Evaluating fault tolerance on asymmetric multicore systems-on-chip using iso-metrics

Chalios, C., Nikolopoulos, D. S., Catalan, S., & Quintana-Orti, E. S. (2016). Evaluating fault tolerance on asymmetric multicore systems-on-chip using iso-metrics. IET Computers and Digital Techniques, 10(2), 85-92. DOI: 10.1049/iet-cdt.2015.0056

Published in:
IET Computers and Digital Techniques

Document Version:
Early version, also known as pre-print

Evaluating Fault Tolerance on Asymmetric Multicore Systems-on-Chip using iso-metrics

Charalampos Chalios[1*], Dimitrios S. Nikolopoulos[1], Sandra Catalán[2], Enrique S. Quintana-Ortí[2]

[1]School of EEECS, Queen's University of Belfast, United Kingdom
[2]Depto. de Ingeniería y Ciencia de Computadores, Universitat Jaume I, Castellón, Spain
[*]cchalios01@qub.ac.uk

Abstract: The end of Dennard scaling has promoted low power consumption into a first-order concern for computing systems. However, conventional power conservation schemes such as voltage and frequency scaling are reaching their limits when used in performance-constrained environments. New technologies are required to break the power wall while sustaining performance on future processors. Low-power embedded processors and near-threshold voltage computing (NTVC) have been proposed as viable solutions to tackle the power wall in future computing systems. Unfortunately, these technologies may also compromise per-core performance and, in the case of NTVC, xreliability. These limitations would make them unsuitable for HPC systems and datacenters. In order to demonstrate that emerging low-power processing technologies can effectively replace conventional technologies, this study relies on ARM's big.LITTLE processors as both an actual and emulation platform, and state-of-the-art implementations of the CG solver. For NTVC in particular, the paper describes how efficient algorithm-based fault tolerance schemes preserve the power and energy benefits of very low voltage operation.

1. Introduction

The performance of today's computing systems is limited by the end of Dennard scaling [1] and the cooling capacity of CMOS technology [5]. To address these challenges, processor vendors turned towards multicore designs more than a decade ago. To further curb power consumption, power-saving techniques that were originally designed for battery operated embedded and mobile appliances, such as dynamic voltage and frequency scaling (DVFS) and sleep states, were integrated into mainstream desktop and server systems. Unfortunately, these efforts seem unable to support higher performance under reasonable power budgets. Computing systems are threatened by the Dark Silicon phenomenon, wherein large portions of hardware real estate remain dormant in order to avoid power emergencies. Exploring alternative methods to better utilize hardware real estate is thus paramount.

The simple, low-power processors in smartphones and tablets, as well as emerging low-voltage processors that operate near or even below threshold voltage, are promising

alternatives to tackle the power wall in computing systems. Broadly, these technologies attempt to modestly or drastically reduce supply voltage, while sustaining processor core clock frequencies to the greatest extent possible. However, these technologies may compromise performance and, in the case of near-threshold voltage computing (NTVC), hardware reliability [4]. When these technologies are adopted, it is hoped that expected reductions in performance caused by frequency decay might be compensated by cramming additional cores into the same power budget. In turn, this increase in hardware concurrency can be leveraged to improve performance and, in the case of NTVC, provide error tolerance by allocating cores to implement resilience techniques that address eventual data corruption.

This paper uses the principle of iso-metrics [11] to evaluate and compare conventional server class processors against low-power embedded architectures and NTVC processors that support Algorithm-Based Fault Tolerance (ABFT). We compare these three classes of architectures under iso-power and iso-performance scenarios [8]. Specifically, we use the heterogeneous ARM cores on the big.LITTLE system-on-chip (SoC) as an actual embedded platform and an emulated NTVC platform. We use a high performance Intel Xeon E5-2650 server as a baseline. To provide fair comparisons and in-depth insights, we use an important and well-studied algorithm, the Conjugate Gradient (CG) method [6]. CG is a memory-bound algorithm for solving linear systems; it is gaining prevalence in High Performance Computing, because it represents the type of operations and performance exhibited by many other scientific and engineering programs executing on supercomputers [2].

As an additional contribution, we describe the energy-saving potential of NTVC under a realistic application execution scenario. For this purpose, we leverage two fault-tolerant variants of CG, respectively enhanced with (i) an inner-outer detection and correction iteration [9], and (ii) and a self-stabilizing (SS) recovery mechanism [7], to assess the practical energy trade-offs between hardware concurrency, CPU frequency, and hardware error rate. We again use the ARM big.LITTLE core architectures as an emulation platform.

Our central conclusion from both studies is that emerging low-power processor technologies can reliably sustain performance and significantly improve energy-efficiency compared with the state-of-the-art. For NTVC in particular, we demonstrate that highly optimized ABFT approaches effectively preserve the energy benefits of extreme low-power hardware.

The remainder of the paper is structured as follows: Section 2 reviews related work on algorithmic fault tolerance, power saving techniques, and iso-metrics. Section 3 describes our experimental setup. Section 4 compares high performance and low-power processors using iso-metrics for performance and power. Section 5 investigates NTVC processors and the effect of NTVC on the CG method. We conclude the paper in Section 6.

2. Related Work

2.1. Power and energy minimization techniques

Power and energy minimization techniques in heterogeneous platforms have been extensively studied. Pao et al. [14] studied the opportunities for energy optimization via DVFS in integrated CPU-GPU architectures. Bailey et al. [15] designed models for power and execution time, and applied them to optimize performance in power-capped execution scenarios by selecting the appropriate CPU or GPU and the optimal hardware-configuration for execution. Klenk et al. analyzez communication patterns to identify energy optimization opportunities [16]. All these approaches utilize application characteristics to select an optimal, energy-aware execution configuration. We explore the potential of leveraging algorithmic information, fault-tolerance, and NTVC to further reduce energy consumption beyond the state-of-the-art.

2.2. Algorithmic fault-tolerance

Several recent works have considered the effect of soft errors (also known as silent data corruption) on iterative Krylov subspace methods. Chen [12] proposed an on-line orthogonality test to detect soft errors during the execution of Krylov subspace solvers combined with check-pointing. Hoemmen and Heroux [9] applied selective reliability to assemble a fault-tolerant version of GMRES that primarily operates in unreliable mode, to avoid expensive restarts from checkpoints. Elliott, Hoemmen, and Mueller [13] introduced a low-cost fault detection mechanism for GMRES, expanding this to limit the magnitude of the error that the method may return. Pao and Vuduc [7] adopted “self-stabilization” to obviate the need for full state saving and fault detection, proposing a method that, in an error-prone scenario, can reach a correct state in a finite number of steps, ensuring convergence. These methods point out the importance of implementing algorithmic-level error-recovery techniques; however, none of them analyzed the energy overhead of errors and the recovery

mechanisms they propose. Our approach measures these overheads on a platform that can operate in two modes: reliable or unreliable. In “unreliable” mode our platform reduces power consumption, which is in par with the projections in [4].

2.3. Iso-metrics

Iso-energy-efficiency models were introduced by Song et al [8] to predict and balance energy and performance in large power-aware clusters, by taking into account software characteristics. We extend their approach with additional metrics and a different, application-driven experimental methodology to explore the trade-offs between performance, power, and reliability on high-end multicore processors and low-power SoCs. Our goal is to investigate whether it is possible to build systems out of unconventional low-power architectures that can match the performance of current throughput-oriented architectures. In a work similar to ours, Göddeke et al. [3] studied the use of low-power architectures in scientific applications. In this study, we take Göddeke's work one step further to make projections about the energy efficiency of platforms operating under NTVC conditions with lower reliability, and use efficient fault tolerance techniques to tackle unreliability.

2.4. Contributions

A preliminary evaluation of iso-metrics and the cost of fault tolerance was presented in our own prior work [11]. This paper extends our prior work by: *(i)* targeting a collection of sparse problems, arising from actual scientific and engineering applications, instead of simple synthetic dense benchmarks; *(ii)* including a fault-tolerant version of CG based on a conventional detection and correction strategy embedded into an inner-outer Generalized Minimal Residual (GMRES) solver [9], in addition to the SS approach, which was the only option considered in our previous study; and *(iii)* employing a more realistic implementation of the fault-tolerant versions of CG that switches between cores with different levels of reliability at execution time, thus capturing the overhead of these changes.

3. Experimental setup

3.1. The CG method

The CG method is a key algorithm for the numerical solution of linear systems of the form $Ax = b$, where $A \epsilon R^{nxn}$ is symmetric positive definite (SPD) and sparse, $b \epsilon R^n$ contains the independent terms, and $x \epsilon R^n$ is the sought-after solution [6]. The cost of this iterative method is dominated by a sparse matrix-vector multiplication involving A, which must be computed once per iteration. For a matrix A with n_z nonzero entries, this operation requires roughly $2n_z$ floating-point arithmetic operations (flops). Additionally, each iteration involves a few vector operations that cost $O(n)$ flops each.

For our evaluation, we employ IEEE 754 real double-precision arithmetic and terminate the CG loop after 2,000 iterations. Furthermore, for simplicity, we do not exploit the symmetric structure of the matrix. Under these conditions, we estimate the cost of CG to be $2n_z$ flops per iteration (i.e., we disregard the lower cost of the vector operations). Moreover, for efficiency, we leverage multi-threaded implementations of the matrix-vector multiplication kernel in Intel MKL (version 11) for the Intel-based CPU. For the ARM-based cores, we rely on our own multi-threaded implementation of this operation, which is based on the CSR sparse matrix layout [9] and built upon the OpenMP parallel programming interface.

3.2. Target architectures and scenarios

The experiments in this paper were performed using three different multicore processors. The first processor, hereafter referred to as Xeon, is a high-performance Intel Xeon E5-2650 processor with 8 cores, connected to 16 GBytes of DDR3-1333 MHz RAM. The alternative low-power architectures are two ARM quad-core clusters, based on Cortex-A15 and Cortex-A7 cores, in an Exynos5 SoC of an ODROID-XU board. These two clusters share 2 GBytes of DDR3-800 MHz RAM. Table 1 lists the most important features of the processor architectures considered in this paper. The column labeled “stream memory bandwidth” reports the memory bandwidth measured using the triad test of the stream benchmark (http://www.cs.virginia.edu/stream), when executed with all cores available in the sockets. The column labeled “Roofline GFLOPS” corresponds to the theoretical upper bound on computational performance (in terms of GFLOPS, or billions of flops per second) derived from the roofline model [10].

Table 1. Hardware specifications of the target architectures.

Acron.	CPU socket/cluster	#Cores	Frequency range (GHz)	LLC: level, type, size (Mbytes)	TDP (W)	Peak mem. bandwidth (GBytes/s)	Stream mem. bandwidth (GBytes/s)	Roofline GFLOPS
XEON	Intel Xeon E5-2650	8	1.2-2.0	L3, shared, 20	95	51.2	44	11
A15	ARM Cortex-A15	4	0.8-1.6	L2, shared, 2	N/A	N/A	5.4	1.35
A7	ARM Cortex-A7	4	0.5-1.2	L2, shared, 0.5	N/A	N/A	2.07	0.51

For our experimental analysis, we investigate scenarios that vary the number of cores, the core clock frequency, and the application benchmarks. For simplicity, we consider the lowest and highest clock frequencies (disregarding Intel's turbo-mode) for each architecture. We also consider a collection of inputs that represent “on-chip” execution, whereby the target working data fill the last-level cache (LLC) of each architecture. Our previous study in [11] exposed the poor scalability attained with CG when the key working sets reside off-chip. We purposely omit off-chip working sets to avoid blurring the iso-metric comparison of processor architectures with memory effects. As part of future work, we intend to perform full system comparisons that also consider alternative memory technologies.

The performance of CG, as well as the fault-tolerant methods built upon it, strongly depend on the implementation of the kernel for the sparse matrix-vector multiplication. The throughput of this operation, in turn, is governed by the sparsity structure and storage layout of the matrix, which dictates the memory access pattern. To capture the behavior of different scenarios, for each architecture we use a set of eight sparse matrices, as well as one dense matrix, all of which fit into the LLC. The sparse cases in Table 2 were obtained from the University of Florida Sparse Matrix Collection (UFMC). For each metric (GFLOPS, power and GFLOPS/W), we present the average value obtained for the cases selected, for each architecture.

Table 2. Sparse matrices from UFMC employed in the evaluation.

XEON			A15			A7		
Name	Rows	#nonzeros	Name	Rows	#nonzeros	Name	Rows	#nonzeros
apache1	80,800	542,184	aft01	8,205	125,567	bcsstk21	3,600	26,600
cbuckle	13,681	676,515	bcsstk13	2,003	83,883	bcsstm12	1,473	19,659

denormal	89,400	1156,224	bloweybq	10,001	49,999	ex33	1,733	22,189
finan512	74,752	596,992	ex10hs	2,548	57,308	mhd4800b	4,800	27,520
G2_circuit	150,102	726,674	ex13	2,568	75,628	msc00726	726	34,518
Pres_Poisson	14,822	715,804	nasa4704	4,704	104,756	nasa1824	1,824	39,208
thremal1	82,654	574,458	s1rmq4m1	5,489	262,411	plat1919	1,919	32,399
thermomech_TK	102,158	711,558	sts4098	4,704	104,756	plbuckle	1,824	39,208

4. High Performance vs Low Power

In this section, we perform an experimental evaluation of the target CPU architectures, assuming they operate in nominal regions. We use the CG method, implemented on top of optimized multi-threaded versions of the sparse matrix-vector kernel from MKL and our ad-hoc OpenMP routine. The purpose of this analysis is to expose the trade-offs between performance, power dissipation, and energy efficiency, for a memory-bound method such as CG, with the ultimate goal of answering two key questions:

- *Q1 (Iso-performance)*: Can we attain the performance of the Intel Xeon processor with the low-power ARM clusters while yielding a more power-efficient solution?

- *Q2 (Iso-power)*: What level of performance that be attained using the low-power ARM clusters within the power budget of the Intel Xeon socket?

4.1. Trade-offs

Figure 1 illustrates the results from the evaluation of the multithreaded CG implementations, in terms of performance (GFLOPS), power dissipation (W), and energy efficiency (GFLOPS/W). An evaluation in terms of GFLOPS and GFLOPS/W allows a comparison of these metrics for problems that vary in size and number of FLOPS performed. Table 3 quantifies the trends captured by the figure. Our analysis of these results is organized along three axes: number of cores (#cores), frequency, and architecture (configuration parameters), as well as three metrics. From the point of view of concurrency (#cores), increasing the number of these resources produces fair speed-ups for irregular memory-bound problems, which are slightly superior for XEON processors and similar for both ARM

architectures, independent of their frequency. For example, the use of four cores on A15 and A7 produces speedups between 2.0 and 2.4 for any of the two frequencies. The equivalent speedup on XEON is 3.1 with 4 cores and both frequencies. From the perspective of power, a linear regression fit to the data shows a high *y*-intercept for XEON, which corresponds to static power, and can be explained by its large LLC, the complex pipeline, the large area dedicated to branch prediction, and other complex micro-architecture structures of the XEON. By comparison, A15 and A7 exhibit much lower static power, reflecting the simpler design of these CPU clusters. This difference between the Intel and ARM-based architectures has a major impact on energy. Increasing the number of XEON cores results in shorter execution time. Moreover, the large static power can be detrimental for energy efficiency (GFLOPS/W). This is a clear indicator of the potential benefits of a "race-to-idle" policy, which can be applied to this architecture in an effort to amortize its high static power consumption. The effect of increasing number of cores for A15 and A7 is the opposite, owing to their low static power.

We proceed with the analysis of frequency. Independent of the number of cores, the effect of this parameter on performance is perfectly linear for XEON but sub-linear for A15, where doubling the frequency only improves performance by a factor of approximately 1.8; the improvement is slightly higher for A7, where raising the frequency from 0.5 to 1.2 GHz (a factor of 2.4*x*) results in a performance increase of 2.1*x*. The effect of frequency on power is sub-linear for XEON (a factor between 1.28 - 1.62*x*, depending on the number of cores) and super-linear for both A15 (3.24 - 3.59*x*) and A7 (3.58 - 3.75*x*). The net effect of the variations of time and power with the frequency is that, on XEON, increasing the frequency slightly improves energy efficiency (again pointing to a race-to-idle strategy for energy-aware execution), while on the ARM-based clusters, it reduces energy consumption by a factor close to 48% for A15 and 55% for A7.

Finally, we observe some additional differences between the processor architectures: the 8-core Intel processor produces significantly higher performance rates (and, therefore, shorter execution times) than the ARM clusters, at the expense of a much higher power dissipation rate and much lower energy efficiency (GFLOPS/W). The differences between the two types of ARM clusters also follow a similar pattern, offering higher performance with the A15 in exchange for higher power and lower energy efficiency.

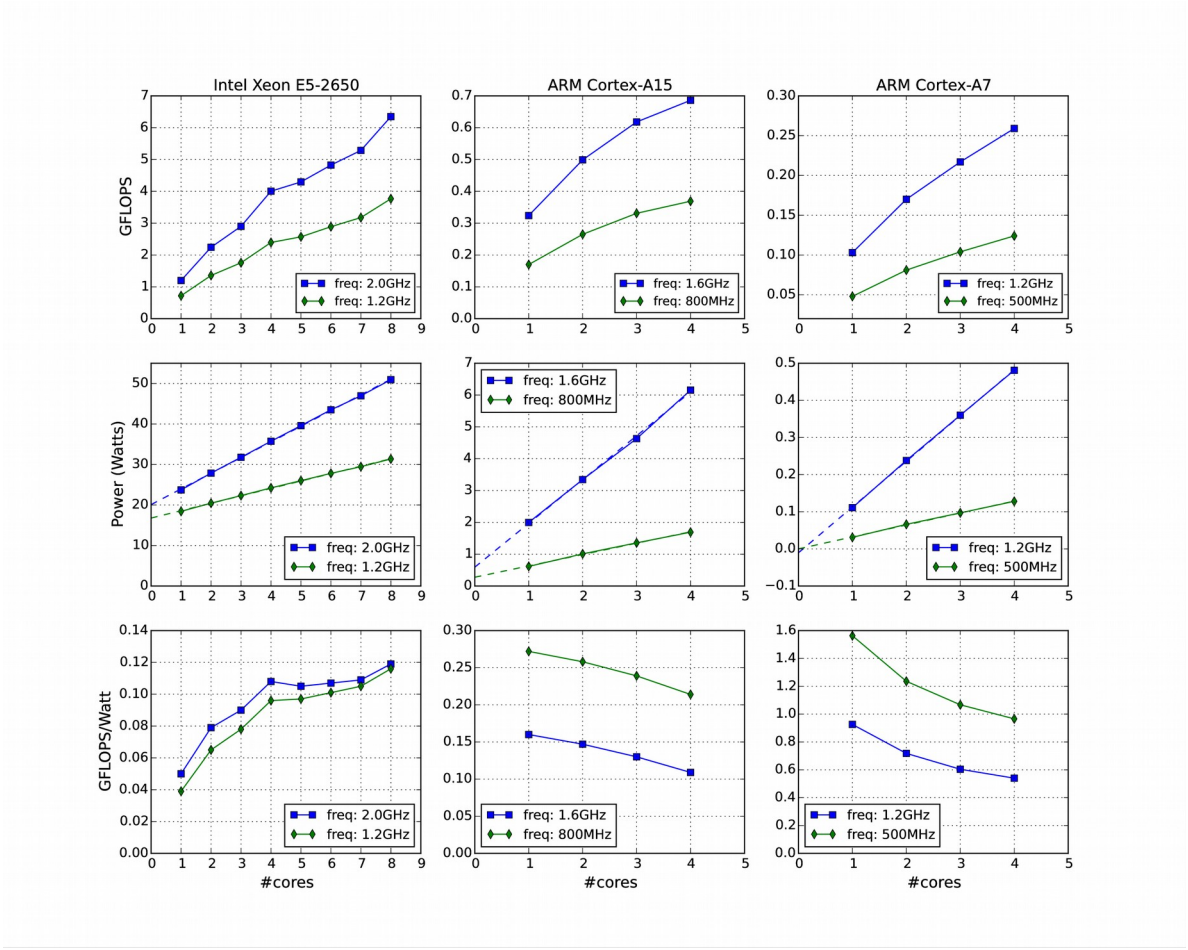

Figure 1. Evaluation of performance, power, and energy on the target architectures using multi-threaded implementations of the CG method for the on-chip problem.

Table 3. Evaluation of performance, power, and energy on the target architectures using multi-threaded implementations of the CG method on the on-chip problems.

CPU	Freq. (GHz)	#cores	Time per iter. (ms)	Performance (GFLOPS)	Speed-up	Power (W)	Energy (GFLOPS/W)
XEON	1.2	1	3.11	0.72	1.0	18.4	0.039
		2	1.73	1.35	1.8	20.4	0.065
		4	1.02	2.39	3.1	24.2	0.096
		6	0.78	2.88	4.0	27.8	0.101
		8	0.66	3.76	4.8	31.3	0.116
	2.0	1	1.87	1.19	1.0	23.7	0.050
		2	1.05	2.24	1.8	27.9	0.079
		4	0.61	4.00	3.1	35.7	0.108
		6	0.47	4.82	4.0	43.4	0.107
		8	0.39	6.34	4.8	50.9	0.119:q

A15	0.8	1	1.20	0.17	1.0	0.61	0.272
		2	0.76	0.26	1.5	1.01	0.258
		4	0.55	0.36	2.1	1.69	0.214
	1.6	1	0.71	0.31	1.0	1.98	0.154
		2	0.46	0.47	1.5	3.32	0.140
		4	0.34	0.64	2.0	6.08	0.103
A7	0.5	1	1.38	0.048	1.0	0.031	1.563
		2	0.84	0.081	1.6	0.066	1.236
		4	0.58	0.124	2.4	0.128	0.966
	1.2	1	0.65	0.103	1.0	0.111	0.926
		2	0.40	0.170	1.6	0.238	0.718
		4	0.27	0.259	2.4	0.481	0.540

4.2. Analysis of iso-metrics

We start the study of iso-metrics by noting that the questions Q1 (iso-performance) and Q2 (iso-power), formulated at the beginning of this section, can be explored in a different number of configurations/scenarios. Here we select one that is relevant for design space exploration. Concretely, for Q1 we set as the baseline the performance of 1 to 8 XEON cores, clocked at 2.0 GHz; and then we evaluate how many clusters (consisting of A15 or A7 cores, operating at either the lowest or highest frequencies) are necessary to match the reference XEON performance. Question Q2 is the iso-power counterpart of Q1, with the baseline power budget fixed by the values obtained with 1 - 8 XEON cores operating at 2.0 GHz.

The left plot in Figure 2 reports the results from the iso-performance study. In order to achieve the performance of eight XEON cores (2.0 GHz), it is necessary to use almost 10 A15 clusters (i.e., quad-cores) at 1.6 GHz or more than 51 A7 clusters at 0.5 GHz (note the different scales of the *y*-axis depending on the type of cluster). In this comparison we implicitly introduce a simplification that favors the ARM processors. Specifically, on the multi-socket ARM platform, data and operations must be partitioned between the clusters, incurring an overhead associated with communication. For the CG method, we can expect that this additional cost comes primarily from the reduction vector operations (which are analogous to synchronization). Furthermore, there is additional overhead caused by the relatively small problem sizes assigned to each core. These sources of overhead are disregarded in our study.

The right-hand side plot in Figure 2 illustrates the ratio between the power rates dissipated by four configuration “pairs” that attain the same performance. Each pair contains a XEON core and either a A15 or A7 core, at the lowest or highest frequency. Using the

previous examples, eight XEON cores (2.0 GHz) deliver the same performance as 9.9 clusters consisting of A15 cores at 1.6 GHz; however the A15 clusters consume 18% more power. On the other hand, using 51.1 clusters of A7 cores at 0.5 GHz only requires a fraction of the power rate dissipated by XEON; concretely 12%.

Figure 3 displays the results from the complementary study on iso-power. The plot on the left illustrates that the power budget of 1 - 8 XEON cores can accommodate a moderate number of A15 clusters or a very large volume of A7 clusters. (Note again the different scales in the *y*-axis.) The performance ratio between these ARM-based clusters with respect to the XEON, shown on the right-hand side plot, reveals decreasing gains with the increasing number of A15 clusters and equal performance with respect to seven or more XEON cores. The ratio also decays for the A7 clusters; however in this case it stabilizes around a factor of eight.

We note that not all ARM-based configurations considered in the iso-performance and iso-power study contain the same on-chip memory capacity (iso-capacity) as XEON. In particular, at least 10 A15 clusters and 40 A7 clusters would be required to achieve an iso-capacity scenario, from the perspective of on-chip memory, with respect to the LLC in XEON.

We close this section by noting that a study of the energy efficiency ratio under the conditions imposed by Q1 or Q2 does not contribute new information. For example, given that Q1 compares the GFLOPS/W of the architectures with equal GFLOPS rates, an evaluation of energy efficiency is equivalent to the analysis of the power ratio.

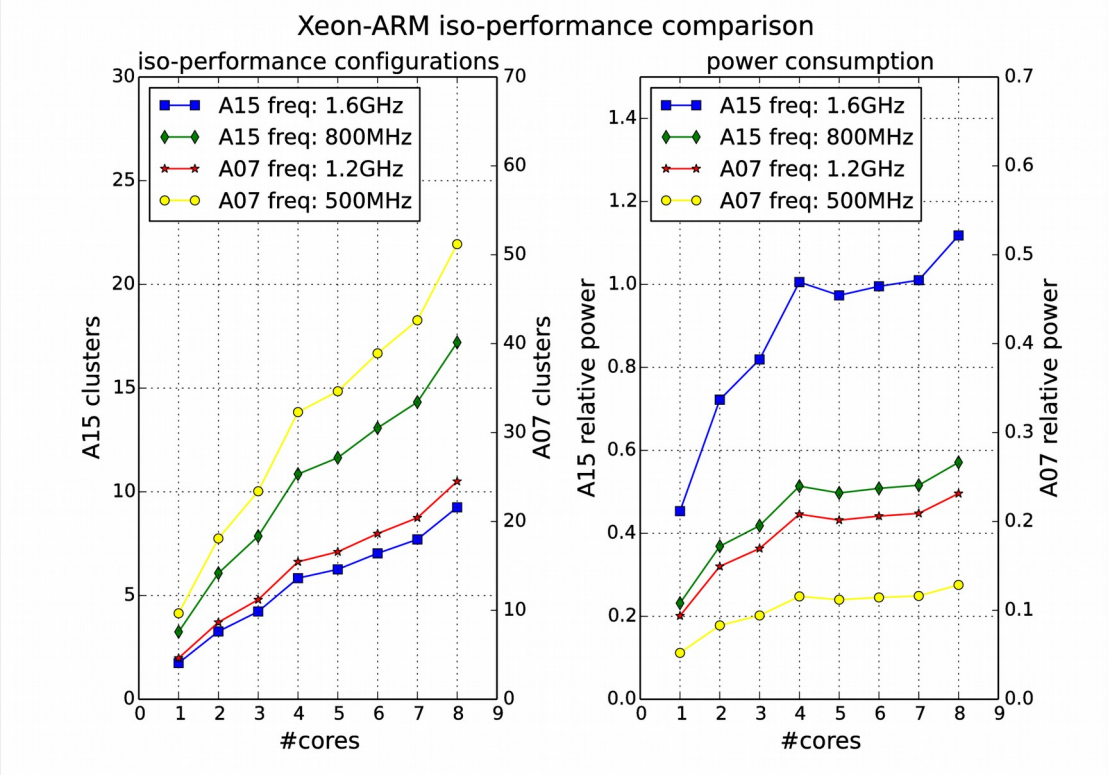

Figure 2. Evaluation of iso-performance. Left: Number of A15 or A7 clusters required to match the performance of a given number of XEON cores at 2.0 GHz. Right: Comparison of power rates dissipated for configurations delivering the same performance.

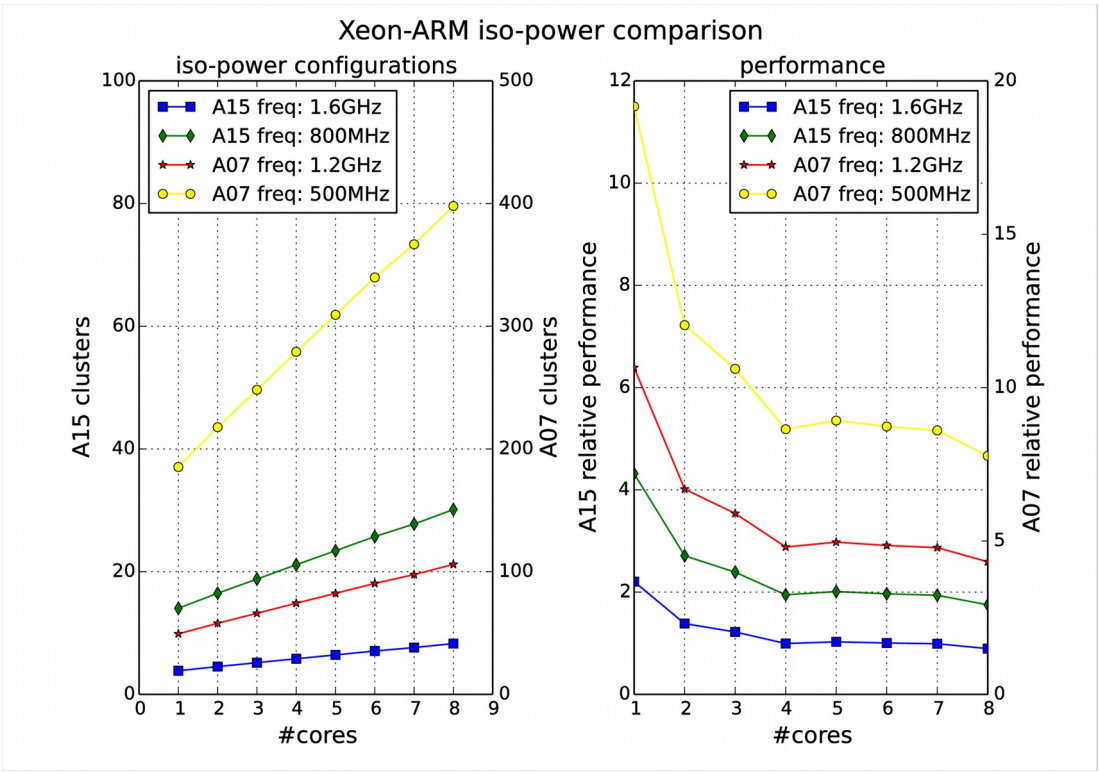

Figure 3. Evaluation of iso-power. Left: Number of A15 or A7 clusters that match the power disipated by a given number of XEON cores at 2.0 GHz. Right: Comparison of performance rates attained for configurations dissipating the same power rate.

5. Energy Cost of Reliability

The experiments and analysis in this section aim to explore the potential impact of NTVC on energy, a technique that trades lower processor voltage and frequency for higher concurrency, but also a higher failure rate. In order to perform this study, we make the following assumptions:

- To emulate a reliable/unreliable execution, we consider a big.LITTLE SoC consisting of several (N) quad-core A15 cluster plus the same number of A7 clusters. Here, the A15 clusters operate at the highest frequency, are reliable, and apply the fault tolerance mechanism (i.e., the stabilizing part in SS or the computations other than CG in the GMRES-based solver). On the other hand, the A7 clusters operate at the lowest frequency, operate in the NTVC region, and are de facto less reliable than the A15 clusters. The A7 clusters are thus used to compute the CG iterations. We will refer to this SoC as NA15/A7, and we will use performance and energy efficiency data for the corresponding on-chip problems for all experiments. Furthermore, we assume that idle clusters (e.g., the N A7 clusters during the execution of the stabilizing operations) still contribute to the total consumption with an idle power. Safety measures integrated in the hardware prevent us from using these architectures below the minimum nominal frequency/voltage. The A7 consumes 0.128 Watts when being used at the highest frequency. Given that these operating values are significantly lower than state-of-the-art high performance cores, we use the A7 power and performance regions as representative of NTVC cores.

- We employ a tuned variant of our multi-threaded implementations of the CG method equipped with an SS recovery mechanism [7] to cope with silent data corruption introduced by unreliable hardware. Following the experiments in [7], the SS part is activated once every 10 iterations in the CG method, and must be performed on reliable cores. From the computational point of view, the major difference between an SS iteration and a “normal” CG iteration (baseline routine) is that the former performs

two matrix-vector products instead of only one. However, these two operations can be performed simultaneously, as they both involve matrix *A*. Therefore, for a memory-bound operation such as the matrix-vector product, we assume that, in practice, the two types of iterations share the same computational cost.

- Additionally, we consider an outer GMRES iteration that leverages CG as the inner solver [9] and integrates a conventional detection/correction mechanism for fault tolerance. The cost of this inner-outer iteration is dominated by the CG solver plus two additional matrix-vector products involving matrix *A*. The CG iteration can be performed on unreliable cores, but the remaining computations should be executed on reliable cores. Following the experiments in [9], we perform 20 iterations of the outer method, with the inner solver executed for 50 iterations each time it is invoked. With these numbers, 96.2% of the FLOPS are performed on unreliable cores and only 3.8% are performed on reliable cores.

- The convergence rate of the CG iteration depends on the condition number of matrix *A* [6]. The convergence of the fault-tolerant variants degrades logarithmically with the error rate [7]. SDC occurs during the matrix-vector product, introducing bit flips into any of its results, and propagates from there to the remainder of the computations. The convergence rate of the fault-tolerant variants also depends to some extent on whether the bit flips are bounded to the sign/mantissa or can also affect the exponent.

Under the aforementioned conditions, we perform an experimental analysis of the energy gains that a hybrid reliable/unreliable big.LITTLE SoC can achieve, against a reliable single quad-core A15 cluster operating at the highest frequency. We compare the architectures again under iso-performance and iso-power conditions. Note that for the latter, we still consider the power of a single idle quad-core A7 cluster.

Under iso-performance assumptions, we aim to determine how many NA15/A7 clusters must be involved during the execution of the fault-tolerant CG iterations so that, when combined, the hybrid clusters match the performance of the baseline CG solver running on a single A15 cluster operating at the highest frequency. Table 4 lists these values, which were determined experimentally for each problem (matrix) and fault-tolerant solver. For example, for the dense case, 7.06 NA15/A7 clusters are required to run the SS variant at the same

GFLOPS rate as the baseline CG solver executed on a single A15 cluster. For simplicity, we will round this number to seven NA15/A7. Next we can compare the power dissipation rate of the two configurations (for the dense case and SS): 4.28 W for one A15 cluster (plus one idle A7 cluster) and 2.1 W for seven NA15/A7. This implies that, for this benchmark case and fault-tolerant solver, it is possible to accommodate an increase in the number of iterations (decay of convergence) that is close to a factor of 2 and still attain the same energy-to-solution (ETS). Figures 4 and 5 report the percentage of increase in the number of fault-tolerant solver iterations (executed on unreliable clusters) that would produce the same ETS as the baseline CG executed on the reliable platform. These results demonstrate the energy gains that can be expected from operating with simpler low-power cores, at low frequencies, for these particular applications. Concretely, depending on the benchmark case, NA15/A7 outperform a single A15 in terms of ETS when the degradation occurs in up to 7 - 129% more iterations for SS and 178 - 359% for the fault-tolerant version of GMRES furnished with an inner-outer iteration.

Table 4. Iso-performance of reliable A15 cluster vs unreliable NA15/A7.

benchmark	iso-performance CG-SS	iso-performance GMRES
dense	7.06	5.02
nasa1824	9.24	7.42
bcsstk21	9.78	8.23
bcsstm12	13.1	9.35
plat1919	8.53	6.74
msc00726	13.09	9.02
ex33	11.67	8.33
plbuckle	11.94	8.71

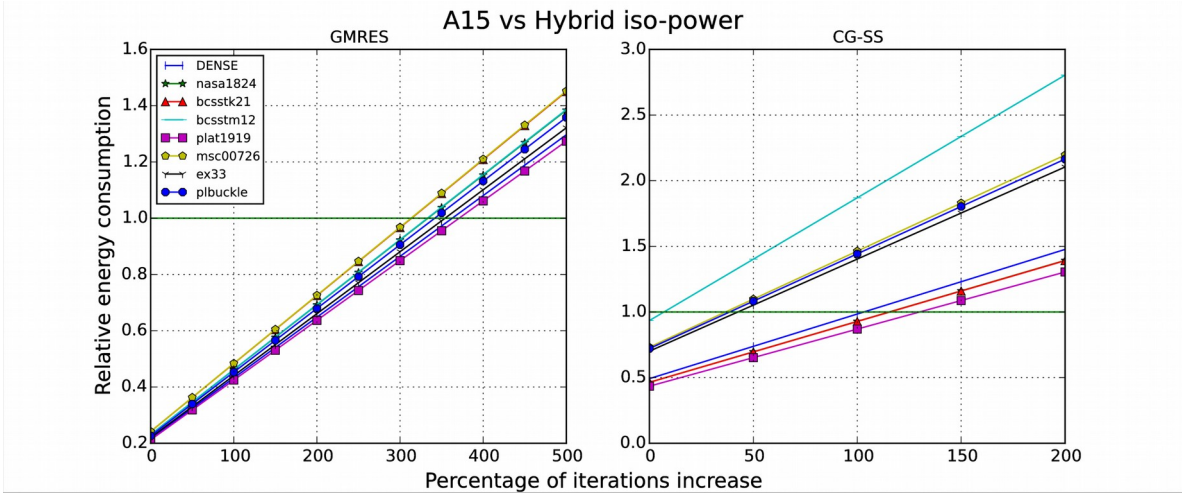

Figure 4. Iso-performance ETS for the original CG (right) and GMRES (left) methods executed by A15 at the highest frequency (reliable mode) and the SS variant of CG executed by NA15/A7 under unreliable conditions that degrade convergence.

We also conducted an iso-power study. We set the power dissipated by the A15 cluster, operating at the highest frequency, as the baseline. We derive how many NA15/A7 clusters, operating at the lowest frequency, fit within the same power budget. This exercise produces the same ETS as the iso-performance analysis. This is to be expected, because any increase in NA15/A7 clusters yields a proportional increase in its GFLOPS rate, or equivalently an inversely proportional decrease in execution time. Simultaneously, the power dissipation will be increased in the same proportion, yielding the same ETS.

To conclude this section, we focus on the iso-capacity problem. For this case study, we require the aggregated LLC of the A7 clusters in NA15/A7 to equal the capacity of A15. Now, the A15 includes a 2MB LLC cache and four A7 clusters, to match the LLC capacity of a single A15 cluster (see Table 1). In conclusion, we can build an NA15/A7 system that can solve problems that are same in size to those tackled by a single A15.

6. Conclusions and Future Work

The computation and data processing requirements of future systems demand more energy-efficient processors. In this study, we utilize processors designed for the mobile computing market and future processors that operate outside the nominal supply voltage regions to investigate whether they can be used to build HPC systems and datacenters with better energy-to-performance ratios. We concretely show that it is possible to use power-efficient ARM clusters in order to match the performance of a high-end Intel Xeon processor while operating, in a worst-case scenario, within the same power budget. Conversely, it is possible to use a rather large number of ARM clusters, fit within the power budget of one Intel Xeon processor, and attain higher performance.

As a further contribution, we tested a reliable CG execution in an A15 cluster, and compared it with an execution of fault-tolerant variants of this method using a hybrid configuration of A15 and A7 clusters to emulate an unreliable processor that operates in the NTVC region. From this study, we find that it is possible to improve ETS, even when errors significantly slow down the convergence of CG.

Because the cornerstone of the CG method is the sparse matrix-vector product, we believe that the significance of this study carries over to many other numerical methods for

scientific and engineering applications. On the other hand, the study has certain limitations. For example, we did not consider factors such as the cache hierarchy, interconnection networks, memory buses, and bandwidth, which can be relevant in large-scale designs and affect both performance and power consumption. We made this choice in order to be able to extract some first-order conclusions about the potential of employing NTVC; however we intend to investigate those matters in more depth in the future.

7. Acknowledgments

We thank F. D. Igual, from Universidad Complutense de Madrid, for his help with the Odroid board. Sandra Catalán and Enrique S. Quintana-Ortí were supported by projects TIN2011-23283 and TIN2014-53495-R of the MINECO and FEDER, and the EU project FP7 318793 "EXA2GREEN". This work was partially conducted while this author was visiting Queen's University of Belfast. This research has also been supported in part by the European Commission under grant agreements FP7-323872 (ScoRPiO), FP6-610509 (NanoStreams) and by the UK Engineering and Physical Sciences Research Council under grant agreements EP/L000055/1 (ALEA), EP/L004232/1 (ENPOWER) and EP/K017594/1 (GEMSCLAIM)

8. References

[1] R.H. Dennard, F.H. Gaensslen, V.L. Rideout, E. Bassous, and A.R. LeBlanc. Design of ion-implanted MOSFET's with very small physical dimensions. IEEE J. Solid-State Circuits, 9(5):256–268, 1974.

[2] K. Asanovic et al. The landscape of parallel computing research: A view from Berkeley. Technical Report UCB/EECS-2006-183, University of California at Berkeley, EECS, 2006.

[3] D. Göddeke, D. Komatitsch, M. Geveler, D. Ribbrock, N. Rajovic, N. Puzovic, and A. Ramirez. Energy efficiency vs. performance of the numerical solution of PDEs: An application study on a low-power ARM-based cluster. J. Computational Physics, 237(0):132–150, 2013.

[4] U.R. Karpuzcu, Nam Sung Kim, and J. Torrellas. Coping with parametric variation at near-threshold voltages. Micro, IEEE, 33(4):6–14, 2013.

[5] R. Lucas et al. Top ten Exascale research challenges, 2014. http://science.energy.gov//media/ascr/ascac/pdf/meetings/20140210/ Top10reportFEB14.pdf.

[6] Y. Saad. Iterative Methods for Sparse Linear Systems. SIAM, 2003.

[7] P. Sao and R. Vuduc. Self-stabilizing iterative solvers. In Workshop Latest Advances in Scalable Algorithms for Large-Scale Systems, pages 4:1–4:8, 2013.

[8] S. Song, C. Su, R. Ge, A. Vishnu, and K. Cameron. Iso-energy-efficiency: An approach to power-constrained parallel computation. In IEEE Int. Parallel Distr. Proc. Symp. (IPDPS), pages 128–139, 2011.

[9] M. A. Heroux and M. Hoemmen, Fault-tolerant iterative methods via selective reliability, Tech. Rep. SAND2011-3915 C, Sandia National Laboratories, 2011. Available at http://www.sandia.gov/~maherou/.

[10] Williams, Samuel, Andrew Waterman, and David Patterson. "Roofline: an insightful visual performance model for multicore architectures." Communications of the ACM 52.4 (2009): 65-76.

[11] Charalampos Chalios, Dimitrios S. Nikolopoulos, Enrique S. Quintana-Orti. Evaluating Asymmetric Multicore Systems-on-Chip using Iso-Metrics. arXiv:1503.08104 [cs.DC], 2015

[12] Z. Chen. Online-ABFT: An online algorithm based fault tolerance scheme for soft error detection in iterative methods. In Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP ’13, pages 167–176, 2013.

[13] J. Elliott, M. Hoemmen, and F. Mueller. Evaluating the impact of SDC on the GMRES iterative solver. In Proc. 2014 EEE 28th Int. Parallel and Distributed Processing Symp., IPDPS’14, pages 1193–1202, 2014.

[14] Indrani Paul, Vignesh Ravi, Srilatha Manne, Manish Arora, and Sudhakar Yalamanchili, “Coordinated Energy Management in Heterogeneous Processors,” Scientific Programming, vol. 22, no. 2, pp. 93-108, 2014. doi:10.3233/SPR-140380

[15] Bailey, P.E.; Lowenthal, D.K.; Ravi, V.; Rountree, B.; Schulz, M.; de Supinski, B.R., "Adaptive Configuration Selection for Power-Constrained Heterogeneous Systems," *Parallel Processing (ICPP), 2014 43rd International Conference on* , vol., no., pp.371,380, 9-12 Sept. 2014. doi: 10.1109/ICPP.2014.46

[16] Klenk, B.; Oden, L.; Froning, H., "Analyzing communication models for distributed thread-collaborative processors in terms of energy and time," *Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium on* , vol., no., pp.318,327, 29-31 March 2015. doi: 10.1109/ISPASS.2015.7095817