Ultra Efficient Embedded SOC Architectures based on *Probabilistic* CMOS (PCMOS) Technology

Lakshmi N. Chakrapani, Bilge E. S. Akgul, Suresh Cheemalavavgu, Pinar Korkmaz, Krishna V. Palem, Balasubramanian Seshasayee

1. ABSTRACT

The major impediments to technology scaling towards nanometer regime include power dissipation and "erroneous" behavior due to process variations and noise susceptibility. In this paper, we demonstrate that CMOS *devices whose behavior is rendered probabilistic through noise (yielding probabilistic* CMOS *or* PCMOS*) can be harnessed for ultra low energy computation.* PCMOS *devices are inherently probabilistic in that they are guaranteed to compute correctly with a probability p, and by design, they are expected to compute incorrectly with a probability* $(1-p)$*. In this paper, we show for the first time that* PCMOS *technology not only yields energy savings at the device level, but also yields significant savings, simultaneously, in the energy consumed as well as in the performance for probabilistic applications drawn from the embedded computing domain. These benefits are derived using a novel algorithm-technology co-design methodology for* PCMOS *based co-processors. All of our application level savings are quantified using the product of the energy and the performance denoted energy* $\times$ *performance: the* PCMOS *based savings range from a substantial multiplicative factor of over* 560 *when compared to a competing conventional* CMOS *based realization.*

School of Electrical and Computer Engineering, Georgia Institute of Technology, USA
{nsimhan,bilge}@ece.gatech.edu
This work is supported in part by DARPA under Seedling #F30602-02-2-0124

2. INTRODUCTION

As CMOS technology scales down into the nanometer region, noise and other perturbations (see Sano [12, 20], Kish [9] and Shepard [21]) pose increasing number of challenges. The surprising premise that noise can be harnessed as a resource, rather than viewed as an impediment has been shown using foundational principles and theoretical models [14, 13, 15]. In prior work [3, 2], we have designed and studied CMOS devices based on these principles though analytical models and simulations. In this work, we demonstrate for the first time that computing platforms based on such CMOS devices can yield orders of magnitude improvements simultaneously to the *energy consumed* as well as to the *running time*—collectively characterized as the energy-performance product (EPP)—of an application. A singular innovation through which these savings are accomplished is the particular form of CMOS that is affected by ambient (thermal) noise—we refer to it as *probabilistic* CMOS or PCMOS. The two significant contributions of this paper are (i) the development of a methodology for using PCMOS to realize ultra efficient embedded computing platforms in the energy-performance sense, and (ii) the demonstration of the value of this novel technology in the context of a range of embedded applications of interest.

To demonstrate the utility and the efficacy of PCMOS, we first develop a methodology (akin to hardware software co-design), described in Section 4 that we refer to as *algorithm-technology* co-design. Our methodology is aimed at realizing an extremely efficient *probabilistic system-on-a-chip* (PSOC) architectures using PCMOS devices. As shown in Figure 2, a canonical PSOC architecture consists of a (conventional) host processor used to compute most of the control-intensive *deterministic* components of an application, whereas the co-processor realized using PCMOS devices will be used as an energy-performance (EPP) accelerator. The reasons for emphasizing the development of this co-design methodology as a significant contribution is based on the following two observations. First, the "probabilistic content" (formalized later as *flux*) of the algorithm becomes a novel resource to be managed and treated, much as space requirements, flexibility and IP-reuse are treated in the traditional co-design context. Furthermore, as we will see in the sequel, considerations of design efficiency differ significantly in the context of PCMOS when compared to those arising in the context of conventional CMOS.

Applications based on *probabilistic algorithms* benefit the most from PSOC implementations. Probabilistic algorithms find wide use in a range of embedded applications ranging from speech and pattern recognition to security. To evaluate the benefits of PCMOS based architectures, we consider a set of applications (Section 4) and four alternate implementations of the probabilistic applications (Section 3) and present the gains in Section 5. In addition, in Section 6, we study another crucial aspect of computing platforms that implement probabilistic algorithms. In application domains employing probabilistic algorithms, independent probabilistic bits are needed in copious quantities. This trend is very favorable to our design approach since it increases the flux. Nevertheless, techniques for producing independent random bits are difficult and are an extensive area of study [17] with several complex approaches yielding poor results [6]. We show that, while yielding significant gains in the EPP, PCMOS technology also yields significantly better quality random bits, verified by applying the tests provided by the National Institute of Standards and Technology (NIST) [19]. Concluding remarks and directions for future research are presented in Section 7.

3. PROBABILISTIC SYSTEM ON A CHIP ARCHITECTURES

As mentioned in the introduction, the surprising premise that CMOS devices rendered probabilistic due to noise, are not only useful but also yield energy and performance benefits at the application level, will be demonstrated using probabilistic system on a chip architectures (PSOCs). For completeness, we first present a brief overview of *probabilistic* CMOS (PCMOS) technology (for a detailed description, please see [3, 2]).

3.1 PCMOS Technology

It has been established in prior work that CMOS devices have an exponential relationship between the probability of correctness (p) and the switching energy (E). In addition, the relationship between the noise RMS and the switching energy E is quadratic. These two relationships formalized as the two PCMOS laws characterize the behavior of PCMOS

devices. These laws, derived from analytical modeling of noise susceptible switches, have been extensively studied and verified using HSpice simulations as well as actual fabrication and measurement of PCMOS devices in TSMC $0.25\mu m$ technology. In this paper, we use these PCMOS switches as building blocks to demonstrate their benefits to applications in the context of a typical PSOC architecture.

3.2 Canonical PSOC Architectures

To effectively leverage PCMOS technology and to compare with computing platforms based on conventional CMOS technology, algorithm implementation in four scenarios shown in Figure 1 are considered: (a) the *best possible* deterministic algorithm solving the same problem, implemented completely in software and executing on the host processor (in our case a StrongARM SA-1100), (b) the probabilistic counterpart executing completely on the host processor, with pseudo random bits generated by a software implementation of a well known algorithm [17] (c) the probabilistic algorithm executing on the host processor with a conventional CMOS co-processor (referred to as the "conventional CMOS based SOC") or (d) with a functionally identical PCMOS co-processor. Collectively, these four cases encompass all reasonable

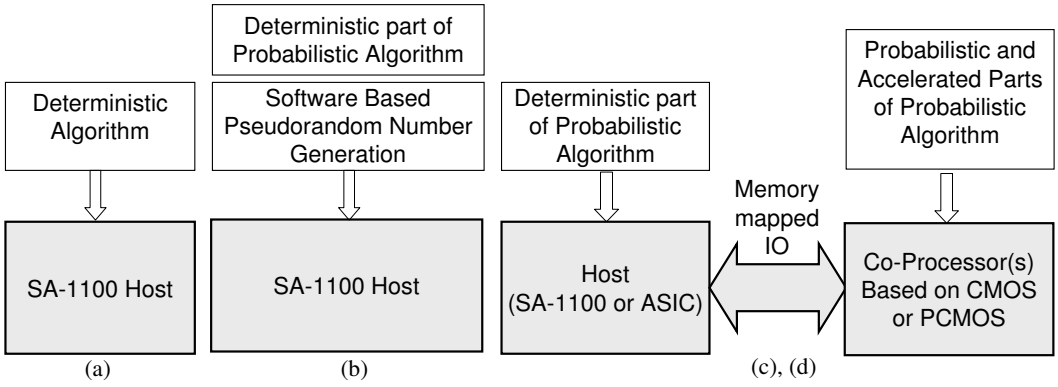

Figure 1. The Host + Co-processor style implementations that are compared

alternate implementations of the application. Throughout this study, the co-processors are application-specific.

3.3 Performance and Energy Modeling of PSOC Architectures

To estimate the performance of these PSOC and SOC architectures, the IMPACT simulator of the Trimaran infrastructure [23] has been modified to measure the number of cycles taken by an application executing on a StrongARM SA-1100 host. The simulator records a trace of the activity of the PCMOS and CMOS co-processors. The combination of this information with the performance models of the co-processors, typically obtained through HSpice simulations or physical chip measurements (of PCMOS switches) yields the PSOC (SOC) performance in terms of execution time.

The energy consumption of an application executing on a PSOC (SOC) architecture is the sum of the energy consumed by the host, the energy consumed by the PCMOS (CMOS) co-processor(s) and the energy cost of communication between the host and the co-processor(s). The co-processors are memory mapped and communication is through load-store instructions executed on the host. To quantify the energy consumed by the SA-1100 host, the JouleTrack model [22] is used. This model is reported to be within 3% of the energy measured on an actual SA-1100 host. The power modeling techniques applied to various components of the PSOC (SOC) architecture are illustrated in Figure 2. Since the design of the co-processors are application-specific, the energy consumed by a particular co-processor is different for each of the applications. The CMOS based co-processors are designed and synthesized into TSMC $0.25\mu m$ process, and the energy cost and performance are derived from HSpice simulations. In the context of extensions based on PCMOS, the energy cost of the co-processor is derived from *physical chip measurements* of functioning PCMOS switches realized in TSMC

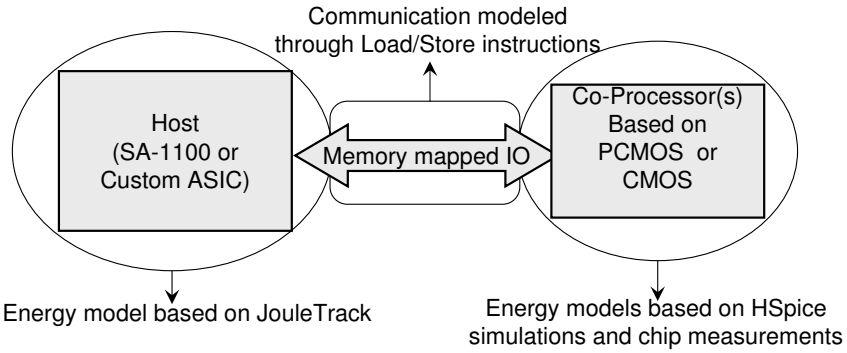

Figure 2. The Host+Co-processor Architecture and Power Modeling.

$0.25\mu m$ process. In all of the styles of PSOC implementations, to account for the probabilistic nature of the applications, several "runs" are averaged.

4. THE PSOC CODESIGN FRAMEWORK

As mentioned in the introduction, applications based on probabilistic algorithms benefit from PSOC implementations. In this study, we consider applications based on probabilistic algorithms that implement *Bayesian Inference* [10], *Random Neural Networks* [8], *Probabilistic Cellular Automata* [7] and *Hyper-Encryption* [4]. Any PSOC implementation of a probabilistic application involves *partitioning* the application between the host and the (application specific) PCMOS based co-processor. Even though the exact host - co-processor partition and the corresponding PCMOS based co-processor architecture for each of these applications vary, they follow a common theme. Common to these applications (and to almost all probabilistic algorithms) is the notion of a *core probabilistic step* with its associated probability parameter p. This core probabilistic step is manually identified and implemented in PCMOS. The deterministic parts of the application are implemented as software executing on the host processor. This co-design methodology is unique in the sense that as opposed to traditional SOC designs, several unique algorithm and technology characteristics need to be considered to obtain efficient designs.

4.1 Algorithm and Technology Characteristics Influencing Codesign

PCMOS is particularly efficient in computing with ultra-low energy. For example, the energy consumed for generating one random bit by PCMOS is 0.4 pico Joules [2]. By contrast, the Park-miller algorithm [17] implemented in custom hardware consumes about 2025 *times* that energy. Given this benefit, it is natural to expect that higher amounts of "probabilistic content" in the algorithm will yield greater opportunities gaining from the use of PCMOS. Thus the amount of "probabilistic content" will be a figure of merit, which we refer to as flux $\mathcal{F}$ defined as the *total number of primitive probabilistic operations* O of the algorithm.

Though PCMOS is extremely energy efficient the operating frequencies of our current design is low [16] at about 1 MHz. By contrast, software and CMOS based pseudo-random bit generators produce random bits at the rate of 3.33 and 4 million bits per second respectively. The peak rate at which an application demands random bits, or *the peak application demand bandwidth* is an application characteristic of interest. If the peak application demand bandwidth exceeds the bandwidth of the PCMOS device, the PCMOS devices need to be replicated. This is captured by the *Replication factor* $\mathcal{R}$. Current realizations of PCMOS devices do not allow on demand activation; by contrast, in SOC designs clock- and data-gating are assumed to be exploited to reduce the energy consumption. In this regard, we are conservative in calculating the EPP gains for PSOC . Given these technology and algorithm characteristics, the applications of interest are partitioned, optimized and implemented as PSOC designs.

4.2 The Suite of Applications

In this section, we (due to space constraints) briefly describe the suite of applications, their partitioning and optimization.

Bayesian Inference (BN) Bayesian inference is a statistical inference technique which models the human decision making process. Hypotheses and their corresponding probability weights are notions central to this technique. The probability weights are interpreted as the *degrees of belief* in their corresponding hypotheses. Based on *evidences* the degree of belief in an hypothesis is incremented (decremented) till it approaches 1 (or 0) in which case the hypothesis is very likely (unlikely). A Bayesian network is used to perform Bayesian inference and is a directed acyclic graph of nodes V representing variables and edges E representing dependence relations between the variables. The variable represented by a node u can take a value from a finite set of values Σ_u. Each value σ in the set Σ_u has a conditional probability $p(\sigma/\Sigma')$ associated with it, where $\Sigma' \in \Sigma \times \Sigma \times \Sigma \cdots$ is the string of values of the variables represented by the parents of u. Variables whose values are known apriori are called *evidences* and based on evidence, other variables are *inferred*. The particular bayesian networks considered in this study implements hospital patient management and windows printer troubleshooting.

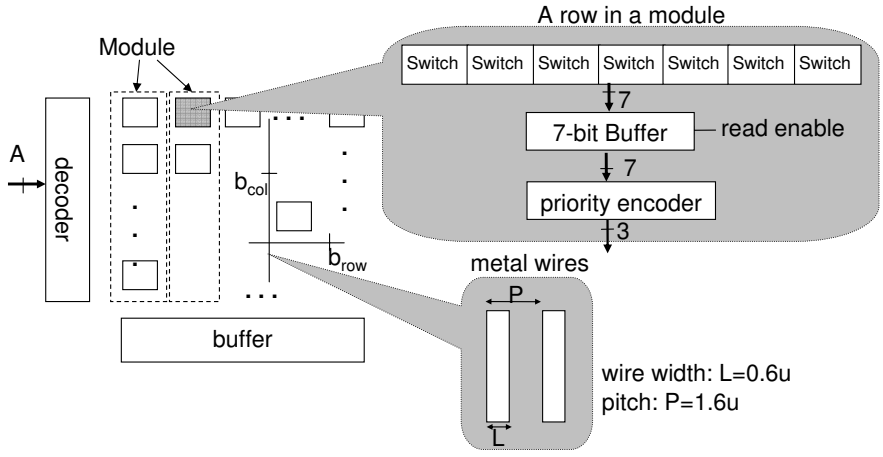

Figure 3. The co-processor architecture for a PSoC implementing Bayesian Inference

Partitioning and Optimization In the likelihood weighting algorithm [18] for bayesian inference, the random experiment (used for inference) is implemented in the PCMOS co-processor (consisting of several *modules*), with the rest of the algorithm implemented as software executing on the host. For a Bayesian network G, the conditional probabilities associated with each value of the variable of a node are known apriori and are used to design a module of PCMOS switches (inverters) for each node in the graph. As an example, consider a node u with $\Sigma_u = \{0,1,2\}$. Let Σ' be the string of values associated with the parents of u. Let $0 \leq p(0/\Sigma'), p(1/\Sigma'), p(2/\Sigma') \leq 1$ be the conditional probabilities associated with $0,1,2 \in \Sigma_u$ respectively. Inference can be performed by designing three PCMOS switches A,B,C corresponding to $0,1,2$ respectively. The inputs are fixed at 0 and the probability of correctness of A,B,C is specialized to $p(0/\Sigma')$, $\frac{p(1/\Sigma')}{1-p(0/\Sigma')}$, $\frac{p(2/\Sigma')}{1-p(0/\Sigma')-p(1/\Sigma')}$ respectively. When the switches are inspected in the order A,B,C the value which corresponds to the first switch whose output is a 1, is the inferred value. Henceforth, the set of switches A,B,C will be referred to as a *row* and each switch referred to as an *element*. Since a row is required for each Σ', many rows are required to implement the random experiments which correspond to each of the possible values of Σ'. These set of rows will be referred to as a *table*.

As shown in Figure 3, the PCMOS module corresponding to a node u implements a table whose row is indexed by the string of values of the parents of u. The number of columns in the table is the cardinality of Σ_u, where each column corresponds to a value from the set Σ_u. An element in the table, identified by a (row,column) pair corresponds to a value $\sigma \in \Sigma_u$ and is implemented by a specialized PCMOS switch whose probability of correctness is computed as explained above. Finally a priority encoder connected to the outputs of a row determines the result of the random experiment.

Random Neural Network [8] (RNN) A random neural network consists of *neurons* and *connections* between the neurons. Information travels between the neurons in the form of bipolar signal trains. Neurons have potentials associated with them which are the sums of incoming signals. The potential in turn, influences the rate of firing. The particular neural network considered in this study solves the vertex-cover [8] of an input graph.

Partitioning and Optimization The Poisson process which models neuron firing is implemented in the PCMOS co-processor with the rest of the computation implemented as software in the host processor. To approximate the Poisson process modeling the firing of a neuron, the Bernoulli approximation of a Poisson process [5] is used. Since the rate at which random bits are requested by the host exceeds the PCMOS operating frequency, PCMOS switches in the co-processor are replicated to match the required rate. Application level optimization is performed to reduce the replication factor $\mathcal{R}$, by interleaving demand for random bits from PCMOS co-processor and the processing of these bits—collecting and distributing these firings. This has the effect of reducing the *peak* application demand bandwidth.

Probabilistic Cellular Automata [7] (PCA) are a class of cellular automata used to model stochastic processes. Cellular automata consist of *cells* with local (typically nearest neighbor) communication. Each cell is associated with a *state* and a simple *transition rule* which specifies the next state of a state transition based on its current state and the states of its neighbors. In the *probabilistic* string classification algorithm [7], the state of each cell is either 0 or 1, giving rise to 8 possible transition rules (each rule has two possible outcomes, 0 or 1). In addition, the transition rules are probabilistic: for a transition rule τ_i ($0 \leq i \leq 7$) the probability that the output state of the rule is 0 is denoted by $p_{i,0}$ and the probability that the output state is 1 is denoted by $p_{i,1}$.

Partitioning and Optimization Each transition rule is implemented by a PCMOS switch whose input is a 0. The probability of correctness associated with the i^{th} switch is $p_{i,1}$. The control-intensive part of choosing transition rule (based on the state of a cell and the states of its neighbors) and updating the states are implemented on the host processor. Since the rate at which the transition rules are evaluated exceeds the frequency of operation of the PCMOS devices, this structure is replicated many times.

Hyper-Encryption (HE) is a provably secure encryption technique [4] in the bounded storage model. This scheme consists of generating an *encryption pad* based on a publicly available random string α and a shared (between the sender and the receiver) secret key. The secret key S is a sequence of whole numbers $S = s_1, s_2, s_3 \ldots s_k$ such that each number $0 \leq s_i < |\alpha|$. If $\alpha[j]$ is the j^{th} bit of alpha, the encryption pad is generated by $\oplus\alpha[s_i]$ where $1 \leq i \leq k$. Message encryption is performed by a bitwise XOR operation of the encryption pad with the message.

Partitioning and Optimization In the host plus co-processor architecture, the random string is generated by PCMOS while the encryption pad generation and the encryption are performed by the host. In the full custom implementation as shown in Figure 4, the entire algorithm is implemented in custom hardware, through k instances of $|\alpha|$ to 1 multiplexers whose select inputs are from the elements of the secret key.

5. METRICS, RESULTS AND ANALYSIS

In order to characterize and quantify PCMOS benefits at the application level, we now define a variety of metrics. Subsequently we will summarize the application level benefits using these metrics.

5.1 Metrics for Quantifying the Application Level Benefits

Energy Performance Product: EPP described earlier, is defined as the product of the application level energy and the execution time, and will be used as the chief metric of interest to evaluate various implementations. Given the EPP of two alternate realizations, they can be compared by computing the ratio of their individual EPP values.

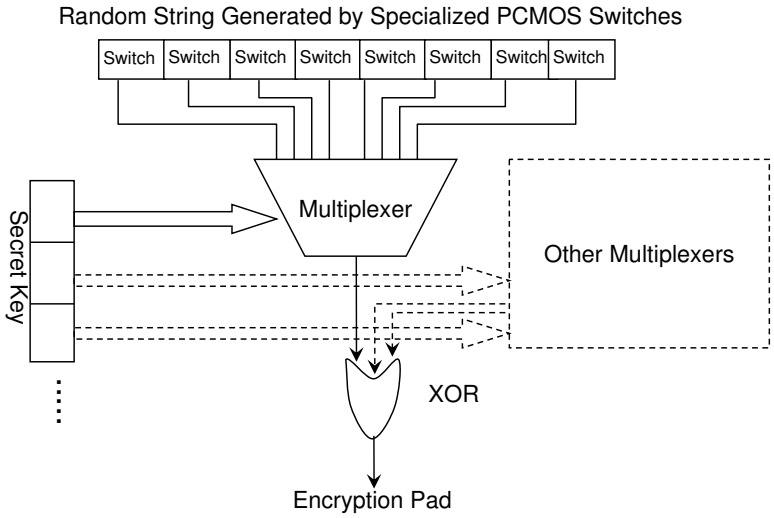

Figure 4. The Custom ASIC host and PCMOS co-processor architecture for a PSoC implementing Hyper-Encryption

Energy Performance Product Gain: Γ is the ratio of the EPP of the baseline to the EPP of a particular implementation I (e.g., a PSOC or an SOC) using a technology T (e.g., PCMOS or CMOS). This ratio is calculated as follows:

$$\Gamma_T = \frac{Energy_{Baseline} \times Time_{Baseline}}{Energy_I \times Time_I} \tag{1}$$

For calculating Γ, *in the sequel, the baseline always corresponds to the case when the entire computation is done on the host processor and therefore, there is no co-processor*. For example, in the context of the RNN application, the baseline is the StrongARM SA-1100 computing the deterministic as well as the probabilistic content and I is the combination of the StrongARM SA-1100 as the host computing the deterministic content and the co-processor computing the probabilistic content implemented with technology T. Now, to compare pairs of alternate choices for realizing the co-processor, for example, CMOS and PCMOS, we introduce the notion of a "relative energy performance product" as follows.

Relative Energy Performance Product Gain with Technology Parameters T, T': REPP of an application is defined as the ratio of its EPP gain when it is implemented on a on a PSOC with a co-processor realized using technology T' to the EPP gain when it is implemented on a (functionally identical, conventional) SOC with a co-processor realized using technology T. The host processor is invariant in both cases. Given the EPP gains Γ_T and $\Gamma_{T'}$ respectively, $\text{REPP}_{T,T'}$ is calculated as follows.

$$\text{REPP}_{T,T'} = \frac{\Gamma_{T'}}{\Gamma_T} = \frac{\text{EPP}_T}{\text{EPP}_{T'}} \tag{2}$$

Quality of Probabilistic Implementation: is defined empirically based on the statistical tests from the NIST suite [19] and is detailed in Section 6.

5.2 Application Level Gains of PCMOS

We will now describe the gain Γ_T where the technology T is PCMOS. Since the applications of interest are probabilistic, these gains in the scope of an entire application vary with varying inputs (and input sizes). As illustrated in Figure 5, the EPP gains are attributed to gains in energy as well as performance. Due to space constraints, the gains are summarized in Table 1. As shown in the table, these gains at the scope of an entire application range from a factor of (about) one order of magnitude for the HE application, to a factor of about 300 in the context of the RNN application kernel.

A range of EPP gains are observed whenever multiple data points are available, for example, in the context of the

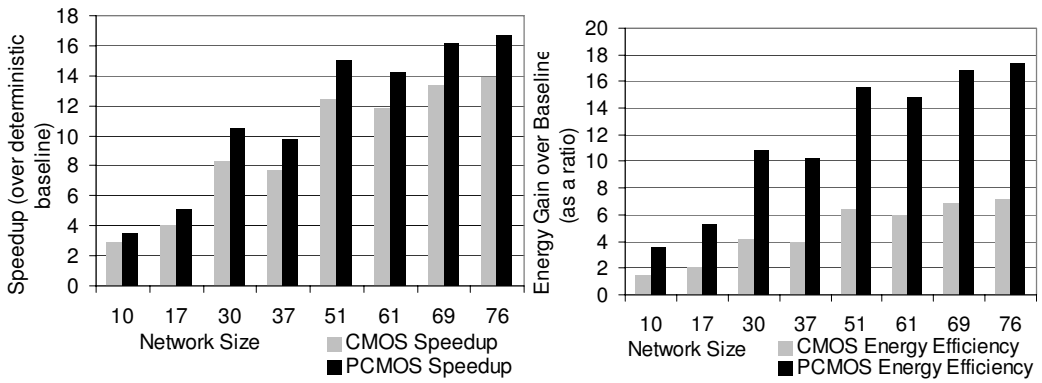

Figure 5. Energy and performance gains of PCMOS based SOC over the baseline when compared to CMOS based SOC

Algorithm	Flux (as percentage of total operations)	EPP Gain =Γ	
		Min	Max
BN	0.25%-0.75%	12.5	291
RNN	16.4%-19.7%	226.5	300
PCA	4.19%-5.29%	61	82
HE	12.5%	1.12	1.12

Table 1. Application level max and min EPP gains of PCMOS over the baseline implementation with increasing flux: The baseline for HE is an ASIC implementation of the host processor, with the StrongARM SA-1100 serving as the baseline in the remaining three cases.

Application	gain over SA-1100	gain over CMOS
BN	9.99×10^7	2.71×10^6
RNN	1.25×10^6	2.32×10^4
PCA	4.17×10^4	7.7×10^2
HE	1.56×10^5	2.03×10^3

Table 2. The EPP gain of PCMOS over SA-1100 and over CMOS for the core probabilistic step for different applications.

bayesian network where different data points correspond to different networks, the flux varies from 0.25 % to 0.75 %, the corresponding gain increases from a factor of 12.5 to an impressive factor of 291 due to increase in flux. Similar increases are observed for the other applications as well, caused by an increase in the flux values as shown in the table.

5.3 Analysis of Application Level Gains

The application level gains in energy and performance (when compared to the baseline case where there is no co-processor) is attributed to the efficiency of the co-processor while executing the probabilistic operations. We summarize these gains of PCMOS over StrongARM SA-1100 and over CMOS for the *core probabilistic step* for each of the applications in Table 2. Each row of this table corresponds to one of the four distinct applications of interest to us and presents the gains achieved per core probabilistic step. As can be readily seen from Table 2, these gains are substantially more for PCMOS—orders of magnitude greater—than those for CMOS. These per-operation gains would of course be valuable at the level of an entire application, only if the application embodies significant opportunity, characterized by its flux $\mathcal{F}$.

A more interesting case is the comparison of two alternate choices for realizing the co-processor—specifically, PCMOS and CMOS—using the concept of REPP. This comparison exposes another interesting characteristic influencing design

efficiency for PSOC based designs. Specifically we wish to delineate the (less obvious) impact of the efficiency of the host processor. In the interests of staying within the mandated space limits, we will restrict our analysis to the HE application: Starting with Figure 1, wherein the host processor is a StrongARM, we note that the host is a COTS processor. In this rather inefficient regime, $\text{REPP}_{T,T'}$ where $T \equiv$ CMOS and $T' \equiv$ PCMOS is nearly 1. That is, the PCMOS and the conventional CMOS based SOC designs achieve nearly the same performance in terms of energy and running time. By contrast and moving away from StrongARM to a host processor realized from custom ASIC logic (illustrated in Figure 4), the corresponding proportion of energy and running time spent in the host is considerably lesser. This enhances the impact of PCMOS efficiency at the application level—the corresponding REPP value to improve significantly to a factor of 9.38 with a custom ASIC host; *signifying that* PCMOS *based* PSOC *is many orders of magnitude better (in the* EPP *sense) not only over a baseline non-co-processor design, but even over conventional* CMOS *based* SOC. We note that these trends are extremely favorable for PSOC based designs as host processors become more efficient with future technology generations, thereby increasing the gains of PSOCs over conventional SOCs. The Table 3 below summarizes the REPP for a StrongARM based host and custom ASIC host for HE and PCA applications where the technologies of interest are PCMOS based PSOC design and conventional CMOS based SOC design.

Algorithm	REPP with SA host	REPP with ASIC Host
Hyper-Encryption	1.00	9.48
Probabilistic Cellular Automata	1.09	561

Table 3. REPP gains for the HE and PCA applications.

6. THE VALUE OF PCMOS TO QUALITY OF RANDOMNESS

While EPP gains for applications have been our foremost metric of concern to demonstrate the utility of PCMOS the *quality* of the implementation of a probabilistic algorithm is a characteristic of interest as well. Generation of random sequences of "high quality" and their impact on the application level quality metrics is an extensive area of study. Poor quality random bits impact application behavior—from the correctness of Monte Carlo simulations [6] to the strength of encryption schemes like Hyper-Encryption, which can be severely compromised if sequence of random bits can be "guessed". To assess to quality of randomization afforded by PCMOS, we employ statistical tests from the NIST suite [1]. The random sequences in the case of PCMOS have been produced from the actual chip measurements of a PCMOS inverter from $0.25\mu m$ TSMC prototype and those of CMOS from HSpice simulations of the hardware implementation of the Park-Miller [17] random number generator; for both cases, p is considered to be 0.5.

Figure 6, presents the results. Among these tests and to highlight a few, the *runs* test determine contiguous sequence of 1s in a block. The *rank* test checks the linear dependence, while the *FFT* and *approximate entropy* tests detect periodicity and frequency of overlapping patterns. The *template matching* tests detect repetitions of non-periodic patterns. In evaluating the test results, we use the same testing strategy and criteria as recommended by the NIST suite. Specifically, the test results shown in parenthesis in the table are compared against a threshold (which is 0.93) used to determine whether the sequence passes or fails a test. The tests are run on random bit sequences of length $20,000,000$. The result indicates the proportion of subsequences (tested through iterations) that pass from the random sequence being tested. As seen from the figure, the quality of random sequences generated by PCMOS is higher than that of those generated by CMOS.

7. CONCLUSION AND REMARKS

We have demonstrated the value of the novel PCMOS technology within the context of embedded applications, which, by their nature often admit probabilistic behaviors. In embedded applications, probability plays a role both in the context

[1]For an explanation and use of these tests, see [19]

Test	PCMOS	CMOS
Frequency	PASS (0.98)	FAIL (0.84)
Block-frequency	PASS (1.00)	PASS (0.98)
Cumulative sum	PASS (0.98)	FAIL (0.86)
Runs	PASS (0.98)	PASS (0.96)
FFT	PASS (1.00)	PASS (1.00)
Approximate entropy	PASS (0.98)	FAIL (0.92)
Long-run	PASS (1.00)	PASS (1.00)
Rank	PASS (1.00)	FAIL (0.00)
Non-overlapping template	PASS (0.93)	PASS (0.9375)
Overlapping template	FAIL (0.8889)	FAIL (0.00)
Lempel-Ziv	FAIL (0.8125)	FAIL (0.0625)
Linear complexity	PASS (1.00)	PASS (1.00)
Universal Statistical	FAIL (0.725)	FAIL (0.8889)
Serial	PASS (1.00)	PASS (1.00)

(result > 0.93) → PASS

(result < 0.93) → FAIL

Figure 6. Comparison of quality of randomization for PRNG and PCMOS.

of the "quality of solution" as well as in context of the algorithm used to solve the problem. The improvements that we were able to demonstrate were *orders of magnitude* over application specific CMOS designs. We find this sufficiently promising and wish to extend this work along two directions. First, we wish to explore a larger suite of applications, significantly from the signal processing (DSP) domain wherein the probability of correctness p at the device level manifests itself naturally as the *signal-to-noise* ratio at the level of a computational kernel such as a filter. An independent and equally interesting direction, involves investigating the applicability of the ideas, methods and constructs presented here to the overarching question of realizing reliable computing from unreliable elements—such "probabilistic designs" are considered central to sustaining Moore's law in the nanometer regime of CMOS designs [1, 11].

REFERENCES

[1] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De. Parameter variations and impact on circuits and microarchitecture. *40th Design Automation Conference*, pages 338–342, 2003.

[2] S. Cheemalavagu, P. Korkmaz, and K. V. Palem. Ultra low-energy computing via probabilistic algorithms and devices: CMOS device primitives and the energy-probability relationship. In *Proc. of The 2004 International Conference on Solid State Devices and Materials*, pages 402 – 403, Tokyo, Japan, Sept. 2004.

[3] S. Cheemalavagu, P. Korkmaz, K. V. Palem, B. E. S. Akgul, and L. N. Chakrapani. A probabilistic CMOS switch and its realization by exploiting noise. *To appear in the Proceedings of the IFIP international conference on very large scale integration, 2005*.

[4] Y. Z. Ding and M. O. Rabin. Hyper-encryption and everlasting security. *Lecture Notes In Computer Science; Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science*, 2285:1–26, 2002.

[5] W. Feller. *An Introduction to Probability Theory and its Applications*. Wiley Eastern Limited, 1984.

[6] A. M. Ferrenberg, D. P. Landau, and Y. J. Wong. Monte carlo simulations: Hidden errors from "good" random number generators. *Phys. Rev. Let*, 69:3382–3384, 1992.

[7] H. Fuks. Non-deterministic density classification with diffusive probabilistic cellular automata. *Physical Review E, Statistical, Nonlinear, and Soft Matter Physics*, 66(066106), 2002.

[8] E. Gelenbe and F. Batty. Minimum graph covering with the random neural network model. In *Neural Networks: Advances and Applications*, volume 2, 1992.

[9] L. B. Kish. End of Moore's law: thermal (noise) death of integration in micro and nano electronics. *Physics Letters A*, 305:144–149, Dec. 2002.

[10] D. MacKay. Bayesian interpolation. *Neural Computation*, 4(3), 1992.

[11] Moore's Law. http://www.intel.com/technology/silicon/mooreslaw/.
[12] K. Natori and N. Sano. Scaling limit of digital circuits due to thermal noise. *Journal of Applied Physics*, 83:5019–5024, May 1998.
[13] K. V. Palem. Energy aware algorithm design via probabilistic computing: from algorithms and models to Moores law and novel (semiconductor) devices. In *Proc. Intl. Conf. on Compilers, Architecture and Synthesis for Embedded Systems*, pages 113–117, San Jose, California, 2003.
[14] K. V. Palem. Proof as experiment: Probabilistic algorithms from a thermodynamic perspective. In *Proc. Intl. Symposium on Verification (Theory and Practice)*,, Taormina, Sicily, June 2003.
[15] K. V. Palem. Energy aware computing through probabilistic switching: A study of limits. *IEEE Trans. Comput.*, to appear September 2005.
[16] K. V. Palem, L. N. Chakrapani, B. E. S. Akgul, and P. Korkmaz. Realizing ultra-low energy application specific soc architectures through novel probabilistic cmos (pcmos) technology. In *To appear at International Conference on Solid State Devices and Materials (SSDM)*, Sept. 2005.
[17] S. Park and K. W. Miller. Random number generators: good ones are hard to find. *Communications of the ACM*, 31, Oct. 1988.
[18] A. Pfeffer. *Probabilistic Reasoning for Complex Systems*. PhD thesis, Stanford Univeristy, 2000.
[19] Random Number Generation and Testing. http://csrc.nist.gov/rng/.
[20] N. Sano. Increasing importance of electronic thermal noise in sub-0.1mm Si-MOSFETs. *The IEICE Transactions on Electronics*, E83-C:1203–1211, Aug. 2000.
[21] K. L. Shepard and V. Narayanan. Conquering noise in deep-submicron digital ICs. *IEEE Design and Test of Computers*, 15:51–62, Jan. 1998.
[22] A. Sinha and A. P. Chandrakasan. Jouletrack a web based tool for software energy profiling. *Proceedings of the 38th conference on Design automation*, pages 220–225, 2001.
[23] Trimaran Consortium. Trimaran: An infrastructure for research in instruction-level parallelism.