Power Wall and Dark Silicon:

Multicore Solutions

Kuldip Singh Atwal
College of Engineering and Computer Science
University of Central Florida
Orlando, USA
Email:kuldip@eecs.ucf.edu

***Abstract*—In this paper, we present some of the challenges in multicore computing due to dark silicon and other factors, and various techniques to deal with it. Alternative emerging technologies are discussed with their scope and applicability. The transition of computing in last decades is discussed and its impacts are studied. Finally, an Error Resilient System Architecture (ERSA) is explained, which is a robust system architecture to target stochastic applications such as recognition, mining, and synthesis (RMS). Using the concept of configurable reliability, ERSA may also be adapted for general-purpose applications that are less resilient to errors, although it may incur higher cost.**

Keywords—Dark silicon; power wall; reliability; error resilience; redundancy.

I. Introduction

The Dennard scaling trend states that as transistors get smaller, their power density stays constant, so that the power use stays in proportion with area. But, transistor scaling and voltage scaling are not in line with each other that lead to sharp increase in power density (failure of Dennard scaling) that hamper powering-on all the transistors simultaneously at the nominal voltage, while keeping the chip temperature in the safe operating range [1]. “Dark Silicon” refers to the amount of silicon that cannot be powered-on at the nominal operating voltage for a given Thermal Design Power (TDP) constraint [2]. The transistor area is still scaling as per Moore’s law, but the voltage and capacitance scaling have slowed that results in power limited designs. Following are reasons for end of Dennard scaling:

- At small sizes, current leakage poses greater challenges.
- Also cause the chip to overheat, which creates a threat of thermal runaway. Therefore, further increases energy costs.

Fig. 1 shows estimated dark silicon levels, as per ITRS [27] and Borkar [38].

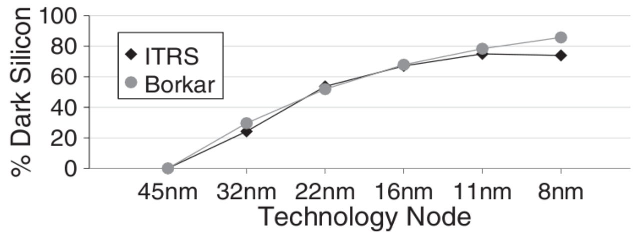
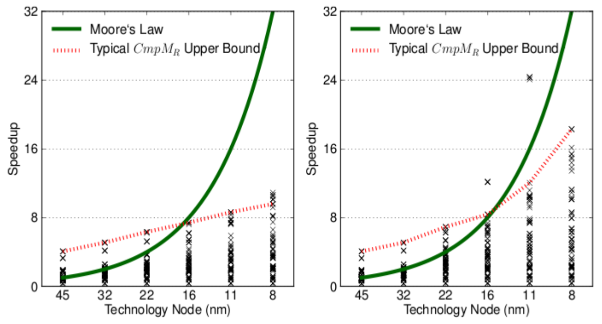


Fig. 1. Estimated dark silicon levels

Moore’s law predicts exponential growth, and clearly exponential growth can’t continue forever. It has become harder and harder to exploit higher clock speeds due to several physical issues, notably: heat, power consumption, and current leakage problem. The problem at 90nm was that transistor gates became too thin to prevent current from leaking out into the substrate [3]. Intel and many other semiconductor manufacturers have pushed many innovations such as strained silicon, hi-k metal gate, FinFET, and FD-SOI, but none of these has re-enabled anything like the scaling we once enjoyed. From 2007 to 2011, maximum CPU clock speed rose from 2.93 GHz to 3.9 GHz, an increase of 33%. From 1994 to 1998, CPU clock speeds rose by 300%. The breakdown of Dennard scaling prompted a switch among some chip manufactures to a greater focus on multicore processors, but the gains offered by switching to more cores are lower than the gains that would be achieved had Dennard scaling continued. A multicore processor is a single computing component with two or more independent actual processing units, CPUs, (called “cores”), which are the units that read and execute program instructions. The instructions are ordinary CPU instructions such as add, move, branch, but the multiple cores can run multiple instructions at the same time increasing overall speed for programs amenable to parallel computing. Utilization of parallelism at several levels in high performance architecture for AI is addressed in [37], along with a distributed performance measurement strategy that emphasizes flexibility while maintaining low cost and reduced complexity. But, there is the fact that adding more CPU cores never results in perfect scaling [4]. Performance is ultimately limited by the amount of serial code (code that can only be executed on one processor), which is known as Amdahl’s law. Parallel execution is limited by following factors:

- Communication cost
- Synchronization cost
- Not all problems are amenable to parallelization (inherently serial problems)
- Hard to think in parallel
- Hard to debug

As depicted in Fig. 2, due to power and parallelism limitations, a significant gap exists between what is achievable and what is expected by Moore's law. The more cores per die, the lower the chip's overall clock speed due to chip's power consumption badly limits its clock speed. The difficulty of software optimization is a further reason why adding more CPU cores does not help much. Multicore let us continue to deliver exponentially increasing compute throughput in mainstream computers, but in a form that was less easily exploitable, because it placed a greater burden on software developers who had to write parallel programs that could use the hardware [5], [6]. It is concluded that: multicore scaling is not likely to continue the historical trends; do not overcome the transistor scaling trends; and the performance gap is significantly large. Therefore, radical departures from conventional approaches are necessary to extract more performance and efficiency from silicon while preserving programmability, and explore other sources of computing [7], [8].



(a) Conservative Scaling (b) ITRS Scaling

Fig. 2. Speedup projections

In this paper, we explore various alternative technologies and discuss ERSA, an error resilient system that targets probabilistic applications. In Section II we discuss the ERSA related research work. Section III is dedicated to alternative technologies. Section IV examines the design issues due to dark silicon. Section V explains the Energysmart technique. Section VI provides detailed description of ERSA. Section VII examines the ERSA design model. Section VIII presents ERSA results. Section IX proposes the prospects of future work followed by conclusion in section X.

II. Related Work

Imprecise computation [9] was used in mission-critical real-time systems to satisfy deadlines. In software engineering, acceptability-oriented computing [10] aims to reduce software development costs by ensuring that errors manifest only within acceptable ranges. Best-effort computing provides a framework to utilize the error-forgiving nature of recognition and mining applications to improve the overall execution time [11]. Breuer suggested ways of improving chip yield by using imperfect chips in applications where degradation of output quality may be acceptable (e.g., multimedia, compression) [12]. Eltawil and Kurdahi also utilized cognitive error resilience and defect maps to implement fault-tolerant embedded memories for system-on-chips (SoCs) targeting wireless applications [13]. Wong and Li [14], [15] analyzed the error resilience of probabilistic applications in the presence of single-event upsets.

Many existing techniques utilize specific characteristics of a given (generally small) set of target applications to improve error resilience. In algorithmic noise-tolerance (ANT) [16], the supply voltage of a DSP system is scaled beyond the critical voltage imposed by critical path delays. Errors were compensated by using algorithmic noise-tolerance techniques in DSP algorithms. Stochastic sensor network on a chip [17] extends ANT by using distributed computational units (or sensors) whose outputs are combined using robust statistical signal processing techniques. Verma et al. suggested a data-driven approach to classification that uses statistical information to extract relevant features using erroneous hardware [18], [19]. Chakrapani et al. suggested an SoC architecture that uses application-specific, probabilistic hardware components for probabilistic applications [20]. However, only a small portion of the application can be mapped to probabilistic hardware components using their approach. EnerJ [21] is a framework to isolate the storage and the computation of error-sensitive data to provide voltage reduction for approximate data. Circuit design techniques to create path slack distribution that are aware of voltage over-scaling are described in [22]. Finally, a large body of literature exists on application dependent error checking (e.g., [23], [24]), which involves encoding of data with algorithm-based techniques prior to the actual computation stage. Table I presents the comparison of ERSA with related work.

III. Alternative Technologies

A. *Heterogeneity*

The end of conventional scaling has sparked a sharp increase in the number of companies researching various types of specialized CPU cores. Prior to that point, general-purpose CPU architectures had eaten through the high-end domains of add-in boards and co-processors at a significant rate. Once that trend slammed into the brick wall of physics, more specialist architectures began to appear. But it is not the solution, since incorporating a specialized many-core processors on-die does not address the underlying problem that transistors can no longer be counted on to scale the way they used to. The fact that transistor scaling density continues to scale while power consumption and clock speed do not, has given rise to "Dark Silicon". It refers to the percentage of silicon on a processor that can't be powered up simultaneously. Despite what some companies like to claim, specialized many-core chips don't "break" Moore's law in any way and are not exempt from the realities of semiconductor manufacturing. What they offer is a tradeoff—a less general, more specialized architecture that's capable of superior performance on a narrower range of problems. They are also less encumbered by socket power constraints. A single computer already typically includes more than one kind of processor cores, as mainstream notebooks, consoles, tables, and smartphones all increasingly have both CPUs and compute-capable GPUs. The open question is not whether a single application will be spread across different kinds of cores, but only "how different" the cores should be—whether they should be basically the same with similar

TABLE I. ERSA COMPARISON WITH RELATED WORK

	Error Rates[5]	High-Order Bit Errors/Control Flow Errors	Nontransient Memory Errors	H/W Prototype	Programmability
Imprecise computation [9]	N/A	No	No	N/A	Yes
Acceptability-oriented computing [10]	N/A	No	No	N/A	Yes
Best-effort computing [11]	N/A	No	No	Yes	No
Imprecise computation for multimedia applications [12]	Single stuck-at-fault	No	No	Yes	No
Increased chip yield through error tolerance[6] [13]	10^{-3} on-chip memory bit-error-rate	No	No	Yes	No
Soft error resilience of probabilistic applications [15]	Singh error injection to a register output	Yes	No	No	Yes
Algorithmic noise-tolerance [16]	10^{-3} (bit-flip to the filter output)	Yes	No	Yes	No
Stochastic sensor network on a chip [17]	2×10^{-2} errors/adder operation	Yes	No	Yes	No
Probabilistic SoC [20]	$0 < p < 1/2$ (each probabilistic gate output)	No control flow errors	No	No	No
Data-driven EEG monitoring [18], [19]	5×10^{-1} (SRAM cell bit-flips)/7×10^{-2} (logic gate stuck-at-faults)	Yes	No	Yes	No
EnerJ [21]	10^{-3} errors/access (SRAM)	No control flow errors	No	No	Yes
Algorithm-based fault tolerance [23]	Single processor error	No control flow errors	No	No	No
ERSA	20 errors/flip-flop/s	Yes (high error rates)	Yes	Yes	Yes

instruction sets but in a mix of a few big cores that are best at sequential code plus many smaller cores, or cores with different capabilities. Heterogeneity amplifies the multicore trend, because if some of the cores are smaller then we can fit more of them on the same chip [25]. The transition to heterogeneous cores is permanent, because different kinds of computations naturally run faster and/or use less power on different kinds of cores – including that different parts of the same application will run faster and/or cooler on a machine with several different kinds of cores. Building heterogeneous chip multiprocessors with different materials is more preferable than conventional designs since it can efficiently utilize the chip level resources and deliver the optimal balance among performance, energy consumption and cost [26].

B. *More-than-Moore (MtM) scaling*

The ITRS (International Technology Roadmap for Semiconductors) is focused on what's referred to as "More-than-Moore" scaling [27]. The goal of MtM scaling is to extend the same design principles that have driven digital device scaling for decades over to analog circuitry, and to integrate those technologies on-die within the SoC/SiP, as shown in Fig. 3. The goal of MtM scaling is to increase system level power efficiency and capabilities, provide a coherent, regular roadmap for the relevant technologies, and increase device complexity. The original proposal that laid out the MtM concept states "it is the heterogeneous integration of digital and non-digital functionalities into compact systems that will be the key driver for a wide variety of application fields, such as communication, automation, environmental control, healthcare, security and entertainment." The use of medical devices capable of interfacing with a mobile phone is an example of exactly the sort of integration MtM is meant to address [31]. Instead of focusing strictly on the CPU as the enabler of experiences, MtM emphasizes integration and efficiently designing every component. As the MtM paper states, "Whereas More Moore may be viewed as the brain of an intelligent compact system, ‘More-than-Moore’ refers to its capabilities to interact with the outside world and the users.”

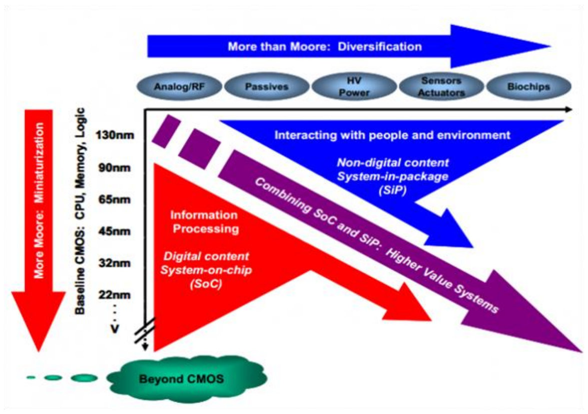

More than Moore: Diversification
Analog/RF
Passives
HV Power
Sensors Actuators
Biochips
More Moore: Miniaturization
Baseline CMOS: CPU, Memory, Logic
130nm
90nm
65nm
45nm
32nm
22nm
V
Interacting with people and environment
Non-digital content System-in-package (SiP)
Information Processing
Digital content System-on-chip (SoC)
Combining SoC and SiP: Higher Value Systems
Beyond CMOS

Fig. 3. MtM scaling

C. *Memristor-based Technologies*

The concept and theory of memristor (short for memory resistor) has been formulated almost four decades ago, but the first practical realization was announced by HP Labs in 2008. It is a circuit element with “memory”: the resistance is a function of past current, in other words, a memristor can remember a voltage applied to it even after the current is cut off. This property makes it a prime candidate for building more compact and faster storage devices, while computer designs based on memristors replacing transistors have also been proposed. If it will materialize, it will represent a unified design base with significant possibilities: 3D designs based on memristors with up to 1,000 layers have been designed [28]. Memristor based commercial processors and memory systems are still some time off, but currently this is one of the most promising basic technologies in the quest for performance improvements. The dark silicon problem can be addressed by adopting memristors as building blocks (instead of transistors) and chip designs where, at any given time, just a subset of components would be active, depending on the type of application.

D. *Approximate computing*

Researchers are developing computers capable of “approximate computing” to perform calculations good enough for certain tasks that do not require perfect accuracy, potentially doubling efficiency and reducing energy consumption. The need for approximate computing is driven by two factors: a fundamental shift in the nature of computing workloads, and the need for new sources of efficiency. Mobile and embedded devices need to process richer media, and getting smarter, being more context-aware and having more natural user interfaces. A growing number of applications are designed to tolerate “noisy” real-world inputs and use statistical or probabilistic types of computations. The nature of these computations is different from the traditional computations where we need a precise answer. Approximate computing could endow computers with a capability similar to the human brain’s ability to scale the degree of accuracy needed for a given task, where we are trying to provide results that are of acceptable quality, but not trying to be perfect. The inability to perform to the required level of accuracy is inherently inefficient and saps energy. Researchers have shown how to apply approximate computing to programmable processors, which are ubiquitous in computers, servers and consumer electronics, because, in order to have a broad impact, we need to be able to apply this technology to programmable processors to perform approximate computing [29]. The approximate computing leverages the intrinsic resilience of applications to inexactness in their computations to achieve a desirable trade-off between efficiency (performance or energy) and acceptable quality of results. Following are classes of approximate applications:

- Programs with analog inputs; e.g., Sensors, scene reconstruction
- Programs with analog outputs; e.g., Multimedia
- Programs with multiple possible answers; e.g., Web search, machine learning
- Convergent programs; e.g., Gradient descent, big data analytics

E. *Perpetual computing*

A key goal of mobile computing is untethering devices from wires, making them truly portable. As computing moves from a handful of mobile devices per user to hundred of specialized devices, fully wireless operation will become essential. While wireless networking has eliminated the need for wires for network communication, nearly all devices depend on frequent wired connections to electricity in order to recharge their batteries. The continual need for human involvement in power management significantly limits the feasibility of having users manage large numbers of mobile devices. The assumption that a person will recharge batteries has also led to a narrow class of solutions to mobile energy management. Whether the system is implemented at the hardware, operating system, or application layer, the goal has been the same: maximize battery lifetime or target a particular lifetime, while minimizing the impact on user-perceived performance [35].

Environmental energy harvesting fundamentally changes these assumptions. Solar, wind, heat differential, and motion-derived energy sources provide the opportunity to build mobile systems that do not require regular human intervention to charge batteries, thus enabling large-scale deployments of mobile systems that are completely untethered.

As two examples of perpetual systems, consider the following: deploying remote outdoor access points and tracking wildlife. In the first case, remote deployments on volcanoes and in forests, solar energy offers the opportunity to cheaply deploy access points that require no maintenance. In case for tracking wildlife, systems have shown that using solar cells to power portable sensors makes perpetual wildlife tracking possible.

IV. DESIGNING FOR DARK SILICON

To gracefully embrace dark silicon, design methodologies must adapt themselves to identify progressive systems that can effectively exploit the growing dark silicon to improve power-performance and reliability [30]. The traditional design metrics may lead to suboptimal design choices with the rise of dark silicon. Fundamentally, design choices that appear superior in one technology generation may become inferior in subsequent technology generations. New metrics are needed to guide a dark silicon aware system design, and measure the dark silicon exploitation. A new Dark Silicon Utilization-Efficiency (DSU) metric for measuring dark silicon exploitation is proposed, and a Dark Silicon-Aware (DSA) design methodology for multicore systems based on this metric is presented [32]. Further, a stochastic optimization algorithm for dark silicon aware multicore systems is proposed for designing a DA multicore to optimize its DSU. 7-23% energy efficiency benefits can be achieved, as compared to conventional design techniques.

When a processing core becomes more specialized, it provides better energy efficiency for a decreasing range of target workloads. This idea is exploited with new DSU metric. The DSU is measured as the product of two sub-metrics: the core-level dark silicon utilization metric and the core-level efficiency metric. Each of these metrics scales the order to provide a trade-off between an individual core's level of specialization and the range of threads for which it is optimally designed.

A DSA multicore system is designed to exploit its underlying dark silicon. By choosing various combinations of active cores, a DSA system is able to operate in various configurations, each providing a high level of efficiency for different workloads. Dark silicon is best exploited by a DSA system that properly designs cores with appropriate ranges of optimality. The optimal configuration of the DSA cores depends on the expected workload characteristics and number of cores, as well as the spectrum of dark silicon that the multicore system is expected to encounter. For a general purpose microprocessor, several applications can combine in various phases creating a very large workload space, making the DSA design problem computationally hard.

To efficiently choose DSA cores, a formalized algorithm is needed. Therefore, the Hungarian Algorithm is used within the simulated annealing framework to activate cores and schedule threads while minimizing the energy efficiency of the system.

V. ENERGYSMART

Near-Threshold Computing: It is a way to improve energy efficiency by reducing the supply voltage (v_{dd}) to a value a bit higher than a transistor's threshold voltage (v_{th}), as opposed to the conventional Super-Threshold Voltage Computing (STC) [36]. One drawback is degradation in frequency, which may be tolerable through more parallelism in the application. Since many more cores can be executing concurrently within the chip's power envelope, the result is a higher throughput for parallel code. According to estimates, NTC can decrease the energy per operation by several times over STC.

Although Near-threshold Computing (NTC) is a promising approach to push back the multicore power wall, it suffers from a high sensitivity to parameter variations, namely the deviation of device parameters from their nominal specifications [33]. One way to deal with variations is to use multiple on-chip voltage (v_{dd}) and frequency (f) domains. With these domains, we can separately adapt to the parameter values in the different chip regions. But this approach if found to be energy inefficient along with substantially increase in complexity.

A multicore organization for NTC is proposed that has a single v_{dd} domain, and relies on multiple frequency domains to tackle variation. To be competitive, it has to be paired with effective core assignment strategies. It is shown that at NTC, a simple chip with a single v_{dd} domain can deliver a higher performance per watt than one with multiple v_{dd} domains. Core assignment algorithms are proposed for EnergySmart architecture that deliver high performance per watt and are simple to implement.

Energysmart keeps a single v_{dd} in the whole chip. DVFS can be used, but is applied globally across the chip. By removing v_{dd} domains, EnergySmart also gains in hardware simplicity and saves chip area. To handle process variations more inexpensively than with fine-grain v_{dd} and frequency (f) domains, EnergySmart uses only *f* domains. EnergySmart is organized in clusters of cores, where each cluster is potentially an *f* domain. A cluster is characterized by the maximum frequency (f_{max}) that it can support at the lowest possible chip-wide safe voltage. With many *f* domains, the chip still has many degrees of freedom to tackle process variation.

In EnergySmart, where multiple v_{dd} domains are given up and rely on fine-grain *f* domains, two challenges appear as follows:

- The need to carefully perform core-to-job assignment.
- The need to effectively support fine-grain DVFS.

EnergySmart targets a several-hundred core chip environment, typically running multiple applications. Its goal is to attain the best variation-aware schedule. Also, it is simple and scalable.

VI. ERSA

Error Resilient System Architecture (ERSA) is a robust system architecture that overcomes shortcomings of traditional, expensive redundancy techniques [34]. It is an energy-efficient system that is resilient to high error rates for emerging applications such as recognition, mining, and synthesis (RMS). It is particularly applicable to applications with inherently high degree of error resilience at low cost. Using the concept of configurable reliability, ERSA may also be adapted for general-purpose applications that are less resilient to errors, but it may incur higher costs. ERSA achieves high error resilience to high-order bit errors and control flow errors, in addition to low-order bit errors, by using a combination of following key ideas:

- Asymmetric reliability in many-core architectures, i.e., mixing processor cores of various "reliability levels" in many-core architectures.
- Error-resilient algorithms at the core of probabilistic applications.

- Light-weight checks such as timeouts and memory bounds violation checks. ERSA does not rely on expensive error detection techniques.
- Software optimization including minimally-intrusive yet effective modifications to the applications' core algorithms.

Error injection experiments on a multicore ERSA hardware prototype demonstrate that even at very high error rates of 20 errors/flip-flop/10^8 cycles (equivalent to 25000 errors/core/s), ERSA maintains 90% or better accuracy of output results, together with minimal impact on execution time, for probabilistic applications such as K-Means clustering, LDPC decoding, and Bayesian network inference.

ERSA leverages two emerging trends in future computing platforms to achieve high degrees of error resilience at low cost:

- Proliferation of multicore & many core systems
- New applications, e.g., data mining, market analysis, cognitive systems and computational biology, which are expected to drive demands for computational capacity. Such applications are also referred to as recognition, mining, synthesis (RMS) applications.

Characteristics of RMS applications include the following:

- Massive parallelism: Massive amounts of data are processed to build mathematical models and to apply these models to help answer real-world questions.
- Algorithm resilience: RMS applications are often tolerant to imprecision and approximation to make analysis of complex systems tractable. Many RMS use probabilistic algorithms, which use probability values or probability densities to compute or represent information. Such probabilistic algorithms can reduce the effects of inaccuracies over many iterations or by using a large number of samples.
- Cognitive resilience: Computation results need not always be correct as long as the accuracy of the computation is "acceptable" to human users.

At high error rates, the errors in high-order bits of data significantly affect accuracy and performance, which is not addressed in previous works. Probabilistic applications often exhibit a high degree of error resilience because approximate answers are allowed for such applications.

ERSA incorporates asymmetric reliability levels for on-chip memory components as well. ERSA interconnect consists of two separate buses, one reliable bus connecting SRC, reliable memory and I/O, and other unreliable bus connecting the SRC with all unreliable components. SRC can access both buses.

VII. ERSA DESIGN

ERSA avoids high overheads of traditional redundancy by combining a multicore hardware platform with an application computational model. This is achieved through the concept of *asymmetric reliability*. Only a small, selected portion of the underlying hardware is required to be highly reliable, while the majority of the system consists of less reliable, low-cost components. The motivation for asymmetric reliability comes from the computation model that decomposes an application into control-intensive and data-intensive operations.

A. ERSA Hardware Design with Asymmetric Reliability

ERSA hardware consists of a small number of highly reliable cores, referred to as super reliability cores (SRCs), together with a large number of cores that are less reliable but account for most of the computation capacity, referred to as relaxed reliability cores (RRCs). With asymmetric design, the need for highly reliable components is minimized. The notion of asymmetric reliability extends to uncore components as well, such as the memory system or interconnects. They can also consume significant on-chip resources. Fig. 4 shows the context diagram of ERSA hardware design.

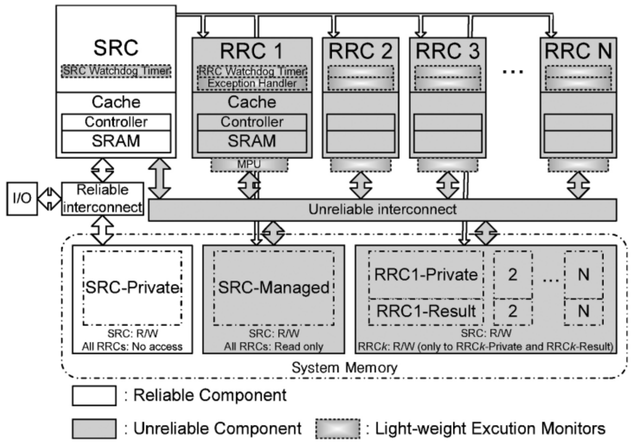


Fig. 4. ERSA hardware architecture.

The SRC is responsible for executing the non-error-resilient portions of the applications, such as computation task creation, computation result reduction, and convergence checks, i.e., control-intensive operations. The operating system kernel is also executed only by an SRC. The SRC assigns computation tasks to RRCs and provides error handling mechanisms in case an RRC exhibits unexpected behaviors. The SRC is also capable of restarting any RRC in the system by clearing all the internal states of that RRC. To achieve high reliability levels, the SRC requires conservative design choices, similar to traditional fault-tolerant designs. However, because the SRC constitutes only a relatively small portion of the system, the overall system-level power, performance, and area costs can still be low. Fig. 5 shows the ERSA taxonomy.

RRCs constitute the majority of the components in ERSA. Together, they provide inexpensive and massive computation capacity. This is possible because ERSA does not require RRCs to be highly reliable. Various tradeoffs exist between achieving high levels of hardware reliability versus the associated cost, such as performance or energy overhead. Adjusting supply voltages or clock speed, configuring built-in error protection mechanisms, or using circuit designs with limited computation accuracy are some examples of adjustable "knobs" for exploring the associated tradeoffs. These knobs can allow the reliability levels of individual hardware components to be flexibly adjusted. Such a "configurable reliability" approach enables ERSA to execute applications that may not be inherently resilient to errors unlike probabilistic applications.

Memory components, such as cache memory or scratch pad memory, account for a significant portion of on-chip resources. ERSA incorporates asymmetric reliability levels for on-chip memory components as well. The ERSA interconnect consists of two separate buses. One is a reliable bus dedicated to connecting the SRC, reliable memory components, and I/O peripherals. The other one is an unreliable bus connecting the SRC with all unreliable components. The SRC can access both buses, but the RRCs are connected to the unreliable bus only.

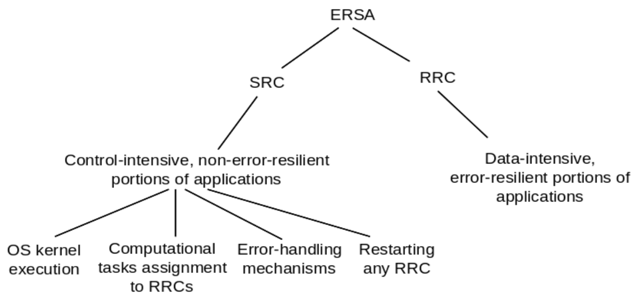

ERSA
SRC
RRC
Control-intensive, non-error-resilient portions of applications
Data-intensive, error-resilient portions of applications
OS kernel execution
Computational tasks assignment to RRCs
Error-handling mechanisms
Restarting any RRC

Fig. 5. ERSA taxonomy

B. Execution Model

The ERSA software layer, the ERSA *runtime*, manages task scheduling and error handling. Fig. 6 shows the computation flow of K-means clustering application. During each iteration, the application's main thread creates computation tasks and requests the ERSA runtime to execute the tasks. The task scheduler in the ERSA runtime distributes computation tasks to RRCs. Each RRC executes computation tasks assigned to it and stores the output results in the designated result buffer; the ERSA runtime then gathers these output results and delivers them to the application's main thread.

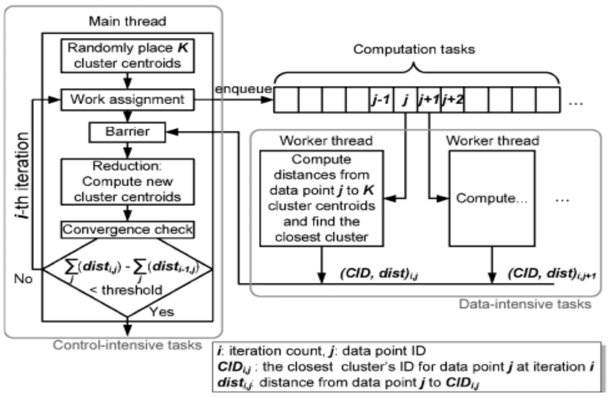

Main thread
Randomly place K cluster centroids
Work assignment
Barrier
Reduction: Compute new cluster centroids
Convergence check
$\sum_j(dist_{i,j}) - \sum_j(dist_{i-1,j})$ < threshold
No
Yes
i-th iteration
Control-intensive tasks
enqueue
Computation tasks
j-1
j
j+1
j+2
Worker thread
Compute distances from data point j to K cluster centroids and find the closest cluster
Worker thread
Compute...
$(CID, dist)_{i,j}$
$(CID, dist)_{i,j+1}$
Data-intensive tasks
i: iteration count, j: data point ID
$CID_{i,j}$: the closest cluster's ID for data point j at iteration i
$dist_{i,j}$: distance from data point j to $CID_{i,j}$

Fig. 6. Computation flow diagram of the K-means clustering application.

In addition to asymmetric reliability, ERSA uses "light-weight" error handling techniques, such as task re-scheduling, watchdog timers, and exception handling, to achieve error resilience.

C. Enhanced ERSA

The ERSA system described so far, i.e., incorporating asymmetric reliability and light-weight execution monitoring, is referred to as Basic ERSA. The basic ERSA can prevent crashes, but it can result in highly inaccurate output results and significant execution time overheads. To overcome these challenges, a set of application-specific software techniques are designed, which when used in conjunction with Basic ERSA result in a system referred to as Enhanced ERSA.

VIII. ERSA Results

The effectiveness of ERSA is demonstrated by comparing results obtained from a variety of system configurations (Table II). The area and power overheads of the ERSA design are compared to No-ERSA design. The area overhead of ERSA compared to No-ERSA is 0.7%, and the power overhead is 1.9% (at the same nominal voltage). This overhead is calculated for logic area only, excluding memory area and power. Several error injection experiments are conducted on ERSA prototype to show its effectiveness of high degrees of error resilience. Random bit flips are injected into the flip-flops inside the RRCs to emulate the effect of logic errors. Errors are injected at a full system speed of 100 MHz. Flip-flop error injection rates from zero to 20 errors per flip-flop, per 10^8 is varied. This error rate of 20 errors/flip-flop/10^8 cycles corresponds to 25000 errors/cores/s.

TABLE II. Overview of various system configurations

System	Description
No-ERSA	The baseline system without ERSA mechanisms
Asymmetric reliability only	Baseline system + asymmetric reliability
Basic ERSA	Baseline system + asymmetric reliability + light-weight execution monitors
Enhanced ERSA	Basic ERSA + application-specific enhancements (same H/W with the basic ERSA)

A. Completion rate

The completion rate (the percentage of runs that complete execution and produce results, regardless of output quality) of ERSA is compared with No-ERSA and Basic ERSA. For the K-means clustering application, the system either hangs or crashes even at low error injection rates without ERSA (Fig. 7). At around 3×10^{-3} errors/flip-flop/10^8 cycles—which corresponds to about ten error injections to the entire system per run—almost no run survived. Even for a small fraction of the runs that survived the error injection, the resulting output quality is highly unacceptable.

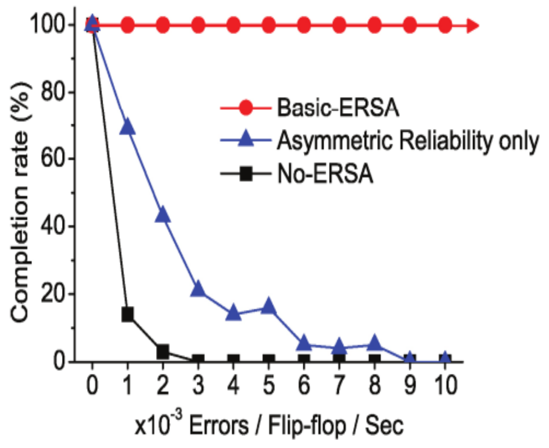


Fig. 7. Completion rate of the K-means clustering application.

Asymmetric reliability architecture overcomes some challenges. However, asymmetric reliability alone is not sufficient. With asymmetric reliability only, crashes could be prevented. However, many runs could not complete execution because of nonresponsive RRCs. Without light-weight execution monitoring mechanisms in Basic-ERSA, we cannot expect such a system to reliably deliver results.

B. *Output result quality*

Fig. 8 (a)-(c) shows the output results accuracies of ERSA applications. While, Basic ERSA successfully prevents crashes, computation accuracies degrade very significantly with increasing error injection rates because more and more RRCs produce results with large deviations. Enhanced ERSA continues to achieve better than 90% computation accuracies even at extremely high error injection rates of more than 20 errors/flip-flop/10^8 cycles. While Enhanced ERSA does not produce 100% correct results at very high error rates, the overall accuracy may be acceptable due to cognitive resilience.

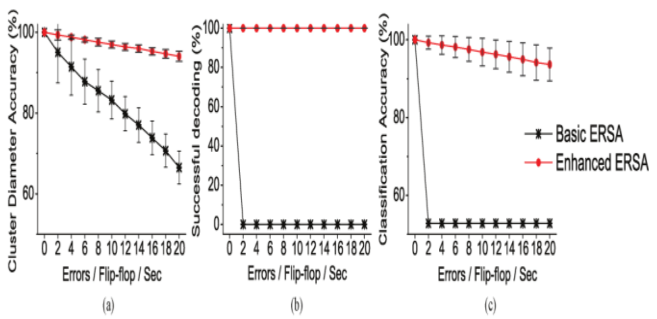


Fig. 8. ERSA computation accuracy results (mean and standard deviation calculated for 1000 runs per error rate point). (a) K-means clustering. (b) LDPC decoding. (c) Bayesian network inference.

C. *Computation time*

In Fig. 9 (a)-(c), ERSA execution time is compared for various error injection rates. An application's original execution time is measured on the baseline system (No-ERSA) without error injection. With Basic ERSA, execution time overheads can be very significant. This is primarily because of increased iteration counts resulting from RRC results with large deviations. Enhanced ERSA incurs execution time overheads even in the presence of no errors because of the additional algorithmic modification. However, as error injection rates increase, execution time overheads continue to be low with Enhanced ERSA, unlike Basic ERSA.

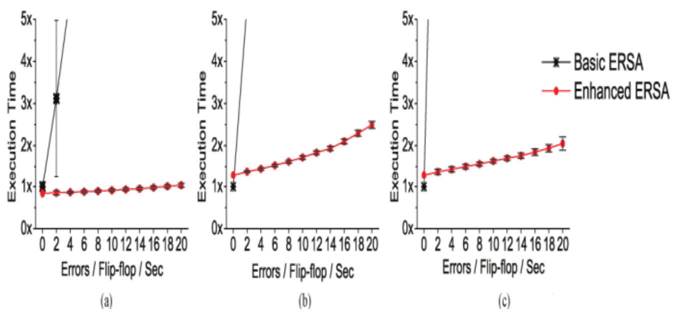


Fig. 9. ERSA execution time overheads (mean and standard deviation calculated for 1000 runs per error rate point). (a) K-means clustering. (b) LDPC decoding. (c) Bayesian network inference.

IX. Future Work

ERSA has many potential improvements and optimizations. Quantification of the ERSA benefits in actual hardware is needed, i.e., intrinsic evaluation should be done, using a variety of error models, such as voltage droops or delay fault models. Formalized techniques are to be followed for convergence damping and filtering, rather that trial-and-error based on error injection experiments. Synergies with traditional error detection (e.g., parity and residue checking, timing error detection), application-specific error detection, and algorithm-based fault-tolerance techniques should be done. Further, use of ERSA for a broader range of applications including speech recognition and multimedia applications and for a wide variety of hardware architectures is a field worth exploration. Finally, adoption of a scalable memory subsystem to expand the system beyond 8 RRCs, and automated design tools for ERSA should also be considered.

X. Conclusion

We study that the emergence of dark silicon, a fundamental design constraint absent in the past generations, caused a significant challenge in computing. It has lead to a transition from past technologies, and many promising alternatives are appearing. Further, we evaluated ERSA for designing energy-efficient systems that are resilient to high error rates.

References

[1] Shanbhag, N. R.; Mitra, S; de Veciana, G.; Orshansky, M.; Marculescu, R.; Roychowdhury, J.; Jones, D.; Rabaey, J. M., "The search for alternative computational paradigms," Design & Test of Computers,

IEEE, vol. 25, no. 4, pp.334,343, July-Aug. 2008, DOI=10.1109/MDT.2008.113

[2] Michael B. Taylor. 2012. “Is dark silicon useful?: harnessing the four horsemen of the coming dark silicon apocalypse”. In Proceedings of the 49th Annual Design Automation Conference (DAC ‘12). ACM, New York, NY, USA, 1131-1136. DOI=10.1145/2228360.2228567

[3] Hruska, Joel, February 1, 2012. “The death of CPU scaling: From one core to many – and why we’re still stuck”. URL http://www.extremetech.com/. Retrieved April 3, 2015.

[4] Hadi Esmaeilzadeh, Emily Blem, Renée St. Amant, Karthikeyan Sankaralingam, Doug Burger, “Dark Silicon and the end of Multicore Scaling”, in the Proceedings of the 38th International Symposium on Computer Architecture (ISCA ‘11). DOI=10.1145/2000064.2000108

[5] Manferdelli, J. L., Govingaraju, N. K., Crall, C, “Challenges and opportunities in Many-Core Computing”, in Prceedings of the IEEE, vol. 96, Issue 5, pp. 808-815, May 2008. DOI=10.1109/JPROC.2008.917730

[6] Sutter, H., 2005. “The free lunch is over: A fundamental turn toward concurrency in software”. URL http://www.gotw.ca/publications/concurrency-ddj.htm. Retrieved April 3, 2015.

[7] Shafique, M., Garg, S., Henkel, J., Marculescu, D., “The EDA challenges in the dark silicon era,” Design Automation Conference (DAC), 2014 51th ACM/EDAC/IEEE, vol., no., pp. 1-6, June 2014. DOI = 10.1145/2593069.2593229

[8] Villa, O., Johnson, D.R. et. al., “Scaling the Power Wall: A Path to Exascale”. In International Conference for Hight Performance Computing, Networking, Storage Analysis, SC14. IEEE, New Orleans, LA, USA. pp. 830-841, Nov 2014. DOI = 10.1109/SC.2014.73

[9] J. Liu, K.-J. Lin, W.-K. Shih, A.-S. Yu, J.-Y. Chung, and W. Zhao, “Algorithms for scheduling imprecise computations,” Computer, vol.24, no. 5, pp. 58-68, May 1991.

[10] M. Rinard, “Acceptability-oriented computing,” in Proc. Companion 18th Annu. ACM SIGPLAN Conf. OOPSLA, 2003, pp. 221-239.

[11] J. Meng, S. Chakradhar, and A. Raghunathan, “Best-effort parallel execution framework for recognition and mining applications,” in Proc. IEEE Int. Symp. Parallel Distributed Process., May 2009, pp. 1-12.

[12] M. Breuer, “Multi-media applications and imprecise computation,” in Proc. 8th Euromicro Conf. Digit. Syst. Des., Aug.-Sep. 2005, pp. 2-7.

[13] A. Eltawil and F. Kurdahi, “Improving effective yield through error tolerant system design,” in Proc. 12th IEEE ICECS, Dec. 2005, pp. 1-4.

[14] X. Li and D. Yeung, “Exploiting soft computing for increased fault tolerance,” in Proc. Workshop Architectural Support Gigascale Integr., 2006.

[15] V. Wong and M. Horowitz, “Soft error resilience of probabilistic inference applications,” in Proc. Workshop Silicon Errors Logic-Syst. Effects, 2006.

[16] N. Shanbhag, “Reliable and energy-efficient digital signal processing,” in Proc. 39th Des. Autom. Conf., 2002, pp. 830-835.

[17] G. Varatkar, S. Narayanan, N. Shanbhag, and D. Jones, “Sensor network-on-chip,” in Proc. Int. Symp. System Chip, Nov. 2007, pp. 1-4.

[18] N. Verma, K. H. Lee, and A. Shoeb, “Data-driven approaches for computation in intelligent biomedical devices: A case study of EEG monitoring for chronic seizure detection,” J. Low Power Electron. Applicat., vol. 1, no. 1, pp. 150-174, 2011.

[19] K. H. Lee, K. J. Jang, S. Mohammed, A. H. Shoeb, and N. Verma, “A data-driven modeling approach to stochastic computation for low-energy biomedical devices,” in Proc. 33rd Annu. Int. IEEE EMBS Conf., Aug. 2011.

[20] L. Chakrapani, B. Akgul, S. Cheemalavagu, P. Korkmaz, K. Palem, and B. Seshasayee, “Ultraefficient (embedded) SoC architectures based on probabilistic CMOS (PCMOS) technology,” in Proc. DATE, vol. 1. Mar. 2006, pp. 1-6.

[21] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and G. Grossman, “EnerJ: Approximate data types for safe and general low-power computation,” in Proc. ACM SIGPLAN Conf. PLDI, Jun. 2011, pp. 164-174.

[22] S. Ghosh, S. Bhunia, and K. Roy, “CRISTA: A new paradigm for low-power, variation-tolerant, and adaptive circuit synthesis using critical path isolation,” IEEE Trans. Comput.-Aided Des. Integr. Circuits syst., vol. 26, no. 11, pp. 1947-1956, Nov. 2007

[23] K. Pattabiraman, G. Saggase, D. Chen, Z. Kalbarczyk, and R. Iyer, “Automated derivation of application-specific error detectors using dynamic analysis,” IEEE Trans. Dependable Secure Comput., vol. 8, no. 5, pp. 640-655, Sep.-Oct. 2011.

[24] K.-H. Huang and J. Abraham, “Algorithm-based fault tolerance for matrix operations,” IEEE Trans. Comput., vol. C-33, no. 6, pp. 518-528, Jun. 1984.

[25] Sutter, H., 2012. “Welcome to the Jungle”. URL http://www.herbsutter.com/welcome-to-the-jungle/. Retrieved April 3, 2015.

[26] Ying Zhang; Lu Peng; Xin Fu; Yue Hu, “Lighting the dark silicon by exploiting heterogeneity on future processors,” Design Automation Conference (DAC), 2013 50th ACM/EDAC/IEEE, vol., no., pp. 1, 7, May 29 2013-June 7 2013

[27] International Technology Roadmap for Semiconductors, 2010 update Tech. rep., ITRS, 2011. URL http://www.itrs.net/. Retrieved April 3, 2015.

[28] Valda Andrάs, “Programming Many-Core Chips.” Springer 2011. ISBN 978-1-4419-9739-5

[29] Ahari, A., Khaleghi, B., Ebrahimi, Z., Asadi, H., Tahoori, M.B., “Towards dark silicon era in FPGAs using complementary hard logic design”, in 24th International Conference on Field Programmable Logic and Applications (FPL), pp 1-6, 2014. DOI=10.1109/FPL.2014.6927504

[30] Allred, J.M., Roy, S., Chakraborty, K., “Dark Silicon aware multicore systems: Employing design automation with architectural insight”, in IEEE Transactions on Very Large Scale Integration (VLSI), vol. 22, Issue 5, pp. 1192-1196, June 2013. DOI = 10.1109/TVLSI.2013.2265338

[31] Hruska, Joel, February 28, 2012. “The future of CPU scaling: Exploring options on the cutting edge”. URL http://www.extremetech.com/. Retrieved April 3, 2015.

[32] Jason Allred, Sanghamitra Roy, and Koushik Chakraborty, “Designing for dark silicon: a methodological perspective on energy efficient systems”. In Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design (ISLPED ‘12). ACM, New York, NY, USA. 255-260. DOI=10.1145/2333660.2333720

[33] Karpuzcu, Ulya R.; Sinkar, Abhishek; Kim, Nam Sung; Torrellas, Josep, “EnergySmart: Toward energy-efficient manycores for Near-Threshold Computing,” High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on, vol., no., pp.542,553, 23-27 Feb. 2013, DOI = 10.1109/HPCA.2013.6522348

[34] Hyungmin Cho; Leem, L.; Mitra, S, “ERSA: Error resilient system architecture for probabilistic applications,” Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 31, no.4, pp.546, 558, April 2012, DOI = 10.1109/TCAD.2011.2179038

[35] J. Sorber, A. Kostadinov, M. Garber, M. D. Corner, and E. D. Berger, “eFlux: A Lanugage and Runtime System for Perpetual Mobile Systems”, in Technical Report 06-61, University of Massachusetts, Amherst, December 2006.

[36] R. A. Ashraf, A. Alzahrani, and R. F. DeMara, “Extending Modular Redundancy to NTV: Costs and Limits of Resiliency at Reduced Supply Voltage,” Workshop on Near Threshold Computing (WNTC-2014), Minneapolis, MN, USA, June 14, 2014.

[37] R. F. DeMara and D. I. Moldovan, “The SNAP-1 Parallel AI Prototype,” IEEE Transactions on Parallel and Distributed Systems, vol. 4, no. 8, pp. 841-854, August, 1993. DOI=10.1109/71.238620

[38] Borkar, S., “The Exascale challenge”, in International Symposium on VLSI Design Automation and Test (VLSI-DAT), vol., Issue, pp. 2-3, April 2010. DOI = 10.1109/VDAT.2010.5496640