HetCore: TFET-CMOS Hetero-Device Architecture for CPUs and GPUs

Bhargava Gopireddy, Dimitrios Skarlatos, Wenjuan Zhu, and Josep Torrellas
University of Illinois at Urbana-Champaign
http://iacoma.cs.uiuc.edu

***Abstract*—**
Tunneling Field-Effect Transistors (TFETs) attain much higher energy efficiency than CMOS at low voltages. However, their performance saturates at high voltages and, therefore, cannot replace CMOS 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.

To approach this goal, this paper judiciously integrates both TFET units and CMOS units in a single core, effectively creating a *hetero-device* core. We call it *HetCore*, and present CPU and GPU versions. In HetCore, TFETs are used in units that consume high power under CMOS, are amenable to pipelining or are not very latency sensitive, and use a sizable area. HetCore powers CMOS and TFET units at different voltage levels, so they operate optimally. However, all units are clocked at the same frequency. Our results based on simulations running standard applications show the potential of this approach, even with conservative assumptions. A HetCore CPU consumes on average 39% less energy than a CMOS CPU, while delivering an average performance that is within 10% of the CMOS CPU. In addition, under a fixed power budget, a multicore with HetCore CPUs can employ twice as many cores as a multicore with CMOS CPUs, resulting in average performance gains of 32% while, at the same time, improving the energy efficiency (ED^2) by an average of 68%. Similar results are obtained with HetCore GPUs.

***Keywords*-TFET; Hybrid TFET-CMOS architecture; Core architecture; CPU; GPU.**

I. Introduction

In pursuit of higher energy efficiency, researchers try to lower the operating voltage of CMOS transistors. Unfortunately, CMOS is, intrinsically, a poor switch [1]. If one reduces the threshold voltage as the supply voltage goes down, leakage power soars, negating the energy savings.

Steep slope (SS) devices are a class of devices that are much better switches [1]. 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)* [2] are one of the most promising [3], 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 [4], [5], [6], [7]. Circuits with a combination of CMOS and TFET transistors have been used to build SRAM cells [8], [9], voltage reference circuits [10], level converters [11], multiplexers [12], 32-bit adders [12], power management circuits [13], analog circuits [14], and benchmark circuits [15].

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 [16], [17], [18]. 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 paper, 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 paper, 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.

Our results based on simulations running standard applications show the potential of this approach, even with conservative assumptions. An AdvHet CPU consumes on average 39% less energy than a CMOS CPU, while delivering a performance that is on average within 10% of the CMOS CPU. Further, under a fixed power budget, a multicore with AdvHet CPUs can employ twice as many cores as a multicore with CMOS CPUs, resulting in average performance gains of

32% while, at the same time, improving the energy efficiency (ED^2) by an average of 68%. Similarly, an AdvHet GPU consumes on average 40% less energy and performs on average 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 reducing ED^2 by an average of 60%.

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 paper 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.

II. Background

A. 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 sub-threshold Slope (SS) devices. Among the various SS devices being explored, Tunneling Field-Effect Transistors (TFETs) are one of the most promising [1], [2], [3], [19]. 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 [1]. 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 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.

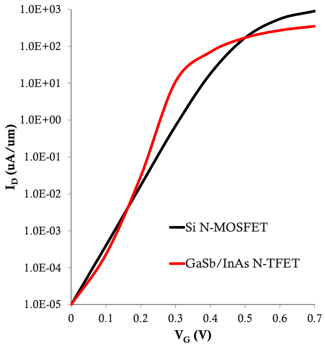

Figure 1: I_D-V_G characteristics of N-HetJTFET and N-MOSFET based on data from Intel [2].

B. 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. [20] 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 [6], [7]. The compatibility of CMOS and TFET process flows has been shown by a number of groups, both through simulation and through fabrication [4], [5], [10], [21]. Recently, mixed MOSFET-HetJTFET SRAM cells and corresponding design layout rules to integrate them at device level have been proposed [8], [9]. Moreover, circuits with a combination of CMOS and HetJTFET transistors have been used to build level converters [11], multiplexers [12], 32-bit adders [12], power management circuits [13], and analog circuits [14]. There is also substantial ongoing research on improving HetJTFET performance and building complementary devices [22], [23], [24], [25].

C. System Architectures with CMOS and TFET

Integration at such fine granularity provides an opportunity for system designers to explore system architectures with

Table I: Characteristics of CMOS and TFET technologies at 15nm, using data from [3], [19].

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

CMOS and TFET. Past work has proposed a heterogeneous multicore with some CMOS cores and some TFET cores [16], [17], [18]. 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 paper, we go beyond in that we judiciously integrate both TFET units and CMOS units in the same core, effectively creating *hetero-device* CPUs and GPUs.

III. Architecture Implications

CMOS remains the choice for high-performance systems, while operating at high $\mathrm{V_{dd}}$. However, at low $\mathrm{V_{dd}}$, the performance and energy efficiency of TFET far exceed those of CMOS. To aid in the analysis, Table I 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 $\mathrm{V_{dd}}$. 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 $\mathrm{V_{dd}}$: 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 [3], [19]. Similar numbers have been reported elsewhere [16], [26].

A. Performance

Row 2 of Table I 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 $\mathrm{V_{dd}}$, we need level converters when we go from a HetJTFET to a Si-CMOS unit. These level converters can be integrated with pipeline latches [27].

B. Energy and Power

Rows 5 and 6 of Table I 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 V-B) shows that HetJTFET still consumes 6.1x lower power than Si-CMOS. However, in this paper, 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 [28] and prior designs [29] 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 I. 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 [29], [30]. Using these numbers, the leakage power of a typical Si-CMOS unit is only about 42% of the value in Table I. This agrees with dual-V_t designs of both logic and SRAM cells in the literature [31], [32], [33], [34]. 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 [35], [36], [37]. They show that the leakage power of these cells is several hundred times lower than a competitive Si-CMOS SRAM cell [35].

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.

C. 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 2 which 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.

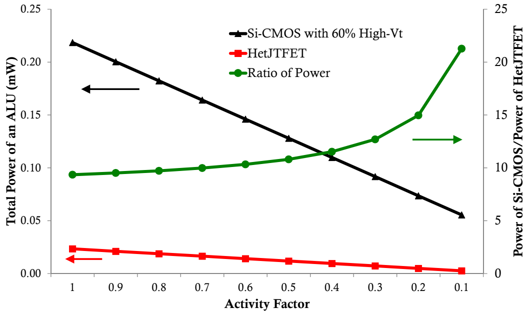

Figure 2: Total power consumption of a Si-CMOS ALU and a HetJTFET ALU with varying activity factors.

D. 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. We generated the Si-CMOS curve from [38], and the HetJTFET curve from [2]. In the curves, we show V^0_{CMOS}=0.73V, V^0_{TFET}=0.40, and f_0=2GHz.

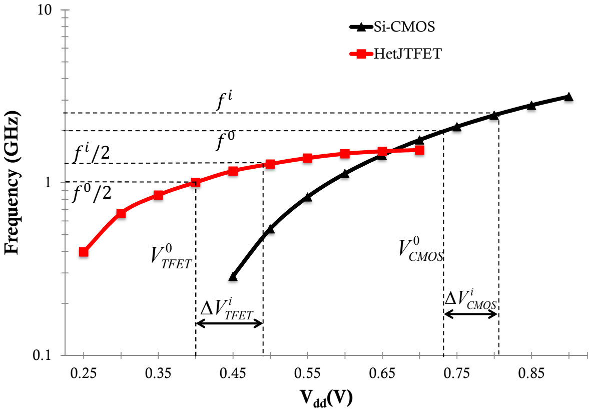

Figure 3: V_{dd}-freq. curves for Si-CMOS and HetJTFET.

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.

E. Process Variation

The main source of variation in both TFET and MOSFETs is the work function [39]. The extent of work function variation in TFETs and MOSFETs is similar, both in logic and SRAM [36], [39]. 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 1) close to the OFF state in TFETs, and in the ON state in CMOS.

As indicated by Avci et al. [39], the performance of the transistors lost to variation can be reclaimed by increasing the V$_{\text{dd}}$ of both Si-CMOS and HetJTFET. We show in Section VII that the result is that HetJTFET loses a small fraction of its energy savings relative to Si-CMOS.

F. 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 [40]. 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 [40] compares the area of standard library cells of vertical HetJTFETs to FinFETs and finds that, for the technology node of 15nm considered in this paper, 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.

IV. 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.

A. 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$_{\text{dd}}$ (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 two effects. First, the slowdown caused by TFET structures presents new opportunities for micro-architectural optimization. Second, TFET structures present different power-performance trade-offs than CMOS ones and, hence, require re-evaluation of certain design decisions. We call this final design *AdvHet*.

B. 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 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 in a CPU and GPU. They are shown in Figure 4.

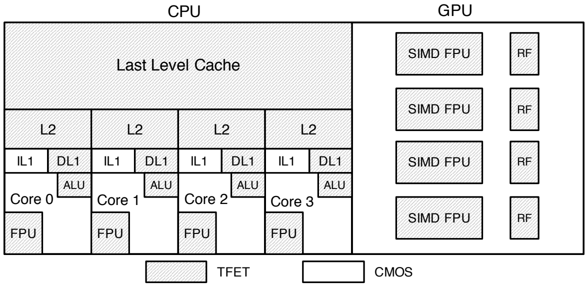

Figure 4: TFET-based units selected for the BaseHet design.

1) 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.

2) 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.

3) 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.

4) 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 [41]). RF access can also be pipelined by partitioning it into multiple stages, such as data array access and source drive [42]. 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.

C. 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 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.

1) 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 5) to alleviate some of the latency penalty introduced by TFETs.

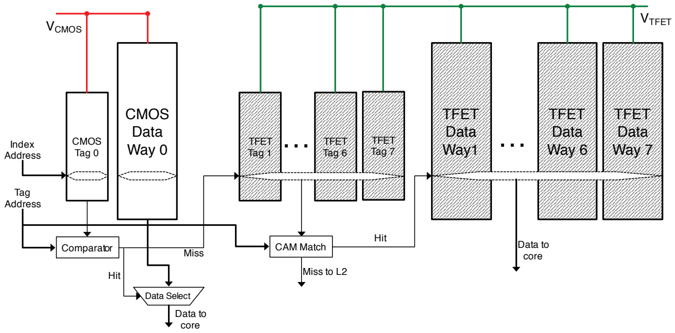

Figure 5: Schematic design of an *Asymmetric Cache*.

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 [43] 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 [44], [45], [46]. Overall, compared to a whole TFET DL1, the asymmetric cache improves performance and reduces energy consumption over the whole program execution.

2) 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 [47]. A simple algorithm suffices for us.

Dual-speed clusters have been studied previously as a mechanism to reduce power consumption [48], [49], [50]. In our design, we employ a simplified version of the Generation Time Gap metric [49] 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.

3) 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 [42]. 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 [42]. 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.

4) 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 [51]. 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.

D. Summary of the Designs

Table II 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 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.

Table II: 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

V. Implementation Considerations

A. 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 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.

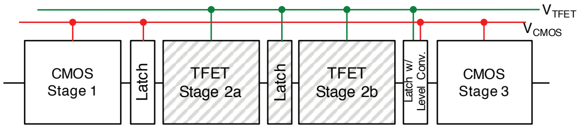

Figure 6: HetCore dual voltage rail design.

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 6, they are augmented with a level converter and take both V_{dd} levels [11], [27].

HetCore employs a level converter design based on Ishihara et al. [27], 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. [11].

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 cyle) [52].

B. 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 [53] 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 [27] work, we estimate a delay impact of ≈5%. The additional area and power of the level converters is negligible [27].

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 [54], 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 [54].

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 I, 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 paper, we set the overall reduction in dynamic power when moving from CMOS to TFET to be only 4x.

VI. Evaluation Setup

We evaluate HetCore using the Multi2Sim [55] 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 III shows the detailed parameters of the modeled CPU and GPU. We obtain the power numbers by using the HP-CMOS process of McPAT [56] and GPUWattch [57] 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 III: Parameters of the simulated architecture.

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 cyles; 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 [16] shows that the insights from Table I hold true at 10nm. CMOS is likely to maintain a performance edge over TFET and, as a result, the HetCore tradeoffs will remain similar.

A. Configurations

Table IV 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 IV: 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 [58]. The latencies of the FPUs and ALUs are shown in Table IV. 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.

B. Applications & Metrics

We use the SPLASH-2 and PARSEC applications to evaluate the CPU designs. 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 [55]. Our metrics of comparison are execution time, energy consumption, energy-delay product (ED), and energy-delay-squared (ED^2). Due to space restrictions, we do not show the ED results.

VII. Evaluation

A. HetCore CPU Evaluation

Figure 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 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 8 shows the energy consumption of the same configurations as Figure 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.

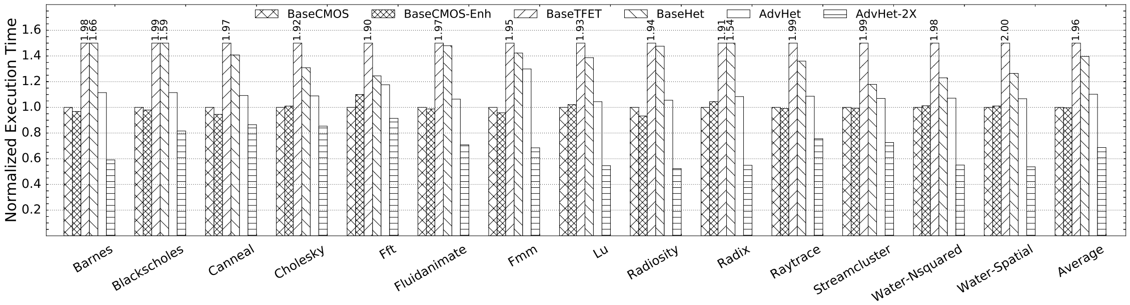

Figure 7: Execution time of different CPU designs, normalized to BaseCMOS.

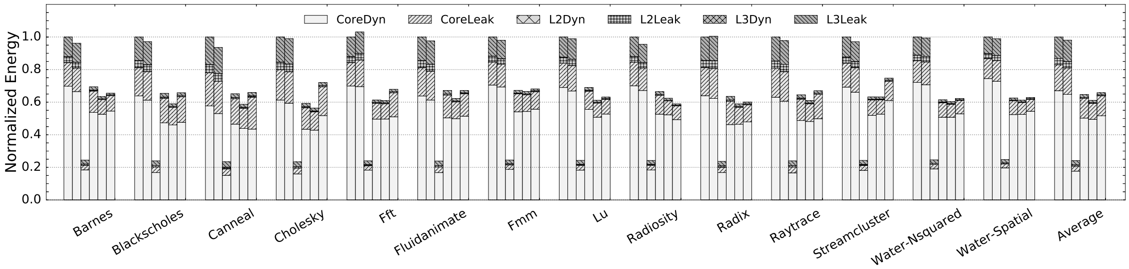

Figure 8: Energy consumption of different CPU designs, normalized to BaseCMOS.

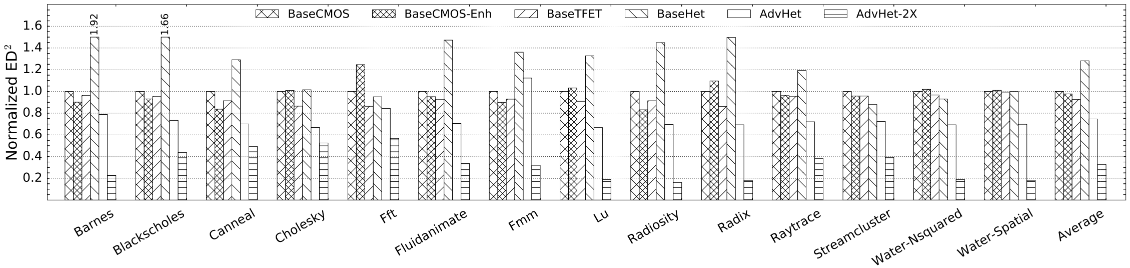

Figure 9: ED^2 of different CPU designs, normalized to BaseCMOS.

BaseCMOS-Enh has similar results as BaseCMOS.

Finally, Figure 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.

1) Comparison Under a Constant Power Budget: AdvHet is especially appealing when comparing chips at a constant power budget. From Figures 8 and 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 7, 8 and 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. 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.

B. HetCore GPU Evaluation

For the GPU architecture, Figure 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 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,

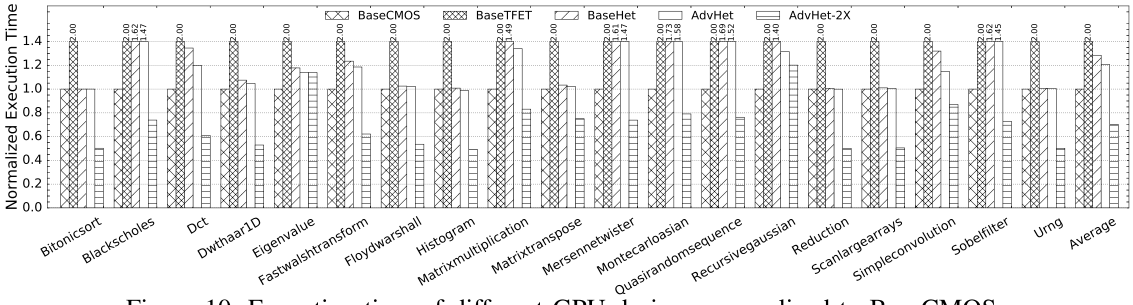

Figure 10: Execution time of different GPU designs, normalized to BaseCMOS.

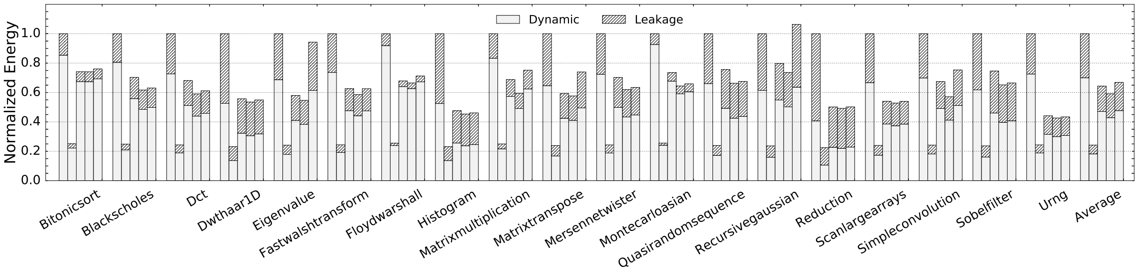

Figure 11: Energy consumption of different GPU designs, normalized to BaseCMOS.

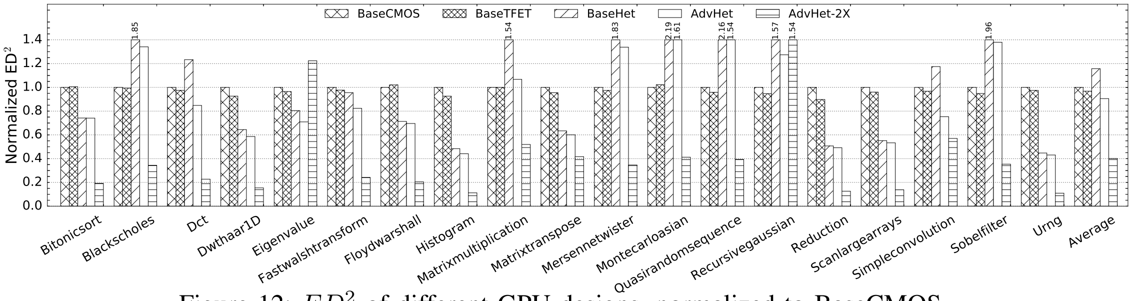

Figure 12: ED^2 of different GPU designs, normalized to BaseCMOS.

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 VII-B1.

Figure 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 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.

1) Comparison Under a Constant Power Budget: The interesting scenario for AdvHet GPUs is comparing against BaseCMOS GPUs under a constant power budget. From Figures 11 and 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 10, 11 and 12.

We see that, under AdvHet-2X, applications take on average 30% less time to execute (Figure 10) and consume on average 34% less energy (Figure 11) than under BaseCMOS. Moreover, Figure 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.

C. Comparison to Alternative CPU Designs

Figure 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.

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

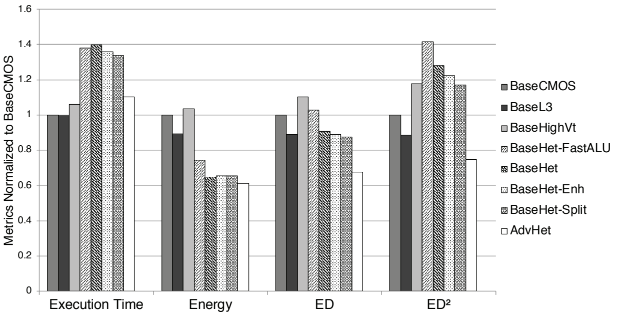

Figure 13: Sensitivity analysis of HetCore CPU designs.

ALUs have slightly higher latencies (Table IV), 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 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.

D. Impact of DVFS and Process Variation

As discussed in Section III-D, the V$_{\text{dd}}$-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 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. 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 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.

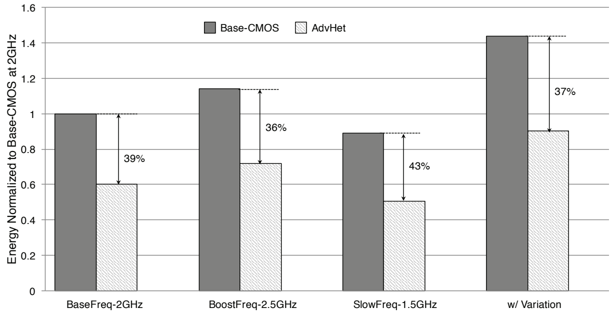

Figure 14: Impact of DVFS and process variation on the energy consumed by BaseCMOS and AdvHet.

We also study the impact of process variation on the energy consumption. According to Avci et al. [39], to protect against all the potential sources of process variation at 15nm, we need to add V$_{\text{dd}}$ guardbands equal to ΔV_{CMOS}=120mV and ΔV_{TFET}=70mV, for the respective operating voltages. These are large guardbands. With these guardbands, the energy consumed by BaseCMOS and AdvHet is shown in the rightmost bars of Figure 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.

VIII. Related Work

Prior work has studied the integration of some TFET cores and some CMOS cores in the same chip [17], as well as in a 3D-stack [16]. HetCore pushes device heterogeneity inside the core for a more energy-efficient design. In [18], 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 [18], 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., [9], [11], [35], [36], [37]) 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 II and III.

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 [44], [45], [46], dual-speed clusters as a mechanism to reduce power consumption [48], [49], [50], and register file caches for GPUs [42]. In this paper, we adapt them to the context of heterogenous-device cores.

As an alternative to using a register file cache, a partitioned register file for GPUs is proposed in [59]. 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.

IX. Conclusion

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, this paper has 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 as 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%.

Acknowledgments

This work was supported in part by NSF under grants CCF-1536795 and CCF-1649432.

References

[1] A. M. Ionescu and H. Riel, “Tunnel field-effect transistors as energy-efficient electronic switches,” *Nature*, Nov 2011.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] M. Fulde *et al.*, “Fabrication, optimization and application of complementary Multiple-Gate Tunneling FETs,” in *International Nanoelectronics Conference*, March 2008.

[11] 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.

[12] 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.

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

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

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

[20] Q. Huang *et al.*, “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.

[21] N. Mukherjee *et al.*, “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.

[22] R. Rooyackers *et al.*, “A new complementary hetero-junction vertical Tunnel-FET integration scheme,” in *IEDM*, Dec 2013.

[23] G. Dewey *et al.*, “Fabrication, characterization, and physics of III–V heterojunction tunneling field effect transistors (H-TFET) for steep sub-threshold swing,” in *IEDM*, 2011.

[24] G. Zhou *et al.*, “Novel gate-recessed vertical InAs/GaSb TFETs with record high ION of 180 uA/um at VDS=0.5V,” in *IEDM*, Dec 2012.

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

[26] 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.

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

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

[29] 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.

[30] X. Chen and N. K. Jha, "Ultra-low-leakage Chip Multiprocessor Design with Hybrid FinFET Logic Styles," *JETC*, Oct. 2014.

[31] 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.

[32] 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.

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

[34] 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.

[35] 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.

[36] 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.

[37] 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.

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

[39] 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.

[40] 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.

[41] S. Hong and H. Kim, "An Integrated GPU Power and Performance Model," in *ISCA*, 2010.

[42] 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.

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

[44] 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.

[45] 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.

[46] J. Kin, M. Gupta, and W. H. Mangione-Smith, "The Filter Cache: An Energy Efficient Memory Structure," in *MICRO*, 1997.

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

[48] J. S. Seng, E. S. Tune, and D. M. Tullsen, "Reducing Power with Dynamic Critical Path Information," in *MICRO*, 2001.

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

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

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

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

[53] 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.

[54] 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.

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

[56] 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.

[57] 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.

[58] T. Skotnicki *et al.*, "Innovative Materials, Devices, and CMOS Technologies for Low-Power Mobile Multimedia," *IEEE T-ED*, Jan 2008.

[59] 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.