SLIC: Statistical Learning in Chip

Ronald D. Blanton, Xin Li, Ken Mai, Diana Marculescu, Radu Marculescu, Jeyanand Paramesh, Jeff Schneider, and Donald E. Thomas
Electrical and Computer Engineering Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{blanton, xinli, kenmai, dianam, radum, paramesh, thomas}@ece.cmu.edu and schneide@cs.cmu.edu

***Abstract*—Despite best efforts, integrated systems are "born" (manufactured) with a unique 'personality' that stems from our inability to precisely fabricate their underlying circuits, and create software a priori for controlling the resulting uncertainty. It is possible to use sophisticated test methods to identify the best-performing systems but this would result in unacceptable yields and correspondingly high costs. The system personality is further shaped by its environment (e.g., temperature, noise and supply voltage) and usage (i.e., the frequency and type of applications executed), and since both can fluctuate over time, so can the system's personality. Systems also "grow old" and degrade due to various wear-out mechanisms (e.g., negative-bias temperature instability), and unexpectedly due to various early-life failure sources. These "nature and nurture" influences make it extremely difficult to design a system that will operate optimally for all possible personalities. To address this challenge, we propose to develop statistical learning in-chip (SLIC). SLIC is a holistic approach to integrated system design based on continuously learning key personality traits on-line, for self-evolving a system to a state that optimizes performance hierarchically across the circuit, platform, and application levels. SLIC will not only optimize integrated-system performance but also reduce costs through yield enhancement since systems that would have before been deemed to have weak personalities (unreliable, faulty, etc.) can now be recovered through the use of SLIC.**

***Keywords*— Integrated system design; low-power design; statistical and machine learning**

I. INTRODUCTION

The most challenging problems in science and engineering are so incredibly complex that many have turned to statistical learning (SL) to derive accurate models from various forms of empirical data. Major advances in SL have resulted in algorithms that can now cope with significant amounts of high-dimensional data, and most importantly, are sufficiently robust to rely upon in critical applications. A popular use of SL is in two-step process optimization. The first step learns a model that approximates the relationship between system parameters and the resulting system performance. This model is constructed on-line from data collected during system operation. The second step uses active learning where the learned performance model is analyzed to determine which parameter settings to try next. Active learning trades off the need to experiment untested areas of the parameter space in order to gain more information for learning, against the objective of selecting parameters that are likely to yield optimal performance. Often the "learner" must accomplish this in the face of non-stationarity.

The design, manufacture and operational characteristics (e.g., yield, performance, reliability, power, security, etc.) of modern integrated systems also exhibit extreme levels of complexity that similarly cannot be easily modeled or predicted from first principles. In this nanoscale era, manufacturers find it increasingly difficult to control fabrication, thus making every aspect of design (circuits, logic, memory, communication networks, cores/uncores, etc.) a grand challenge. Moreover, the operating environment of a system which is characterized by temperature, supply voltage, the amount of noise, etc. also adds a level uncertainty that is extremely difficult to deal with optimally at the time of design. Finally, the fact that the use of an integrated system may vary widely from user to user adds yet another major source of uncertainty. These sources all combine together in the worst possible ways to establish an overall level of uncertainty that leads to systems that exhibit non-optimal performance, or require excessive resources to design and fabricate. For example, modern portable, multimedia devices such as a tablet computer require millions of engineering hours to integrate several SoCs (systems on chip) including multiple radios, DSPs, µPs, application-specific processors, display drivers, and solid-state memories, altogether which execute a variety of applications. The uncertainty exhibited by the integration and use of various heterogeneous sub-systems within diverse environments can be better optimized by learning and then adapting.

II. STATISTICAL LEARNING IN CHIP

We propose to develop new SL algorithms that enable an integrated system to learn and adapt operation across the system stack (i.e., the circuit, platform, and application levels) [1]. Conventional approaches to SL assume learning takes place on server farms characterized by virtually unlimited compute and storage resources. Integrated systems, on the other hand, have stringent constraints on power and security, and thus require a more compressed learning cycle which means a complete re-thinking of SL is necessary for it to be effective within an integrated-system environment.

While there is a great deal of active research that individually addresses each source of uncertainty within an integrated system, we instead want to tackle them all simultaneously using a universal solution that we call a self-evolving system. A self-evolving system has the ability to adapt and evolve to changes and unknowns encountered both

This work has been supported in part by the National Science Foundation of the United States under contract no. 1314876.

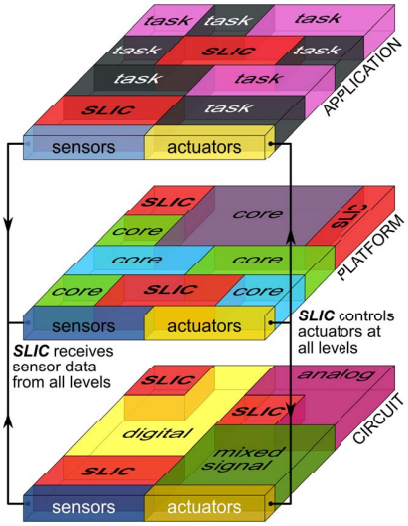

Figure 1: A self-evolving system.

at the time of manufacturing and over the lifetime of the system. Figure 1 shows a hierarchical view of a self-evolving system that consists of a bottom layer of circuits that together form a variety of cores at the middle (platform) layer, which are used by various tasks at the top layer to execute one or more applications. Sensors at the circuit, platform, and application levels collect various forms of data so that the state of the system, from various perspectives, can be learned. This global view is used to evolve the system into a new state (via the various actuators shown) for improving a wide-variety of system attributes (performance, power, reliability, etc.). At the circuit level, temperature, voltage, and frequency are likely quantities to be sensed. For a platform of heterogeneous cores (e.g., computation, memory, etc.), this same data would be useful but in addition, various real-time measures of workload, queue occupancy, communication among different modules, etc. would be collected as well. Finally, for an application viewed as a plethora of interacting tasks, sensors of various sorts are likely to already exist for enabling the application. For instance, the brain-computer interface (BCI) application discussed later has a sensor array for measuring neural signals. All of this sensed data is provided to the SLIC (statistical learning in chip) cores for learning how to improve the behavior of the entire system which is achieved by providing new parameter values to the level-specific actuators shown in Figure 1. SLIC cores (SCs) perform the learning and are employed at the circuit, platform and application levels, and across levels. SCs are implemented in custom hardware, software, or even "in the cloud" depending on the amount of data and the time allotted for active learning. Actuators take on various forms, but in general they are "control knobs" that allow one or more operational parameters to be fine-tuned. For instance, at the circuit level, these are controls for supply voltage, clock frequency and body bias. At the platform level, actuators can take the form of decisions on what memory accesses to grant [2] [3], what communication policy to invoke [4], and what resources to utilize or avoid to ensure both reliability and availability. At the application level, actuators may naturally exist already such as the sensitivity of a robotic prosthetic controlled by a BCI. In other applications, actuators may not be inherently present, but can always take the form of real-time optimizers that improve application-level performance using various statistical-learning techniques.

There have been many publications focused on using hardware to speedup various learning approaches (neural networks, decision trees, etc.) [5] [6] [7] [8] [9] [10]. It cannot be overly emphasized however, that SLIC does not fall into that category. Our objective instead is to integrate a comprehensive learning capability that applies to all levels (circuit, platform and application) within an integrated system.

III. Learning Background

SLIC will have the flexibility to implement several specific learning algorithms but all will implement the forward model illustrated in Figure 2.

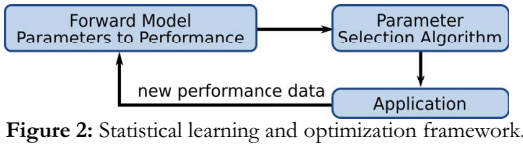

Figure 2: Statistical learning and optimization framework.

The conventional forward model accepts training data in the form of previous parameter settings and the resulting performances. It builds a model that predicts what performance will be achieved by a hypothetical, new parameter selection. A key element of these predictions is that they come with uncertainty estimates derived from the variance in the training data and the amount of relevant training data for each prediction. A trivial example of learning is a simple linear regression. In this case, learning is simply the process of fitting the regression parameters, and the "training data" is just the data used to perform the fit. "Testing data" refers to any data used later to query the fitted model for checking how well its predictions perform relative to the true values in the data. As described next, we propose more sophisticated methods that can fit non-linear and discrete models, but the basic concepts are the same.

The parameter-selection method uses the forward model to select parameters for the next learning cycle. Selection typically involves addressing an exploration/exploitation problem where a tradeoff has to be made by selecting parameters that are expected to perform well versus those that have uncertainty, and thus could provide useful data for improving the forward model. The selected parameters are then applied in the target system and new data is collected on the resulting performance that is used to update the forward model (i.e., included in the training data for the model). The algorithms developed for SLIC depend on features of the target application. Here we outline some scenarios. The highest-level distinction between scenarios is whether the parameters represent a continuous-valued space of possible policies, or if they are a discrete set of alternative policies.

Discrete Policy Choices. Figure 3 shows the performance of a hypothetical forward model where there are four alternative policies the system can choose. This model does not attempt to generalize what it learns between policies. The mean and confidence intervals for the performance of each policy are simply tabulated based on the data collected when the corresponding policy is employed. In this form, the problem is a classic multi-armed bandit [11]. Each alternative is treated independently, and we want to learn the performance of each while concentrating most of the trials on the best performers. For instance, when dealing with memory accesses, policy choices include first-come first-serve (FCFS), round-robin, or row-buffer hit-first strategies to optimize the overall system performance. There are two common methods for parameter selection. The first is the UCB1 (Upper Confidence Bounds) algorithm [12]. It provides regret bounds for arbitrary reward distributions and can be computed quickly based on the means and numbers of samples from each policy. The second is Gittins indices [13]. These have the benefit of yielding optimal (in expectation) choices for the time-discounted case when the distributional assumptions on reward are correct. These two algorithms are ideal for SLIC since they both require little overhead for implementation.

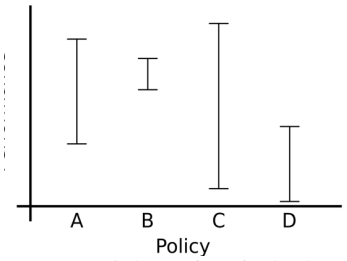

Figure 3: A forward model for four alternative policies.

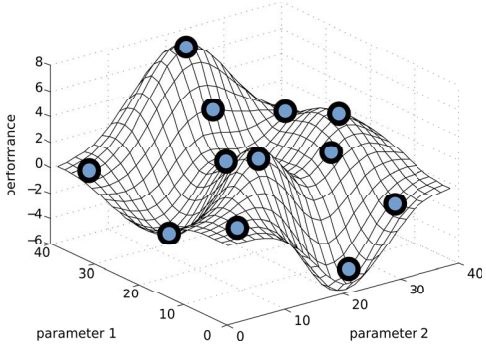

Figure 4: A hypothetical forward model with two continuous parameters.

Continuous Parameters. Figure 4 is a second hypothetical forward model where there are two parameters that control a policy. The circles represent data points collected by choosing specific parameter settings and observing the resulting performance. The surface is a current estimate of the function that maps parameter values to performance. While there are many possible function approximators that could produce this estimate, Gaussian processes are quite interesting [14] since they provide the ability to model nonlinear functions and also yield confidence intervals (not shown in the figure) on their estimates. For example, we could use these and similar approaches to tune the transistor back-bias voltage to balance performance and leakage of an arithmetic-logic unit (ALU) or SRAM (static random access memory). Or in a system that implements dynamic voltage frequency scaling (DVFS), we can use this approach to identify the optimal voltage and frequency combination for a time segment of the specific application workload [15]. For a BCI application, these techniques can be used to deal with neuroplasticity.

It has been shown that good empirical performance can be achieved by modifying bandit selection algorithms for the continuous case even when the theoretical assumptions are violated [16]. With that in mind, the UCB1 and Gittins methods can be extended from the discrete alternatives case for use with continuous parameters. One approach simply evaluates a point in the continuous space as if it were a discrete alternative by retrieving its mean, variance, and effective number of supporting data points from the function approximator. Then a set of candidate points are evaluated as discrete alternatives. Generalization happens through the function approximator, but is not explicitly considered during selection. In higher-dimensional parameter spaces, it may be difficult to explicitly build or search the forward model described here. This is especially true in non-stationarity systems where the performance data becomes stale long before sufficient data has been collected to build a reasonable model. For these scenarios, a "model free" approach can be implemented based on gradient ascent [17]. The only information stored by the forward model will be an estimate of the gradient derived from the most recent data points. The policy selection method will simply choose gradient steps. This method however is subject to being trapped in local minima, but is simpler and faster than the methods based on a full forward model.

Reinforcement Learning. The examples presented so far have assumed that the credit assignment between a newly chosen set of parameters and newly observed performance data is immediate. When time delays in the system mean that performance is the cumulative result of earlier choices, reinforcement learning will be used to deal with the credit-assignment problem. The forward modeling described above may be used to learn value functions and the parameter-selection algorithms may be adapted for Q-learning [18].

IV. Initial Demonstration Projects

To demonstrate the viability of SLIC, several short-term projects have been initiated. One project focuses on SLIC demonstration at the application level involving a brain-computer interface implementation. A second project takes on the challenge of dealing with system aging using an efficient process for test, diagnosis and healing.

Brain-Computer Interface. While statistical learning has been extensively studied for more than three decades, most

learning algorithms were developed for general-purpose computing via a desktop or server. On-chip learning poses a number of unique opportunities and constraints that were not extensively studied in the past. For instance, implementing a learning algorithm with on-chip, fixed-point arithmetic reduces the circuit complexity (e.g., silicon area, power consumption, etc.); however, fixed-point arithmetic often limits the dynamic range and reduces computing accuracy, compared to the double-precision floating-point arithmetic that is often used by a desktop or server. For this reason, there is an immediate need to revise today's learning algorithms to fit the applications where on-chip implementation is essential.

We re-design the learning algorithms by taking into account the circuit-related non-idealities. It is well-known that many learning algorithms (e.g., support vector machine, linear discriminant analysis, logistic regression, etc.) are particularly designed to accommodate input data with embedded large-scale noise. Hence, we take advantage of the inherent error-pruning within these algorithms to robustly handle the circuit-induced errors (e.g., truncation error, quantization error, overflow, etc.) associated with digital computing.

In particular, we will consider the brain computer interface (BCI) as an application example to demonstrate the proposed algorithm design and circuit implementation for on-chip learning. The objective of BCI is to create alternative communication pathways for human patients to interact with the environment via prosthetics and/or wheelchairs when their normal motor function is impaired by amputation, trauma or disease. Towards this goal, neural signals from the brain are sensed and decoded to control a prosthetic limb, an electric wheelchair, or other external devices. Figure 5 shows the major data collection and processing steps for a BCI system.

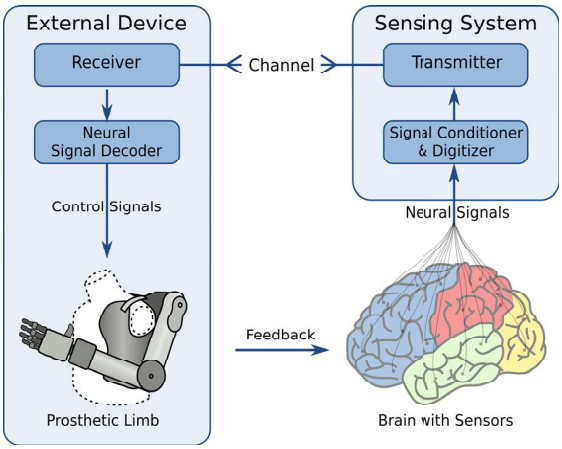

Figure 5: Major data collection and processing steps are shown for a BCI system.

Aging and Wearout Mitigation. Today's underlying technology for integrated systems are susceptible to wearout and aging. While this is not a new phenomenon, the impact of NBTI (negative bias temperature instability) and PBTI (positive bias temperature instability) on transistor threshold voltage is now significant in terms of the amount of circuit slowdown that can be experienced. Additional sources of circuit-timing failure include hot carrier injection (HCI) and time-dependent dielectric breakdown (TDDB). The traditional approach for coping with wearout would be to over design the integrated system so that any slowdown experienced over the lifetime of the system would not cause failure. Over-design typically takes the form of either slowing down the system clock, or sizing-up transistors in critical circuit paths so that no system, regardless of its usage pattern and manufacturing-induced variations (i.e., its personality), is susceptible to the worse-possible case of slowdown. As a result, the system suffers from performance loss and consumes excessive power. Such an approach, while tolerable in the past, is no longer acceptable because of the growing impact of wearout and aging effects.

Another approach for coping with wearout and aging is to aggressively design the system so that it operates optimally, but also include internal capability for the system to monitor and mitigate wearout/aging. Specifically, we plan to develop and evaluate a SLIC-based test and diagnostics methodology for ensuring that aging and wearout is minimized over the lifetime of the system. In this approach, the system is periodically tested using an approach such as the CASP methodology [19]. Any detected slowdown is then pinpointed to a certain system sub-circuit using the on-chip diagnosis methodology developed at Carnegie Mellon [20]. The diagnosis information is then passed to the scheduler so that dynamic task mapping can be invoked so that the affected circuit can be "healed", that is, the aging/wearout process can be significantly reversed by cutting power to the affected circuit and/or scheduling tasks that balances the switching activity experienced by the affected sub-circuit.

Diagnosis is not perfect however since there is often ambiguity in identifying the failing correct sub-circuit. Despite the use of powerful heuristics for improving diagnostic resolution, there is significant likelihood that the wrong sub-circuit is healed. Such a situation is easily brought to light by re-testing the affected circuit to determine if the circuit failure has indeed been healed.

Each test-diagnosis-heal cycle yields a data instance of supervised-learning that can be used by a SLIC core to continuously learn a prediction model for improving diagnosis. We have modified the k-nearest neighbor (kNN) algorithm for efficient, on-chip implementation that is capable of incrementally learning a diagnostic resolution-improving prediction model using the labeled data resulting from the periodic test-diagnosis-heal cycles used to monitor and detect wearout and aging. (Figure 6 shows a block diagram of the hardware implementation.) The expectation is that diagnostic resolution will be significantly improved which means system operation across the hierarchy is improved.

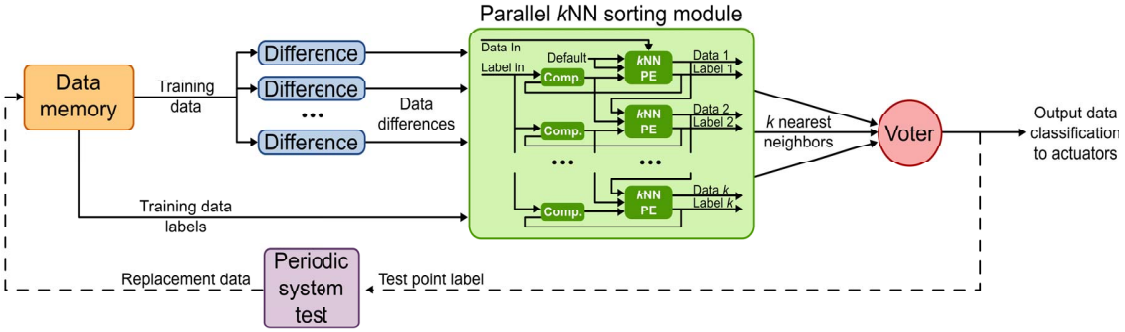

Figure 6: Custom hardware for adaptive learning based on *k* nearest neighbors.

V. SUMMARY

Statistical learning in chip (SLIC) applied to the design and on-line operation of an integrated system has great potential to have significant impact on a number of areas:

- **Design** – It is challenging to design an integrated system so that all of its possible personalities can be seamlessly handled. With SLIC, the burden on the designer is eased since the ever-changing personality of the system can instead be learned and adapted to.
- **Yield** – Currently, an integrated system that does not meet specifications is discarded. With SLIC, it will be possible to increase yield since some flaws in the system personality will be compensated based on learning.
- **Test** –With SLIC, stress testing can be mitigated since changes in operation due to a subtle flaw can be detected and compensated for by learning a model of normal/expected operation.
- **Performance** – SLIC allows the performance optimization across the system stack, allowing the unique system personality to be exploited for maximum gain. This capability is not only critical for mobile integrated systems but will also be quite beneficial for server farms since power for such entities is also paramount.
- **Individualization** – Since SLIC learns the habits of the user, applications that were before learning-agnostic can now be fine-tuned to enhance the overall experience of every individual user. For learning-inherent applications, especially from the medical field, SLIC promises to usher in a new field of personalized medical instrumentation.

REFERENCES

[1] D. Ricketts et al., "Enhancing CMOS using Nanoelectronic Devices, a Perspective on Hybrid Integrated Systems," *Proceedings of the IEEE*, pp. 2061-2075, Dec. 2010.

[2] E. Ipek et al., "Self-Optimizing Memory Controllers: A Reinforcement Learning Approach," *International Symposium on Computer Architecture*, pp. 39-50, 2008.

[3] J. Martinez and E. Ipek, "Dynamic Multicore Resource Management: A Machine Learning Approach," *IEEE Micro*, Sept./Oct. 2009.

[4] C. -L. Chou and R. Marculescu, "User-Aware Dynamic Task Allocation in Networks-on-Chip," *ACM/IEEE Design, Test and Automation in Europe*, 2009, pp. 1232-1237.

[5] V. Prabha and E. Moine, "Hardware Architecture of Reinforcement Learning Scheme for Dynamic Power Management in Embedded Systems," *EURASIP Journal on Embedded Systems*, 2007.

[6] Z. Baker and V. Prasanna, "N Architecture for Efficient Hardware Data Mining using Reconfigurable Computing Systems," *IEEE Symposium on Field-Programmable Custom Computing Machines*, 2006, pp. 67-75.

[7] Z. Baker and V. Prasanna, "Efficient Hardware Data Mining with the Apriori Algorithm on FPGAs," *IEEE Symposium on Field-Programmable Custom Computing Machines*, 2005, pp. 18-20.

[8] R. Narayanan et al., "An FPGA Implementation of Decision Tree Classification," in *ACM/IEEE Design, Automation and Test in Europe*, 2007, pp. 16-20.

[9] Y. Luo, K. Xiang, and S. Li, "Acceleration of Decision Tree Searching for IP Traffic Classification," *ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pp. 40-49, 2008.

[10] S. Chkrabarty and G. Cauwenberghs, "Sub-Microwatt Analog VLSI Trainable Pattern Classifier," *IEEE Journal of Solid-State Circuits*, May 2007.

[11] D. Berry and B. Fristedt, *Bandit Problems: Sequential Allocation of Experiments*.: Chapman and Hall, 1985.

[12] P. Auer, N. Cesa-Bianchi and P. Fischer, "Finite-time Analysis of the Multiarmed Bandit Problem," *Machine Learning*, 2002.

[13] J. Gittins, *Multi-Armed Bandit Allocation Indices*.: Wiley, 1989.

[14] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*.: MIT Press, 2006.

[15] D. C. Juan et al., "Learning the Optimal Operating Point for Many-Core Systems with Extended Range Voltage/Frequency Scaling," CODES+ISSS, Oct. 2013.

[16] J. Schneider and A. Moore, "Active Learning in Discrete Input Spaces," *Interface Symposium*, 2002.

[17] P. C. Pendharkar, "A Comparison of Gradient Ascent, Gradient Descent and Genetic-Algorithm-Based Artificial Neural Networks for the Binary Classification Problem," *Expert Systems*, pp. 65-86, May 2007.

[18] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*.: MIT Press, 1998.

[19] Y. Li, S. Makar, and S. Mitra, "CASP: Concurrent Autonomous Chip Self-Test using Stored Test Patterns," *Design Automation and Test in Europe*, 2008.

[20] M. Beckler and R. D. Blanton, "On-Chip Diagnosis for Early-Life and Wear-Out Failures," *International Test Conference*, 2012.