Research Statement

Society has become so dependent on computing power that any inefficiencies in the way that we process information can considerably impede productivity and quality of life. Three emerging trends pose challenges to the design of efficient computer systems. First, applications—particularly sensory and neural processing—are executing on strictly energy-constrained hardware due to the rise of internet of things (IoT) and mobile computing. Traditional architectures expend a great deal of energy providing perfect precision for these inherently approximate applications. Second, data sets are growing to enormous proportions due to the rapid gathering of user information in modern devices. We can no longer rely on data being readily available in on-chip computer storage. Third, active chip area is diminishing at smaller technology nodes due to thermal and power density limitations in process technology scaling. We can no longer fully utilize all on-chip hardware resources simultaneously.

My research tackles these challenges by recognizing that they stem from fundamental gaps in the way that data is contextualized in computer hardware. The goal of a processor is to process real-world information; yet in modern computers, hardware perceives data as nothing more than bits. Can the hardware tell if an integer is purely numerical or if it will be used as a memory address later? Can it tell that two elements are adjacent in a multidimensional data structure even though they are not stored contiguously? Can it tell if contiguous data elements will be accessed at the same time or accessed independently? Traditionally, hardware cannot interpret data bits on their own; it only interprets data bits by how instructions use them. It lacks an understanding of what information is encoded in the bits (functional context), where that information is located (spatial context), and when that information is needed (temporal context). My research fills these gaps in data context to address the three challenges, recognizing that 1) functional context enables approximation for greater efficiency under tight energy constraints; 2) spatial context demystifies patterns and correlations to more efficiently and concisely process massive data sets; and 3) temporal context infers the criticality of data to allow for better utilization of precious on-chip resources.

Dissertation Work

My ongoing and past research presents new designs and insights that address the lack of data context in computer architecture—the layer that bridges software and hardware.

Functional Context of Data. Is there benefit to making hardware more aware of the information encoded in the data bits? With the growing demand for sensory and neural processing, the perfect precision and exactness enforced by traditional processors is increasingly wasteful. I propose techniques that infer functional context and exploit data that can be approximated to uncover new sources of efficiency in hardware.

Modern processors continue to suffer the high latency and energy of fetching data from off-chip memory. I introduce load value approximation [12, 13], a mechanism for estimating the values of data accesses. The insight is that processor data is merely some digital representation of real-world information. And for a wide variety of applications (e.g., multimedia, machine learning, physical simulation, computer vision), this information exhibits patterns and trends that can be approximated with reasonable accuracy (e.g., periodicity in signals, gradual changes in video frames), eliminating the need to fetch the actual data from memory. I propose hardware approximators that learn and generate estimates of values in lieu of memory accesses. This not only saves energy in off-chip accesses but also takes memory latency off of the critical path of application execution. Furthermore, I mentored a Master's student in implementing approximators using only texture caches in graphics processing units [19], reaping these benefits with commodity hardware. This work has inspired further explorations of value approximation in the research community [8, 20].

Awareness of functional context offers new opportunities for on-chip storage savings. The immense growth of data motivates more on-chip caching, yet the high leakage energy of caches makes them extremely costly in mobile devices. Thus we need new caching methods where storage size does not scale with the increasing sizes of data sets. I explore the notion of approximate value similarity [11]: in many data sets, a large fraction of data blocks contain similar information (e.g., similar pixels in an image data set, similar temperature values in a weather data set). To exploit this, I propose Doppelgänger, a cache architecture that identifies similar data blocks on-the-fly and deduplicates them into the same storage location. This provides a means of on-chip storage that scales not with address uniqueness (i.e. data set size) as in traditional architectures but rather with value uniqueness, which is often very low in real-world data (e.g., a data set of body temperatures is likely to have low variance).

My dissertation includes additional contributions that exploit functional context via approximation. Collaborating with the Sampa Group from University of Washington, I present a taxonomy of approximation techniques [7], which characterizes how approximation offers efficiency gains at all layers of the system stack. This is beneficial for chip designers, providing answers to the following: 1) how observable are the effects of an approximation technique; 2) how well can it be tested and debugged; and 3) how aggressively can it be applied for efficiency gains. Other approximation ideas that also address spatial context (cache spatio-value similarity [10]) and temporal context (anytime automaton model [15, 16]) are discussed in later sections.

Spatial Context of Data. Is there benefit to making hardware more aware of where information is located in the data set? With the immense growth of data, we need new ways to concisely represent information in hardware. I propose techniques that exploit the dimensionality and spatial characteristics of data sets to more efficiently process them on-chip.

Collaborating with Aamer Jaleel from NVIDIA Research, I explore spatio-value similarity [10], addressing the lack of both functional and spatial context in on-chip caching. The insight is that real-world information is often continuous and smooth, which implies that spatially colocated data elements are approximately similar in value. However, in hardware, multidimensional data sets (e.g., two-dimensional set of pixels in an image, three-dimensional set of bodies in space) must be flattened in one-dimensional memory, resulting in similar elements being spread apart. I propose the Bunker Cache, a new design that effectively unflattens data sets to extract spatio-value similarity, saving significant data movement and storage costs. This design requires only small changes to cache hardware, introducing minimal energy overhead and integrating easily into commodity systems.

Dimensionality can also be exploited to alleviate computation costs in the processor microarchitecture. The ability to accurately predict changes in control flow (i.e., branches) is crucial to processor performance. Architects generally follow a simple philosophy: the outcome of a branch is likely to correlate with the outcome of some branches earlier in time. However, it is still very difficult to identify exactly which branches correlate with each other in an application. I introduce the concept of multidimensional branch prediction [1, 2], recognizing that correlations are easier to identify when hardware is aware of spatial structure in the application. I propose the Wormhole branch predictor, which infers the dimension lengths of application data structures by tracking the sizes of loops in the code. This spatial information is then used to determine which branches correlate based on how far apart they are from each other, enabling more accurate prediction. This motivated a collaboration with André Seznec from INRIA in implementing a very low-cost multidimensional branch predictor—the Innermost Loop Iteration (IMLI) counter [17]—which integrates easily into commodity processors. This work is recognized by the computer architecture community as one of the top research contributions of 2015 [18].

Temporal Context of Data. Is there benefit to making hardware more aware of when information is needed by the application? Due to physical limitations in technology scaling, we are in need of new ways to maximize utilization of precious on-chip resources. I propose techniques that exploit temporal criticality to enable more efficient data movement and scheduling of computation in hardware.

The abundance and heterogeneity of compute units integrated on a processor chip are rapidly increasing. Thus the underlying communication fabric—the network-on-chip (NoC)—is becoming more complex in design and contributing more to overall performance and energy. For decades, architects have followed a simple philosophy: data should be delivered no later than needed by the application. However, with the pressure from abundant cores and the emergence of energy-critical devices, this philosophy needs rethinking. I argue that the philosophy is fundamentally incomplete: data should be delivered both no later *and no earlier* than needed by the application. I present a limit study on data criticality [14], demonstrating significant energy wasted due to early data movement in traditional NoC architectures. Recognized as a best paper candidate, the study finds that there is considerable disparity in when contiguous data elements are used by the application, naturally occurring due to common programming paradigms (e.g., unbalanced instruction flows, fragmentation, thread synchronization, preemption). I propose NoCNoC, a design that infers temporal context within contiguous blocks and normalizes fetch speed accordingly to deliver data right on time, eliminating wasted energy. Extending this, I mentored a Master's student in developing the Runahead NoC [6], a lossy accelerator for time-critical data that integrates easily as a lightweight, plug-and-play NoC for energy-constrained, heterogeneous platforms.

Efficiently allocating computation to hardware resources is a major concern considering the shrinking

	data context			efficiency gains		
	functional	spatial	temporal	compute	storage	communication
load value approximation [12, 13]	✓					✓
multidimensional branch prediction [1, 2]		✓		✓		
texture cache approximation [19]	✓					✓
network-on-chip data criticality [14]			✓			✓
cache value similarity [11]	✓				✓	
inner loop iteration branch prediction [17, 18]		✓		✓		
network-on-chip data lossiness [6]			✓			✓
approximate computing taxonomy [7]	✓			✓	✓	✓
anytime automaton model [15, 16]	✓		✓	✓		
cache spatio-value similarity [10]	✓	✓			✓	✓

Table 1: Summary of research contributions in chronological order.

of active chip area due to technology scaling limitations. Collaborating with Viji Srinivasan, Ravi Nair and Dan Prener from IBM Research, I introduce the Anytime Automaton [15, 16], a new computation model that addresses both functional and temporal context. It recognizes that for many applications, intermediate results can yield an acceptable approximation of the application's final output. The model pipelines computations such that an application produces its output with increasing quality over time and can be interrupted if necessary, still yielding a valid result. This is inspired by the progressive thinking of the brain, improving its answer the longer it thinks. With this computing paradigm, time-critical computations are prioritized in hardware, while other computations that work on improving quality can be scheduled on a best-effort basis. This work has been recognized by the computer architecture community as one of the top research contributions (honourable mention) of 2016 and has already inspired new research in applying anytime approximations to distributed computing [4].

Summary. In my dissertation, I present new ideas and insights that address the lack of context in how data is computed, stored and communicated in hardware (Table 1). These contributions broadly span computer architecture, touching on caches, memory systems, branch predictors, computation models and networks-on-chip.

Future Work

I plan to continue exploring the potential of better data contextualization in hardware, developing solutions to ongoing challenges across the system stack. I work towards ideally efficient computation fabrics: systems that are omniscient of the information they process.

Open Challenges in Computer Architecture. Mobile devices are becoming increasingly user-interactive while IoT is relying more on intermittent, harvested energy sources. This motivates a rethinking of traditional architectures to efficiently support interactive, interruptible execution. My work on the anytime automaton marries well with this new execution model. I will design anytime architectures that exploit approximation in a way where users and energy budgets dynamically dictate the amount of time and energy expended versus the quality of the application output. In particular, I will build on the insight that an anytime automaton can leverage past hardware innovations in speculative processing (e.g., thread-level speculation) for the purposes of anytime approximation. Anytime architectures motivate a paradigm of *hold-the-power-button* computing: the architecture progressively works towards higher quality results based on the constraints of its users or environment. This interactive, anytime approach can make approximate computing more practical and industry-friendly, giving users more control over the quality of results without compromising correctness.

Memory continues to be a major consumer of system energy, motivating the need for more efficient utilization of storage capacity. Taking advantage of recent advancements in near-data processing, my research will explore new in-memory transformations and organizations of data. Given the spatial and temporal characteristics of data sets, data elements can be rearranged to maximize DRAM row buffer utilization and prioritize critical information. Inspired by recent work on approximate storage in solid-state memories [5, 9], I will design new context-aware error correction schemes where the degree of detection/correction is inherently proportional to a bit's impact on application quality.

With applications striving for more parallelism, traditional architectures need to support more efficient data sharing on-chip. Recent work demonstrates the performance benefit of exposing functional context via data commutativity to cache coherence [21]. Stretching this further, I will investigate techniques that exploit approximation in the coherence protocol. Specifically, protocols can alleviate both on- and off-chip traffic via 1) approximate data aggregation and reduction in the cache hierarchy, and 2) approximately silent updates to data (i.e., minor modifications that have little impact on application quality). I also aim to develop an information-centric on-chip communication fabric, taking inspiration from internetworking. This involves building NoC architectures that support subscription-based data sharing as opposed to traditional point-to-point routing methods. This can enable more efficient data aggregation and multicasting as chips scale to thousands of cores.

Open Challenges Across the System Stack. Collaborating with researchers in operating systems, compilers and programming languages, I plan to explore how leveraging data context can solve problems that stretch across the system stack. In particular, I will tackle inefficiencies in parallel programming and design formal context-aware interfaces between hardware and software.

Writing high-performance parallel applications continues to be a significant challenge for programmers due to complexities in thread synchronization and load balancing. First, I will investigate how the operating system can exploit data context for more intelligent synchronization primitives. In particular, I will extend my work on value approximation to employ memoization techniques on critical sections. This involves training and inferring the results of critical sections—which are generally small—using machine learning or other predictive models in order to bypass the overheads of locking. I aim to push this further to transactional memory systems. To minimize transaction abort rates as applications scale to thousands of threads, I will design conflict detection schemes that exploit spatial context, selectively allowing and disallowing conflicts based on the expected impact on application quality. Second, I will explore how data criticality information can be used to accurately infer thread criticality. Similar to my limit study on NoC data criticality, I will investigate how variance in the temporal context of data fundamentally impacts load imbalance among threads. Such a study can aid programmers and system designers in reasoning about how data sharing governs which threads fall on the critical path of the application.

Traditional processors are built upon an instruction set architecture (ISA), which formally specifies instruction context from the application to the underlying hardware. For example, opcodes explicitly tell the hardware which functional unit to use for a given instruction without any ambiguity. However, modern processors do not have a corresponding *data set architecture* (DSA). The hardware cannot simply look at a 32-bit word in isolation and determine if it represents an integer, a floating-point number, a pointer address, a bit field, a byte array, or something else. It does not know how to interpret data bits on their own; it only interprets them based on how instructions use them. This fundamentally limits the efficiency of moving and storing data on-chip. The long-term goal of my research is to design DSAs—either as ISA enhancements or as completely new architectures—and consequently develop new programming paradigms and compiler extensions to leverage them. Recent work formalizes constructs for specifying approximate data in programs [3]. Building on this, I will explore ways of specifying spatial and temporal context that are nonintrusive for programmers yet still informative to the hardware. In one approach, programmers can provide hints in their declarations of data, either implicitly in the compiler or explicitly via language extensions. For example, the ordering of members in a struct can hint at 1) their significance to output quality, or 2) their temporal criticality in application execution. Another approach is to support spatial hints by introducing a notion of points of interest in data sets (e.g,. moving objects in video frames, news articles based on date of publication). Such points of interest can inform the hardware of data regions that are likely to demand 1) more computational resources, or 2) higher approximation quality. I will then explore how DSAs can be efficiently represented in hardware; how is data context information explicitly passed down to the processor? This can be done via 1) context reference tables that are tightly coupled to the microarchitecture, 2) virtual data reorganization, or 3) in-memory headers that effectively packetize data sets. By working towards DSAs, my research aims to build computers that are far more intelligent and knowledgeable of the information they process, providing efficient support for the data-intensive processing on today's energy-constrained devices.

References

[1] J. Albericio, J. San Miguel, N. Enright Jerger, and A. Moshovos, “Wormhole branch prediction using multi-dimensional histories,” in *Championship Branch Prediction (CBP-4) held in conjunction with the ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2014.

[2] J. Albericio, J. San Miguel, N. Enright Jerger, and A. Moshovos, “Wormhole: Wisely predicting multi-dimensional branches,” in *ACM/IEEE International Symposium on Microarchitecture (MICRO)*, 2014.

[3] B. Boston, A. Sampson, D. Grossman, and L. Ceze, “Probability type inference for flexible approximate programming,” in *ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, 2015.

[4] N. Ferdinand and S. Draper, “Anytime coding for distributed computation,” in *IEEE Allerton Conference on Communication, Control, and Computing (Allerton)*, 2016.

[5] Q. Guo, K. Strauss, L. Ceze, and H. S. Malvar, “High-density image storage using approximate memory cells,” in *ACM/IEEE International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2016.

[6] Z. Li, J. San Miguel, and N. Enright Jerger, “The runahead network-on-chip,” in *ACM/IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2016.

[7] T. Moreau, J. San Miguel, M. Wyse, J. Bornholt, L. Ceze, N. Enright Jerger, and A. Sampson, “A taxonomy of approximate computing techniques,” UW CSE Technical Report UW-CSE-2016-03-01, Tech. Rep., 2016.

[8] P. V. Rengasamy, A. Sivasubramaniam, M. T. Kandemir, and C. R. Das, “Exploiting staleness for approximating loads on CMPs,” in *ACM/IEEE International Conference on Parallel Architecture and Compilation Techniques (PACT)*, 2015.

[9] A. Sampson, J. Nelson, K. Strauss, and L. Ceze, “Approximate storage in solid-state memories,” in *ACM/IEEE International Symposium on Microarchitecture (MICRO)*, 2013.

[10] J. San Miguel, J. Albericio, N. Enright Jerger, and A. Jaleel, “The bunker cache for spatio-value approximation,” in *ACM/IEEE International Symposium on Microarchitecture (MICRO)*, 2016.

[11] J. San Miguel, J. Albericio, A. Moshovos, and N. Enright Jerger, “Doppelganger: A cache for approximate computing,” in *ACM/IEEE International Symposium on Microarchitecture (MICRO)*, 2015.

[12] J. San Miguel, M. Badr, and N. Enright Jerger, “Load value approximation,” in *ACM/IEEE International Symposium on Microarchitecture (MICRO)*, 2014.

[13] J. San Miguel and N. Enright Jerger, “Load value approximation: Approaching the ideal memory access latency,” in *Workshop on Approximate Computing Across the System Stack (WACAS) held in conjunction with the ACM/IEEE International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2014.

[14] J. San Miguel and N. Enright Jerger, “Data criticality in network-on-chip design,” in *ACM/IEEE International Symposium on Networks-On-Chip (NOCS)*, 2015.

[15] J. San Miguel and N. Enright Jerger, “The anytime automaton,” in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2016.

[16] J. San Miguel, V. Srinivasan, R. Nair, and D. A. Prener, “A systolic approach to deriving anytime algorithms for approximate computing,” IBM Research Report RC25600, Tech. Rep., 2016.

[17] A. Seznec, J. San Miguel, and J. Albericio, “The inner most loop iteration counter: A new dimension in branch history,” in *ACM/IEEE International Symposium on Microarchitecture (MICRO)*, 2015.

[18] A. Seznec, J. San Miguel, and J. Albericio, "Practical multidimensional branch prediction," *IEEE Micro Top Picks from the Computer Architecture Conferences*, 2016.

[19] M. Sutherland, J. San Miguel, and N. Enright Jerger, "Texture cache approximation on GPUs," in *Workshop on Approximate Computing Across the Stack (WAX) held in conjunction with the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, 2015.

[20] D. Wong, N. S. Kim, and M. Annavaram, "Approximating warps with intra-warp operand value similarity," in *ACM/IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2016.

[21] G. Zhang, W. Horn, and D. Sanchez, "Exploiting commutativity to reduce the cost of updates to shared data in cache-coherent systems," in *ACM/IEEE International Symposium on Microarchitecture (MICRO)*, 2015.