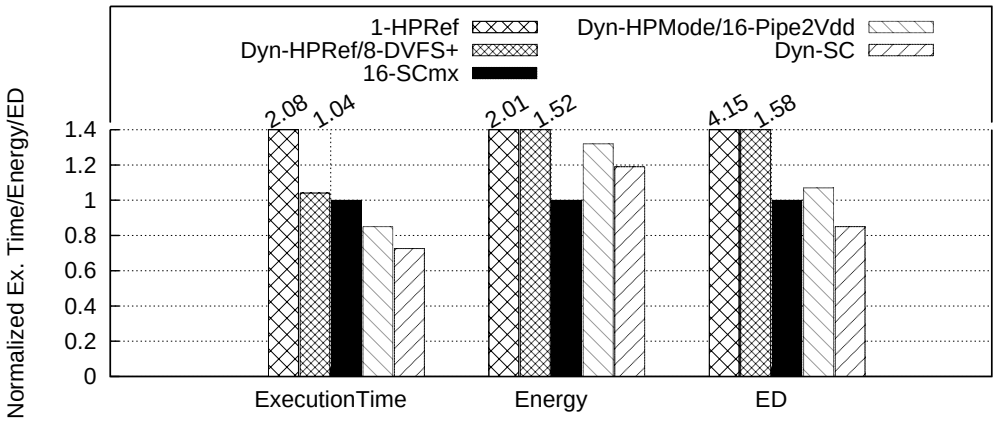

Figure 2.15: Dynamically reconfiguring ScalCore.

Dyn-SC provides the lowest execution time and the lowest ED product. It does not deliver the lowest energy because it runs a high-performance core during the serial sections. Instead, *16-SCmx* consumes less energy; however, it is slower. Compared to *16-SCmx*, *Dyn-*

SC reduces the execution time and the ED by 28% and 15%, respectively, at the cost of 19% more energy. This is a very favorable tradeoff.

Dyn-SC is much better than *Dyn-HPRef/8-DVFS+* and substantially better than *Dyn-HPMode/16-Pipe2Vdd*. The comparison of *Dyn-SC* to *Dyn-HPMode/16-Pipe2Vdd* shows the impact of ScalCore's pipeline stage fusing and storage structure resizing, over simply having two V_{dd} domains in the pipeline. We can see that *Dyn-SC* reduces the execution time, energy, and ED of *Dyn-HPMode/16-Pipe2Vdd* by 14%, 10%, and 21%, respectively. Hence, both contributions of the ScalCore design, namely dual V_{dd} domains in the pipeline, and the resulting pipeline stage fusing and structure resizing, are beneficial.

The transition overhead between modes in *Dyn-SC* is negligible because of the relatively low frequency of transitions. Specifically, *Dyn-SC* performs 6-32 transitions in 130-700 million cycles. Overall, ScalCore with dynamic reconfiguration is a very attractive design for applications that change phases.

2.9 SUMMARY

The goal of this project was to design a voltage scalable core — namely, one that can work in high-performance mode (HPMode) at nominal V_{dd}, and in a very energy-efficient mode (EEMode) at low V_{dd}. We call this core *ScalCore*. ScalCore introduces two ideas to operate energy-efficiently in EEMode. First, it applies two low V_{dd}s to the pipeline: one to the logic stages (V_{logic}) and a higher one to the storage-intensive stages. Second, it increases the low V_{dd} of the storage-intensive stages (V_{op}) even further and exploits the speed differential to the logic ones by either fusing storage-intensive stages or increasing the size of storage structures in the pipeline. Simulations of 16 cores show that a design with ScalCores in EEMode is much more energy-efficient than one with conventional cores and aggressive DVFS: for approximately the same power consumption, ScalCores reduce the average execution time of programs by 31%, the energy consumed by 48%, and the *ED* product by 60%. In addition, dynamically switching between EEMode and HPMode is very effective: it reduces the average execution time and *ED* product by an additional 28% and 15%, respectively, over running in EEMode all the time.

CHAPTER 3: HETCORE: TFET-CMOS HETERO-DEVICE ARCHITECTURE FOR CPUS AND GPUS

3.1 INTRODUCTION

In pursuit of higher energy efficiency, researchers try to lower the operating voltage of CMOS transistors, as we discussed in Chapter 2. Unfortunately, CMOS is, intrinsically, a poor switch [9]. If one reduces the threshold voltage as the supply voltage goes down, leakage power soars, negating the energy savings.

In this work we explore Steep Slope (SS) devices, which are a class of devices that are much better switches [9]. They can turn-off a transistor hard with a small decrease in the voltage applied. This makes these devices attractive when operated at low voltage: they both consume low dynamic energy while working, and leak little. Among the various SS devices being explored, *Tunneling Field-Effect Transistors (TFETs)* [6] are one of the most promising [1], thanks to manufacturing feasibility and ability to integrate with current FinFET CMOS devices.

While TFETs operate efficiently at low voltage, they do not scale well with increasing voltage. Their performance saturates beyond a certain voltage. Hence, they cannot replace CMOS transistors when high performance is needed. Instead, the best course to execute workloads with both high performance and high energy efficiency may be to combine CMOS and TFET transistors.

CMOS and TFET devices can be integrated in the same chip [10, 11, 59, 60]. Circuits with a combination of CMOS and TFET transistors have been used to build SRAM cells [61, 12], voltage reference circuits [62], level converters [63], multiplexers [64], 32-bit adders [64], power management circuits [65], analog circuits [66], and benchmark circuits [67].

Integration at such fine granularity provides an opportunity for system designers to explore novel architectures. Prior work has proposed a heterogeneous multicore with some CMOS cores and some TFET cores [68, 69, 70]. The authors migrate threads across the cores to attain most efficient executions. This is an exciting approach, although it is limited in that a given core delivers either high performance or energy efficiency, but not both.

In this work, our goal is to go one step further and design a core that, ideally, is as energy-efficient as a TFET core, and provides as much performance as a CMOS core. For this, we judiciously integrate both TFET units and CMOS units in the same core, effectively creating a *hetero-device* core. We call it *HetCore*, and present CPU and GPU versions.

At their optimal operating voltage levels, TFET structures switch at half the speed of CMOS ones, but consume about 8x lower power. This high-level tradeoff provides guidance

to select the TFET and CMOS units. TFETs should be used in units that consume high power under CMOS, are amenable to pipelining or are not very latency sensitive, and use enough area to amortize the additional design effort.

HetCore powers CMOS and TFET units at different voltage levels, so they operate at optimal conditions. However, all units are clocked at the same frequency. To make this feasible, HetCore reduces the work done by each TFET pipeline stage, effectively giving to a TFET unit more pipeline stages than an equivalent CMOS unit would have.

In this work, we start by proposing a simple HetCore design called *BaseHet*. While BaseHet reduces energy consumption substantially, it is slow. Hence, we improve it by adapting a few known micro-architecture optimizations, enabled by the presence of the TFET units. The result is the better-tuned *AdvHet* design.

The alternative of simply using high-V_t CMOS transistors in the units that are candidates for TFET implementation is not as good a design. The reason is that high-V_t CMOS transistors consume higher dynamic energy and leak more than TFET transistors. In addition, applying the HetCore micro-architecture optimizations to a CMOS core is of little benefit. The reason is that such core is already highly tuned without the optimizations.

Overall, the contributions of this work are:

- The concept of a hetero-device TFET-CMOS core architecture for high performance and energy efficiency (HetCore).
- The design of the AdvHet core for CPUs and GPUs, which judiciously integrates CMOS and TFET units, and customizes known micro-architecture optimizations.
- An evaluation of BaseHet and AdvHet.

In the following sections, we first provide a background on TFETs and then analyze the TFET devices from an architectural perspective. Later we discuss the BaseHetand AdvHetdesigns, followed by an evaluation of the designs proposed.

3.2 BACKGROUND

3.2.1 Tunneling Field-Effect Transistors (TFETs)

To improve energy efficiency substantially, we need devices that can operate at low voltage (V_{dd}), and that can switch between ON and OFF conditions with little V_{dd} changes. Ideally,

the ON and OFF currents of a device should be separated by four orders of magnitude. Conventional CMOS transistors are inherently limited to needing 60mV to increase the current tenfold — i.e., they need at least a change of 240mV to go from OFF to ON conditions.

The class of devices that have a slope higher than 60mV per decade are called Steep subthreshold Slope (SS) devices. Among the various SS devices being explored, Tunneling Field-Effect Transistors (TFETs) are one of the most promising [9, 2, 1, 6]. They consume low power and have a steep slope. Moreover, they are the closest to being realized industrially, thanks to their manufacturability and ability to integrate with current FinFET-based CMOS devices.

TFETs' steep slope is the result of electron flow being facilitated through a band-to-band tunneling process, as opposed to through a transport channel like in MOSFETs. The materials used in TFETs range from the usual Group IV elements like Si and Ge, to Group III-V materials like InAs, GaSb, InGaAs, and AlGaSb [9]. Various TFET devices have been proposed over the last decade that have successively improved their characteristics.

TFETs are typically classified into HomoJunction TFET (HomJTFET) and HeteroJunction TFET (HetJTFET), based on the materials used for source and drain. A HomJTFET uses the same materials for the source and the drain. However, the ON current is low and, hence, this device exhibits low performance. A HetJTFET uses a different material for the source and the drain — e.g., GaSb for source and InAs for drain. The materials are chosen to allow for a higher ON current and an extremely low OFF current.

Figure 3.1 compares the I-V characteristics of a HetJTFET and a MOSFET transistor. As we can see, HetJTFET has a higher slope than MOSFET. HetJTFET performs better than MOSFET at low V_{dd}, but stops scaling beyond ≈0.6V, when the curve saturates. For higher V_{dd}, MOSFET performs better. As a result, HetJTFET cannot be used as a replacement of MOSFET for high-performance designs.

3.2.2 CMOS-TFET Integration

The structure of HomJTFET is very similar to that of a CMOS FinFET. Hence, it is possible to manufacture both of them using the same fabrication process with minor changes. For example, Huang et al. [71] have recently fabricated Complementary HomJTFET (C-TFET) devices in a standard CMOS foundry, showcasing the readiness for high-volume production and, from an architect's perspective, the feasibility of a hybrid CMOS-TFET system.

There has also been extensive work on fabricating HetJTFET on standard CMOS foundries. For example, InAs-Si HetJTFETs have been fabricated on a silicon substrate [59, 60]. The

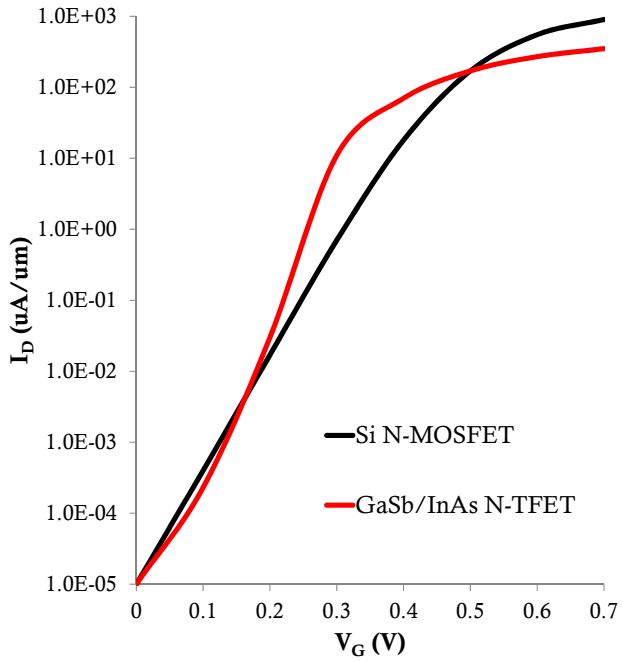

Figure 3.1: I_D- V_G characteristics of N-HetJTFET and N-MOSFET based on data from Intel [6].

compatibility of CMOS and TFET process flows has been shown by a number of groups, both through simulation and through fabrication [10, 11, 62, 72]. Recently, mixed MOSFET-HetJTFET SRAM cells and corresponding design layout rules to integrate them at device level have been proposed [12, 61]. Moreover, circuits with a combination of CMOS and HetJTFET transistors have been used to build level converters [63], multiplexers [64], 32-bit adders [64], power management circuits [65], and analog circuits [66]. There is also substantial ongoing research on improving HetJTFET performance and building complementary devices [73, 74, 75, 76].

3.2.3 System Architectures with CMOS and TFET

Integration at such fine granularity provides an opportunity for system designers to explore system architectures with CMOS and TFET. Past work has proposed a heterogeneous multicore with some CMOS cores and some TFET cores [68, 69, 70]. A core provides either high performance or energy efficiency, but not both at the same time. The authors propose various techniques to manage the migration of threads across the different types of cores. In our work, we go beyond in that we judiciously integrate both TFET units and CMOS units

Table 3.1: Characteristics of CMOS and TFET technologies at 15nm, using data from [1, 2].

Parameter		Si-CMOS	HetJTFET	InAs-CMOS	HomJTFET
Supply voltage (V)		0.73	0.40	0.30	0.20
Performance	Transistor switching delay (ps)	0.41	0.79	3.80	6.68
	Interconnect delay per transistor length (ps)	0.18	0.42	2.50	3.60
	32bit ALU delay (ps)	939	1881	9327	15990
Energy	Transistor switching energy (aJ)	32.71	7.86	3.62	1.96
	Interconnect energy per transistor length (aJ)	10.08	3.03	1.70	0.76
	32bit ALU dynamic energy (fJ)	170.1	43.4	20.5	10.8
Power	32bit ALU leakage power (uW)	90.2	0.30	0.14	1.44
	ALU power density (W/cm^2)	50.4	5.1	0.6	0.2

in the same core, effectively creating *hetero-device* CPUs and GPUs.

3.3 ARCHITECTURE IMPLICATIONS

CMOS remains the choice for high-performance systems, while operating at high V_{dd}. However, at low V_{dd}, the performance and energy efficiency of TFET far exceed those of CMOS. To aid in the analysis, Table 3.1 compares the performance, energy, and power of four types of devices at 15nm: Silicon CMOS (*Si-CMOS*), HetJTFET, HomJTFET, and *InAs-CMOS*. The latter is a futuristic MOSFET built out of InAs (a Group III-V material) that can operate at low V_{dd}. InAs-CMOS would use the same approach as TFET to integrate with Si-CMOS. In HomJTFET, the source and drain use InAs, while in HetJTFET, they use GaSb and InAs, respectively. The table compares each device at its most cost-effective V_{dd}: 0.73V for Si-CMOS, 0.40V for HetJTFET, 0.30V for InAs-CMOS, and 0.20V for HomJTFET. The data is obtained from Nikonov and Young [1, 2]. Similar numbers have been reported elsewhere [68, 77].

3.3.1 Performance

Row 2 of Table 3.1 shows that the switching delay of a HetJTFET, InAs-CMOS, and HomJTFET transistor is about 2x, 10x, and 16x longer, respectively, than the switching delay of a Si-CMOS one. The next row compares the interconnect delay for a distance equal to the transistor length. Since the dimensions of MOSFET and TFET transistors are similar, these delays are directly comparable. These delays follow similar trends as the transistor switching delays. Finally, Row 4 shows the delay of a 32bit ALU operation, which includes both transistor switching and interconnect delay. We can see that the ratios are about the same as for the transistor delays.

Our goal is to implement some of the units in a Si-CMOS CPU or GPU core in TFET technology. Mixing Si-CMOS and HetJTFET units in the core is feasible, as a 2x differential speed can be handled by keeping a single frequency, but pipelining the HetJTFET unit at least twice as deeper. An example can be an HetJTFET functional unit in a CMOS core. However, including InAs-CMOS or HomJTFET units would be too challenging: their speed differential would require unrealistic 10x and 16x deeper pipelines, which would be too disruptive. HomJTFET and InAs-CMOS are better suited for ultra-low power applications in wearables or IoT devices. Note also that, since Si-CMOS and HetJTFET operate at different V_{dd}, we need level converters when we go from a HetJTFET to a Si-CMOS unit. These level converters can be integrated with pipeline latches [55].

3.3.2 Energy and Power

Rows 5 and 6 of Table 3.1 show the switching energy of a transistor, and the interconnect energy for a distance equal to the transistor length for all the technologies. The next row shows the dynamic energy of a 32bit ALU operation, which includes both transistor switching and interconnect energy. We see that a Si-CMOS 32bit ALU operation consumes about 4x, 8x, and 16x as much energy as with HetJTFET, InAs-CMOS, and HomJTFET, respectively. Since HetJTFET is 2x slower than Si-CMOS, the operation with HetJTFET consumes about 8x less power.

Overheads like separate voltage rails for CMOS and TFET units, and timing guardbands reduce the power savings of TFETs. Our conservative estimate of overheads (Section 3.5.2) shows that HetJTFET still consumes 6.1x lower power than Si-CMOS. However, in this work, we impose even stricter guardbands, and evaluate TFETs conservatively assuming that they provide *only* a 4x power savings over CMOS.

The best property of HetJTFET transistors is their low leakage power. Row 8 shows the leakage power of a 32bit ALU. A HetJTFET ALU consumes about 300x lower leakage power than a Si-CMOS ALU. In practice, the reduction is not so high. This is because, in CMOS processors, many logic structures not in the critical path use high-V_t CMOS transistors to reduce leakage. For example, commercial processors like AMD Ryzen [78] and prior designs [79] contain about 60% high-V_t transistors. Such transistors consume about the same dynamic energy as the regular-V_t CMOS transistors assumed in Table 3.1. However, they consume less leakage power.

Specifically, using a Synopsis library for 28/32nm technology, we find that they consume 25-30x less leakage power than regular-V_t transistors. This is in line with numbers reported in prior work [79, 80]. Using these numbers, the leakage power of a typical Si-CMOS unit

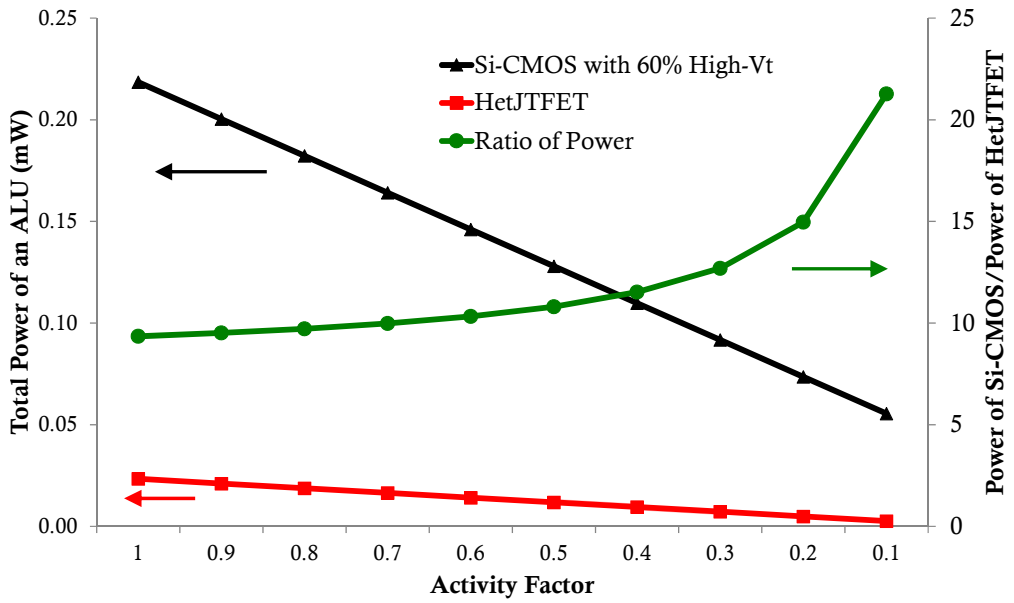

Figure 3.2: Total power consumption of a Si-CMOS ALU and a HetJTFET ALU with varying activity factors.

is only about 42% of the value in Table 3.1. This agrees with dual-V_t designs of both logic and SRAM cells in the literature [81, 82, 83, 84]. Overall, using this figure, a HetJTFET ALU consumes 125x lower leakage power than a dual-V_t Si-CMOS ALU.

6T and 8T HetJTFET-based SRAM cells have been proposed by some authors [85, 86, 87]. They show that the leakage power of these cells is several hundred times lower than a competitive Si-CMOS SRAM cell [85].

Overall, HetJTFET units provide over two orders of magnitude savings in leakage power compared to Si-CMOS. In the worst case, when 100% of the Si-CMOS transistors are high-V_t, the savings reduce to a still sizable 10x. Therefore, we will use HetJTFET devices in logic and memory structures of the core where leakage power dominates.

Finally, row 9 shows the power density of an ALU. A Si-CMOS design has a 10x higher power density than a HetJTFET design. This indicates that HetJTFETs will be a better choice for units that need high computational density, such as SIMD FPUs.

3.3.3 Activity Factor

Because of their low leakage power, HetJTFETs are a good choice for units that have a low activity factor. When there is no activity, the HetJTFET implementation consumes very little, while the Si-CMOS one still consumes a large leakage power. In such a unit, the ratio

of power consumed by the Si-CMOS implementation over the HetJTFET implementation keeps increasing the lower the activity factor is.

Figure 3.2 depicts the total 32bit ALU power of both designs and the ratio of powers, as the activity factor decreases. An activity factor of 1 means that the ALU is used every cycle. In the figure, the Si-CMOS ALU is composed of 60% high-V_t transistors in noncritical paths to minimize leakage. We see that, as activity decreases, the HetJTFET implementation becomes relatively more attractive.

3.3.4 Dynamic Voltage-Frequency Scaling (DVFS)

We envision a core with two V_{dd}, one for the Si-CMOS units (V^0_{CMOS}), and one for the HetJTFET units (V^0_{TFET}). All units are clocked at a single frequency (f_0). To make this possible, we reduce the work that each TFET pipeline stage does, giving at least twice as many pipeline stages to the TFET unit as a CMOS unit would have.

We also envision the ability to apply DVFS. When higher performance needed, both Si-CMOS and HetJTFET units increase their V_{dd}; when more energy efficiency is needed, both decrease their V_{dd}. This means that we need to find pairs of voltages (V^i_{CMOS}, V^i_{TFET}) such that the Si-CMOS circuit is always 2x faster than the HetJTFET circuit to do equivalent work. From the previous discussion, these pairs are such that, if V^i_{CMOS} attains f^i, then we need a V^i_{TFET} that would attain $f^i/2$ to do the same work per pipeline stage for the HetJTFET units.

One challenge is that each technology has a different V_{dd}-frequency curve, with a different slope and a different range. These curves are shown in Figure 3.3. We generated the Si-CMOS curve from [19], and the HetJTFET curve from [6]. In the curves, we show V^0_{CMOS}=0.73V, V^0_{TFET}=0.40, and f_0=2GHz.

If we want to increase Si-CMOS's V_{dd} by ΔV^i_{CMOS} to attain f^i, we need to increase HetJTFET's V_{dd} by an amount ΔV^i_{TFET} that is *different* than ΔV^i_{CMOS}. It is an amount that can deliver $f^i/2$ for the HetJTFET units to do the same work per pipeline stage. Given that the slope of the HetJTFET curve is less steep, ΔV^i_{TFET} will typically be larger than ΔV^i_{CMOS}. For example, to turbo-boost to a f^1=2.5GHz, we need ΔV^1_{CMOS}=75mV and ΔV^1_{TFET}=90mV.

3.3.5 Process Variation

The main source of variation in both TFET and MOSFETs is the work function [88]. The extent of work function variation in TFETs and MOSFETs is similar, both in logic and

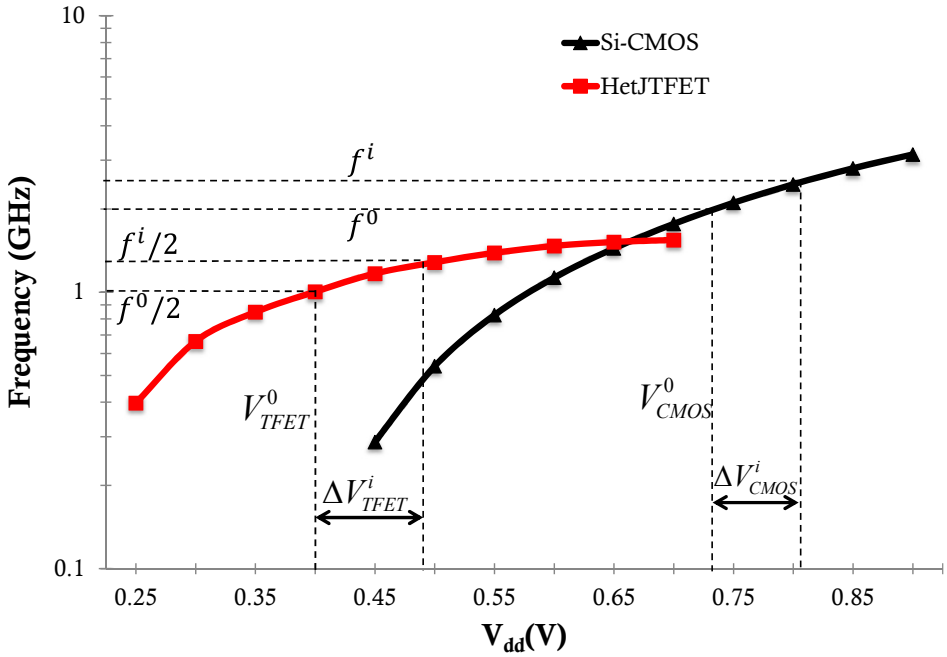

Figure 3.3: V_{dd}-freq. curves for Si-CMOS and HetJTFET.

SRAM [88, 86]. While the variation affects both I_{off} and I_{on}, the impact is higher on I_{off} for TFET, and I_{on} for CMOS. This is due to the steeper slope of the I-V curve (Figure 3.1) close to the OFF state in TFETs, and in the ON state in CMOS.

As indicated by Avci et al. [88], the performance of the transistors lost to variation can be reclaimed by increasing the V_{dd} of both Si-CMOS and HetJTFET. We show in Section 3.7 that the result is that HetJTFET loses a small fraction of its energy savings relative to Si-CMOS.

3.3.6 Area Consumption

A HetJTFET transistor has dimensions similar to a Si-CMOS transistor. Further, the contacted gate pitch, and the pitch of the two lowest metal layers (MP0 and MP1) are the same in both CMOS and TFET devices [89]. The fact that HetJTFETs have asymmetric source and drain materials does impose some layout constraints when placing transistors close to each other. However, a recent study [89] compares the area of standard library cells of vertical HetJTFETs to FinFETs and finds that, for the technology node of 15nm considered in this work, the areas are similar. For older technology nodes, the HetJTFET implementations occupy more area than the FinFET ones, while for future, smaller technology nodes, it is expected that HetJTFETs will have an area advantage over FinFETs.

3.4 HETCORE ARCHITECTURE

Our goal is to design a hetero-device core architecture that integrates CMOS and TFET devices, and that, ideally, is as energy efficient as a TFET implementation and provides the performance of a CMOS implementation. We call the architecture *HetCore*, and provide CPU and GPU designs.

3.4.1 Main Idea

HetCore takes a high-performance CMOS CPU and GPU, and selectively replaces some units with TFET implementations. The TFET units are supplied a V_{dd} (V_{TFET}) that is lower than that of the CMOS units (V_{CMOS}). The TFET units are slower than the CMOS units. This is because TFET devices take about 2x longer to switch than CMOS devices.

HetCore clocks the TFET units at the same frequency as the CMOS units. This is made possible by reducing the work that each TFET pipeline stage does, and at least doubling the number of pipeline stages of the operation. Keeping a single frequency domain in the core reduces the complexity of the design, and eliminates any associated clock synchronization overheads. Overall, through careful selection of TFET units, we substantially reduce the energy consumption of the CPU and GPU. However, we suffer performance degradation. We name this design *BaseHet*.

Since BaseHet is slow, we then introduce mitigation techniques to recover some of the performance lost. These mitigation techniques are enabled by one of the two following effects. First, the slowdown caused by TFET structures presents new opportunities for micro-architectural optimization. Second, TFET structures present different power-performance tradeoffs than CMOS ones and, hence, require re-evaluation of certain design decisions. We call this final design *AdvHet*.

3.4.2 BaseHet Design

An ideal unit to replace with a TFET implementation has the following traits:

- *Is Highly Power Consuming.* The power consumed by the CMOS variant should be significant compared to the total power of the CPU or GPU. Otherwise, any savings will be small — or even negative, due to the program slowdown.

- *Is Amenable to Pipelining and/or is Not Very Latency-Sensitive.* The longer latency

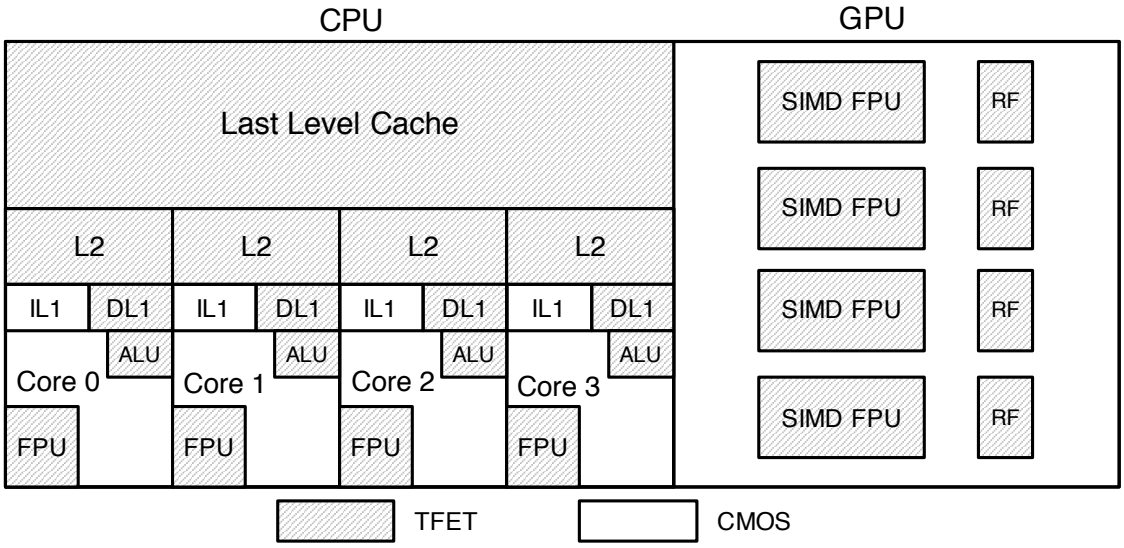

Figure 3.4: TFET-based units selected for the BaseHet design.

induced by TFET devices should not hurt the overall performance too much.

- *Uses a Large Area.* To amortize the design effort, it is preferable that the unit be relatively large. We impose this constraint for BaseHet, and later relax it slightly for AdvHet.

We now discuss the candidate units that have such traits in a CPU and GPU. They are shown in Figure 3.4.

Floating-Point Units in the CPU and GPU Floating-Point Units (FPUs) in both the CPU and the GPU (SIMD FMA units) are power hungry. They are also pipelined for multiply and add operations. While divide and a few other complex operations are not typically pipelined in the CPU, such operations are less common in most applications. In addition, floating-point intensive applications are known to exhibit high Instruction Level Parallelism (ILP). Hence, deeper-pipelined FPUs can still attain high levels of occupancy. As a result, moving to TFET FPUs, and making their pipeline deeper should have modest impact on performance. In case of a SIMD FMA unit in the GPU, due to the inherent throughput-oriented nature of the programs, it is even easier to fill the pipeline with other threads and minimize the performance impact. The FPUs, therefore, are ideal candidates for moving to a TFET design.

ALUs in the CPU The ALUs in a CPU core consume substantial dynamic power and can be pipelined. The more complicated ALU operations such as multiply and divide are usually pipelined. Pipelining an ALU, however, will have a negative impact on the performance, especially in the case of branch mispredictions. Despite the slowdown caused, pipelined ALU designs have been employed in commercial microprocessors since Alpha 21064 to reach high frequencies. Therefore, even though pipelining the ALUs has a performance impact, as we show in our evaluation, the energy savings of implementing the ALUs in TFET is attractive.

Caches in the CPU Caches contribute the majority of the leakage power consumption in a CPU. Since TFETs leak very little, even compared to high-V_t transistors, caches are excellent candidates to move to TFET. Out of the three levels of caches in a modern hierarchy, the latency of L3 has the least impact on performance. Hence, L3 can definitely be implemented in TFET. The latency of L2 has impact on some programs, but it is limited. Note that out of an 8-10 cycle round trip to L2, only 3-5 cycles are actually spent accessing L2. Therefore, by moving to a TFET L2, the additional latency of L2 access is only 3-5 cycles.

In the case of L1, an increase in access latency clearly causes performance degradation. This is especially true for the instruction cache (IL1). Any latency increase of the data cache (DL1) is unwanted as well, but it can be hidden partially in an out-of-order core with enough ILP. The cache accesses are pipelined and may be distributed among multiple banks, allowing multiple accesses to proceed in parallel. Finally, both leakage and dynamic power consumption in DL1 are significant. Hence, even though we induce a performance loss, we move DL1 to TFET.

Register File in the GPU The Register File (RF) in a GPU is big and consumes significant power (up to 10% of the GPU power [90]). RF access can also be pipelined by partitioning it into multiple stages, such as data array access and source drive [91]. The additional latency increase results in a performance degradation, which may be hidden in throughput-oriented workloads. Hence, the RF in GPUs is also a good candidate for implementation in TFETs.

3.4.3 AdvHet Design

BaseHet improves the energy efficiency over a pure-CMOS design at a performance cost. In AdvHet, we adapt known performance-improvement techniques to BaseHet and recover most of the performance lost. BaseHet exposes an opportunity for such techniques by changing the

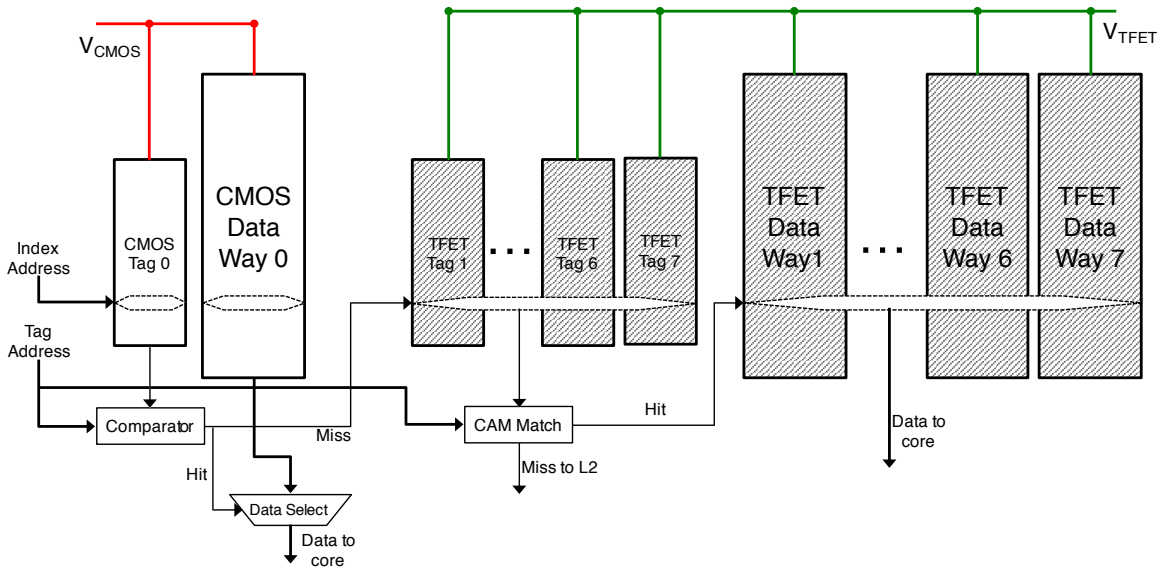

Figure 3.5: Schematic design of an *Asymmetric Cache*.

balanced power/performance design of the baseline CMOS. First, the slowdown due to the TFET units provides avenues for previously-suboptimal micro-architectural design choices. Second, equipped with the lower power consumption of TFET units, a small power penalty might now be a good tradeoff for a big performance gain. This may sometimes result in overall energy savings as well, due to the corresponding reduction in leakage energy.

Asymmetric DL1 Cache The DL1 cache access latency is critical to the performance of most applications. By using a TFET DL1, BaseHet doubles the round trip to 4 cycles from 2 cycles in baseline. We present the design of an *Asymmetric Cache* (Figure 3.5) to alleviate some of the latency penalty introduced by TFETs.

The goal of the asymmetric cache is to reduce the hit latency. To accomplish this, the asymmetric cache partitions the ways in an associative cache. One way is implemented in CMOS (FastCache), and the rest of the ways in TFET (SlowCache). A request from the processor checks the FastCache first. A hit is satisfied in 1 cycle. A miss sends the request to the SlowCache, where a hit takes 4 additional cycles. Hence, the hit latency is either 1 cycle (for FastCache hits) or 5 cycles (for SlowCache hits). Such a tradeoff is attractive in AdvHet because, otherwise, all hits would take 4 cycles. However, it is not as attractive in the baseline CMOS where hits take 2 cycles.

The Most Recently Used (MRU) line from each set is moved to the FastCache to improve the hit rate. The FastCache is partitioned into two banks with two read/write ports to facilitate the data transfer between FastCache and SlowCache. CACTI [52] analysis shows

that the access latency of the FastCache is about one third of the base 32KB DL1.

The access energy of the FastCache is small. On average, this approach of accessing the FastCache first and, potentially, then accessing the SlowCache, and even moving a line between caches saves energy over accessing a whole CMOS DL1, or accessing a whole TFET DL1. In fact, prior work has looked at using similar cache designs for energy reduction [92, 93, 94]. Overall, compared to a whole TFET DL1, the asymmetric cache improves performance and reduces energy consumption over the whole program execution.

Dual-Speed ALU Cluster Increasing the latency of an ALU degrades the overall performance. Notably, it prevents the back-to-back issue of dependent instructions, and also increases the branch misprediction penalty. We mitigate the impact of the first issue by keeping one of four ALUs in the core implemented in CMOS, hence creating a dual-speed ALU cluster. By identifying appropriate producer-consumer instructions and executing them on the CMOS ALU, we enable back-to-back issue of these instructions.

The algorithm to identify such producer-consumer instructions in AdvHet has the following objectives. First, it minimizes the situations where back-to-back dependent instructions are sent to a TFET ALU. Second, it maximizes the power savings by steering the majority of the instructions to a TFET ALU. Finally, it balances the overall utilization of the TFET ALUs and the CMOS ALU. Note that the penalty of mis-steering is only to increase the latency of an ALU operation from 1 to 2 cycles. Due to this reason, the objective of our scheme is different from some of the prior work on identifying the most critical path [95]. A simple algorithm suffices for us.

Dual-speed clusters have been studied previously as a mechanism to reduce power consumption [96, 97, 98]. In our design, we employ a simplified version of the Generation Time Gap metric [97] for steering instructions to slow and fast clusters. Specifically, for each instruction in the *dispatch* stage, we check if any consumer is present in a small window of instructions behind the current one. As the additional latency of a TFET ALU over a CMOS ALU is one cycle, we set the window length as the number of instructions that can be issued in one cycle — i.e., the core's issue width. Intuitively, if a consumer exists in this small window, then executing the current instruction on the CMOS ALU may benefit the consumer. Note that in an out-of-order machine, this is not a necessary condition, and we may mis-steer occasionally. Such scenario could be avoided by performing the check in the issue stage. However, doing so would interfere with the issue process and add to the complexity of the issue stage. Hence, steering is best performed in the dispatch stage, in parallel to its current functionality. This minimizes the additional complexity.

Register File Cache in the GPU Register file access is in the critical path of an arithmetic operation in a GPU. In throughput-oriented workloads, the compiler could customize the binary to hide the additional latency of accessing a TFET register file. However, this would likely not be enough. Therefore, to reduce the access latency, we instead use a register file cache, with 6 entries per thread. This is a very small subset of the 256 registers per thread in the GPU that we model (based on AMD's Southern Islands). The access latency of this small cache is only one cycle.

To maximize the utility of this register file cache and avoid thrashing, we only cache registers that we write. This is because as much as 40% of the writes are consumed by reads within a few instructions [91]. Hence, caching only the writes provides good locality for reads and minimizes thrashing. In our simulations, we observe that this cache is able to recover up to 70% of the performance loss caused by the increase in the register file access latency.

The register file cache was originally proposed to reduce the power consumption of GPUs [91]. In AdvHet, however, we also reap the benefits of a faster register access enabled by such cache. The opportunity for reducing latency is much higher in HetCore than in a CMOS design, in a manner similar to the asymmetric cache.

Discussion The deeper pipelining of the FPUs in BaseHet unbalances the core pipeline. To keep such deeper-pipelined FPUs utilized, we need to sustain more inflight instructions. Hence, we increase the sizes of the FP register file and ROB appropriately. Note that a larger ROB size will also aid in some non FP-intensive applications.

Other optimizations are possible, but we do not consider them due to questionable tradeoffs. For example, there are FPU designs that reduce latency but increase area and/or power [99]. This includes different encoding schemes (Booth 2 versus Booth 3), combining networks (Wallace tree versus OS1), and multiplier types (CMA versus FMA). For example, a CMA design would reduce the latency over an FMA unit when forwarding the output to another multiply/add operation. However, it would take up 15% more area and consume 20% more power. One could also customize the GPU compiler to hide some of the additional FPU latency. We leave the analysis of these techniques to future work.

3.4.4 Summary of the Designs

Table 3.2 shows a summary of the design modifications for HetCore. In the BaseHet design, we implement in TFET the following structures: FPUs, ALUs, DL1, L2, and L3 in a CPU; and SIMD FPUs and register file in a GPU. In the AdvHet design, we additionally

Table 3.2: Design modifications for HetCore.

Design	CPU Structures	GPU Structures
BaseHet	FPUs, ALUs, DL1, L2, and L3 in TFET	SIMD FPUs and RF in TFET
AdvHet	BaseHet + asymmetric DL1 cache + dual-speed ALU + larger ROB and FP RF	BaseHet + register file cache

add the following structures: the asymmetric DL1 cache, the dual-speed ALU cluster, and a larger ROB and FP register file in a CPU; and the register file cache in the GPU.

3.5 IMPLEMENTATION CONSIDERATIONS

3.5.1 Dual Voltage Rails and Level Converters

HetCore integrates CMOS and TFET units operating at different V_{dd} inside a CPU and GPU. Hence, it requires provisioning for separate V_{dd} rails for the two groups of units, and level converters between such units. More specifically, each pipeline stage is powered at a single V_{dd}. This is shown in Figure 3.6, which shows two TFET stages in between two CMOS stages. The former are powered with the lower V_{TFET}, while the latter with the higher V_{CMOS}. A given stage includes both data-path and control-path signals.

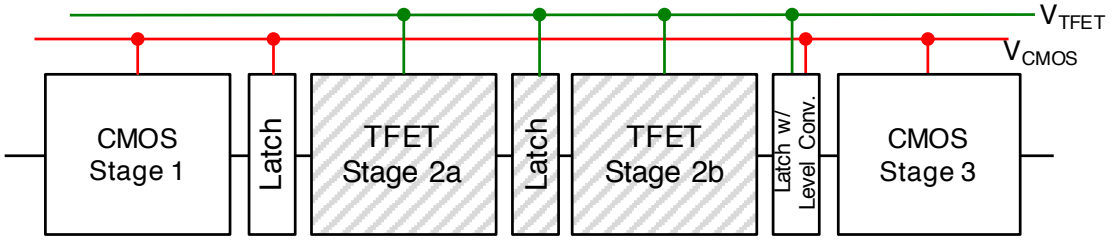

Figure 3.6: HetCore dual voltage rail design.

Latches between two same-device stages are implemented with the same device type. Latches between two different-device stages are implemented in CMOS, and are powered at V_{CMOS}. Additionally, those latches that connect a TFET stage to a CMOS stage need to perform up-conversion. Hence, as shown in Figure 3.6, they are augmented with a level converter and take both V_{dd} levels [55, 63].

HetCore employs a level converter design based on Ishihara et al. [55], which is implemented as part of a latch. This design uses pulsed half-latch level converting flip-flops, which are shown to be more efficient in terms of energy-delay and area when compared to asynchronous level conversion. Moreover, the level converter follows the hybrid CMOS-TFET organization that has recently been proposed by Lanuzza et al. [63].

The fact that the whole pipeline uses a single frequency domain keeps the design simpler. There is no need to perform synchronization across stages. The presence of multiple V_{dd} domains requires careful design of the clock tree, but it has been shown that such tree can be generated with very little skew (<0.5% of the clock cycle) [100].

3.5.2 Overheads of the Multi-V_{dd} Substrate

The multi-V_{dd} substrate of HetCore introduces delay, area, and power overheads. The first issue is the dual V_{dd} rails themselves. Their main overheads are the additional area they take, and the need to customize their layout/routing, as automatic tools may not be able to handle them. One implementation of dual rails [56] estimates the area cost to be ≈5% of the core.

The second issue is the level converters. They require carefully managing the clock skew and timing across V_{dd} domains. Moreover, they add a few gates to the critical path of the pipeline stage. Based on Ishihara et al.'s [55] work, we estimate a delay impact of ≈5%. The additional area and power of the level converters is negligible [55].

A third issue is the deeper pipelining of the TFET structures. It introduces delay in two ways. First, the work in a pipeline stage cannot usually be sliced into two equally-sized portions; instead, one portion takes longer. We estimate that this effect makes TFET stages ≈5% longer than ideal. Second, TFET latches are themselves slower that CMOS ones. Given that latches account for ≈10% of a stage's latency [101], we add one extra 10% stage delay due to slow TFET logic. The latches added for the deeper pipelining also introduce a power overhead of ≈10% of the stage power [101].

The fourth issue is that HetCore introduces design complexity and verification costs, which are hard to quantify.

In summary, TFET stages in HetCore suffer from a delay of up to 15% — resulting from 5% due to unequal work partitioning between stages, and 10% due to a level converter or a slow TFET latch (but not both). Since we do not want to penalize the frequency of the pipeline, HetCore raises V_{TFET} slightly over its value in Table 3.1, to meet CMOS timing constraints. Specifically, to recover this 15% delay, V_{TFET} is increased by 40mV. As a result, the power consumption of TFETs increases by 24%, which lowers the overall dynamic power

savings of moving from CMOS to TFET from 8× to about 6.1×. Further, to be very conservative, in the rest of the work, we set the overall reduction in dynamic power when moving from CMOS to TFET to be only 4x.

3.6 EVALUATION SETUP

We evaluate HetCore using the Multi2Sim [102] architectural simulator, which models CPUs and GPUs. We model a processor with 4 CPUs and 1 GPU. Each CPU is 4-wide and out of order. The GPU hardware is modeled after the AMD Southern Islands, with 8 compute units. Table 3.3 shows the detailed parameters of the modeled CPU and GPU. We obtain the power numbers by using the HP-CMOS process of McPAT [38] and GPUWattch [103] for the CPU and GPU, respectively. Recall that TFET units now operate at a V_{dd} of 0.440 V, and CMOS units at 0.730 V. At these voltages, the frequency reached by all-CMOS and all-TFET CPUs is 2GHz and 1GHz, respectively. While the dynamic power consumption of TFET units is 6.1x lower than HP-CMOS ones, we conservatively use a 4x factor. Further, to calculate the TFET leakage power, we conservatively assume that it is only 10x lower than the CMOS leakage power, as if all the CMOS transistors were high-V_t devices.

Table 3.3: Parameters of the simulated architecture for the evaluation of HetCore.

Parameter	**Value**
CPU Hardware	4 out-of-order cores, 4-issue each, 2GHz
INT/FP RF; ROB	128/80 regs; 160 entries
Issue queue	64 entries
Ld-St queue	48 entries
Branch prediction	Tournament: 2-level, 32-entry RAS, 4way 2K-entry BTB
Functional units:	
4 ALU	CMOS: 1 cycle, TFET: 2 cycles
2 Int Mult/Div	CMOS: 2/4 cycles, TFET: 4/8 cycles
2 LSU	1 cycle
2 FPU	CMOS: Add/Mult/Div 2/4/8 cycles; TFET: 4/8/16 cycles; Add/Mult issue every cycle, Div issues every 8/16 cycles
Private I-Cache	32KB, 2way, 64B line, Round-trip (RT): 2 cycles
Asym. FastCache	4KB, 1way, writeback (WB), 64B line, RT: 1cycle
Private D-Cache	32KB, 8way, WB, 64B line, RT: 2cycles (CMOS) or 4cycles (TFET)
Private L2	256KB, 8way, WB, 64B line, RT: 8cycles (CMOS) or 12cycles (TFET)
Shared L3	Per core: 2MB, 16way, WB, 64B line, RT: 32cycles (CMOS) or 40cycles (TFET)
DRAM latency	RT: 50ns
GPU Hardware	8 CUs with 16 EUs each, 1GHz
FMA unit	CMOS:3 cycles, TFET:6 cycles, pipelined issue every cycle
Vector registers	256 per thread, access: 1 cycle (CMOS) or 2 cycles (TFET)
Register file cache	6 entries per thread, access: 1 cycle
Network	Ring with MESI directory-based protocol

In our evaluation, we use the 15nm process node for the power and performance characteristics of TFET and CMOS. This is because we can obtain reliable parameter data at

15nm technology, but not beyond 15nm. A high-level scaling study of TFETs from 22nm to 10nm [68] shows that the insights from Table 3.1 hold true at 10nm. CMOS is likely to maintain a performance edge over TFET and, as a result, the HetCore tradeoffs will remain similar.

3.6.1 Configurations

Table 3.4 shows the CPU and GPU configurations evaluated. For the CPU, we evaluate 10 configurations. The baseline is an all-CMOS core (*BaseCMOS*). In BaseCMOS, all the caches use high-V_t transistors, and the core units consist of 60% high-V_t transistors. Two other baselines are BaseCMOS enhanced with the techniques of AdvHet in CMOS (*BaseCMOS-Enh*), and an all-TFET core (*BaseTFET*). Note that BaseTFET operates at 2x lower frequency and consumes 8x less dynamic power than BaseCMOS. This is much less dynamic power than *HetCore*, where TFET units consume 4x less dynamic power than CMOS units.

Table 3.4: CPU and GPU configurations evaluated.

CPU Configurations Evaluated	
Configuration	**Notes**
BaseCMOS	All-CMOS core
BaseCMOS-Enh	BaseCMOS + Larger ROB(160→192) & FP-RF (80→128) + CMOS asymm. DL1 (1cycle for 1way & 3cycles for rest)
BaseTFET	All-TFET core
BaseHet	BaseCMOS + FPUs, ALUs, DL1, L2, and L3 in TFET
AdvHet	BaseHet + Larger ROB(160→192) & FP-RF (80→128) + Dual speed ALU (3 ALUs in TFET & 1 ALU in CMOS) + Asymm. DL1 (1way CMOS & rest in TFET)
BaseL3	BaseCMOS + Larger ROB & FP-RF + L3 in TFET
BaseHighVt	BaseCMOS + high-V_t in FPUs & ALUs. Latencies of Add/Mul/Div are: Int 2/3/6 cycles & FP 3/6/12 cycles
BaseHet-FastALU	BaseHet + all ALUs in CMOS
BaseHet-Enh	BaseHet + Larger ROB & FP-RF
BaseHet-Split	BaseHet-Enh + Dual speed ALU
GPU Configurations Evaluated	
Configuration	**Notes**
BaseCMOS	All-CMOS core + Register file cache
BaseTFET	All-TFET core
BaseHet	BaseCMOS + SIMD FPUs & RF in TFET
AdvHet	BaseHet + Register file cache

We compare these baselines to BaseHet and AdvHet. We also evaluate several intermediate design points. *BaseL3* is BaseCMOS with the larger ROB and FP register file, and with a TFET L3. *BaseHighVt* is BaseCMOS plus FPUs and ALUs with only high-V_t transistors. These high-V_t devices have a 1.4-1.6x higher delay than regular-V_t ones [104]. The latencies of the FPUs and ALUs are shown in Table 3.4. Note that all the caches are designed with high-V_t transistors even in BaseCMOS, and so cache latencies remain the same. However, the leakage power of FPUs and ALUs in BaseHighVt is 10x lower than in BaseCMOS.

Finally, other configurations include BaseHet with all the ALUs in CMOS (*BaseHet-FastALU*), BaseHet with the larger ROB and FP register file (*BaseHet-Enh*), and BaseHet-Enh with the dual speed ALU cluster (*BaseHet-Split*).

For the GPU, we evaluate 4 configurations. The baseline is an all-CMOS core *with* the register file cache (*BaseCMOS*). We add the register file cache for fairness. We compare it to an all-TFET core (*BaseTFET*) and our proposed BaseHet and AdvHet designs.

3.6.2 Applications & Metrics

We use the SPLASH-2 and PARSEC applications to evaluate the CPU designs, similar to the ones we used in the evaluation of ScalCore. From SPLASH-2, we use Barnes (16K particles), Cholesky (tk29.O), FFT (2^{20}), FMM (16K), LU (512x512), Radiosity (batch), Radix (2M keys), Raytrace (teapot.env), Water-Nsquared (random.in), and Water-Nspatial (512). From PARSEC, we use Blackscholes(16K), Canneal(10000), Streamcluster (4K), and Fluidanimate(15K). For the GPU evaluation, we use all the applications from the AMD-SDK-APP suite provided along with the Multi2Sim simulator, with the suggested input sizes [102]. Our metrics of comparison are execution time, energy consumption, and energy-delay-squared (ED^2).

3.7 EVALUATION

3.7.1 HetCore CPU Evaluation

Figure 3.7 compares the execution time of BaseCMOS, BaseCMOS-Enh, BaseTFET, BaseHet, and AdvHet running our applications. The bars are normalized to BaseCMOS. There is an extra bar (AdvHet-2X) that we discuss later.

On average, BaseHet experiences a slowdown of 40%. This is mostly due to the increased latencies of the FPUs, ALUs, and DL1. Applications that often hit in the DL1 suffer the

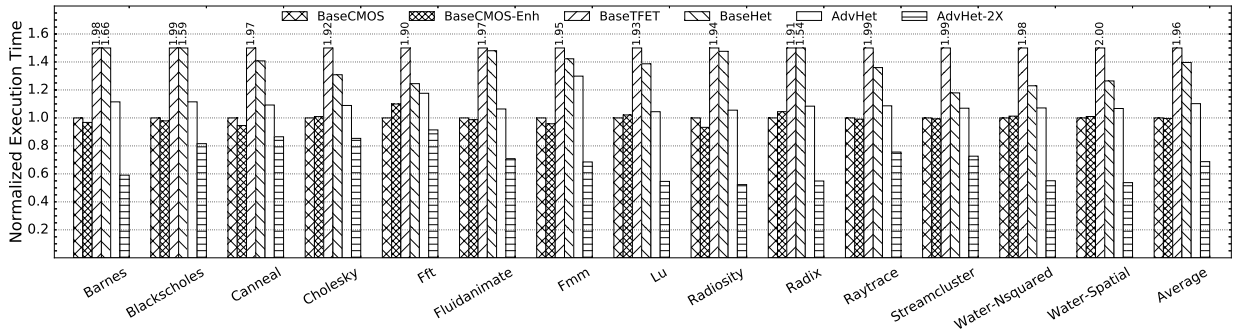

Figure 3.7: Execution time of different CPU designs, normalized to BaseCMOS.

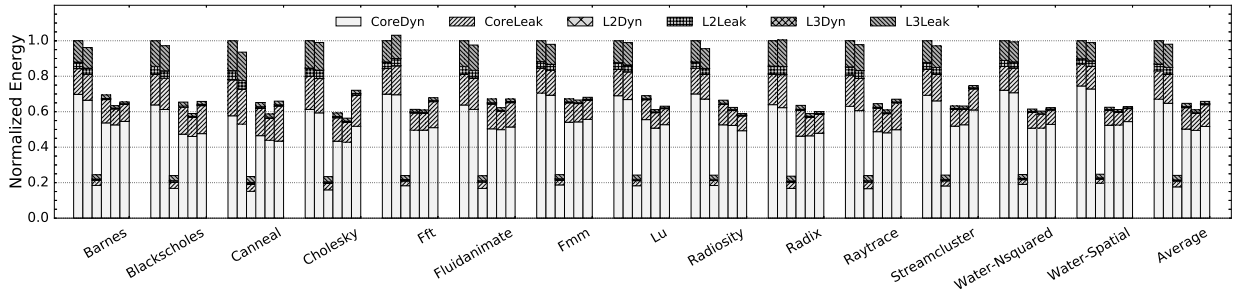

Figure 3.8: Energy consumption of different CPU designs, normalized to BaseCMOS.

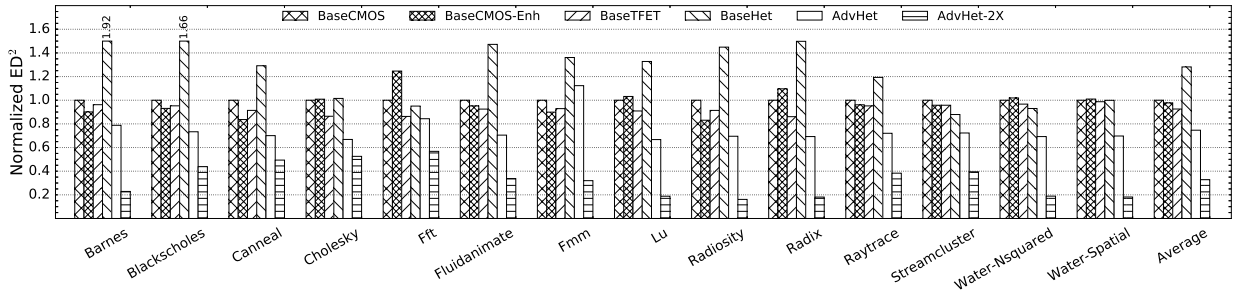

Figure 3.9: ED^2 of different CPU designs, normalized to BaseCMOS.

most, due to the higher access latency in BaseHet. The deeper-pipelined FPU and ALU units also hurt BaseHet's performance. Overall, BaseHet is not a very good design.

The performance enhancement techniques used in AdvHet prove effective, and recover most of the performance losses in BaseHet. Specifically, AdvHet's average execution time is only 10% higher than that of BaseCMOS.

BaseTFET shows a large slowdown of 96%. This is because its frequency is half of BaseCMOS' frequency. We also see that BaseCMOS-Enh does not improve over BaseCMOS on

average. This is because the pipeline changes in BaseCMOS-Enh largely unbalance the already balanced BaseCMOS design. These changes are only effective in AdvHet, due to unbalanced nature of BaseHet.

Figure 3.8 shows the energy consumption of the same configurations as Figure 3.7, broken down into the contributions of core (including the L1s), L2, and L3, and separating the dynamic and leakage energy. The bars are normalized to BaseCMOS. We see that BaseTFET reduces the energy consumption by 76%, thanks to the excellent energy efficiency of TFETs. The HetCore designs also provide very good energy savings over BaseCMOS. Specifically, BaseHet and AdvHet reduce the energy by 35% and 39%, respectively. The reductions come from both dynamic and leakage energy.

AdvHet saves slightly more energy than BaseHet for two reasons. First, AdvHet is faster and, hence, has lower leakage. Second, an access to the fast CMOS way of the asymmetric DL1 cache in AdvHet consumes less dynamic energy than an access to the TFET DL1 cache in BaseHet. Since a large fraction of DL1 accesses in AdvHet hit in the fast CMOS way, and never access the slow TFET ways, the overall dynamic energy consumption is low. Overall, AdvHet is an attractive design: it consumes on average 39% less energy than BaseCMOS, while performing within 10% of it.

BaseCMOS-Enh has similar results as BaseCMOS.

Finally, Figure 3.9 compares the ED^2 of all the designs. Although BaseHet consumes less energy than BaseCMOS, it has a worse average ED^2 because it is slower. AdvHet has the lowest ED^2, because it is nearly as fast as BaseCMOS and consumes much less energy. On average, its ED^2 is 26% lower than BaseCMOS, and 20% lower than BaseTFET.

Comparison Under a Constant Power Budget AdvHet is especially appealing when comparing chips at a constant power budget. From Figures 3.8 and 3.7, one can deduce that an AdvHet core consumes half the power of a BaseCMOS one. Hence, under the same power budget, we can power twice as many AdvHet cores as BaseCMOS ones in the chip.

The last column (AdvHet-2X) in Figures 3.7, 3.8 and 3.9 corresponds to this design. AdvHet-2X executes with 8 cores, with the same power budget as BaseCMOS with 4 cores. We can see that AdvHet-2X reduces the average execution time by 32% relative to BaseCMOS, while consuming 34% less energy. The result is a large 68% average ED^2 reduction.

Overall, combining CMOS and TFET in AdvHet delivers a compelling solution for upcoming energy-constrained environments. Workloads consume 39% less energy that CMOS designs, while running only 10% slower. Moreover, if they have substantial parallelism, they can execute much more energy efficiently as well as faster than CMOS designs.

Note that BaseTFET is also able to employ more cores within the same power budget.

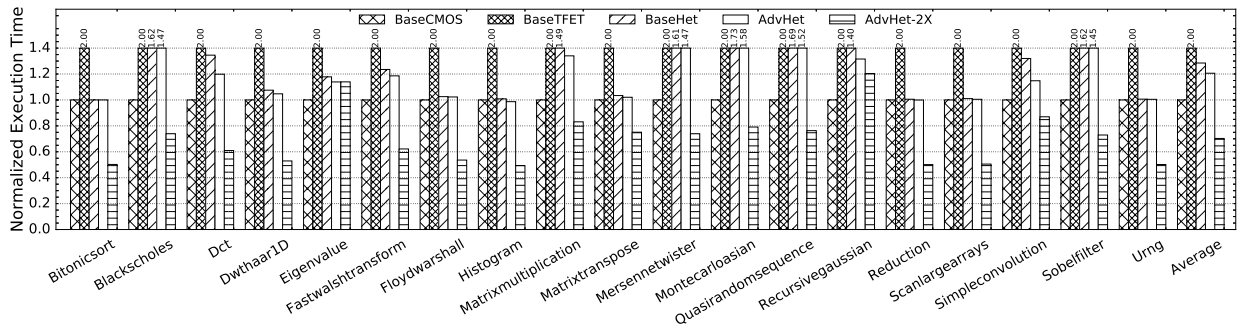

Figure 3.10: Execution time of different GPU designs, normalized to BaseCMOS.

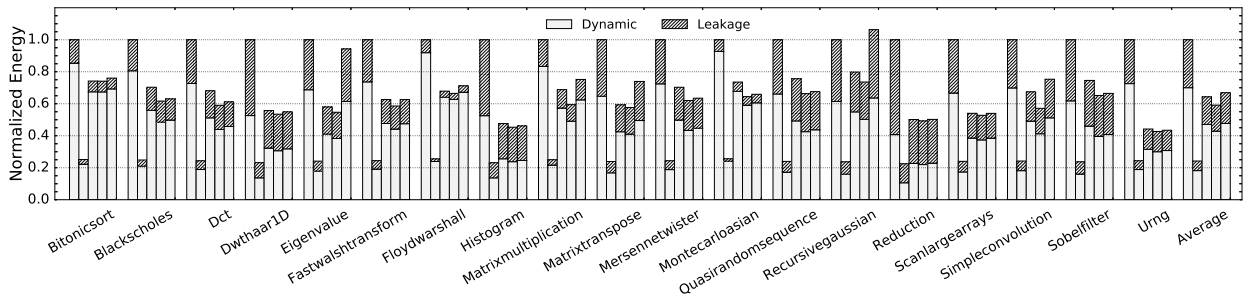

Figure 3.11: Energy consumption of different GPU designs, normalized to BaseCMOS.

Specifically, it can power 7-8 times more cores than BaseCMOS. The result is an efficient execution for very parallel workloads. However, with the same thread count as BaseCMOS, BaseTFET runs at half the BaseCMOS speed, which makes BaseTFET unattractive.

The goal of HetCore was to build a design that performs as well as a CMOS core while providing the energy efficiency of a TFET core. With AdvHet the performance of HetCore is within 10% of a CMOS core and exceeds the energy efficiency (ED^2) of TFET core by 20%. When compared to a CMOS core, HetCore reduces the energy by 39% and ED^2 by 26%. Under the same power budget, HetCore improves the performance by 32% and reduces energy and ED^2 by 34% and 68% respectively.

3.7.2 HetCore GPU Evaluation

For the GPU architecture, Figure 3.10 compares the execution time of BaseCMOS, BaseTFET, BaseHet, AdvHet, and AdvHet-2X running our applications. The bars are normalized to BaseCMOS. AdvHet-2X will be discussed later.

The execution time of BaseTFET is about twice that of BaseCMOS, as BaseTFET runs

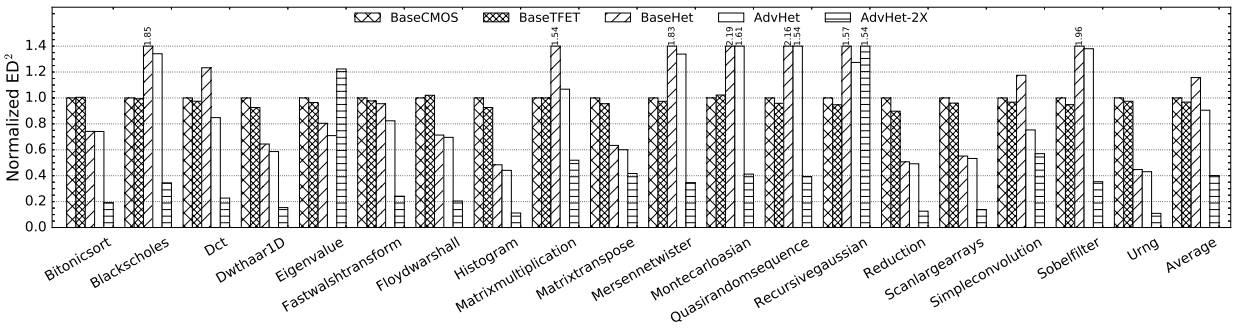

Figure 3.12: ED^2 of different GPU designs, normalized to BaseCMOS.

at half the frequency. Among the HetCore designs, BaseHet suffers an average performance loss of 28%. This is due to the slower SIMD FMA unit and register file. In AdvHet, we take BaseHet and add the register file cache. With this support, AdvHet improves the performance, but the average execution time is still 20% higher than BaseCMOS.

This performance loss appears, mostly, because we do not perform any compiler optimizations on the code to hide some of the longer latencies of the SIMD FPUs and register file. Such optimizations would help speed-up the programs, especially those with short-distance dependencies. In reality, however, since GPU workloads are throughput oriented, we are more interested in the performance under a fixed power budget. We consider this case in Section 3.7.2.

Figure 3.11 shows the energy consumption of the same configurations, broken down into dynamic and leakage energy contributions. The bars are normalized to BaseCMOS. We see that BaseTFET reduces the average energy consumption by 75%. BaseHet and AdvHet are also effective, reducing the average energy over BaseCMOS by 35% and 40%, respectively. The reductions come from both dynamic and leakage energy. The savings of AdvHet over BaseHet are due to the hits in the register file cache. Recall that, for fairness, BaseCMOS also includes the register file cache.

Finally, Figure 3.12 shows the ED^2 of the different configurations. While BaseHet consumes less energy than BaseCMOS, it has a worse average ED^2 because it is slower. AdvHet, thanks to the register file cache, is able to reduce the average ED^2 by 9% over BaseCMOS.

Comparison Under a Constant Power Budget The interesting scenario for AdvHet GPUs is comparing against BaseCMOS GPUs under a constant power budget. From Figures 3.11 and 3.10, one can see that an AdvHet GPU consumes half the power of a BaseCMOS one. Hence, we compare the execution of the 8 compute units in BaseCMOS to 16 compute

units in AdvHet. We call the latter AdvHet-2X in Figures 3.10, 3.11 and 3.12.

We see that, under AdvHet-2X, applications take on average 30% less time to execute (Figure 3.10) and consume on average 34% less energy (Figure 3.11) than under BaseCMOS. Moreover, Figure 3.12 shows that AdvHet-2X's ED^2 is on average 60% lower than that of BaseCMOS. Overall, AdvHet-2X is an attractive design for a GPU.

3.7.3 Comparison to Alternative CPU Designs

Figure 3.13 compares the execution time, energy, ED, and ED^2 of BaseHet and AdvHet to several other CPU configurations. Specifically, the figure includes three designs related to the baseline (BaseCMOS, BaseL3, and BaseHighVt), and three designs related to BaseHet (BaseHet-FastALU, BaseHet-Enh, and BaseHet-Split). The bars are normalized to BaseCMOS.

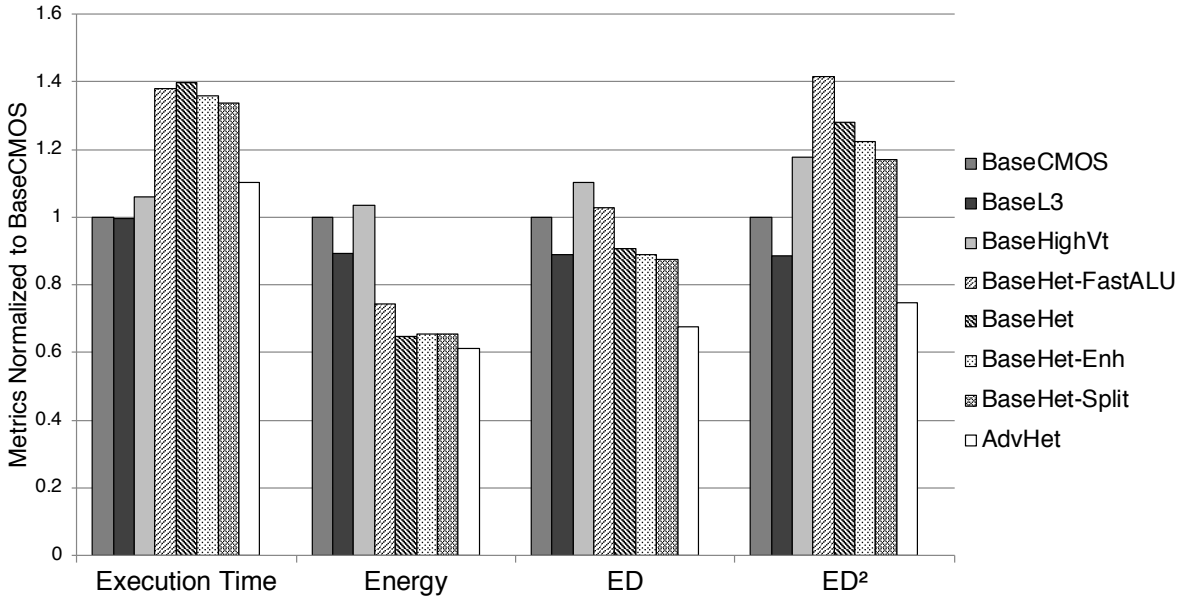

Figure 3.13: Sensitivity analysis of HetCore CPU designs.

BaseL3 has a performance similar to BaseCMOS, and saves leakage by using TFET devices in L3. Hence, it reduces about 10% of the energy, ED and ED^2 relative to BaseCMOS. AdvHet is better than BaseL3 because it saves 40% of the energy, reducing ED and ED^2 substantially.

BaseHighVt has FPUs and ALUs that use *only* high-V_t transistors. The rest is like BaseCMOS, which uses 60% high-V_t transistors in all the core units. Since the FPUs and ALUs have slightly higher latencies (Table 3.4), BaseHighVt is slightly slower than BaseCMOS. In

addition, we see that it consumes higher energy. This is because the leakage energy saved by having high-V_t FPUs and ALUs does not compensate for the increase in overall leakage energy due to longer execution. Since BaseHighVt is less cost-effective than BaseCMOS, it is also less cost-effective than AdvHet.

BaseHet-FastALU is BaseHet except that all its ALUs use CMOS. Compared to BaseHet-FastALU, BaseHet is 2% slower, but saves 10% of the energy. This tradeoff justifies using TFETs in ALUs as in BaseHet.

BaseHet-Enh takes BaseHet and increases the sizes of ROB and FP register file. This provides a marginal 3% performance improvement at comparable energy. On top of that, BaseHet-Split adds the dual-speed ALU cluster. This provides another 2% speedup at similar energy. Finally, if we add the asymmetric DL1 cache to BaseHet-Split, we obtain AdvHet. Adding the asymmetric DL1 cache reduces execution time by a substantial 17%, with marginal energy reduction (Figure 3.13). This speed-up is due to the high hit rate of the fast CMOS way of the asymmetric cache. Such hit rate is only 5-20% lower than that of a whole 32KB DL1.

3.7.4 Impact of DVFS and Process Variation

As discussed in Section 3.3.4, the V_{dd}-frequency curves for TFET and CMOS devices around their operating points are different. As a result, the energy consumption changes due to DVFS are different in AdvHet and BaseCMOS. In Figure 3.14, we show the energy consumed by BaseCMOS and AdvHet as we increase/decrease the frequency by 500 MHz. The figure shows bars for 2GHz (BaseFreq-2GHz), 2.5GHz (BoostFreq-2.5GHz), and 1.5GHz (SlowFreq-1.5GHz). The bars are normalized to the energy of BaseCMOS at 2GHz.

At 2GHz, AdvHet saves 39% of the BaseCMOS energy. As we move to f=2.5GHz, AdvHet needs to increase the voltages by ΔV_{CMOS}=75mV and ΔV_{TFET}=90mV. This was shown in Figure 3.3. The larger increase in V_{TFET} is needed because of the shape of the curve. As a result of the relatively higher ΔV_{TFET}, AdvHet is relatively less efficient and, as shown in Figure 3.14, only saves 36% of the BaseCMOS energy.

If we move to f=1.5GHz, AdvHet changes the voltages by ΔV_{CMOS}= -70mV and ΔV_{TFET}= -80mV. The larger V_{TFET} reduction makes AdvHet relatively more efficient, and now saves 43% of the BaseCMOS energy.

We also study the impact of process variation on the energy consumption. According to Avci et al. [88], to protect against all the potential sources of process variation at 15nm, we need to add V_{dd} guardbands equal to ΔV_{CMOS}=120mV and ΔV_{TFET}=70mV, for the respective operating voltages. These are large guardbands. With these guardbands, the

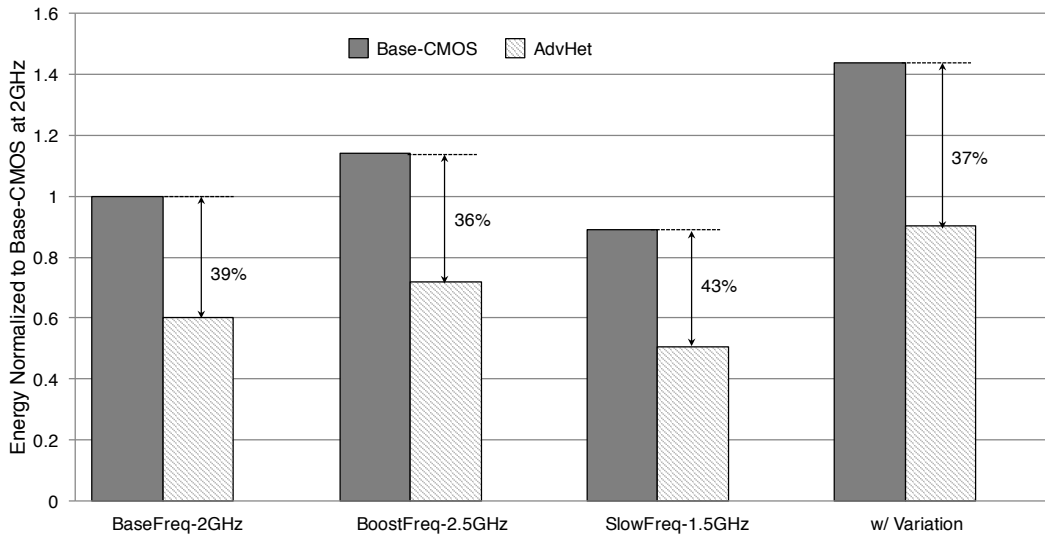

Figure 3.14: Impact of DVFS and process variation on the energy consumed by BaseCMOS and AdvHet.

energy consumed by BaseCMOS and AdvHet is shown in the rightmost bars of Figure 3.14. We see that the energy of both configurations goes up. Compared to BaseCMOS, AdvHet now saves more energy in absolute terms, but slightly less (37%) in relative terms.

3.8 SUMMARY

Ideally, we desire CPU and GPU cores that operate as energy-efficiently as a TFET core, while providing the performance of a CMOS core. To this end, we proposed the *HetCore* architecture, which judiciously integrates both TFET and CMOS units in a single core, creating a *hetero-device* core. We suggested that TFETs are used in units that consume high power under CMOS, are amenable to pipelining or are latency insensitive. We further improved the design through some microarchitectural optimizations. Our results show that such a design is very promising, even with conservative assumptions. An AdvHet CPU consumes on average 39% less energy than a CMOS CPU, while delivering a performance that is within 10% of the CMOS CPU. In addition, under a fixed power budget, a multicore with AdvHet CPUs attains average performance gains of 32% over a multicore with CMOS CPUs, while reducing ED^2 by 68%. Similarly, an AdvHet GPU consumes on average 40% less energy and performs within 20% of a CMOS GPU. Under a fixed power budget, an AdvHet GPU with twice as many compute units as a CMOS GPU improves average performance by 30% while lowering ED^2 by 60%.

CHAPTER 4: DESIGNING VERTICAL PROCESSORS IN MONOLITHIC3D: OPPORTUNITIES, CHALLENGES AND TRADEOFFS

4.1 INTRODUCTION

Vertical processors — i.e., processors laid out vertically in stacked layers — can reap major benefits in reduced wire delays, low energy consumption, and small footprint. Currently, 3D integration consists of stacking dies and using TSVs for inter-die communication (*TSV3D*) [15, 16, 105]. Unfortunately, TSV3D is a poor match for vertical processors. Specifically, the thick TSVs inhibit fine-grained logic partitioning across dies. Further, the challenges of cooling the layers away from the heat sink limit the flexibility of 3D designs [105, 16].

Monolithic 3D (*M3D*) [13, 17, 18] is a new 3D integration technology that allows high-bandwidth communication across layers and ultra high-density integration. Rather than bonding together pre-fabricated dies as in TSV3D, an M3D chip is built by sequentially fabricating different layers of devices on top of one another.

Using M3D to build vertical processors is attractive for three reasons. First, the active layers in M3D are separated by a distance of less than 1μm, which is one to two orders of magnitude shorter than in TSV3D [106, 14, 5]. This reduces the communication latency between the layers of a processor and allows for very compact designs.

Second, heat flows vertically very easily, thanks to a low thermal resistance. This is in contrast to TSV3D designs, which include thick, thermally-resistive layers such as the die-to-die layers [107]. As a result, temperatures away from the heat sink in M3D can be kept moderate.

Third and most importantly, the layers communicate using Monolithic Interlayer Vias (MIVs), which have diameters that are two orders of magnitude smaller than TSVs [108, 106, 109, 18, 14, 5]. The tiny diameters of MIVs allow designers to use many of them, dramatically increasing the bandwidth of inter-layer communication. This enables the exploitation of fine-grain partitioning of processor structures across layers, reducing wire length, energy consumption, and footprint.

There is broad consensus that M3D is the most promising approach to continue increasing transistor integration as Moore's law sunsets. As a result, there has been significant recent interest by both industry and research agencies [110, 18, 4, 111] to surmount the challenges of fabricating M3D chips. The 2017 IRDS roadmap [112] predicts that vertical nanowires will be realized in several years' time, followed by M3D. Prototypes of M3D systems have been demonstrated, signaling that this technology is feasible [18, 17, 113]. Finally, CAD tools required for 3D floorplanning are being actively developed as well [114, 115, 111].

As M3D becomes feasible, it is essential for computer architects to understand the opportunities and challenges of building vertical processors with this technology. As hinted above, M3D offers short wire lengths, good thermal properties, and high integration. However, an important constraint of current M3D technology is that different layers in an M3D stack have different performance. Specifically, the bottom layer is built out of high-performance transistors. However, any subsequent layer built on top of it must be fabricated at low temperature, to avoid destroying the bottom-layer devices. As a result, the transistors of any subsequent layer have a lower performance [116, 117, 4]. This imbalance has implications on how to design processor structures.

This work is the first one to show how to partition a processor for M3D. We design a vertical processor by taking logic, storage, and mixed logic-storage pipeline stages, and partition each of them into two layers. Our partition strategy uses the fact that the top layer has lower-performance transistors. Specifically, for logic structures, we place the critical paths in the bottom layer and the non-critical ones in the top one. For multiported storage structures, we asymmetrically partition the ports, assigning to the top layer fewer ports with larger access transistors. For single-ported storage structures, we asymmetrically partition the bitcell array, assigning to the top layer a shorter subarray with larger bitcells.

Overall, our contributions are:

- First work to partition cores for an M3D stack.
- Novel partition strategies of logic and storage structures for an environment with heterogeneous layers.
- Performance, power, and thermal evaluation of single and multiple M3D cores.

4.2 3D MONOLITHIC INTEGRATION

3D Monolithic Integration (M3D or 3DMI) is a new device integration technology that allows ultra-high density, fine-grained 3D integration. It involves fabricating two or more silicon layers sequentially on the same substrate. The bottom layer of transistors is fabricated first, using the same techniques as in a traditional die. Later, a layer of active silicon is grown on top of the bottom layer using novel techniques at a lower temperature [13, 106, 18]. Transistors are then formed on the top layer using a low-temperature process. The resulting top layer is often very thin, namely 100nm or less [13].

The integration process is fundamentally different from the conventional 3D integration, where dies are pre-fabricated and later connected using Through-Silicon-Vias (TSVs). For

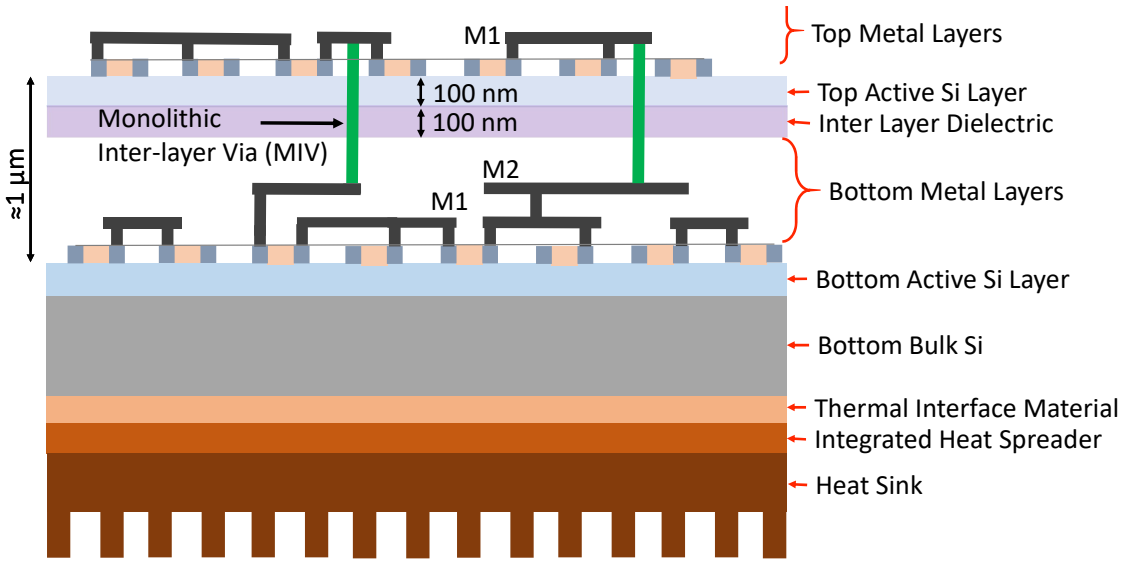

Figure 4.1: M3D integration of two layers.

this reason, M3D is also referred to as sequential 3D, while TSV3D is known as parallel 3D. Figure 4.1 shows a cross-section of an M3D stack. The layers of an M3D stack are connected by *Monolithic Inter-layer Vias* (MIVs).

4.2.1 Comparing M3D to TSV3D

To understand the architectural implications of using monolithic3D integration, we compare it with the TSV-3D integration. We contrast their utility based on electrical, thermal properties and the associated area overheads. We later discuss the opportunities and challenges of using the M3D technology for designing a multi-core chip.

Physical Dimensions of Vias A major advantage of M3D is the very small size of the MIVs. According to CEA-LETI [108, 106, 109, 18], they have a side equal to ≈50nm at the 15nm technology node. This is in contrast to TSVs, which are very large in comparison. Specifically, ITRS projects that TSVs will have a diameter greater than 2.6μm in 2020 [14]. Hence the granularity and placement of TSVs is heavily constrained, whereas MIVs provide great flexibility. To be conservative in our analysis, this work will assume an aggressive TSV with half the ITRS diameter, namely 1.3μm.

Figure 4.2 shows the relative area of an FO1 inverter, an MIV, an SRAM bitcell, and a TSV at 15nm technology. An MIV uses 0.07x the area of the inverter, while a TSV uses 37x the area of the inverter.

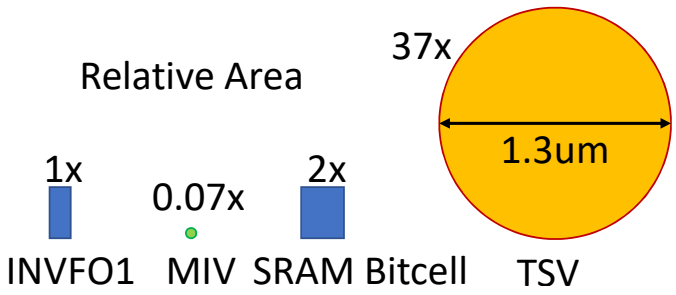

Figure 4.2: Relative area of an FO1 inverter, an MIV, an SRAM bitcell, and a TSV.

Ultra thin MIVs are possible due to two unique characteristics of M3D integration. First, as shown in Figure 4.1, the inter-layer dielectric (ILD) and the active silicon layer are very thin (≈100nm) [106]. This is a direct consequence of the sequential manufacturing of the top silicon layer. Second, M3D enables very precise alignment of layers through the use of standard lithography tools [17, 18]. Hence, the diameter of an MIV is equal to the pitch of the lowest metal layer.

Table 4.1 compares the area overhead of an MIV and a TSV to a 32-bit adder and a 32-bit SRAM cell at 15nm technology. The areas of the adder and SRAM cell are obtained from Intel [1, 118]. Note that, because an MIV is so small, it is assumed to be a square. For the TSV, we consider our aggressive design with a 1.3μm diameter, and the most recent design produced in research [5], which has a 5μm diameter. For the TSV, we add the area of the Keep Out Zone (KOZ) around it for mechanical strength and electrical isolation. The MIVs do not need any additional KOZ.

Table 4.1: Area overhead of an MIV and a TSV compared to a 32-bit adder and a 32-bit SRAM cell at 15nm.

Structure	**MIV(50nm)**	**TSV(1.3um)**	**TSV(5um)**
32bit Adder (77.7 um^2)	<0.01%	8.0%	128.7%
32bit SRAM Cell (2.3 um^2)	0.1%	271.7%	4347.8%

As we can see from Table 4.1, the MIV area accounts for a negligible overhead for both the 32-bit adder and the 32-bit SRAM cell. In contrast, even the most aggressive TSV implementation has noticeable overheads: its area (plus the KOZ) is equivalent to 8.0% of an adder's area or 272% of an SRAM cell area. Therefore, unlike TSV3D, M3D can provide connectivity to support the ultra-fine partition of components of a core across layers [108, 17, 18].

Table 4.2: Physical dimensions and electrical characteristics of typical copper MIV and TSVs [3, 4, 5].

Parameter	**MIV**	**TSV**	
Diameter	$50nm$	$1.3\mu m$	$5\mu m$
Via Height	$310nm$	$13\mu m$	$25\mu m$
Capacitance	$\approx 0.1fF$	$2.5fF$	$37fF$
Resistance	5.5Ω	$100m\Omega$	$20m\Omega$

Electrical Properties Table 4.2 shows the capacitance and resistance of an MIV and the two designs of TSV. We obtain the numbers for the 5μm TSV from the literature [5, 3], and use them to estimate the numbers for the 1.3μm TSV.

MIVs are shorter and thinner than TSVs. As a result, they have a significantly smaller capacitance but a higher resistance. The overall RC delay of the MIV and TSV wires is roughly similar. However, the wire power and the gate delay to drive the wire are mostly dependent on the capacitance of the wire. Both are much smaller in the case of MIVs. For example, Srinivasa et al. [119] show that the delay of a gate driving an MIV is 78% lower than one driving a TSV.

Thermal Properties TSV3D stacks have die-to-die (D2D) layers in between the dies. Such layers have ≈13-16x higher thermal resistance than metal and silicon layers [107]. Therefore, vertical thermal conductance is relatively limited in TSV3D, and the temperature differences across the layers are substantial.

M3D integration requires only a few metal layers in the bottom layer, as they route mostly local wires. Hence, the two active layers in M3D are physically close to each other — typically less than $1\mu m$ apart, even with 3-4 metal layers [120, 41]. Therefore, thermal coupling between the layers is high. In addition, the inter-layer dielectric is only 100nm thick. As a result, vertical thermal conduction is higher than in TSV3D. Hence, the temperature variation across layers is small.

4.2.2 Partitioning Granularity and Trade-offs

M3D technology is capable of supporting the partitioning of logic and memory structures across layers in a very fine-grained manner [108, 17, 18]. We briefly discuss the trade-offs in selecting the partitioning granularity.

Transistor Level Partitioning Transistor level (or N/P) partitioning places "N-type" and "P-type" transistors on two different layers. It allows independent optimization of each layer for the type of transistor. It also allows the use of largely standard 2D CAD tools for place and route. Finally, it also does not require any metal layers in between the two layers, simplifying the manufacturing process. However, it requires a redesign of standard library cells to use 3D stacked transistors. Further, static CMOS designs require a via for each N/P transistor pair, which results in a ≈10-20% area overhead [121, 122].

Gate Level or Intra-Block Partitioning Gate level or intra-block partitioning partitions logic or memory blocks at a gate level granularity. Adjacent gates can either be in the same layer or in a different layer. This approach allows the use of standard CMOS libraries and also has a lower via area overhead (at most 0.5% [123]).

Intra-block partitioning as a result of cell-on-cell stacking can reduce the footprint of a core by up to 50%. A smaller footprint reduces the intra-block wire length, and reduces the latency of some critical paths in the core such as the results bypass path, the load-to-use, and the notification of branch misprediction. It also reduces the span of the clock tree and power delivery networks, and their power consumption.

Block Level Partitioning Block level partitioning partitions the design by placing individual blocks such as ALUs, register files, or instruction decoders, as units in the different layers. It primarily has the same trade-offs as intra-block partitioning. However, there is much less flexibility in 3D routing and, correspondingly, the wire length reductions are much smaller. Further, it delivers no gains when the critical path is within a block as opposed to across blocks.

In this work, based on the capabilities of M3D, and our desire to keep the analysis at the architecture level, we focus on intra-block partitioning.

4.2.3 Prior Work on 3D Partitioning

Partitioning for TSV3D Prior architectural work on partitioning for TSV3D has examined placing cores on top of other cores [16], block level partitioning [15], and intra-block partitioning [105, 124]. In this section, we discuss the intra-block partitioning work, and leave the other, less relevant work, for Section 5.3.

Puttaswamy and Loh [105] examine several modules within a core and partition them into up to four layer. The partition is based on the activity of the gates. Since the gates with the highest activity are likely to consume the most power, the authors place them in the

highest layer, which is closest to the heat sink. Their goal is to alleviate thermal issues. For example, the least significant bits of an adder are placed in the top layer. However, such partition is not desirable with TSV technology. As we show in Table 4.1, the area of a single 1.3μm-diameter TSV, together with its KOZ, is equal to 8.0% of an adder's area. Hence, the overhead of the 16 TSVs proposed in [105] would be 128% of the area of the adder itself. This approach would negate any wire delay benefits. The same conclusion is reached by other researchers using 3D floor-planning tools on general logic stages [125, 111]. They find no wire delay reductions due to the high area overhead of TSVs.

Puttaswamy and Loh [124] also examine the 3D partitioning of SRAM array structures to reduce the wire delay of an access. The proposed strategies are bit partitioning (BP), word partitioning (WP), and port partitioning (PP). They are shown in Figure 4.3. These techniques partition the bits, words and ports, respectively, into two or more layers. For a two-layer design, BP has the lowest access latency, thanks to having the decoder drive only half of the bit length. Port partitioning could provide maximum gains due to the quadratic decrease in area with the reduction in number of ports. However, it requires two vias per bit cell (Figure 4.3). As indicated above, TSVs take too much area to make these designs attractive. For example, the area of a single SRAM bitcell is $\approx 0.05\mu m^2$ at 14nm [118], whereas the area of a single 1.3μm-diameter TSV, together with its KOZ, is $\approx 6.25\mu m^2$.

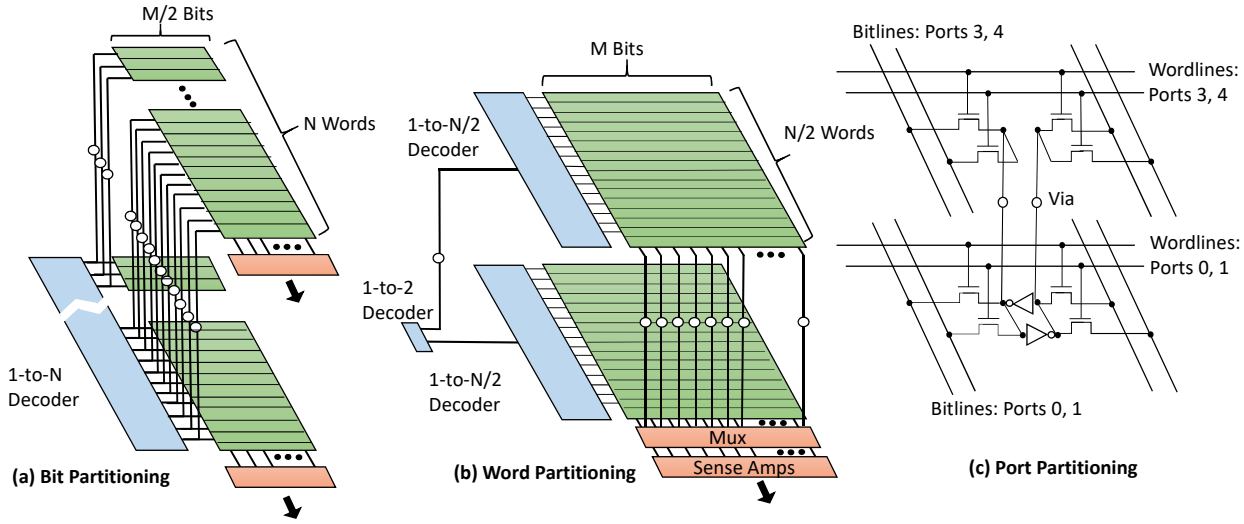

Figure 4.3: Partitioning an SRAM array using bit partitioning (a), word partitioning (b), and port partitioning (c).

Partitioning for M3D Some researchers have proposed to exploit the multi-layer capabilities of M3D integration to enhance the SRAM structures. Specifically, Srinivasa et

al. [119] use the second layer in an M3D design to add column access capability to a regular SRAM cell. Further, in [126], they place a few transistors on top of the SRAM cell either to improve the robustness/noise margins or to provide compute in memory support by performing simple operations such as AND and OR.

Several designs propose to partition the SRAM cell into two levels by placing n-type and p-type transistors on different levels [127, 128]. As we discuss in Section 4.2.2, we choose to partition the design at a gate level, which has different tradeoffs than transistor-level partitioning.

Kong et al. [129] study the benefits of using M3D integration to build large SRAM arrays such as the last-level cache. They focus on large single-ported structures. In this work, we focus on partitioning the processor core, where several key SRAM structures are small and multi-ported.

Most of these works [119, 126, 129] use the CACTI tool [52] to obtain the access energy and delay of SRAM structures. We use the same tool.

4.2.4 M3D: Opportunities and Challenges

Opportunities Modern core designs are constrained due to the historically slower scaling of wire delay relative to transistor delay. This is evident in wire-dominated structures such as SRAM arrays and wire-dominated critical paths such as the results bypass path. M3D integration provides a great opportunity to reduce the wire lengths and therefore the delays by partitioning at gate-level granularity.

M3D integration allows the optimization of the manufacturing process of each layer separately, to attain different power-performance points. For example, the bottom layer can use bulk transistors for a high-performance (HP) process, whereas the top layer can use the slower but lower power FDSOI transistors. This offers an opportunity for power savings beyond the simple choice of transistors with different V_t.

Challenges The primary challenge for M3D is manufacturability issues. The top layer in M3D is fabricated sequentially on top of the bottom one. This step usually involves high temperature, and may damage the bottom layer's devices and interconnects. One option is to use a tungsten-based interconnect in the bottom layer, as it has a higher melting point than copper [106, 18]. Unfortunately, tungsten has 3x higher resistance than copper and results in a significant wire delay increase. Further, the use of tungsten may still not be sufficient, as the bottom layer transistors can still be damaged.

Alternatively, the top layer can be processed at a significantly lower temperature, using

laser-scan annealing techniques [116, 117]. However, the M3D IC manufactured using this process showed a performance degradation of 27.8% and 16.8% for PMOS and NMOS devices, respectively [117]. A more recent study estimates that the delay of an inverter in the top layer degrades by 17% [4]. As a result, the authors found that gate-level partitioning LDPC and AES blocks causes their frequency to go down by 7.5% and 9%, respectively. Overall, the lower performance of the top layer poses challenges to the partitioning of the core.

A second challenge is a scarcity of CAD tools for 3D, which are currently being actively developed [114, 115]. In this work, we do not address this challenge. In the following Section, we first present the design of core assuming iso-performance in both tiers and later address the hetero-performance case in Section 4.4.

4.3 M3D CORE DESIGN

M3D integration allows a fine grained partitioning of core at an intra block granularity. As we discussed in Section 4.2, the primary beneficiary is the wire length and wire delay reduction. In this section, we discuss how to partition a core in M3D. For now, we assume that both M3D layers have the same performance. We present a hetero-layer design in the next section. We consider in turn the logic stages, storage structures, and other structures.

4.3.1 Logic Stages

The wires in a logic pipeline stage can be classified into local, semi-global, and global. Local wires connect gates that are close to each other. These wires comprise most of the intra-stage critical path wire delay. To optimize these wires, the best approach is to use CAD tools for place and route. It has been shown that 3D floor-planners customized for M3D integration reduce the lengths of local wires by up to 25% [111, 130]. There is little scope for further optimization of local wires using micro-architectural insights.

Semi-global wires connect one logic block to another logic block within a stage. These wires are often critical for performance from a micro-architectural viewpoint. Some examples are wires in the micro-architectural paths that implement the ALU plus bypass, the load to use, and the branch misprediction notification. Semi-global wires are not only critical to the frequency attained by the specific pipeline stage, but are also critical to the IPC of the core. M3D integration of a pipeline stage can reduce the footprint of the stage by up to 50% — therefore requiring the semi-global wires to traverse only half the distance. By

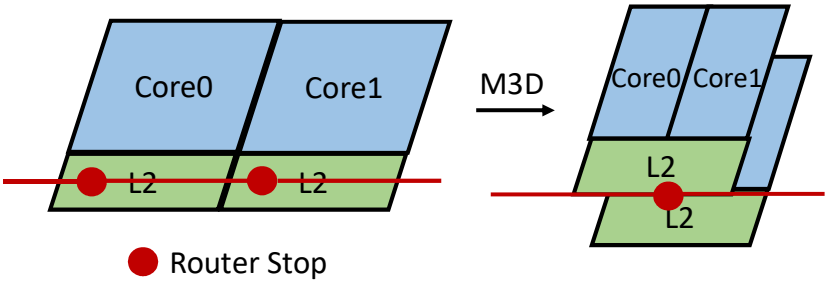

Figure 4.4: Two cores sharing the L2 and the router stop.

exploiting this, we reduce the latency of load to use and branch misprediction notification (similar to [15]).

Global wires span a significant section of the chip and may take multiple clock cycles. At best, the global footprint reduction halves the length of a global wire. An example of global wire is a link in an Network-on-Chip (NoC). If we manage to fold each core into half of its original area, two cores can share a single NoC router stop (Figure 4.4). In this case, we halve the distance between neighboring routers and reduce the number of hops. This reduces the average network delay for the same number of cores.

To verify the impact of wirelength reduction in logic stages, we synthesize and lay out a 64-bit adder along with a bypass path in 45nm technology. We use the M3D place and route tools developed by Lim et al. [115, 111]. The results show that a two-layer M3D implementation achieves a 15% higher frequency. Moreover, the footprint reduction observed is 41%. This reduction is in line with numbers reported elsewhere [111, 130]. If we had laid out multiple ALUs with their bypass paths, the contribution of wire delay toward the stage delay would be higher, since the length of the bypass path increases quadratically with the number of ALUs. In the case of four ALUs with bypass paths, we estimate a 28% higher frequency, 10% lower energy, and 41% lower footprint than a 2D design. Further, at the 15nm technology node that we consider in this work, the wire delay contribution is higher and, therefore, the frequency gain would be higher.

For the case of TSV3D we use the partitioning technique and parameters in [105] to estimate that the four ALUs and bypass paths enable 27% higher frequency, 10% lower energy, and 38% lower footprint than 2D designs.

4.3.2 Storage Structures

The storage structures in a core consist of SRAM structures such as the register file and branch prediction table, and CAM structures such as the issue queue and load/store queue. CAM and RAM structures are structurally very similar in their layout. Therefore, we treat them similarly for the purpose of partitioning them in 3D.

An SRAM array is given by its height and width. The height is the number of words (N_{words}). The width is the number of bits per word (N_{bits}). A wordline is as long as the width of the array; a bitline is as long as the height of the array.

Equations 4.1 and 4.2 give the width and height of the array, where N_r,N_w are the number of read and write ports.

$$Width = N_{bits} * (BitcellWidth + K_1 * (N_r + N_w)) \tag{4.1}$$

$$Height = N_{words} * (BitcellHeight + K_2 * (N_r + N_w)) \tag{4.2}$$

$$Area = Width * Height \propto (N_r + N_w)^2 \tag{4.3}$$

As we partition an array into two layers, we keep in mind some basic rules. The area is proportional to the square of the number of ports. Both the array access latency and the energy consumed depend in part on the length of the wordline and bitline. The wordline and bitline length depend on the width and height of the array, respectively.

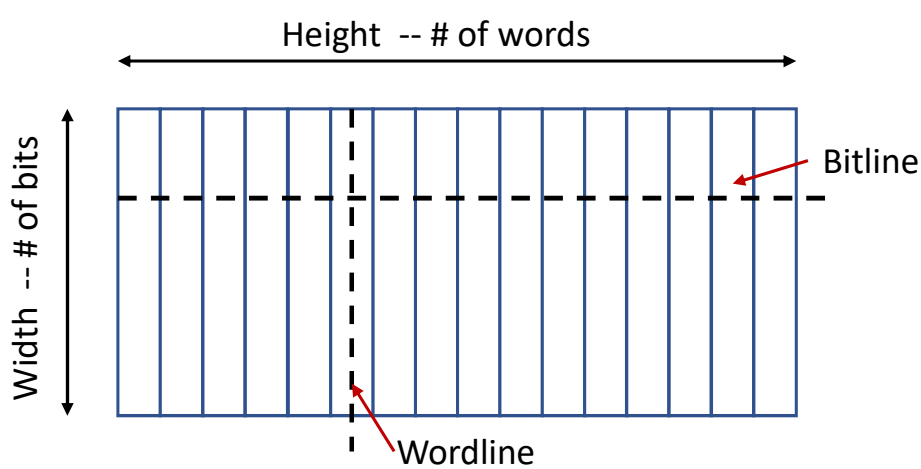

Figure 4.5: Basic structure of an SRAM array.

We model the partitioning of the SRAM arrays using CACTI [52]. We use high performance (HP) transistors and, to be both conservative and reasonably accurate, use 22nm technology parameters. MIV and TSV overheads are modeled using 50nm and 1.3μm diam-

eters, respectively, as per Section 4.2.1.

We partition the following SRAM arrays in the core: register file (RF), issue queue (IQ), store queue (SQ), load queue (LQ), register alias table (RAT), branch prediction table (BPT), BTB, data and instruction TLB, data and instruction L1, and L2 cache. We partition them using bit partitioning (BP), word partitioning (WP), and port partitioning (PP) (Section 4.2.2), and measure the reduction (or increase) in access latency, access energy, and area footprint compared a 2D design. As examples, we describe the partitioning of a register file and a branch predictor table. The former has 160 words of 64 bits, 12 read ports, and 6 write ports. The latter has 4096 words of 8 bits and 1 port.

Bit Partitioning (BP) BP spreads half of each word in a layer, which reduces the wordline length. Each word requires a via across the layers (Figure 4.3(a)). Table 4.3 shows the percentage of improvement we attain by bit partitioning our two structures using M3D and TSV3D, compared to a 2D structure.

Table 4.3: Percentage reduction in access latency, access energy, and area footprint through bit partitioning.

	Register File (RF)			**Branch Pred. Table (BPT)**		
	Latency	Energy	Footprint	Latency	Energy	Footprint
M3D	28%	22%	40%	14%	15%	37%
TSV3D	25%	19%	31%	4%	-3%	4%

From the table, we observe that M3D performs better than TSV3D in all metrics. This is expected, as the diameter of an MIV is much smaller than that of a TSV. Furthermore, we see that the gains in the multiported register file are much higher than in the single-ported branch prediction table. This is because of two reasons related to the much larger area required by multiported structures. First, when the area is large, the wire component of the SRAM access delay is relatively higher; hence partitioning will be relatively more beneficial. Second, when the area is large, the overhead of the vias becomes less noticeable, which enables higher improvements. This is especially evident in the case of TSV3D, which only marginally improves the BPT.

Word Partitioning (WP) WP spreads half of the words in each layer, which reduces the bitline length. The number of vias needed is equal to the array width (Figure 4.3(b)). Table 4.4 shows the percentage of improvement we attain by word partitioning our two structures using M3D and TSV3D, compared to a 2D structure.

Table 4.4: Percentage reduction in access latency, access energy, and area footprint through word partitioning.

	Register File (RF)			**Branch Pred. Table (BPT)**		
	Latency	Energy	Footprint	Latency	Energy	Footprint
M3D	27%	35%	43%	14%	36%	57%
TSV	24%	32%	39%	-6%	9%	19%

The observations from BP hold true for WP as well. They are both affected in similar ways by the larger area induced by multiple ports, and the larger size of TSVs. However, in general, BP partitioning is preferable over WP, as wordlines are responsible for more energy and delay than bitlines. Interestingly, the case of the branch prediction table is an exception: WP proves to be a better design than BP in M3D. The reason is the aspect ratio of the branch prediction table's array. The array's height is much higher than its width. Hence, WP's ability to halve the bitlines delivers significant savings.

Port Partitioning (PP) PP places the SRAM bit cell with half of its ports in one layer and the rest of the ports with their access transistors in the second layer. It needs two vias per SRAM bit cell as shown in Figure 4.3(c). Table 4.5 shows the percentage of improvement we attain by port partitioning our two structures using M3D and TSV3D, compared to a 2D structure.

Table 4.5: Percentage reduction in access latency, access energy, and area footprint through port partitioning.

	Register File (RF)			**Branch Pred. Table (BPT)**		
	Latency	Energy	Footprint	Latency	Energy	Footprint
M3D	41%	38%	56%	-	-	-
TSV	-361%	-84%	-498%	-	-	-

Generally, halving the number of ports is an excellent strategy: it reduces both the wordline length and the bitline length nearly by half (Equations 4.1 and 4.2). In M3D, this reduces the latency, energy, and area by a large amount. As can be seen in Table 4.5, the improvements in the RF are large. Of course, PP cannot be applied to the BPT because the latter is single ported.

PP can be applied to M3D because the MIVs are very thin. It is possible to place two vias per RF SRAM cell — especially since a multiported SRAM cell is larger than the typical SRAM cell. However, TSVs are too thick to be used in PP. As shown in Table 4.5, the cell

area increases by 498%, creating large increases in access latency and energy.

Overall, we conclude that, for M3D, PP is the best design for multiported structures, while BP is usually the best one for single-ported ones. The exception to the latter is when the SRAM array has a much higher height than width, in which case WP is best. TSV3D is much less effective, and is not compatible with PP. Table 4.6 shows the best partitioning strategy for each SRAM structure in the core we evaluate in Section 4.6 — both for M3D and TSV3D. The table shows the percentage of improvement we attain in access latency, access energy, and footprint compared to a 2D structure.

Table 4.6: Best partition for each structure and percentage reduction in latency, energy and area footprint.

Structure with per bank Words;Bits/Word	**Best Partition M3D—TSV**	**Latency Reduc.(%) M3D—TSV**	**Energy Reduc.(%) M3D—TSV**	**Footprint Reduc.(%) M3D—TSV**
RF[160;64]	PP—BP	41—25	38—19	56—41
IQ[84;16]	PP—BP	26—17	35—5	50—32
SQ[56;48]	PP—BP	14—(-3)	21—(-18)	44—0
LQ[72;48]	PP—BP	15—2	36—8	48—10
RAT[32;8]	PP—WP	20—10	32—5	45—(-11)
BPT[4096;8]	WP—BP	14—4	36—(-3)	58—4
BTB[4096;32]	BP—BP	15—(-6)	20—(-10)	37—(-20)
DTLB[64*8;64]	BP—BP	26—18	28—20	35—22
ITLB[64*4;64]	BP—BP	20—7	28—11	36—11
IL1[256*4;256]	BP—BP	30—14	36—23	41—25
DL1[256*4;256]	BP—BP	41—31	40—33	44—34
L2[1024*8;512]	BP—BP	32—24	47—42	53—46

Since the distance from a core to another core and its L2 cache is now reduced, two cores can now share their L2 caches without increasing the overall latency. (Figure 4.4).

4.3.3 Clock Tree and Power Delivery Network

The clock-tree network and the power delivery network (PDN) only have to cover half of the footprint of a 2D design. Since the clock tree consumes substantial dynamic power, the power savings due to the reduced footprint can be significant. There are two options for designing the power delivery network in M3D chips. One option is to give each of the two layers its own PDN. This design increases the number of metal wires, which increases both the via routing complexity and the cost. Alternatively, one can use a single PDN that is

present in the top layer and then supply power to the bottom layer through MIVs. Billoint et al. [131] suggests that this second approach is preferable.

4.4 HETERO LAYER M3D DESIGN

Low temperature processing of the top layer in M3D causes the top layer to have lower performance than the bottom one. As indicated in Section 4.2.4, current projections suggest that the overall frequency reduction is 9% [4]. Consequently, we modify the core partitioning strategies of Section 4.3 to alleviate the impact of the slowdown. With these strategies, we design a *hetero-layer M3D core*.

Our approach is shown in Table 4.7. In logic pipeline stages, we identify the critical paths in the stage and place them in the bottom layer. The non-critical paths are placed in the top layer and do not slow down the stage. This is possible because more than 60% of the transistors in a typical stage are high V_t, and fewer than 25% are low V_t [78] (the rest are regular V_t). Hence, there are always many non-critical paths.

In storage structures, the critical path spans the entire array. Hence, we cannot use the same approach as the logic. Instead, we use two separate techniques based on the partitioning strategy applied in Section 4.3.2. In port partitioning, we exploit the fact that the two inverters in the SRAM bit cell are in the bottom layer (Figure 4.3(c)). Specifically, we partition the ports asymmetrically between the two layers, and increase the sizes of the access transistors in the ports in the top layer, which makes them relatively faster. In bit/word partitioning, we partition the array asymmetrically between the layers, giving a smaller section to the top layer. Further, we use the area headroom in the top layer to increase the sizes of the bit cells. Finally, in mixed stages, we combine the two techniques. In this section, we present these techniques.

Table 4.7: Partitioning techniques for a hetero-layer M3D core.

Logic Stages	Critical paths in bottom layer; non-critical paths in top	
Storage Structures	Port Partitioning	Asymmetric partitioning of ports, and larger access transistors in top layer
	Bit or Word Partitioning	Asymmetric partitioning of array, and larger bit cells in top layer
Mixed Stages	Combination of the previous two rows	

4.4.1 Logic Stages

We analyze an out-of-order superscalar processor and identify three primarily logic oriented stages: decode, dispatch, and execute in integer and FP units. We partition them as per Table 4.7. We give two examples below.

Partitioning an Integer Execution Unit Figure 4.6 shows a 64-bit carry skip adder. The critical path is shown shaded. It consists of a carry propagate block, a sum block, 15 muxes and a final sum block. The majority of the blocks, i.e., the remaining 15 4-bit carry propagate blocks and sum blocks are not in the critical path. The farther away a propagate block is from the LSB, the higher slack it has. Therefore, we place the carry propagate blocks of bits {32:63} and the sum blocks of bits {28:59} in the top layer. There is no impact on the critical path and hence the stage delay.

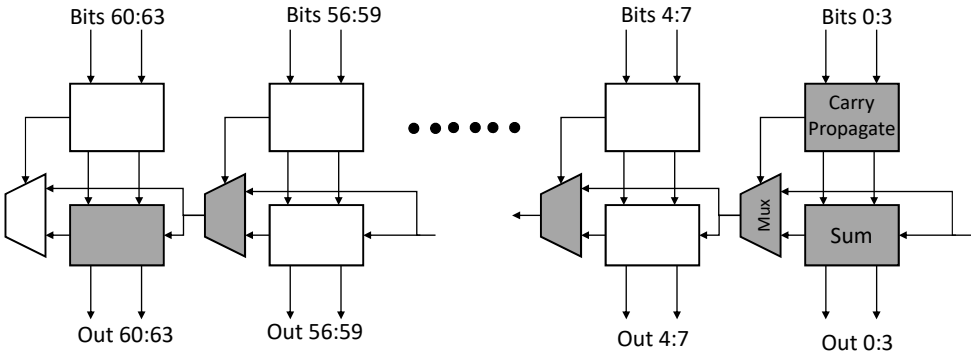

Figure 4.6: ALU with shaded critical path blocks.

Using our M3D place and route tools of Section 4.3.1, we find that only 1.5% of the gates in the 64-bit adder are in the critical path. We place them in the bottom layer. Even if we assumed that the top layer was 20% slower — and, hence, we need a 20% slack — we would only have 38% of the gates in the critical path. Hence, we can always find 50% of gates that are not critical and place them in the top layer.

Partitioning Decode Modern x86 processors have a set of simple decoders and a complex decoder. Most common operations that translate into a single μop are processed by the simple decoders. More complex and uncommon instructions utilize the complex decoder and, occasionally, a special μcode ROM to generate multiple μops. In our hetero-layer M3D design, we place the simple decoders in the bottom layer. The complex decoder and the

μcode ROM are placed in top layer and take one additional cycle. The μcode ROM access already takes multiple cycles.

4.4.2 Storage Structures

Port Partitioning (PP) As shown in Table 4.6, PP is the best strategy for multiported arrays such as the RF, IQ, SQ, LQ, and RAT. In a port-partitioned cell, the two inverters are left in the bottom layer, while the ports are divided into the top and bottom layers. In a hetero-layer M3D, we assign fewer ports to the top layer than to the bottom one, and double the width of transistors of the ports in the top layer. The goal is to make the top layer's transistors as fast as the bottom layer's ones and still use the same footprint in both layers.

We measure that the area of the two inverters in a bitcell is comparable to that of two ports. However, the optimal port partitioning depends on the number of ports. For example, consider a register file with 12 read and 6 write ports. We find that the least-footprint partition places 10 ports in the lower layer and 8 ports (with double-width transistors) in the top one. In this case, Table 4.8 shows that the register file uses about 47% less area than in a 2D layout. This is 9 fewer percentage points than the partition for same-performance M3D layers (Table 4.6).

The wider access transistors alleviate the impact of the bitline delay in the top layer. However, they increase the capacitance on the wordlines slightly. This increases the cell access energy and wordline delay slightly. We measured the resulting access latency, energy and footprint of the RF, IQ, SQ, LQ, and RAT structures. Table 4.8 shows the savings compared to a 2D structure. These are substantial reductions. Compared to the partition for same-performance layers in Table 4.6, the numbers are only slightly lower.

Table 4.8: Percentage reduction in access latency, access energy, and area footprint with our hetero-layer partitioning compared to a 2D layout.

	RF (%)	IQ (%)	SQ (%)	LQ (%)	RAT (%)	BPT (%)	BTB (%)	DTLB (%)	ITLB (%)	IL1 (%)	DL1 (%)	L2 (%)
Latency	40	24	13	13	20	13	13	23	18	27	37	29
Energy	32	30	17	30	24	30	16	25	25	33	36	42
Area	47	47	43	47	44	40	26	25	28	30	31	42

Bit/Word Partitioning (BP/WP) Our technique to alleviate the impact of the slower top layer in structures using BP/WP consists of two steps. First, we perform BP/WP asymmetrically across layers, giving a larger section of the array to the bottom layer. Next,

we use larger transistors in the top layer. We tune these two operations to obtain the minimum access latency, while tolerating a slight increase in access energy and footprint. In general, a partition that gives 2/3 of the array to the bottom layer, together with doubling the transistor width in the top layer works best. Table 4.8 shows the reductions of access latency, energy, and footprint of these structures compared to a 2D layout. Again, these are large reductions, only slightly smaller than those under the same-performance partition (Table 4.6).

In Table 4.8, we see that L1 and L2 caches have large latency reductions. Since the core's frequency is determined by the slowest pipeline stage, we can tune the caches' partitions to save more on footprint at the expense of less on latency.

4.4.3 Stages with Logic and SRAM Structures

Most storage structures are part of a stage that also contains logic components. We discuss the modifications to such stages in two parts: this subsection for SRAMs and the next one for CAMs.

Rename The rename stage reads from and writes to the Register Alias Table (RAT), which is a multiported structure. We use PP for the RAT as per Section 4.4.2. In parallel to the RAT access, a dependency check is performed among the registers being renamed. This check is not in the critical path [45]. Hence, we place this checking logic and the shadow RAT tables used for checkpointing in the top layer. We place other critical structures such as the decoder to the RAT's RAM array in the bottom layer.

Fetch & Branch Prediction The fetch unit mainly consists of accessing the IL1 cache and computing the next PC. The IL1 cache uses BP as per Section 4.3.2. Computing the next PC has a few different parallel paths: BTB access, branch prediction, Return Address Stack (RAS) access, and incrementing PC. Of these, only branch prediction and BTB access are critical to stage delay. Hence, we place RAS and PC increment in the top layer.

Since the BTB is critical, we use asymmetric BP coupled with larger transistors in the top layer. As shown in Table 4.8, we reduce its access energy by 16% compared to a 2D layout, which is 4 percentage points fewer than the partition for same-performance M3D layers (Table 4.6).

Our core employs a tournament branch predictor, which contains a selector table indexed by a hash of PC and global branch history. The selector output drives a mux that chooses between a local and global predictor. We observe that the critical path is formed by the

selector and mux, and not by the local or global predictors. Therefore, we propose an organization where we use asymmetric BP for the selector, local predictor, and global predictor. However, we place the larger section of the selector array in the bottom layer, and the larger sections of the two predictors in the top layer. We save 40% of footprint relative to a 2D layout (Table 4.8).

4.4.4 Stages with Logic and CAM Structures

CAM arrays are similar to SRAM arrays, except that they include an additional wire per cell called the *Match* line, which is connected to the cell. A few additional transistors associated with the cell (usually 4) pull the match line low when the bit stored in the cell is different than the one being compared to it (i.e., the *Tag* bit). The critical path in this structure is the time it takes to drive the tag line, plus the time for the match line to go low, plus any peripheral logic that operates on the output of the match lines.

In a core, CAM structures are found in the IQ, LQ, SQ, and the tag arrays of caches. For the tag arrays of caches, we use the same organization as the associated cache, namely, BP. The IQ is multiported with as many ports as the issue width, and the LQ and SQ have two ports each. These structures use asymmetric PP and larger transistors in the top layer.

Issue Issue in modern processors consists of two pipeline stages, namely Wakeup and Select. During the Wakeup stage, the IQ is accessed to mark as ready the dependent operands. They will determine the instructions that can be executed next.

During the Select stage, the select logic is activated to pick the instructions to execute. The selection logic is made up of multi-level arbitration steps, and consists of two phases: a *Request* phase in which the ready signal is propagated forward from each arbiter, and a *Grant* phase in which one of the requesters is selected by the arbiter. At a particular arbitration level, the generation of the grant signal is in turn decomposed into two parts. First, the local priorities at this level are compared, and one requester is selected for the grant signal. Second, the signal from this requester is ANDed with an incoming grant signal from a higher level in the arbitration hierarchy, and an output is generated. Such an organization minimizes the critical path delay in selection logic [132].

The first part of grant phase, when each arbiter level generates its local grant signal, is not critical. Hence, we place this logic in the top M3D layer. The second part, when the grant signals are ANDed and propagated, is critical. We place this logic along with the forward request propagation logic in the bottom layer. With this, the Select stage has the same latency as in the partition for same-performance layers, without increasing the area or

power.

Load Store Unit The LQ and SQ in the Load Store Unit (LSU) are CAM structures searched by incoming store and load requests. When a load searching the SQ hits, the value from the youngest matching store is forwarded to the load. This comprises the critical path in a LSU [133]. The search in the LQ, and the corresponding squash on a match are not critical to the stage delay. Therefore, the critical path in the stage consists of the CAM search of the SQ, a priority encoder to find the youngest store, and the read from the store buffer. For the CAM SQ, we use the usual PP methodology with bigger transistors in the top layer. We place the priority encoder in the bottom layer. The store buffer uses asymmetric BP with more bits in the bottom layer. Finally, the less critical LQ uses asymmetric PP occupying more area in the top layer. Compared to the partition for same-performance layers in Table 4.8, this design attains a roughly similar footprint reduction and only a slight increase in access latency and energy (Table 4.6).

4.5 ARCHITECTURES ENABLED BY M3D

M3D enables several types of architectures. We briefly discuss them now.

4.5.1 Exploiting Wire Delay Reduction in Conventional Cores

One can use conventional cores in M3D and exploit the reduction in wire delay in both logic and storage stages. This can be done in three possible ways. First, wire delay reductions can be translated into reductions in cycle time, which allows a higher core frequency. However, this approach increases the power density.

Alternatively, one can increase the sizes or the number of ports for some of the storage structures, while maintaining the same frequency as a 2D core. Note that most of the structures that are bottlenecks in wide-issue cores benefit significantly from M3D. These include multiported issue queues, multiported register files, and bypass networks. Therefore, one can increase the issue width of the core while maintaining the same frequency.

Finally, another alternative design is to simply operate the M3D design at the same frequency as the 2D core, and lower the voltage. Reducing the voltage lowers the power consumption and the power density. The M3D design can now operate more cores, using leeway in power consumption, for the same power budget. We evaluate these three designs in Section 4.7.2.

4.5.2 Hetero M3D design with HP and LP layers

It is possible that M3D with equal-performance layers may be manufacturable in the future. Even in this case, our partitioning techniques may be applicable. Specifically, to save overall energy, one may choose to build the two layers with different technology: high performance (HP) bulk transistors in the bottom layer and low performance (LP) FDSOI transistors in the top layer. In this case, the top layer has lower performance and power, while the bottom layer has high performance and power. Our proposed partitioning techniques apply directly in this environment. We evaluate such a scenario in Section 4.7.1.

4.5.3 Novel Architectures

Specialization is increasingly common in architectures, and is necessary to provide tight integration between specialized engines and general-purpose cores, to support fine-grain communication between the two. However, such architectures have to make compromises in a 2D design. M3D integration facilitates such tight integration, e.g., by supporting a set of accelerators in the top layer without compromising the layout of general-purpose cores in the bottom layer.

Further, M3D integration allows heterogeneous integration of process technologies, such as non-volatile memories on top of regular logic cells [113]. This allows new computing paradigms that can take advantage of huge amounts of persistent memory very close to the compute engines. Heterogeneous M3D integration is expected to provide 50x improvement in performance at the same power [110].

4.5.4 Other Techniques

A different class of techniques such as time borrowing, clock skew, and speculative setting of stage boundaries [134, 135, 136] can be used to address the problem of slower top tier. However this approach has a few caveats. First its applicability is limited. For example, it cannot help in the case of an intra stage loop such as ALU+bypass. Second, it constrains the 3D routing for elements such as clock tree, instead of exploiting them. Finally, the complexities of such a design to carefully manage clock skew etc., would have more overheads. Therefore, we refrain from applying fancier speculative circuit techniques in our M3D core design.

4.6 EVALUATION METHODOLOGY

We evaluate the performance of our designs using the Multi2Sim [102] architectural simulator. We model a processor with 4 cores. Each core is 6-wide and out of order. Table 4.9 shows the detailed parameters of the modeled architecture. We model the 3D SRAM and CAM arrays using CACTI as discussed in Section 4.3.2. We obtain the power numbers of the logic stages by using the HP-CMOS process of McPAT [38]. We set the voltage to the nominal V_{dd} at 22nm, namely 0.8V [52, 38]. We set the frequency by analyzing all the core structures we discussed, and using CACTI to find the maximum access time of any structure. Based on this, the frequency is set to 3.3GHz.

Table 4.9: Parameters of the simulated architecture for the evaluation of an M3D Core.

Parameter	**Value**
Cores	4 out-of-order cores, f=3.3GHz, V_{dd}=0.8V
Core width	Dispatch/Issue/Commit: 4/6/4
Int/FP RF; ROB	160/160 regs; 192 entries
Issue queue	84 entries
Ld/St queue	72/56 entries
Branch prediction	Tournament, with 4K entries in selector, in local, and in global predictor, 32-entry RAS, 4way 4K-entry BTB
Functional units	
4 ALU	1 cycle
2 Int Mult/Div	2/4 cycles
2 LSU	1 cycle
2 FPU	Add/Mult/Div 2/4/8 cycles; Add/Mult issue every cycle; Div issues every 8 cycles
Private I-cache	32KB, 4way, 64B line, Round-trip (RT): 3 cycles
Private D-cache	32KB, 8way, WB, 64B line, RT: 4cycles
Private L2	256KB, 8way, WB, 64B line, RT: 10cycles
Shared L3	Per core: 2MB, 16way, WB, 64B line, RT: 32cycles
DRAM latency	RT: 50ns
Network	Ring with MESI directory-based protocol

We model M3D with the layers shown in Figure 4.1. The layers are: the heat sink, Integrated Heat Spreader (IHS), Thermal Interface Material (TIM), bottom bulk silicon, bottom active silicon layer, bottom metal layer, Inter Layer Dielectric (ILD), top active silicon layer, and top metal layer. Note that the bottom part of the picture ends up being on top in practice; we show the figure this way for the sake of consistency with manufacturing terminology.

The core blocks are partitioned across the two silicon layers. Note that M3D requires only

3-4 stacked metal wires in the bottom metal layer [120]. Moreover, the inter layer dielectric is very thin. As a result, the distance between the two active silicon layers is only ≈1μm [41]. The thermal conductivity of the different layers is shown in Table 4.10. These values are obtained from recent work on thermal models for TSV3D [107].

Table 4.10: Thermal modeling parameters for M3D & TSV3D.

Layer	**M3D Dimensions**	**TSV3D Dimensions**	**Thermal Conductivity**
Top Metal	12 μm	12 μm	12 W/m-K
Top Silicon	100nm	20μm	120 W/m-K
ILD	100nm	20 μm	≈1.5 W/m-K
Bottom Metal	<1μm	12μm	12 W/m-K
Bottom Silicon	100μm	100μm	120 W/m-K
TIM	50 μm	50 μm	5 W/m-K
IHS	3.0x3.0x0.1 cm^3	3.0x3.0x0.1 cm^3	400 W/m-K
HeatSink	6.0x6.0x0.7cm^3	6.0x6.0x0.7cm^3	400 W/m-K

We use Hotspot's extension [137, 138] to model the effects of heterogeneous components in the same layer. We model both lateral and vertical thermal conduction using the more accurate grid-model. We first obtain the steady state temperature and then run the trace to obtain the operating temperature.

We evaluate M3D and TSV3D designs for both single cores and multicores. For the single-core experiments, we run 21 SPEC2006 applications. For the multicore experiments, we run 12 SPLASH2 applications and 3 PARSEC ones; the same applications used for the evaluation of HetCore in Section 3.6.

4.6.1 Architecture Configurations Evaluated

We compare a variety of architecture configurations. In the following, we refer to M3D with same-performance layers as *iso-layer* M3D, and M3D with different-performance layers as *hetero-layer* M3D.

2D Baseline. Traditionally, the clock cycle time of a microprocessor has been limited by the wakeup plus select operations in the issue stage, or by the ALU plus bypass paths [45]. Reduction in these stage latencies has been used by previous work [105], as the potential frequency gains. However, in recent processors, the wakeup and select steps are split into two stages [44]. Further, the register file access has emerged as a key bottleneck for wide-issue cores in addition to the ALU plus bypass paths. Of all the core structures we discussed, we measure with CACTI that the one that limits the core cycle time is the access time

of the register file at 293ps. Hence, we set the frequency of our *Base* 2D core to 3.3 GHz (Table 4.11).

Table 4.11: Core configurations evaluated.

Name	**Configuration**
Single Core	
Base	Baseline 2D, f=3.3GHz
M3D-Iso	Iso-layer M3D, f=3.83GHz
M3D-HetNaive	Hetero-layer M3D without modifications, f=3.5GHz
M3D-Het	Hetero-layer M3D with our modifications, f=3.76GHz
M3D-HetAgg	Aggressive M3D-Het, f=4.34GHz
TSV3D	Conventional TSV3D, f=3.3GHz
MultiCore	
M3D-Het	M3D-Het + Shared L2, 4 cores, f=3.76GHz
M3D-Het-W	M3D-Het + Shared L2, Issue width=8, 4 cores, f=3.3GHz
M3D-Het-2X	M3D-Het + Shared L2, 8 cores, f=3.3GHz, V_{dd}=0.75V
TSV3D	Conventional TSV3D + Shared L2, 4 cores, f=3.3GHz

Iso-layer M3D. Table 4.6 shows the reduction in the access latency of different structures in iso-layer M3D relative to 2D. We see that the RF and IQ access latency reduce by 41% and 26%, respectively. Further, in Section 4.3.1, we estimate that four ALUs with bypass paths can sustain a 28% higher frequency. Hence, the potential of M3D is very high. If we consider the traditional frequency-critical structures (similar to [105]), we find the frequency is limited by reductions of IQ access at 26%. This corresponds to 3.3/(1-0.26)=4.46 GHz. We call this design, *M3D-IsoAgg*, but do not evaluate due to space considerations.

However, to be very conservative, we assume that all the array structures in Table 4.6 are in the critical path. In particular, we assume that the BPT and BTB arrays need to be accessed in a single cycle. Based on this assumption, we identify the structure with the least reduction in access time, i.e., SQ and BPT with 14%. With this estimate, we set the frequency of our *M3D-Iso* core to 3.3/(1-0.14)=3.83GHz (Table 4.11).

TSV3D. The corresponding numbers for TSV in Table 4.6 are sometimes negative. Hence, TSVs may result in a core slowdown. The large footprint of TSVs makes intra-block 3D partitioning undesirable. Therefore, we keep the frequency of the *TSV3D* core the same as the 2D *Base* (Table 4.11). However, compared to 2D, we apply latency reductions in load-to-use and branch misprediction paths saving 1 cycle and 2 cycles of the 4 and 14 cycles respectively. M3D designs have same reductions.

Hetero-layer M3D. We consider three designs. The first one, called *M3D-HetNaive*, simply takes the *M3D-Iso* design and slows its frequency by 9% — which is the loss in frequency estimated by Shi et al. [4] due to the slower top layer (Section 4.2.4). Hence, we set the frequency of the *M3D-HetNaive* core to 3.83*0.91≈3.5GHz (Table 4.11).

The second design, called *M3D-Het*, is the result of our asymmetric partitioning of structures in Section 4.4. We consider all the array structures in Table 4.8, and take the one that reduces the access latency the least. Specifically, the SQ, LQ, BPT, and BTB only reduce the access latency by 13% relative to 2D. Consequently, we set the frequency of the *M3D-Het* core to 3.3/(1-0.13)≈3.76GHz (Table 4.11). Finally, we evaluate another design, *M3D-HetAgg* that is derived in a manner similar to M3D-IsoAgg. In this case, the frequency is limited by the reductions in IQ access at 24%. The corresponding frequency is 3.3/(1-0.24)≈4.34GHz (Table 4.11).

Advanced hetero-layer M3D. We consider several multi-core designs, shown in Table 4.11. In this case, pairs of cores share their L2 cache as in Figure 4.4. We consider a 4-core *M3D-Het* and two related designs we describe next: a 4-core *M3D-Het-W* (where W stands for wide) and an 8-core *M3D-Het-2X* (where 2X stands for twice the cores). We also evaluate a 4-core *TSV3D*.

To configure M3D-Het-W, we take M3D-Het, set its frequency to that of the 2D Base core (3.3GHz), and increase the core's width as much as possible. The maximum width is 8. To configure M3D-Het-2X, we again take M3D-Het and set its frequency to 3.3GHz, and reduce the voltage as much as possible. Following curves from the literature [25, 19], the maximum reduction is 50mV, which sets the voltage to 0.75V. At this point the multicore consumes relatively little power. Hence, we increase the number of cores as much as possible until it reaches the same power consumption as the 2D *Base*. The number of cores is in between 7 and 8. We pick 8 as some parallel applications require a power-of-two core count.

4.7 EVALUATION

4.7.1 M3D Performance, Energy and Thermals

Speedup Figure 4.7 shows the speedup of different M3D designs for benchmarks from the SPEC2006 suite over the *Base*. An iso-layer M3D design *M3D-Iso*, in which both layers have the same performance, is on average 27% faster than Base across the applications. The performance improvement is due to two reasons. First, M3D-Iso executes at a 16% higher frequency over Base. Second, some of the critical pipeline paths, specifically, *load-to-use*

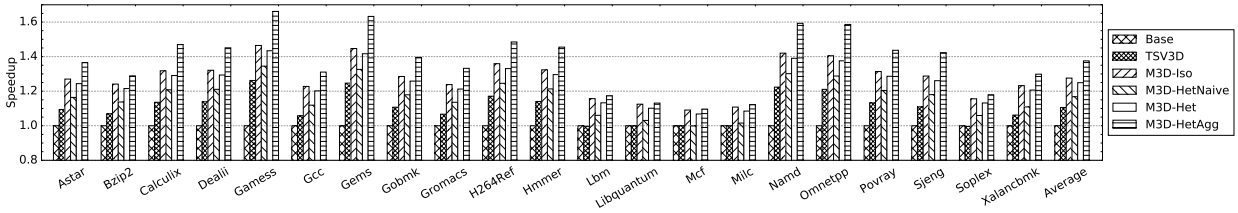

Figure 4.7: Speedup of different M3D designs over the Base (2D).

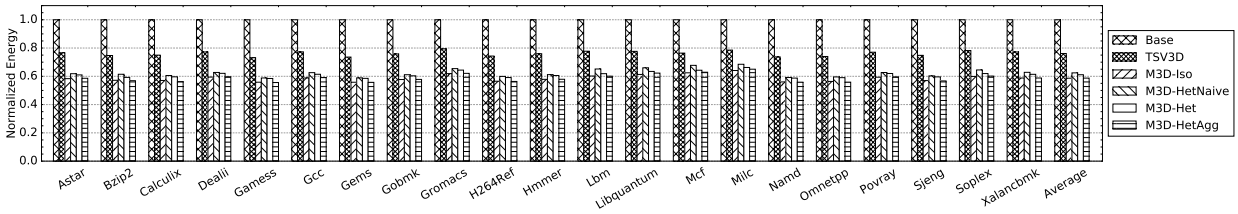

Figure 4.8: Energy of different M3D designs, normalized to Base (2D).

and *branch-misprediction* are shorter. Therefore the IPC is higher as well. Note that this is despite the increase in memory latency in terms of core clocks. *M3D-HetNaive*, with a slower top layer, has a higher stage delay, and operates only at 6% higher frequency than Base. Therefore the speedup is only 17% over the Base.

The critical path optimizations that we proposed in Section 4.4 prove useful for *M3D-Het*. The average stage delay is now 13% less than Base. As a result M3D-Het, with only 2% lower frequency than M3D-Iso, recovers most of the performance lost due to the slower top layer. Overall, M3D-Het is, on average, 25% faster than the Base. Finally, the aggressive M3D configuration, *M3D-HetAgg*, operating at 31% higher frequency than the Base, provides a speedup of 38% over the Base on average. This improvement is substantial.

TSV3D, which operates at the same frequency as Base, is only 10% faster than Base. The gains are due to the reduction in critical path latencies that we discussed above.

Energy Figure 4.8 shows the energy consumption of different designs normalized to the Base. M3D designs' consume lower energy due to a few factors. First, the power consumption of many SRAM structures is significantly smaller (Table 4.6). Second, the clock tree network's footprint is only half that of a 2D design. Finally, they execute faster saving leakage energy on top of the previous reductions. As a result of these different factors, M3D-Iso design consumes 41% lower energy than the Base design.

A simple M3D-HetNaive design has a similar power consumption as the M3D-Iso design, but executes longer. Therefore, the energy consumed is 3 percentage points higher. The

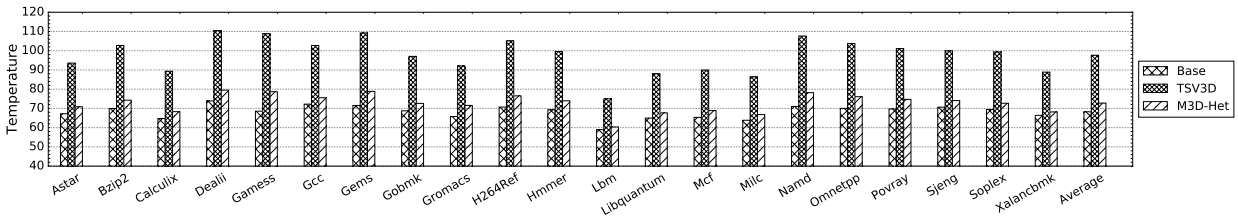

Figure 4.9: Peak Temperature of different CPU designs.

performance oriented design decisions made for M3D-Het increase the power consumption of several structures. However, the power consumption increases only in the top layer and only due to larger transistors. Further, it executes faster than M3D-HetNaive. As a result, the total energy consumption decreases slightly when compared to M3D-HetNaive and is 2 percentage points higher than iso-layer M3D-Iso design. When compared to the 2D Base design, the overall energy is reduced by 39%. The more aggressive M3D-HetAgg executes faster and lowers the energy consumption further by 2 percentage points bringing the total energy savings to 41%.

In comparison, energy reductions from TSV3D are smaller at 24%. Similar to M3D designs, the energy savings in TSV3D design are due to the reductions in SRAM arrays and clock tree power. However, the magnitude of array savings is smaller (Table 4.6).

Hetero-layers using LP Process for Top layer As we discussed in Section 4.5.2, when it is feasible to manufacture M3D chips with iso-performance in both layers, we can utilize our techniques to reduce energy consumption by having the top layer in LP process and the bottom layer in HP process. Such a design will have the same performance as M3D-Het. We evaluate that this design reduces the energy further by 9 percentage points over the M3D-Het design. Therefore, if the manufacturing process doesn't pose any limitations on top layer performance, the techniques we described can be used to obtain additional power savings.

Thermal Behavior To study the thermal behavior of M3D designs, we measure the maximum temperature reached by each design using the hotspot tool as discussed in Section 4.6. The average power consumption across all applications in Base is 6.4W for a single core excluding the L2/L3 caches. The floorplan our chip is based on AMD Ryzen [78]. We conservatively assume 50% footprint reduction only for calculating the peak temperature.

Figure 4.9 shows the peak temperature reached within the core across all the applications for Base, M3D-Het and TSV3D designs. The values for other M3D designs are very similar.

The hottest point in the core varies across applications. For example, the hottest point is in Issue Queue for dealII, whereas, it is in FPU for gems. The peak temperature in M3D-Het is on average 5°C higher than Base. The maximum increase across any application is 10°C, and is observed in the Issue Queue for Gamess. These are considerably smaller than the 30°C average temperature increase observed for TSV3D. In fact, TSV3D gets extremely hot exceeding T_{jmax} for a few applications. The high temperatures in TSV3D are due to the thermal conduction bottlenecks [107, 105].

The temperature increases in M3D-Het are small due to two reasons. First, as we discussed in Section 4.2, the vertical thermal conduction across the layers is very high. Second, in M3D-Het, we observe that some of the hotspots such as IQ, RAT, RF have relatively larger power reductions. For example, IQ consumes 34% less power in M3D-Het, which is higher than the 24% power reduction for the whole core. Therefore, even though the overall power density of M3D increases by up to 52%, the increase in power density of hotter regions is smaller (32% in this example). Overall, M3D design is thermally much more efficient than TSV3D.

4.7.2 Multicore Design Choices

In this section, we explore different multicore designs enabled by M3D, as we discussed in Sections 4.3, 4.5.1 and 4.6 using parallel applications. These are (i) M3D-Het with shared L2 cache/NoC router stop, (ii) M3D-Het with wider issue cores and (iii) Operating more cores at iso-frequency and power. The bars M3D-Het, M3D-Het-W and M3D-Het-2X in Figures 4.10 and 4.11 represent these configurations respectively. The speedup and energy consumption are normalized to the Base. We also show TSV3D for comparison.

Multicore M3D and Wider Issue M3D M3D-Het design provides a speedup of 26% while reducing energy consumption by 33% over the Base. The performance gains include the benefits of a shared L2 and NoC router on top of the gains seen in single core environment. With respect to the energy, we observe a slight reduction in dynamic power due to the reduction in the network traffic in addition to the energy saving factors discussed previously.

The bars titled *M3D-Het-W* in Figures 4.10 and 4.11 show the speedup and energy consumption of a core whose issue width is increased from 6 to 8 while operating at the same frequency as the Base (Section 4.6.1). Overall, the average speedup and the reduction in energy consumption are 25% and 27% respectively. These are lower than M3D-Het which simply increases the frequency.

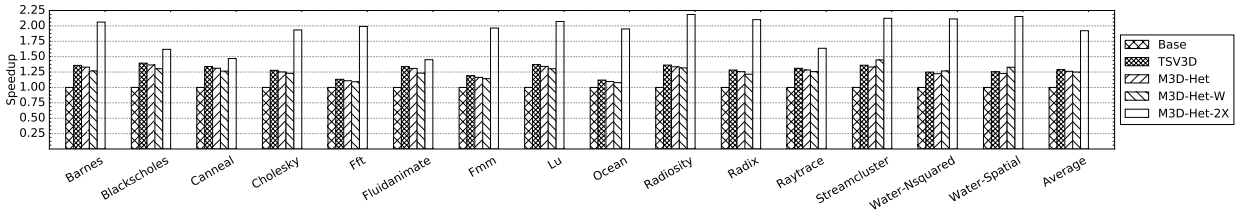

Figure 4.10: Speedup of MultiCore CPU designs, over the Base (2D).

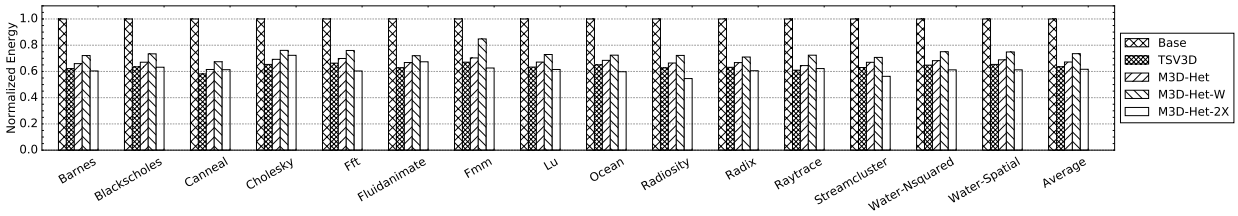

Figure 4.11: Energy of MultiCore CPU designs, normalized to Base (2D).

Iso Power Environment The *M3D-Het-2X* configuration builds upon M3D-Het design. Instead of increasing the frequency or making the core wider, M3D-Het-2X operates at iso-frequency at a reduced voltage. At this lower voltage and power, it can operate more cores within the same power budget and power density. In particular, we find that M3D-Het can operate $\approx 2X$ as many cores as Base for the same power. The design is shown in Figures 4.10 and 4.11 as *M3D-Het-2X*. The speedup of the parallel applications employing twice as many cores in *M3D-Het-2X* is 92% over the Base. At the same time, it consumes 39% lower energy. Overall, the power consumption is higher than the Base at 13%. Note that these results are for the hetero-layer design that includes the slowdown of the top layer with conservative assumptions. In the absence of this degradation, a similar *M3D-Iso-2X* design consumes 4% percentage points lower energy than the Base and consumes only 5% more power than the Base, while operating twice as many cores.

Overall, M3D integration presents exciting opportunities for core partitioning and design. Of the several design choices that we evaluated for a multicore environment, M3D-Het-2X, operating twice as many cores under the same power budget and at the same frequency as Base, provides the best energy efficiency as well as speedup.

4.8 SUMMARY

In this work, we analyzed Monolithic3D, a promising 3D integration technology, that has area, latency, power and thermal benefits over die stacking with TSVs. This is the first work

to partition a processor for M3D. We partitioned logic and storage structures into two layers, taking into account that the top layer has lower-performance transistors. For logic structures, we place the critical paths in the bottom layer. For storage structures, we asymmetrically partition them, assigning to the top layer fewer ports with larger access transistors, or a shorter bitcell subarray with larger bitcells. With very conservative assumptions on M3D technology, the M3D core executed applications on average 25% faster than a 2D core while consuming 39% less energy. The aggressive design was 38% faster while consuming 41% lower energy than a 2D core. Moreover, under a similar power budget, an M3D multicore could use twice as many cores as one with 2D cores, effectively executing applications on average 92% faster with 39% less energy. Finally, the M3D core was thermally efficient, with the peak temperatures marginally higher than a 2D core and substantially lower than TSV3D design.

CHAPTER 5: RELATED WORK

5.1 BUILDING VOLTAGE SCALABLE CORES

The prior work related to ScalCore can be divided into two broad categories for comparison. The first category summarizes designs using variable latency and resizing techniques while the other one summarizes designs focused on scaling cores to the NTV domain.

There are several prior works on variable-latency pipelines within a core. For example, when process variation makes a pipeline stage too slow, ReCycle [134] uses cycle-time stealing between stages, and ReVIVaL [139] makes a transparent latch opaque. On the other hand, Collapsible Pipelines [49] dynamically make a latch transparent when there is a bubble in the pipeline. Pipeline Stage Unification [50] combines every two or every four pipeline stages when the frequency is low. In this work, we focus only on combining storage-intensive stages under dual-V_{dd} pipelines. Another variable latency pipeline technique is GALS [140, 141]. GALS operates different stages at different f, while we keep the same f for all stages.

Many works propose reconfigurable architectures that adapt to resource occupancy (e.g., [142, 143, 144]). The goal is to improve either performance or power efficiency. While these ideas are similar to ours, we apply the changes only to adapt to a low-V_{dd} mode, and have only two settings. In general, these techniques and our work are complementary.

There is past work on dual-V_{dd} architectures. Dreslinski et al. [37] apply a different V_{dd} to the core and to the caches. The same approach is followed in the Intel Claremont prototype [25, 26]. A detailed comparison to Claremont is provided in Section 2.5.3. Miller et al. [145, 146] provide two V_{dd} rails and allow a core to dynamically switch between them. In the presence of process variation, this technique hides speed heterogeneity and tolerates slow functional units. Our work is different in that different stages of the same pipeline have different V_{dd}s. In addition, pipeline stages do not switch between V_{dd} rails; ScalCore merely changes the rails' V_{dd}.

5.2 BUILDING CORES USING CMOS-TFET DEVICES

We classify the prior work related to HetCore into (i) heterogeneous systems that utilize CMOS and TFET cores and (ii) techniques that exploit the combination of fast and slow units. HetCore employs several techniques of the second kind that have been well studied in prior work and adapts them to design a core that is better than a baseline hetero-device core, BaseHet resulting in AdvHet.

Prior work has studied the integration of some TFET cores and some CMOS cores in the same chip [69], as well as in a 3D-stack [68]. HetCore pushes device heterogeneity inside the core for a more energy-efficient design. In [70], the authors proposed a barrier-aware thread migration scheme to move threads from a TFET core to a CMOS core and vice-versa to minimize the ED. We performed an iso-area comparison with such barrier-aware thread migration scheme. It can be shown that AdvHet provides, on average, higher performance while consuming lower energy. This is because, in [70], the threads on the TFET cores slow down the program, while the threads on the CMOS cores consume more power than in AdvHet.

Several researchers (e.g., [85, 86, 87, 12, 63]) have developed TFET or mixed TFET-CMOS circuits, including SRAM cells, that lay the groundwork for building TFET-CMOS hybrid cores. We discussed such work in Sections 3.2 and 3.3. HetCore builds upon such work by proposing CPU and GPU designs that use CMOS- and TFET-based units inside a single core to obtain a high performance core that also has high energy efficiency. In addition, one can envision a heterogeneous chip that consists of a mix of TFET cores and HetCores that can take advantage of the prior work.

The architectural techniques employed by AdvHet to alleviate the performance penalties present in BaseHet have been proposed and studied in prior work in other contexts. Such techniques include designs similar to the asymmetric data cache [92, 93, 94], dual-speed clusters as a mechanism to reduce power consumption [96, 97, 98], and register file caches for GPUs [91]. In this work, we adapt them to the context of heterogeneous-device cores.

As an alternative to using a register file cache, a partitioned register file for GPUs is proposed in [147]. It consists of a fast partition operating at nominal voltage and a slow partition operating at near-threshold voltage. Such a design can readily be adapted to AdvHet, by implementing the slow partition in TFET and the fast one in CMOS. Similar complementary optimizations can indeed be applied to AdvHet resulting in even better energy efficiency.

5.3 3D PARTITIONING OF CORES

Prior architectural work on partitioning for TSV3D has examined placing cores on top of cores [16], block level partitioning [15], and intra-block partitioning [105, 125, 111, 124]. We discussed such work on intra-block 3D partitioning of core in Section 4.2.3.

In addition to the TSV based intra-block 3D partitioning , prior work has also considered partitioning the core at a block-level granularity using TSVs. Black et al. [15] study the benefits of such core partitioning as well as benefits of placing DRAM/SRAM on top of a

core. They observe that 3D core can reduce some of the critical paths with in the core such as load-to-use delay and branch misprediction delay. DRAM on top of the core can reduce the power requirements of memory and provide higher bandwidth. They also note that a TSV3D based core can have thermal challenges. We compared the M3D partitioned core against a finer partitioned TSV3D core which in turn has more benefits than a block level partitioned core.

Emma et al. [16] limit themselves to a core level partitioning across the different layers and share the different resources such as caches, NoC etc. They focus on the impact of 3D partitioning from a thermal and yield perspective and discuss the tradeoffs between the power and performance in 3D setting. Our analysis of M3D core design is at a much finer granularity and within the core.

CHAPTER 6: CONCLUSION

In this thesis, we presented the design of energy efficient cores that are suited for the upcoming process technology trends. The designs took advantage of process technologies that are currently available as well as ones that would be feasible in the medium to long term.

In the first part of thesis, we presented the design of a voltage scalable core — namely, one that can work in high-performance mode (HPMode) at nominal V_{dd}, and in a very energy-efficient mode (EEMode) at low V_{dd}. ScalCore introduced two ideas to operate energy-efficiently in EEMode. First, it applied two low V_{dd}s to the pipeline: one to the logic stages (V_{logic}) and a higher one to the storage-intensive stages. Second, it increased the low V_{dd} of the storage-intensive stages (V_{op}) even further and exploited the speed differential to the logic ones by either fusing storage-intensive pipeline stages or increasing the size of storage structures in the pipeline. Simulations of 16 cores showed that a design with ScalCores in EEMode is much more energy-efficient than one with conventional cores and aggressive DVFS: for approximately the same power consumption, ScalCores reduced the average execution time of programs by 31%, the energy consumed by 48%, and the ED product by 60%. In addition, dynamically switching between EEMode and HPMode is very effective: it reduced the average execution time and ED product by an additional 28% and 15%, respectively, over running in EEMode all the time.

In the next part, we considered another upcoming technology i.e. TFETs. While TFETs are much more energy efficient than CMOS, they are not competitive in terms of the performance. Ideally, we desire CPU and GPU cores that operate as energy-efficiently as a TFET core, while providing the performance of a CMOS core. To this end, we proposed the HetCore architecture, which judiciously integrates both TFET and CMOS units in a single core, creating a hetero-device core. Our results show that such a design is very promising, even with conservative assumptions. An AdvHet CPU consumes on average 39% less energy than a CMOS CPU, while delivering a performance that is within 10% of the CMOS CPU. In addition, under a fixed power budget, a multicore with AdvHet CPUs attains average performance gains of 32% over a multicore with CMOS CPUs, while reducing ED^2 by 68%. Similarly, an AdvHet GPU consumes on average 40% less energy and performs within 20% of a CMOS GPU. Under a fixed power budget, an AdvHet GPU with twice as many compute units as a CMOS GPU improves average performance by 30% while lowering ED^2 by 60%.

Finally, we analyzed Monolithic3D technology, that is by a broad consensus considered the most promising approach to continue increasing transistor integration. We presented the first

analysis of the architecture implications of using Monolithic3D, and showed how to partition a processor for M3D. We partitioned logic and storage structures into two layers, taking into account that the top layer has lower-performance transistors. For logic structures, we placed the critical paths in the bottom layer. For storage structures, we asymmetrically partitioned them, assigning to the top layer fewer ports with larger access transistors, or a shorter bitcell subarray with larger bitcells. With very conservative assumptions on M3D technology, the M3D core executed applications on average 25% faster than a 2D core while consuming 39% less energy. The aggressive design was 38% faster while consuming 41% lower energy than a 2D core. Moreover, under a fixed power budget, an M3D multicore could use twice as many cores as one with 2D cores, effectively executing applications on average 92% faster with 39% less energy. Finally, the M3D core was thermally efficient with the peak temperatures marginally higher than a 2D core and substantially lower than TSV3D design.

In summary, we presented the design of cores for three upcoming process technology trends. The process trends that we considered span a range of timelines, beginning with the most current one i.e. voltage scalable cores. We later considered TFETs that are expected to supplement the current CMOS devices in the near future. Finally, we studied Monolithic3D integration technology which is necessary to continue integrating more transistors on a chip. This is expected to be realized at a slightly farther point in time. For each of the technologies, we studied the properties in detail, analyzed the implications on architecture and proposed a core design that maximizes the energy efficiency. We further evaluated all these designs based on conservative assumptions on the technology parameters. In particular, we showed that all of our proposed designs excel in an environment with a fixed power budget. Across all our designs, we were able to operate twice as many cores in the same power budget as an existing baseline design, while consuming lower energy at the same time. Our results show that fine tuning core architectures to specific technology is very beneficial and can provide significantly more energy efficient cores. We believe that our work will lay the foundation for more exciting work in this domain and serve as a stepping stone in our quest to build ever more energy efficient cores without compromising the performance.

REFERENCES

[1] D. E. Nikonov and I. A. Young, "Benchmarking of Beyond-CMOS Exploratory Devices for Logic Integrated Circuits," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, Dec 2015.

[2] D. E. Nikonov and I. A. Young, "Overview of Beyond-CMOS Devices and a Uniform Methodology for Their Benchmarking," *Proceedings of the IEEE*, Dec 2013.

[3] G. V. der Plas, P. Limaye, I. Loi, A. Mercha, H. Oprins, C. Torregiani, S. Thijs, D. Linten, M. Stucchi, G. Katti, D. Velenis, V. Cherman, B. Vandevelde, V. Simons, I. D. Wolf, R. Labie, D. Perry, S. Bronckers, N. Minas, M. Cupac, W. Ruythooren, J. V. Olmen, A. Phommahaxay, M. de Potter de ten Broeck, A. Opdebeeck, M. Rakowski, B. D. Wachter, M. Dehan, M. Nelis, R. Agarwal, A. Pullini, F. Angiolini, L. Benini, W. Dehaene, Y. Travaly, E. Beyne, and P. Marchal, "Design Issues and Considerations for Low-Cost 3-D TSV IC Technology," *IEEE Journal of Solid-State Circuits*, Jan 2011.

[4] J. Shi, D. Nayak, S. Banna, R. Fox, S. Samavedam, S. Samal, and S. K. Lim, "A 14nm FinFET transistor-level 3D partitioning design to enable high-performance and low-cost monolithic 3D IC," in *2016 IEEE International Electron Devices Meeting (IEDM)*, Dec 2016, pp. 2.5.1–2.5.4.

[5] S. V. Huylenbroeck, M. Stucchi, Y. Li, J. Slabbekoorn, N. Tutunjyan, S. Sardo, N. Jourdan, L. Bogaerts, F. Beirnaert, G. Beyer, and E. Beyne, "Small Pitch, High Aspect Ratio Via-Last TSV Module," in *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)*, May 2016, pp. 43–49.

[6] U. E. Avci, D. H. Morris, and I. A. Young, "Tunnel Field-Effect Transistors: Prospects and Challenges," *IEEE J-EDS*, May 2015.

[7] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark Silicon and the End of Multicore Scaling," in *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ser. ISCA '11. New York, NY, USA: ACM, 2011. [Online]. Available: http://doi.acm.org/10.1145/2000064.2000108 pp. 365–376.

[8] H. Kaul, M. Anders, S. Hsu, A. Agarwal, R. Krishnamurthy, and S. Borkar, "Near-threshold Voltage (NTV) Design: Opportunities and Challenges," in *DAC*, June 2012.

[9] A. M. Ionescu and H. Riel, "Tunnel field-effect transistors as energy-efficient electronic switches," *Nature*, Nov 2011. [Online]. Available: http://dx.doi.org/10.1038/nature10679

[10] S. Datta, G. Dewey, J. M. Fastenau, M. K. Hudait, D. Loubychev, W. K. Liu, M. Radosavljevic, W. Rachmady, and R. Chau, "Ultrahigh-Speed 0.5 V Supply Voltage In0.7 Ga0.3As Quantum-Well Transistors on Silicon Substrate," *IEEE Electron Device Letters*, Aug 2007.

[11] R. Asra, M. Shrivastava, K. V. R. M. Murali, R. K. Pandey, H. Gossner, and V. R. Rao, "A Tunnel FET for Vdd Scaling Below 0.6V With a CMOS-Comparable Performance," *IEEE T-ED*, July 2011.

[12] Y. N. Chen, M. L. Fan, V. P. H. Hu, P. Su, and C. T. Chuang, "Evaluation of Stability, Performance of Ultra-Low Voltage MOSFET, TFET, and Mixed TFET-MOSFET SRAM Cell With Write-Assist Circuits," *IEEE JETCAS*, Dec 2014.

[13] P. Batude, M. Vinet, C. Xu, B. Previtali, C. Tabone, C. L. Royer, L. Sanchez, L. Baud, L. Brunet, A. Toffoli, F. Allain, D. Lafond, F. Aussenac, O. Thomas, T. Poiroux, and O. Faynot, "Demonstration of low temperature 3D sequential FDSOI integration down to 50 nm gate length," in *2011 Symposium on VLSI Technology - Digest of Technical Papers*, June 2011.

[14] International Technology Roadmap for Semiconductors (ITRS), "ITRS 2.0." [Online]. Available: http://www.itrs2.net.

[15] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. H. Loh, D. McCaule, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb, "Die Stacking (3D) Microarchitecture," in *2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*, Dec 2006, pp. 469–479.

[16] P. Emma, A. Buyuktosunoglu, M. Healy, K. Kailas, V. Puente, R. Yu, A. Hartstein, P. Bose, and J. Moreno, "3D stacking of high-performance processors," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2014, pp. 500–511.

[17] P. Batude, B. Sklenard, C. Fenouillet-Beranger, B. Previtali, C. Tabone, O. Rozeau, O. Billoint, O. Turkyilmaz, H. Sarhan, S. Thuries, G. Cibrario, L. Brunet, F. Deprat, J. E. Michallet, F. Clermidy, and M. Vinet, "3D sequential integration opportunities and technology optimization," in *IEEE International Interconnect Technology Conference*, May 2014, pp. 373–376.

[18] L. Brunet, P. Batude, C. Fenouillet-Beranger, P. Besombes, L. Hortemel, F. Ponthenier, B. Previtali, C. Tabone, A. Royer, C. Agraffeil, C. Euvrard-Colnat, A. Seignard, C. Morales, F. Fournel, L. Benaissa, T. Signamarcheix, P. Besson, M. Jourdan, R. Kachtouli, V. Benevent, J. . Hartmann, C. Comboroure, N. Allouti, N. Posseme, C. Vizioz, C. Arvet, S. Barnola, S. Kerdiles, L. Baud, L. Pasini, C. . V. Lu, F. Deprat, A. Toffoli, G. Romano, C. Guedj, V. Delaye, F. Boeuf, O. Faynot, and M. Vinet, "First demonstration of a CMOS over CMOS 3D VLSI CoolCube^{TM} integration on 300mm wafers," in *2016 IEEE Symposium on VLSI Technology*, June 2016, pp. 1–2.

[19] B. Gopireddy, C. Song, J. Torrellas, N. Kim, A. Agrawal, and A. Mishra, "ScalCore: Designing a Core for Voltage Scalability," in *HPCA*, March 2016.

[20] B. Gopireddy, D. Skarlatos, W. Zhu, and J. Torrellas, "HetCore: TFET-CMOS Hetero-Device Architecture for CPUs and GPUs," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, June 2018.

[21] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A dynamic voltage scaled microprocessor system," in *ISSCC*, Feb 2000.

[22] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for Reduced CPU Energy," in *OSDI*, 1994.

[23] R. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits," *Proceedings of the IEEE*, Feb 2010.

[24] E. Seevinck, F. List, and J. Lohstroh, "Static-noise margin analysis of MOS SRAM cells," *Solid-State Circuits, IEEE Journal of*, Oct 1987.

[25] S. Jain, S. Khare, S. Yada, V. Ambili, P. Salihundam, S. Ramani, S. Muthukumar, M. Srinivasan, A. Kumar, S. Gb, R. Ramanarayanan, V. Erraguntla, J. Howard, S. Vangal, S. Dighe, G. Ruhl, P. Aseron, H. Wilson, N. Borkar, V. De, and S. Borkar, "A 280mV-to-1.2V wide-operating-range IA-32 processor in 32nm CMOS," in *ISSCC*, Feb 2012.

[26] G. Ruhl, S. Dighe, S. Jain, S. Khare, and S. Vangal, "IA-32 Processor with a Wide-Voltage-Operating Range in 32-nm CMOS," *IEEE Micro*, 2013.

[27] G. Chen, D. Sylvester, D. Blaauw, and T. Mudge, "Yield-Driven Near-Threshold SRAM Design," *IEEE TVLSI*, Nov 2010.

[28] P. Greenhalgh, "Big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7," September 2011, ARM White Paper. [Online]. Available: http://www.arm.com/files/downloads/big_LITTLE_Final_Final.pdf

[29] The Tech Report, "The Exynos 5433 SoC." [Online]. Available: http://techreport.com/review/27539/samsung-galaxy-note-4-with-the-exynos-5433-processor/2

[30] H. D. Cho, K. Chung, and T. Kim, "Benefits of the big.LITTLE Architecture," February 2012, Samsung White Paper. [Online]. Available: http://www.arm.com/files/downloads/Benefits_of_the_big.LITTLE_architecture.pdf

[31] L. Chang, D. J. Frank, R. K. Montoye, S. J. Koester, B. L. Ji, P. W. Coteus, R. H. Dennard, and W. Haensch, "Practical Strategies for Power-Efficient Computing Technologies," *Proceedings of the IEEE*, February 2010.

[32] D. Markovic, C. C. Wang, L. P. Alarcon, T.-T. Liu, and J. M. Rabaey, "Ultralow-Power Design in Near-Threshold Region," *Proceedings of the IEEE*, February 2010.

[33] N. S. Kim, S. Draper, S.-T. Zhou, S. Katariya, H. R. Ghasemi, and T. Park, "Analyzing the Impact of Joint Optimization of Cell Size, Redundancy, and ECC on Low-Voltage SRAM Array Total Area," in *VLSI*, December 2012.

[34] S. Hsu, A. Agarwal, M. Anders, H. Kaul, S. Mathew, F. Sheikh, R. Krishnamurthy, and S. Borkar, "A 2.8GHz 128-Entry 152b 3-Read/2-Write Multi-Precision Floating-Point Register File and Shuffler in 32nm CMOS," in *VLSIC*, June 2012.

[35] Y. Zhao, J. Li, and K. Mohanram, "Multi-port FinFET SRAM Design," in *GLSVLSI*, New York, NY, USA.

[36] M. Sinangil, N. Verma, and A. Chandrakasan, "A Reconfigurable 8T Ultra-Dynamic Voltage Scalable (U-DVS) SRAM in 65 nm CMOS," *Solid-State Circuits, IEEE Journal of*, Nov 2009.

[37] R. Dreslinski, B. Zhai, T. Mudge, D. Blaauw, and D. Sylvester, "An Energy Efficient Parallel Architecture Using Near Threshold Operation," in *PACT*, Sept 2007.

[38] S. Li, J.-H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *MICRO*, Dec 2009.

[39] U. R. Karpuzcu, K. B. Kolluru, N. S. Kim, and J. Torrellas, "VARIUS-NTV: A Microarchitectural Model to Capture the Increased Sensitivity of Manycores to Process Variations at Near-Threshold Voltages," in *DSN*, June 2012.

[40] U. R. Karpuzcu, A. Sinkar, N. S. Kim, and J. Torrellas, "EnergySmart: Toward Energy-Efficient Manycores for Near-Threshold Computing," in *HPCA*, February 2013.

[41] C.-H. Jan, U. Bhattacharya, R. Brain, S.-J. Choi, G. Curello, G. Gupta, W. Hafez, M. Jang, M. Kang, K. Komeyli, T. Leo, N. Nidhi, L. Pan, J. Park, K. Phoa, A. Rahman, C. Staus, H. Tashiro, C. Tsai, P. Vandervoorn, L. Yang, J.-Y. Yeh, and P. Bai, "A 22nm SoC platform technology featuring 3-D tri-gate and high-k/metal gate, optimized for ultra low power, high performance and high density SoC applications," in *IEDM*, Dec 2012.

[42] E. Karl, Z. Guo, J. Conary, J. Miller, Y.-G. Ng, S. Nalam, D. Kim, J. Keane, U. Bhattacharya, and K. Zhang, "A 0.6V 1.5GHz 84Mb SRAM Design in 14nm FinFET CMOS Technology," in *ISSCC*, Feb 2015.

[43] M. Qazi, M. Sinangil, and A. Chandrakasan, "Challenges and Directions for Low-Voltage SRAM," *Design Test of Computers, IEEE*, Jan 2011.

[44] A. Gonzalez, F. Latorre, and G. Magklis, "Processor Microarchitecture: An Implementation Perspective," *Synthesis Lectures on Computer Architecture*, 2010.

[45] S. Palacharla, N. P. Jouppi, and J. E. Smith, "Complexity-effective Superscalar Processors," in *ISCA*, 1997.

[46] J. Sartori, B. Ahrens, and R. Kumar, "Power balanced pipelines," in *HPCA*, Feb 2012.

[47] I. Burcea and A. Moshovos, "Phantom-BTB: A Virtualized Branch Target Buffer Design," in *ASPLOS*.

[48] M. Evers, P.-Y. Chang, and Y. N. Patt, "Using Hybrid Branch Predictors to Improve Branch Prediction Accuracy in the Presence of Context Switches," in *ISCA*.

[49] H. M. Jacobson, "Improved Clock-Gating Through Transparent Pipelining," in *ISLPED*, August 2004.

[50] H. Shimada, H. Ando, and T. Shimada, "Pipeline Stage Unification: A Low-Energy Consumption Technique for Future Mobile Processors," in *ISLPED*, August 2003.

[51] A. Buyuktosunoglu, D. Albonesi, S. Schuster, D. Brooks, P. Bose, and P. Cook, "A Circuit Level Implementation of an Adaptive Issue Queue for Power-aware Microprocessors," in *GLSVLSI*, 2001.

[52] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "CACTI 6.0: A tool to understand large caches."

[53] I. Park, C. L. Ooi, and T. N. Vijaykumar, "Reducing Design Complexity of the Load/Store Queue," in *MICRO*, 2003.

[54] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: a tutorial and survey," *Solid-State Circuits, IEEE Journal of*, March 2006.

[55] F. Ishihara, F. Sheikh, and B. Nikolic, "Level conversion for dual-supply systems," *IEEE TVLSI*, 2004.

[56] K. U. Mutsunori, M. Igarashi, T. Ishikawa, M. Kanazawa, M. Takahashi, M. Hamada, H. Arakida, T. Terazawa, and T. Kuroda, "Design Methodology of Ultra Low-power MPEG4 Codec Core Exploiting Voltage Scaling Techniques," in *DAC*, June 1998.

[57] D. M. Tullsen, S. J. Eggers, J. S. Emer, H. M. Levy, J. L. Lo, and R. L. Stamm, "Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor," in *ISCA*.

[58] J. Renau, B. Fraguela, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, S. Sarangi, P. Sack, K. Strauss, and P. Montesinos, "SESC simulator," 2005.

[59] H. Riel, K. E. Moselund, C. Bessire, M. T. Bjork, A. Schenk, H. Ghoneim, and H. Schmid, "InAs-Si heterojunction nanowire tunnel diodes and tunnel FETs," in *IEDM*, Dec 2012.

[60] H. Schmid, K. E. Moselund, M. T. Bjork, M. Richter, H. Ghoneim, C. D. Bessire, and H. Riel, "Fabrication of vertical InAs-Si heterojunction tunnel field effect transistors," in *DRC*, June 2011.

[61] M. Alioto and D. Esseni, "Tunnel FETs for Ultra-Low Voltage Digital VLSI Circuits: Part II - Evaluation at Circuit Level and Design Perspectives," *IEEE TVLSI*, Dec 2014.

[62] M. Fulde, A. Heigl, M. Weis, M. Wirnshofer, K. v. Arnim, T. Nirschl, M. Sterkel, G. Knoblinger, W. Hansch, G. Wachutka, and D. Schmitt-Landsiedel, "Fabrication, optimization and application of complementary Multiple-Gate Tunneling FETs," in *International Nanoelectronics Conference*, March 2008.

[63] M. Lanuzza, S. Strangio, F. Crupi, P. Palestri, and D. Esseni, "Mixed Tunnel-FET/MOSFET Level Shifters: A New Proposal to Extend the Tunnel-FET Application Domain," *IEEE T-ED*, Dec 2015.

[64] R. Mukundrajan, M. Cotter, V. Saripalli, M. J. Irwin, S. Datta, and V. Narayanan, "Ultra Low Power Circuit Design Using Tunnel FETs," in *VLSI*, Aug 2012.

[65] D. Cavalheiro, F. Moll, and S. Valtchev, "TFET-Based Power Management Circuit for RF Energy Harvesting," *IEEE J-EDS*, Jan 2017.

[66] B. Sedighi, X. S. Hu, H. Liu, J. J. Nahas, and M. Niemier, "Analog Circuit Design Using Tunnel-FETs," *IEEE CAS*, Jan 2015.

[67] M. Hemmat, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Hybrid TFET-MOSFET circuits: An approach to design reliable ultra-low power circuits in the presence of process variation," in *VLSI-SoC*, Sept 2016.

[68] K. Swaminathan, H. Liu, J. Sampson, and V. Narayanan, "An Examination of the Architecture and System-level Tradeoffs of Employing Steep Slope Devices in 3D CMPs," in *ISCA*, 2014.

[69] V. Saripalli, A. Mishra, S. Datta, and V. Narayanan, "An energy-efficient heterogeneous CMP based on hybrid TFET-CMOS cores," in *DAC*, June 2011.

[70] K. Swaminathan, E. Kultursay, V. Saripalli, V. Narayanan, M. Kandemir, and S. Datta, "Improving Energy Efficiency of Multi-threaded Applications Using Heterogeneous CMOS-TFET Multicores," in *ISLPED*, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=2016802.2016859

[71] Q. Huang, R. Jia, C. Chen, H. Zhu, L. Guo, J. Wang, J. Wang, C. Wu, R. Wang, W. Bu, J. Kang, W. Wang, H. Wu, S. W. Lee, Y. Wang, and R. Huang, "First foundry platform of complementary tunnel-FETs in CMOS baseline technology for ultralow-power IoT applications: Manufacturability, variability and technology roadmap," in *IEDM*, Dec 2015.

[72] N. Mukherjee, J. Boardman, B. Chu-Kung, G. Dewey, A. Eisenbach, J. Fastenau, J. Kavalieros, W. K. Liu, D. Lubyshev, M. Metz, K. Millard, M. Radosavljevic, T. Stewart, H. W. Then, P. Tolchinsky, and R. Chau, "MOVPE III-V material growth on silicon substrates and its comparison to MBE for future high performance and low power logic applications," in *IEDM*, Dec 2011.

[73] R. Rooyackers, A. Vandooren, A. S. Verhulst, A. Walke, K. Devriendt, S. Locorotondo, M. Demand, G. Bryce, R. Loo, A. Hikavyy, T. Vandeweyer, C. Huyghebaert, N. Collaert, and A. Thean, "A new complementary hetero-junction vertical Tunnel-FET integration scheme," in *IEDM*, Dec 2013.

[74] G. Dewey, B. Chu-Kung, J. Boardman, J. Fastenau, J. Kavalieros, R. Kotlyar, W. Liu, D. Lubyshev, M. Metz, N. Mukherjee et al., "Fabrication, characterization, and physics of III–V heterojunction tunneling field effect transistors (H-TFET) for steep subthreshold swing," in *IEDM*, 2011.

[75] G. Zhou, R. Li, T. Vasen, M. Qi, S. Chae, Y. Lu, Q. Zhang, H. Zhu, J. M. Kuo, T. Kosel, M. Wistey, P. Fay, A. Seabaugh, and H. Xing, "Novel gate-recessed vertical InAs/GaSb TFETs with record high ION of 180 uA/um at VDS=0.5V," in *IEDM*, Dec 2012.

[76] M. G. Pala and S. Brocard, "Exploiting Hetero-Junctions to Improve the Performance of III-V Nanowire Tunnel-FETs," *IEEE J-EDS*, May 2015.

[77] E. Kultursay, K. Swaminathan, V. Saripalli, V. Narayanan, M. T. Kandemir, and S. Datta, "Performance Enhancement Under Power Constraints Using Heterogeneous CMOS-TFET Multicores," in *International Conference on Hardware/Software Codesign and System Synthesis*, 2012. [Online]. Available: http://doi.acm.org/10.1145/2380445.2380487

[78] "AMD Ryzen Micro Architecture," https://arstechnica.com/gadgets/2017/03/amds-moment-of-zen-finally-an-architecture-that-can-compete/, 2017, [Online].

[79] A. Calimera, R. I. Bahar, E. Macii, and M. Poncino, "Temperature-Insensitive Dual-V_{th} Synthesis for Nanometer CMOS Technologies Under Inverse Temperature Dependence," *IEEE TVLSI*, Nov 2010.

[80] X. Chen and N. K. Jha, "Ultra-low-leakage Chip Multiprocessor Design with Hybrid FinFET Logic Styles," *JETC*, Oct. 2014. [Online]. Available: http://doi.acm.org/10.1145/2629576

[81] T. Karnik, Y. Ye, J. Tschanz, L. Wei, S. Burns, V. Govindarajulu, V. De, and S. Borkar, "Total Power Optimization by Simultaneous dual-Vt Allocation and Device Sizing in High Performance Microprocessors," in *DAC*, 2002. [Online]. Available: http://doi.acm.org/10.1145/513918.514042

[82] A. Agarwal, K. Kang, S. Bhunia, J. D. Gallagher, and K. Roy, "Device-Aware Yield-Centric Dual-Vt Design Under Parameter Variations in Nanoscale Technologies," *IEEE TVLSI*, June 2007.

[83] N. Verma, "Analysis towards minimization of total SRAM energy over active and idle operating modes," *IEEE TVLSI*, 2011.

[84] B. Amelifard, F. Fallah, and M. Pedram, "Leakage Minimization of SRAM Cells in a Dual-V_t and Dual-T_{ox} Technology," *IEEE TVLSI*, 2008.

[85] J. Singh, K. Ramakrishnan, S. Mookerjea, S. Datta, N. Vijaykrishnan, and D. Pradhan, "A novel Si-Tunnel FET based SRAM design for ultra low-power 0.3V VDD applications," in *ASP-DAC*, Jan 2010.

[86] V. Saripalli, S. Datta, V. Narayanan, and J. P. Kulkarni, "Variation-tolerant ultra low-power heterojunction tunnel FET SRAM design," in *2011 IEEE/ACM International Symposium on Nanoscale Architectures*, June 2011.

[87] D. H. Morris, U. E. Avci, and I. A. Young, "Variation-tolerant dense TFET memory with low VMIN matching low-voltage TFET logic," in *Symposium on VLSI Technology*, June 2015.

[88] U. E. Avci, D. H. Morris, S. Hasan, R. Kotlyar, R. Kim, R. Rios, D. E. Nikonov, and I. A. Young, "Energy efficiency comparison of nanowire heterojunction TFET and Si MOSFET at Lg= 13nm, including P-TFET and variation considerations," in *IEDM*, 2013.

[89] M. S. Kim, W. Cane-Wissing, X. Li, J. Sampson, S. Datta, S. K. Gupta, and V. Narayanan, "Comparative Area and Parasitics Analysis in FinFET and Heterojunction Vertical TFET Standard Cells," *JETC*, May 2016. [Online]. Available: http://doi.acm.org/10.1145/2914790

[90] S. Hong and H. Kim, "An Integrated GPU Power and Performance Model," in *ISCA*, 2010. [Online]. Available: http://doi.acm.org/10.1145/1815961.1815998

[91] M. Gebhart, D. R. Johnson, D. Tarjan, S. W. Keckler, W. J. Dally, E. Lindholm, and K. Skadron, "Energy-efficient Mechanisms for Managing Thread Context in Throughput Processors," in *ISCA*, 2011. [Online]. Available: http://doi.acm.org/10.1145/2000064.2000093

[92] R. G. Dreslinski, G. K. Chen, T. Mudge, D. Blaauw, D. Sylvester, and K. Flautner, "Reconfigurable energy efficient near threshold cache architectures," in *MICRO*, Nov 2008.

[93] A. Sakanaka, S. Fujii, and T. Sato, "A Leakage-energy-reduction Technique for Highly-associative Caches in Embedded Systems," *SIGARCH Computer Architecture News*, Sep. 2003. [Online]. Available: http://doi.acm.org/10.1145/1024295.1024302

[94] J. Kin, M. Gupta, and W. H. Mangione-Smith, "The Filter Cache: An Energy Efficient Memory Structure," in *MICRO*, 1997. [Online]. Available: http://dl.acm.org/citation.cfm?id=266800.266818

[95] E. Tune, D. Liang, D. M. Tullsen, and B. Calder, "Dynamic prediction of critical path instructions," in *HPCA*, 2001.

[96] J. S. Seng, E. S. Tune, and D. M. Tullsen, "Reducing Power with Dynamic Critical Path Information," in *MICRO*, 2001. [Online]. Available: http://dl.acm.org/citation.cfm?id=563998.564013

[97] A. Baniasadi and A. Moshovos, "Asymmetric-frequency clustering: a power-aware back-end for high-performance processors," in *ISLPED*, 2002.

[98] R. Pyreddy and G. Tyson, "Evaluating Design Tradeoffs in Dual Speed Pipelines," in *In Workshop on Complexity-Effective Design in conjunction with ISCA 2001*.

[99] S. Galal and M. Horowitz, "Latency sensitive FMA design," in *ARITH*, 2011.

[100] C. Sitik and B. Taskin, "Multi-voltage domain clock mesh design," in *ICCD*, Sept 2012.

[101] V. Srinivasan, D. Brooks, M. Gschwind, P. Bose, V. Zyuban, P. N. Strenski, and P. G. Emma, "Optimizing Pipelines for Power and Performance," in *MICRO*, 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=774861.774897

[102] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli, " Multi2Sim: A Simulation Framework for CPU-GPU Computing ," in *PACT*, Sep. 2012.

[103] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. S. Kim, T. M. Aamodt, and V. J. Reddi, "GPUWattch: Enabling Energy Optimizations in GPGPUs," in *ISCA*, 2013.

[104] T. Skotnicki, C. Fenouillet-Beranger, C. Gallon, F. Boeuf, S. Monfray, F. Payet, A. Pouydebasque, M. Szczap, A. Farcy, F. Arnaud, S. Clerc, M. Sellier, A. Cathignol, J. P. Schoellkopf, E. Perea, R. Ferrant, and H. Mingam, "Innovative Materials, Devices, and CMOS Technologies for Low-Power Mobile Multimedia," *IEEE T-ED*, Jan 2008.

[105] K. Puttaswamy and G. H. Loh, "Thermal Herding: Microarchitecture Techniques for Controlling Hotspots in High-Performance 3D-Integrated Processors," in *2007 IEEE 13th International Symposium on High Performance Computer Architecture*, Feb 2007, pp. 193–204.

[106] P. Batude, T. Ernst, J. Arcamone, G. Arndt, P. Coudrain, and P. E. Gaillardon, "3-D Sequential Integration: A Key Enabling Technology for Heterogeneous Co-Integration of New Function With CMOS," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, Dec 2012.

[107] A. Agrawal, J. Torrellas, and S. Idgunji, "Xylem: Enhancing Vertical Thermal Conduction in 3D Processor-memory Stacks," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-50 '17. New York, NY, USA: ACM, 2017. [Online]. Available: http://doi.acm.org/10.1145/3123939.3124547 pp. 546–559.

[108] P. Batude, M. Vinet, A. Pouydebasque, L. Clavelier, C. LeRoyer, C. Tabone, B. Previtali, L. Sanchez, L. Baud, A. Roman, V. Carron, F. Nemouchi, S. Pocas, C. Comboroure, V. Mazzocchi, H. Grampeix, F. Aussenac, and S. Deleonibus, "Enabling 3D Monolithic Integration," *ECS Transactions*, vol. 16, no. 8, pp. 47–54, 2008. [Online]. Available: http://ecst.ecsdl.org/content/16/8/47.abstract

[109] C. Liu and S. K. Lim, "A design tradeoff study with monolithic 3D integration," in *Thirteenth International Symposium on Quality Electronic Design (ISQED)*, March 2012, pp. 529–536.

[110] DARPA, "Electronics Resurgence Initiative." [Online]. Available: https://www.darpa.mil/work-with-us/electronics-resurgence-initiative

[111] S. K. Samal, D. Nayak, M. Ichihashi, S. Banna, and S. K. Lim, "Monolithic 3D IC vs. TSV-based 3D IC in 14nm FinFET technology," in *2016 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, Oct 2016, pp. 1–2.

[112] International Roadmap for Devices and Systems (IRDS), "IRDS 2017." [Online]. Available: https://irds.ieee.org/roadmap-2017

[113] M. M. Shulaker, T. F. Wu, A. Pal, L. Zhao, Y. Nishi, K. Saraswat, H. . P. Wong, and S. Mitra, "Monolithic 3d integration of logic and memory: Carbon nanotube fets, resistive ram, and silicon fets," in *2014 IEEE International Electron Devices Meeting*, Dec 2014, pp. 27.4.1–27.4.4.

[114] S. Bobba, A. Chakraborty, O. Thomas, P. Batude, and G. d. Micheli, "Cell Transformations and Physical Design Techniques for 3D Monolithic Integrated Circuits," *J. Emerg. Technol. Comput. Syst.*, vol. 9, no. 3, Oct. 2013. [Online]. Available: http://doi.acm.org/10.1145/2491675

[115] S. Panth, K. Samadi, Y. Du, and S. K. Lim, "Design and CAD methodologies for low power gate-level monolithic 3D ICs," in *2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, Aug 2014, pp. 171–176.

[116] C. Ortolland, T. Noda, T. Chiarella, S. Kubicek, C. Kerner, W. Vandervorst, A. Opdebeeck, C. Vrancken, N. Horiguchi, M. D. Potter, M. Aoulaiche, E. Rosseel, S. B. Felch, P. Absil, R. Schreutelkamp, S. Biesemans, and T. Hoffmann, "Laser-annealed junctions with advanced CMOS gate stacks for 32nm Node: Perspectives on device performance and manufacturability," in *2008 Symposium on VLSI Technology*, June 2008, pp. 186–187.

[117] W. Pease, R. Fabian, G. S. Tompa, R. L. D. Leon, N. S. Chokshi, D. J. Witte, R. S. Shenoy, B. Rajendran, W. Pease, R. Fabian, G. S. Tompa, R. L. D. Leon, N. S. Chokshi, D. J. Witte, R. S. Shenoy, and B. Rajendran, "Low Thermal Budget Processing for Sequential 3-D IC Fabrication," *IEEE Transactions on Electron Devices*, vol. 54, no. 4, pp. 707–714, April 2007.

[118] C. H. Jan, F. Al-amoody, H. Y. Chang, T. Chang, Y. W. Chen, N. Dias, W. Hafez, D. Ingerly, M. Jang, E. Karl, S. K. Y. Shi, K. Komeyli, H. Kilambi, A. Kumar, K. Byon, C. G. Lee, J. Lee, T. Leo, P. C. Liu, N. Nidhi, R. Olac-vaw, C. Petersburg, K. Phoa, C. Prasad, C. Quincy, R. Ramaswamy, T. Rana, L. Rockford, A. Subramaniam, C. Tsai, P. Vandervoorn, L. Yang, A. Zainuddin, and P. Bai, "A 14 nm SoC platform technology featuring 2nd generation Tri-Gate transistors, 70 nm gate pitch, 52 nm metal pitch, and 0.0499 um2 SRAM cells, optimized for low power, high performance and high density SoC products," in *2015 Symposium on VLSI Circuits (VLSI Circuits)*, June 2015, pp. T12–T13.

[119] S. Srinivasa, X. Li, M. Chang, J. Sampson, S. K. Gupta, and V. Narayanan, "Compact 3-D-SRAM Memory With Concurrent Row and Column Data Access Capability Using Sequential Monolithic 3-D Integration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 4, pp. 671–683, April 2018.

[120] F. Andrieu, P. Batude, L. Brunet, C. Fenouillet-Beranger, D. Lattard, S. Thuries, O. Billoint, R. Fournel, and M. Vinet, "A review on opportunities brought by 3d-monolithic integration for cmos device and digital circuit," in *2018 International Conference on IC Design Technology (ICICDT)*, June 2018, pp. 141–144.

[121] Y. J. Lee, D. Limbrick, and S. K. Lim, "Power benefit study for ultra-high density transistor-level monolithic 3D ICs," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, May 2013, pp. 1–10.

[122] B. W. Ku, T. Song, A. Nieuwoudt, and S. K. Lim, "Transistor-level monolithic 3D standard cell layout optimization for full-chip static power integrity," in *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, July 2017, pp. 1–6.

[123] S. Panth, K. Samadi, Y. Du, and S. K. Lim, "Power-performance study of block-level monolithic 3D-ICs considering inter-tier performance variations," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2014, pp. 1–6.

[124] K. Puttaswamy and G. H. Loh, "3D-Integrated SRAM Components for High-Performance Microprocessors," *IEEE Transactions on Computers*, vol. 58, no. 10, pp. 1369–1381, Oct 2009.

[125] D. H. Kim, K. Athikulwongse, and S. K. Lim, "A study of Through-Silicon-Via impact on the 3D stacked IC layout," in *2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, Nov 2009, pp. 674–680.

[126] S. Srinivasa, A. K. Ramanathan, X. Li, W.-H. Chen, F.-K. Hsueh, C.-C. Yang, C.-H. Shen, J.-M. Shieh, S. Gupta, M.-F. M. Chang, S. Ghosh, J. Sampson, and V. Narayanan, "A Monolithic-3D SRAM Design with Enhanced Robustness and In-Memory Computation Support," in *Proceedings of the International Symposium on Low Power Electronics and Design*, ser. ISLPED '18. New York, NY, USA: ACM, 2018. [Online]. Available: http://doi.acm.org/10.1145/3218603.3218645 pp. 34:1–34:6.

[127] C. Liu and S. K. Lim, "Ultra-high density 3D SRAM cell designs for monolithic 3D integration," in *2012 IEEE International Interconnect Technology Conference*, June 2012, pp. 1–3.

[128] M. Brocard, R. Boumchedda, J. P. Noel, K. C. Akyel, B. Giraud, E. Beigne, D. Turgis, S. Thuries, G. Berhault, and O. Billoint, "High density sram bitcell architecture in 3d sequential coolcube 14nm technology," in *2016 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, Oct 2016, pp. 1–3.

[129] J. Kong, Y. Gong, and S. W. Chung, "Architecting large-scale SRAM arrays with monolithic 3D integration," in *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, July 2017, pp. 1–6.

[130] S. Panth, K. Samadi, Y. Du, and S. K. Lim, "High-density integration of functional modules using monolithic 3D-IC technology," in *2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2013, pp. 681–686.

[131] O. Billoint, H. Sarhan, I. Rayane, M. Vinet, P. Batude, C. Fenouillet-Beranger, O. Rozeau, G. Cibrario, F. Deprat, A. Fustier, J. E. Michallet, O. Faynot, O. Turkyilmaz, J. F. Christmann, S. Thuries, and F. Clermidy, "A comprehensive study of Monolithic 3D cell on cell design using commercial 2D tool," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2015, pp. 1192–1196.

[132] S. Palacharla, N. P. Jouppi, and J. E. Smith, "Quantifying the Complexity of Superscalar Processors," 1996. [Online]. Available: ftp://ftp.cs.wisc.edu/sohi/trs/complexity.1328.pdf

[133] L. Baugh and C. Zilles, "Decomposing the load-store queue by function for power reduction and scalability," *IBM Journal of Research and Development*, vol. 50, no. 2.3, pp. 287–297, March 2006.

[134] A. Tiwari, S. R. Sarangi, and J. Torrellas, "ReCycle:: Pipeline Adaptation to Tolerate Process Variation," in *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ser. ISCA '07. New York, NY, USA: ACM, 2007. [Online]. Available: http://doi.acm.org/10.1145/1250662.1250703 pp. 323–334.

[135] X. Liang and D. Brooks, "Mitigating the Impact of Process Variations on Processor Register Files and Execution Units," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 39. Washington, DC, USA: IEEE Computer Society, 2006. [Online]. Available: https://doi.org/10.1109/MICRO.2006.37 pp. 504–514.

[136] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: a low-power pipeline based on circuit-level timing speculation," in *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, Dec 2003, pp. 7–18.

[137] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: a compact thermal modeling methodology for early-stage VLSI design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 5, pp. 501–513, May 2006.

[138] J. Meng, K. Kawakami, and A. K. Coskun, "Optimizing energy efficiency of 3-D multicore systems with stacked DRAM under power and thermal constraints," in *DAC Design Automation Conference 2012*, June 2012, pp. 648–655.

[139] X. Liang, G.-Y. Wei, and D. Brooks, "ReVIVaL: A Variation-Tolerant Architecture Using Voltage Interpolation and Variable Latency," in *ISCA*, 2008.

[140] G. Semeraro, G. Magklis, R. Balasubramonian, D. Albonesi, S. Dwarkadas, and M. Scott, "Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling," in *HPCA*, Feb 2002.

[141] Q. Wu, P. Juang, M. Martonosi, and D. W. Clark, "Formal Online Methods for Voltage/Frequency Control in Multiple Clock Domain Microprocessors," in *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems*.

[142] D. H. Albonesi, R. Balasubramonian, S. G. Dropsho, S. Dwarkadas, E. G. Friedman, M. C. Huang, V. Kursun, G. Magklis, M. L. Scott, G. Semeraro, P. Bose, A. Buyuktosunoglu, P. W. Cook, and S. E. Schuster, "Dynamically Tuning Processor Resources with Adaptive Processing," *Computer*, Dec. 2003.

[143] Y. Kora, K. Yamaguchi, and H. Ando, "MLP-aware Dynamic Instruction Window Resizing for Adaptively Exploiting Both ILP and MLP," in *MICRO*.

[144] P. Petrica, A. M. Izraelevitz, D. H. Albonesi, and C. A. Shoemaker, "Flicker: A Dynamically Adaptive Architecture for Power Limited Multicore Systems," in *ISCA*.

[145] T. Miller, X. Pan, R. Thomas, N. Sedaghati, and R. Teodorescu, "Booster: Reactive Core Acceleration for Mitigating the Effects of Process Variation and Application Imbalance in Low-Voltage Chips," in *HPCA*, February 2012.

[146] T. Miller, R. Thomas, and R. Teodorescu, "Mitigating the Effects of Process Variation in Ultra-low Voltage Chip Multiprocessors using Dual Supply Voltages and Half-Speed Units," in *CAL*, 2012.

[147] M. Abdel-Majeed, A. Shafaei, H. Jeon, M. Pedram, and M. Annavaram, "Pilot Register File: Energy Efficient Partitioned Register File for GPUs," in *HPCA*, Feb 2017.