Value Locality Based Approximation With ODIN

Rahul Singh, Gokul Subramanian Ravi, Mikko Lipasti, and Joshua San Miguel

Abstract—Applications suited to approximation often exhibit significant value locality, both in terms of inputs as well as outcomes. In this early stage proposal - the *ODIN: Outcome Driven Input Navigated* approach to value locality based approximation, we hypothesize that value locality based optimizations for approximate applications should be driven by outcomes i.e., the result of the computation, but navigated with the help of inputs. An outcome-driven approach can enable computation slices, whose outcomes are deemed (approximately) redundant or derivable, to be entirely eliminated resulting in large improvements to execution efficiency. While such an approach provides large potential benefits, we address its design challenges by aiding the outcome-driven approach with input-navigation - attempting to map the value locality characteristics within inputs to that of the outcomes. To enable this, we build a novel taxonomy to categorize value locality and use it to analyze benchmarks from the PERFECT suite. We show that with oracle prediction and an ideal design, more than 80 percent of computations can be eliminated at an SNR of 17.8 or a 90 percent accuracy, thus capable of tremendous performance and energy benefits. Finally, we discuss directions towards achieving optimal benefits.

Index Terms—Value locality, approximate computing, value prediction

1 ODIN: Introduction & Motivation

Approximate Computing. Workloads from several prevalent and emerging application domains, such as vision, machine learning, and data analytics, possess the ability to produce outputs of acceptable quality in the presence of inexactness or approximations [1]. Allowing computation to be approximate can lead to significant improvements in processor efficiency. In this work, we are particularly interested in the potential for hardware approximation stemming from value locality.

Value Locality. Over past decades, the program attributes of value locality have been examined and exploited through a variety of proposals. Fundamentally, value locality describes the likelihood of the influence of previously-seen program values on newly-occurring ones. In a time of steeply declining benefits from classical dataflow acceleration, value locality is still of prime importance in attempting to exceed the classical dataflow limit, which is defined as the program performance obtained when machine instructions execute as soon as their operands are available [2]. While characteristics of value locality are beneficially exploitable in multiple ways, targeting both precise and approximate computing, in this work we are focused on the latter. More specifically, we make the case for approximate computing centered around the outcome-driven exploitation of value locality instead of the conventional input-driven approach. We describe these forms of exploitation next.

Input-Driven Approximation. The conventional Von Neumann wisdom in general purpose processing is that a processor must fetch and wait for all input data elements on which it needs to compute. In most designs, these data values are brought from the main memory into the local cache (on a "cache miss"), and then read and computed upon by the core. Further, a cache is agnostic to the values of these elements before reading them from memory. Prior work from San Miguel *et al.* [3] recognized that not all "new" data values are required to be read from the memory (i.e., into the cache and then for compute). Instead values of input data can often be *approximately* estimated from a) the values of previously seen data elements and/or b) the values of adjacent data elements in memory. The effectiveness of such schemes is a function of program characteristics, domain error tolerance, redundancy in input data, as well as the hardware design (e.g., cache sizes, policies). To exploit this possibility, the authors propose methods to approximately infer these "new" data values from existing "old" values in the cache, thus alleviating the high latency and energy overheads of retrieving data from memory.

Outcome-Driven Approach. Over multiple contributions targeting precise computing, Lepak and Lipasti [2] hypothesized that value locality should be focused on outcomes. They suggested that while value locality based optimizations have generally focused on efficient and rapid processing of instructions, which is *the means* of modern computing, they should instead be driven by the result of the computation, which is *the end goal* of computing. Specifically, Lepak and Lipasti [2] focused on reducing the redundant memory traffic generated by *silent stores*, i.e., stores that write *unchanged* values to memory (and thus create no change to system state).

Comparison. Below we summarize the benefits and challenges of outcome-driven value locality compared to an input-driven approach, in the context of approximate compute.

Merits. ① An outcome driven approach has potential for higher savings by eliminating entire computation slices (i.e., sequences of operations that potentially consist of multiple inputs/loads and compute operations, but ending in a single outcome/store). If executed, these slices would end up producing only *redundant or approximately redundant* outcomes. ② An outcome driven approach is optimum for approximate computing since controlling the efficiency-error trade-off should ideally be driven by the results of computations (and not inputs to computations). ③ As shown later on, slices of computation often have a large number of inputs leading to a single outcome. Moreover, many inputs often do not directly impact the outcomes. Convolutions are classical examples wherein multiplication of two matrices (weights / activations) produces a single output. Further, the changing activations (or weights) would not impact outcomes if, say in sparse scenarios, the corresponding weights (or activations) are negligible/zero. Thus, for better control of efficiency and error, assimilating the value locality of multiple and/or non-impacting inputs is less favorable (and more challenging) compared to that of a single outcome.

Challenges. At the time of dynamic execution in hardware, decisions regarding the (locality of an) outcome of a computation slice might be made too late to provide considerable benefit. This is especially true in high ILP processors in which outcomes of proximate older computations might only become available on cycles very close to the computation of interest - thus making any outcome based decisions impractical or less beneficial.

Our Proposal: ODIN. The discussion above highlights the merits and challenges of an output-driven approach to value locality based approximation. In this work, we pursue an *outcome-driven + input-navigated* philosophy to value locality based approximation, so as to benefit from the merits of being outcome driven, while overcoming the challenges it imposes. In ODIN, computation slices' outcomes are early-identified to be *redundant or locally derivable, approximately or otherwise*. This is possible thorough input-navigation - the idea of learning some form of mapping between the value locality characteristics within inputs and that of the slice's outcome. If an outcome is deemed to be redundant or derivable, the corresponding

• *The authors are with the Department of Electrical and Computer Engineering, University of Wisconsin–Madison, Madison, WI 53706.*
E-mail: {rsingh76, gravi, jsanmiguel}@wisc.edu, mikko@engr.wisc.edu.

Manuscript received 24 Mar. 2020; revised 10 May 2020; accepted 8 June 2020. Date of publication 15 June 2020; date of current version 22 July 2020.
(Corresponding author: Rahul Singh.)
Digital Object Identifier no. 10.1109/LCA.2020.3002542

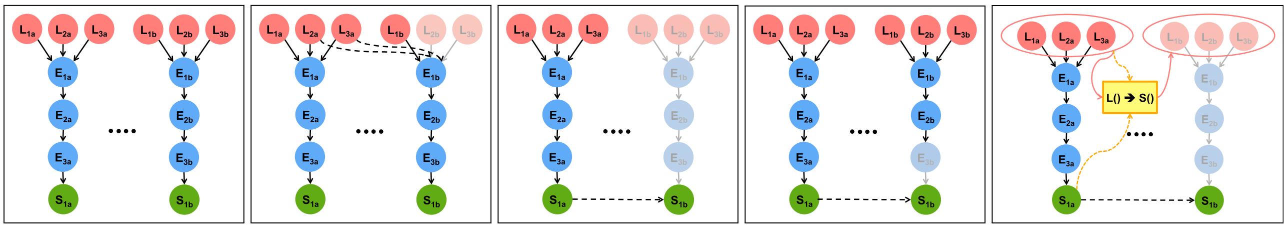
Fig. 1. Left to Right: (a) Two computation slices fully executed, (b) Input-driven approximation eliminates 2 out of 3 loads in 2nd slice, (c) Optimal outcome-driven approximation eliminates the 2nd slice entirely and derives its store from the first slice, (d) Sub-optimal outcome-driven approximation is unable to eliminate much of the 2nd slice due to late information flow from the 1st slice, (e) Optimal outcome-driven approach via intelligent learning of Load-Store value locality mappings.

computation slice can be eliminated and eliminating slices of operations (loads, compute and stores) results in significant improvements to execution efficiency, in terms of both energy reduction and performance speedup. The outcomes of these slices can instead be derived nominally (for example, in memory) with minimal overhead. Further, the locality of these outcomes and thus, the ability to eliminate slices of computation, is dependent on application characteristics and the prescribed error tolerance. The different scenarios discussed here i.e., outcome and input based approximation, as well as the ODIN approach, are illustrated in Fig. 1.

Contributions:

① We introduce a novel taxonomy for fundamentally categorizing different types of value locality, applicable to both input values and output values (Section 2).

② We characterize the value locality of two approximate applications: dwt53 and 2dconvolution based on our taxonomy, highlighting the potential for the ODIN approach (Section 3).

③ Assuming an ideal design, we present initial observations on the benefits of value locality based approximation with ODIN, for the aforementioned applications (Section 4).

④ We briefly discuss future directions to build an optimum design to achieve the goals of ODIN. (Section 5).

Fig. 2 clearly highlights the potential benefits of our proposal, at an outcome-driven approximation of 10 percent. Here, more than 80 percent of the total program operations are eliminated, resulting in proportional performance and energy efficiency gains.

2 Characterizing Value Locality

Without context, the meaning and benefit of value locality is often lost. In this section, we put forth four properties that classify value locality, which have wide spread potential benefits. These properties are further illustrated via Fig. 3.

A. Dimension. Along what locality dimension are the values correlated?

① *Spatial:* Meaningful correlation across (potentially sparse) data elements in memory. Examples include multi-dimensional patterns in data, specifically for image and video processing.

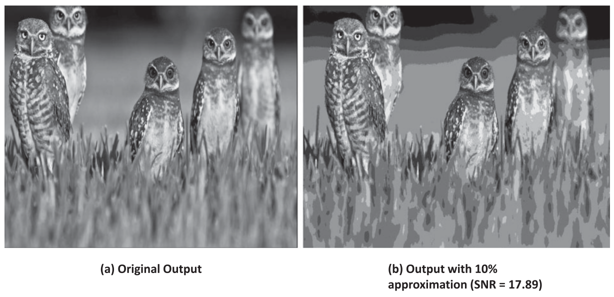

Fig. 2. DWT53: Precise versus Approximation w/ ODIN.

② *Temporal:* Meaningful correlation between values loaded or stored over time by the same instruction. For example, iterative applications such as kmeans perform repeated nearly-redundant computations once optimal solutions are reached.

B. Distribution. Over what granularity or periodicity over space/time does the correlation exist/repeat?

The granularity can be adjacent values, alternate values or any period N. For example, prior work [4] discussed that applications like 2dconvolution show two periodic patterns, one with granularity of ’1’ and the other with granularity of the row-size of the processed image, when the image is stored in row-major order (i.e., x-dimension is contiguous while y-dimension is spread across rows).

C. Derivability. In what functional form are data elements correlated or how can a specific element be derived from others?

① *Zero-order (same):* The trivial scenario of the values being constant - for example, always zero or one.

② *First order (Value is correlated):* The values are correlated by a simple function which makes next values predictable. Examples are values repeating at certain stride.

③ *Second order (Change is correlated):* The values themselves are not correlated but the rate of change is correlated through simple functions (like constant slope).

D. Degree. At what amount/degree of approximation do the particular value locality characteristics exist?

For instance, adjacent pixels in an image might exhibit spatial value locality only under some approximation. Consider that these pixels are close color shades of blue (sky). The value locality might not exist at full accuracy since the pixels may not have the exact same RGB value, but might exist under some approximation. In our work we define approximation via bit quantization, i.e., dropping lower significant bits.

3 Analyzing Application Value Locality

We analyze 2 approximate benchmarks from the PERFECT suite [5], *DWT53* and *2d-convolution*, to demonstrate locality characterization as well as benefits from ODIN.

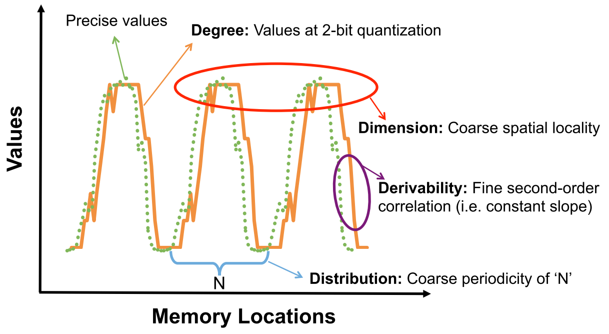

Fig. 3. Illustrating the classes of value locality.

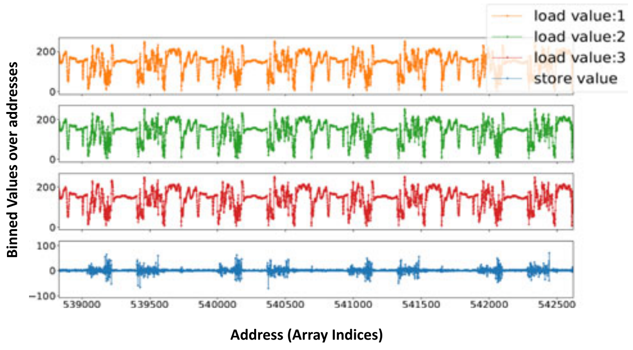

Fig. 4. Load-Store Correlation of dwt53 at *1 percent approximation*.

3.1 DWT53

DWT or Discreet Wavelet Transform computes a combination of 1-D transforms on image pixels, first on rows and then on columns. The transform is a linear combination of neighboring pixels. Each pixel is loaded along with its two neighbors, the transform is performed and stored to memory. Hence, for each pixel three loads are required.

We plot all three Loads and Stores for each pixel of the dwt53 computation slice against the address of each pixel (1080x1920) of the image. We show a zoom into a particular range of addresses in Fig. 4. These values are obtained with 1 bit quantization (discussed in Section 2). We analyze the characteristics of the computation slice of interest in terms of the value locality parameters defined previously:

① *Dimension.* We observe strong spatial locality as both loads and stores show correlation across data elements in memory - periodic repetitions of values at coarse granularity and constant (equalling zero) values at fine granularities (for the store). Notice that there is far less 'noise' in the store locality, a common characteristic of many approximate applications wherein a lot incoming signal noise is filtered out. Thus the locality of outcomes i.e., stores is far more pronounced than that of loads i.e., inputs. The input noise as well as the sheer number of inputs are tougher to manage in terms of locality as these computation slices become complex with more inputs.

② *Distribution.* We notice that the coarse grained repetitive aspect of spatial locality is periodic in nature and in this particular case the period is the row size of image, i.e., the similar values repeat themselves in each row. Note that the periodicity of locality can be different for different inputs for more complex applications (as we will see with 2D-Convolution), and tracking or learning the potentially different periodicity of multiple inputs is more challenging compared to that of a single output.

③ *Derivability.*Within the shown address range, we observe the coarse grained spatial locality to be of the first order i.e., the same memory values repeat at the periodicity rate. The fine-grained spatial locality over some regular portions of the address range are of the zeroth order (constant, equalling zero). These are artifacts of repeating pixel values across rows and columns over a significant region of a image which become a constant single value under some level of quantization (1 percent shown here).

*Takeaways.*First, while locality exists across both inputs and outcomes, the locality is more pronounced in the outcomes, as irregularities in inputs often get filtered out. Second, streams with different locality characteristics are often intermingled with each other and should be tracked/learned independently. Third, locality can become more prominent under higher degrees of approximation by smoothing out system noise.

3.2 2d-Convolution

Next, we look at 2d-convolution which involves a convolution of two matrices. In our case it is the input image with a 9x9 Gaussian filter. It requires to load both filter values as well the input image pixels, so as to perform the convolutional computation slice. The benchmark uses 9x9 filter matrix, resulting in a total of 81*2=162 loads (both filter and pixels), per computation slice, and of course only one outcome (i.e., store). Note that the characteristics of these loads can be very different - while the filter is fixed throughout, the image pixels can change from one frame to the next - presenting more challenging input locality.

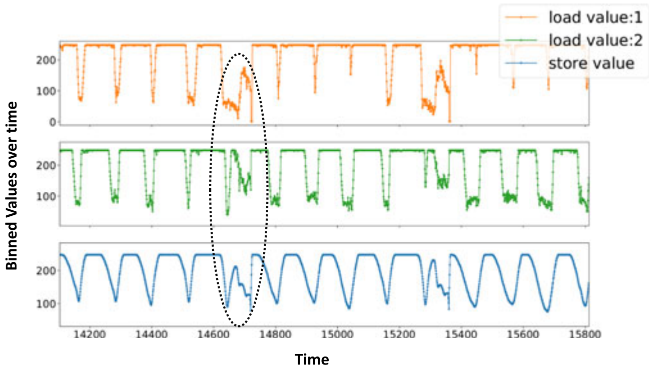

Fig. 5. Load-Store Correlation of 2d-conv at *1 percent approximation*. Dotted circle highlights the closer correlation of store to 2nd load.

In Fig. 5, we show two loads and the store from the slice and plot over time. Their characteristics are discussed below:

① *Dimension.* There is a strong temporal locality across both the loads and the store (i.e., values are repeated over time) but note that store (i.e., outcome) locality is more significantly correlated to the 2nd load than to the 1st load (dotted circle in Fig. 5). Moreover, as mentioned earlier there are many filter-value loads which can be of constant value, and hence do not exhibit similar locality characteristics as the outcomes of the computation. This is in contrast to DWT53 where all loads had a strong correlation with stores.

② *Distribution.* We again notice a coarse-grained periodicity with a period equivalent to the size of rows.

③ *Derivability.* Above, we noticed a coarse grained periodicity (at the granularity of row size). In order to predict values at sub-row granularity, we look to Fig. 6 which depicts the second order of value locality i.e., slope (Rate of change) of loads and stores wrt time. It is clear that slopes are constant over much of the fine grained periods (i.e., at sub-row granularity), and these values can be predicted based on their fixed rate of change.

Takeaways. First, in the presence of many inputs to a slice, it is likely that some inputs are highly correlated to outcomes, while some are not. This can also be dependent on the input values. Second, in applications with simple computations such as accumulation, it is possible to identify higher degrees of derivability in the value locality such as constant rate of change.

4 ODIN: Initial Observations

Based on the above characterization of outcome value locality, we can potentially *derive* the outcome of the slices without actually executing the slice, thus gaining in terms of both performance and execution efficiency. In this initial evaluation, we analyze the *upper bound of the fraction of computation slices that can be saved/eliminated* assuming a maximally ideal design. The ideal design is assumed to be capable of a) identifying the derivability or redundancy of the outcome of the slice prior to any operation within the slice (i.e., prior to the earliest load in the slice) and b) updating the outcome of the slice to the appropriate value with no computational effort.

We study two scenarios. The first scenario assumes an *infinitely deep history* i.e., any slice outcome can be derived from any prior instance of the same outcome (if it exists), through an unlimited history buffer. The second assumes a *4-deep history* i.e., only the most recent unique 4 outcomes are held in a 4 entry buffer. Thus, if

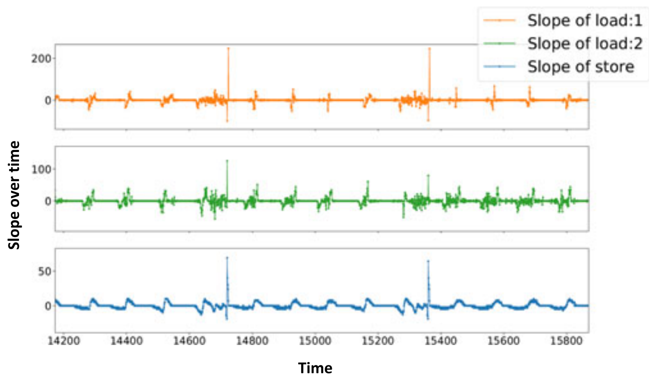

Fig. 6. Rate of change of Loads/Store of 2d-conv at *1 percent approx.*

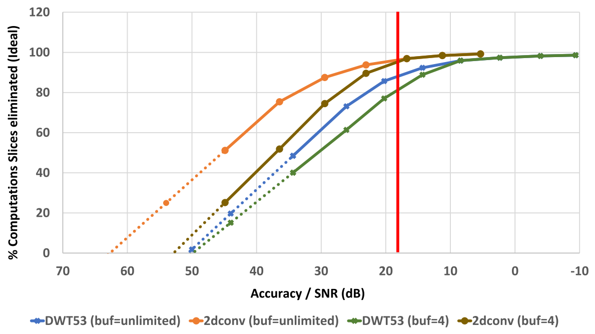

Fig. 7. Computational slices eliminated versus application error. Red line corresponds to the SNR shown in Fig. 2b.

any new slice's outcome is expected to (approximately) match any entry in the history buffer, that computation slice is eliminated. Further, we sweep the degree of approximation, achieved via bit quantization here, from 0 bit (precise) to 8 bits (complete approximation). The number of possible outcomes is influenced by the degree of approximation - thus higher the degree, the greater is the overall application error, but higher is the number of computational slices that can be ideally eliminated.

The analysis is presented in Fig. 7 which shows the computation slices that can be eliminated/saved versus the accuracy of the application shown in SNR. We show results DWT53 and 2d-convolution. It is evident that roughly *more than 50 percent of the application slices can be eliminated, even for a limited buffer size, at reasonable accuracy* (e.g., at SNR = 30). At lower accuracy requirements, these numbers grow higher. As noted at the start of this section these results are still optimistic, but the opportunity for outcome-based approximation is clearly evident.

5 Discussion and Future Work

Target Applications and Domains. While we have performed a deep-dive into two specific approximate applications, we observe the applicability of ODIN to a variety of applications and domains. In terms of the *dimension* characteristic, generally all error-tolerant applications processing images have abundant spatial locality while video processing applications show both spatial and temporal locality. Coming to the *distribution* characteristic, at the minimum, these applications have finely periodic distributions of locality i.e., similar adjacent pixels of images and correlated adjacent pixels/frames of videos. The *derivability* is usually very application specific - ML applications which cause input features to vanish or saturate might show zero-order derivability, streaming applications performing independent operations over inputs might show first-order correlation, while applications with accumulation operations might show second-order derivability. Finally, the *degree* is dependent on the requirements of the domain, the characteristics of the inputs and the program structure of the applications as well. Outside of image/video processing, we also observe potential in domains such as graph processing, IOT, robotics and so on.

Challenges and Opportunity. We briefly outline challenges and potential to build novel mechanisms to achieve our goals:

Output Locality Prediction. The main challenge is that we wish to predict the locality of the outcomes of computation slices. For maximum benefit, this prediction should be performed early enough so that the largest possible fraction of the computation slice can be eliminated. There is potential to estimate the outcome locality based on locality in slice input(s), with prediction / ML-based mechanisms.

Slice Elimination. Earlier the identification and removal of the redundant slice, higher the possible benefits, but early action requires a low overhead, fast reacting mechanism. Moreover, the mechanism should be overlaid on existing execution schemes with minimal non-invasive instrumentation. There is potential to explore instruction-removal mechanisms akin to prior work on slipstream processors [6].

Output Derivation. Elimination of slices means that in scenarios when outcomes are non-redundant but derivable from value locality, some from of computation is required, potentially at some level in the memory hierarchy. Ideas from in-cache compute, memoization etc. are applicable.

6 Conclusion

In this paper, we propose an outcome driven approach to value locality based approximation, which can provide tremendous benefits to execution efficiency by eliminating entire slices of computations. At the same time there are challenges to such an approach and we propose navigation/assistance from inputs to achieve an optimal outcome-driven mechanism. We define a novel taxonomy to classify value locality and analyze two approximate applications based on this taxonomy. This taxonomy enables us to demonstrate that the locality in outcomes i.e., stores, is more pronounced and potentially easier to track in comparison to the inputs/loads. Moreover, it shows that correlation between specific inputs and outcomes of the analyzed computations slices - which supports the premise of input navigation. For our chosen applications, we quantify the benefits of our proposal using an ideal mechanism with oracle prediction. Finally, we discuss a preliminary design and future directions to realistically implement this proposal.

Acknowledgments

This work was supported by the Office of the Vice Chancellor for Research and Graduate Education at the University of Wisconsin.

References

[1] S. Venkataramani, V.K. Chippa, S.T. Chakradhar, K. Roy, A. Raghunathan "Quality programmable vector processors for approximate computing." in *Proc. 46th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2013, pp. 1–12.
[2] K. M. Lepak and M. H. Lipasti, "On the value locality of store instructions," in *Proc. 27th Int. Symp. Comput. Archit.*, 2000, pp. 182–191.
[3] J. S. Miguel, M. Badr, and N. E. Jerger, "Load value approximation," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2014, pp. 127–139.
[4] J. S. Miguel, J. Albericio, N. E. Jerger, and A. Jaleel, "The bunker cache for spatio-value approximation," in *Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2016, pp. 1–12.
[5] K. Barker *et al.*, "PERFECT (power efficiency revolution for embedded computing technologies) Benchmark Suite Manual," Pacific Northwest National Laboratory and Georgia Tech Research Institute, Dec. 2013. [Online]. Available: http://hpc.pnnl.gov/projects/PERFECT/
[6] Z. Purser, K. Sundaramoorthy, and E. Rotenberg, "A study of slipstream processors," in *Proc. 33rd Annu. IEEE/ACM Int. Symp. Microarchit.*, 2000, pp. 269–280.