Cherry-Picking: Exploiting Process Variations in Dark-Silicon Homogeneous Chip Multi-Processors

Bharathwaj Raghunathan[1], Yatish Turakhia[2], Siddharth Garg[1,*] and Diana Marculescu[3]

[1]Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON
[2]Department of Electrical Engineering, Indian Institute of Technology Bombay, Mumbai, India
[3]Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA
[*]Email ID of corresponding Author s6garg@ecemail.uwaterloo.ca

***Abstract*—It is projected that increasing on-chip integration with technology scaling will lead to the so-called dark silicon era in which more transistors are available on a chip than can be simultaneously powered on. It is conventionally assumed that the dark silicon will be provisioned with heterogeneous resources, for example dedicated hardware accelerators. In this paper we challenge the conventional assumption and build a case for *homogeneous* dark silicon CMPs that exploit the inherent variations in process parameters that exist in scaled technologies to offer increased performance. Since process variations result in core-to-core variations in power and frequency, the idea is to cherry pick the best subset of cores for an application so as to maximize performance within the power budget. To this end, we propose a polynomial time algorithm for optimal core selection, thread mapping and frequency assignment for a large class of multi-threaded applications. Our experimental results based on the Sniper multi-core simulator show that up to 22% and 30% performance improvement is observed for homogeneous CMPs with 33% and 50% dark silicon, respectively.**

I. Introduction

Technology scaling has enabled increasing on chip integration to the extent that, in the near future, a chip will have more transistors than can be simultaneously powered on within the peak power and temperature budgets. This has been referred to as the dark silicon era [5] where, at any given point in time, only a percentage of transistors on the die are operational. Dark silicon multi-core systems have typically been thought of in the context of heterogeneous computing using, for example, a multitude of dedicated hardware accelerators to assist the on-chip cores [6]. However, we take a different view of the problem and attempt to the answer the following question — is there a case for traditional *homogeneous* multi-core chips in the dark silicon era? In other words, is there any benefit in provisioning a chip multi-processor (CMP) with more homogeneous cores than can be simultaneously powered on? In this paper, we demonstrate that by exploiting the core-to-core variations in leakage power dissipation and clock frequency introduced by process variations, this question can be answered in the *affirmative*.

Process variations have typically been thought of as a major design concern for CMOS ICs. The magnitude of process variations increases with technology scaling and with decreasing supply voltages, transistors are more susceptible to variations in their process parameters. In their seminal work, Bowman et al. demonstrated that up to 30% of performance can be lost due to process variations alone in scaled technologies [2]. This result is based on the observation that the performance of a synchronous digital system is dependent on the slowest critical path on a chip. By the same token, the performance of a multi-threaded application is typically determined by the slowest core on a chip, if all cores are be utilized.

Dark silicon chips, on the other hand, offer a new opportunity to *exploit* process variations. In particular, since we are allowed to *pick and choose* which cores on the chip to turn on, we can potentially harness process variations to our benefit by picking the subset of cores that best fit the application characteristics. We refer to this intuitive idea as **cherry-picking**. In this paper, we propose a framework to evaluate the benefits of cherry-picking in dark silicon homogeneous CMPs. In particular, we show that as the number of redundant homogeneous cores increases, i.e., the percentage of dark silicon increases, there is an increasing performance benefit from the ability to cherry pick cores from a larger set.

Figure 1 shows an overview of the proposed approach using an example of a 16 core homogeneous CMP with 4 redundant cores (25% dark silicon). Multi-threaded applications are mapped on to the CMP by choosing the optimal *subset* of cores, i.e., cherry-picking cores, that maximize application performance under a power budget. The unmapped cores are left dark. We validate our proposed ideas on multi-threaded applications from the SPLASH-2 and PARSEC benchmark suites and perform detailed simulations using a modified version of the Sniper multi-core simulation infrastructure [3].

II. Related Work

Goulding et al. [6] and Esmailzadeh et al. [5] were amongst the first to observe, using power performance projections, the impending onset of the dark silicon computing era. Goulding et al. propose populating the dark silicon area with dedicated hardware accelerators optimized for common application templates [6]. Esmailzadeh et al. [5] explore the design space of dark silicon CMPs but do not account for process variations. Karpuzcu et al. have proposed Bubblewrap [9], a technique that

978-3-9815370-0-0/DATE13/©2013 EDAA

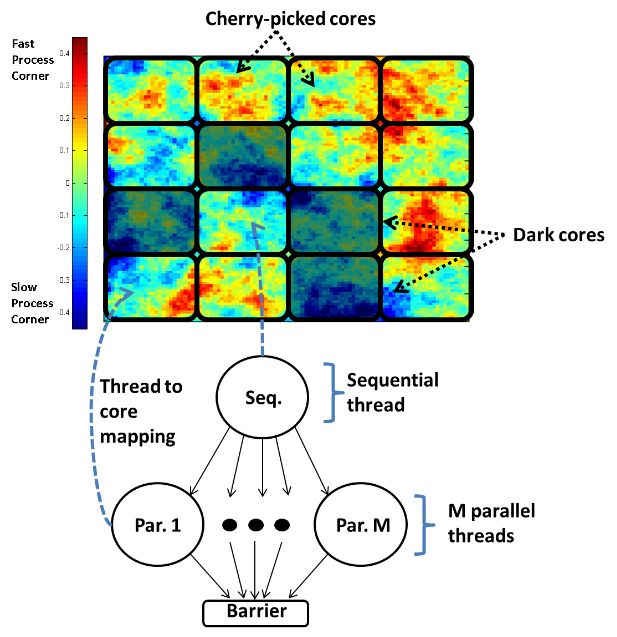

Fig. 1. A homogeneous CMP with N cores overlaid on top of a process variation map. Multi-threaded applications with M parallel threads ($M \leq N$) are mapped to the CMP by cherry-picking the best cores within a power budget. The remaining cores are dark.

makes uses of the extra cores available on a chip to extend its lifetime. In fact, extended product lifetime is *another* reason for designing homogeneous dark CMPs. However, the focus of this paper is on exploiting process variations in dark silicon chips for performance improvement under a power budget.

A number of recent papers have addressed the problem of process variation aware scheduling on homogeneous CMPs, although none in the dark silicon context. In addition, we note that all of the thread scheduling techniques proposed so far focus on either multi-programed workloads where each thread is a separate application [12], [11], or a simple performance model where the application throughput is computed as the sum throughput of each thread [7], [8], [13]. These models are inappropriate for a large class of multi-threaded applications that are based on barrier synchronization. We note that ours is the first paper to solve the problem of variation-aware core selection and thread scheduling for these application classes. Dighe et al. [4] have demonstrated the benefits of variability-aware scheduling on an 80-core hardware prototype, but make use of micro-benchmarks and synthetic workloads.

III. Paper Contributions

Compared to the prior state of the art, in this paper we make the following novel contributions:

- We build a case for *homogeneous* CMPs in the dark silicon regime that *exploit* process variations to improve performance within a power budget, relative to a baseline homogeneous CMP with no dark silicon.
- We propose a simple yet accurate performance model for barrier synchronization based multi-threaded applications and experimentally validate this model using benchmarks from the SPLASH-2 and PARSEC benchmark suites.
- Based on the proposed performance model, we propose a polynomial time algorithm for optimally picking a subset of cores, mapping threads to cores in this subset and assigning operating frequencies to each core in order to maximize performance under a power budget. This is referred to as core cherry-picking.
- Our experimental results using the Sniper multi-core simulation platform highlight the benefits of cherry picking cores in homogeneous dark silicon CMPs — we report up to a 22% increase in performance with 33% dark silicon resources and a 30% performance increase with 50% dark silicon.

IV. Process Variation Models

In this paper, we focus on tiled homogeneous chip multiprocessors consisting of N_{cores} identical tiles. Each tile consists of a processing core, a private L1 cache, a section of the shared L2 cache and an on-chip router to communicate with neighboring cores. As a result of manufacturing process variations, the power dissipation and maximum frequency of each core will be different, depending on the statistics of the process variation distribution. We start by discussing the process variation model used in this paper and the resulting core frequency and power distributions.

To model process variation, the chip surface is modeled as a fine grid of dimensions $N_{chip} \times N_{chip}$. Let $p_{ij}(i, j \in [1, N_{chip}])$ represent the value of the process parameter at grid cell (i, j), for example, the channel gate length of the gates that correspond to that grid cell. In the presence of process variations, p_{ij} can be modeled as a Gaussian random variable with mean μ_p and standard deviation σ_p. In addition, the process parameters at two different grid points are correlated with a correlation coefficient, $\rho_{ij,kl}$, that reduces with increasing distance. Based on the experimentally validated model proposed by [15], we express the spatial correlation between two grid points as

$$\rho_{ij,kl} = e^{-\alpha\sqrt{(i-k)^2+(j-l)^2}} \forall i, j, k, l \in [1, N_{chip}], \qquad (1)$$

where the value of α determines how quickly the spatial correlations die out.

In [2], the authors have shown that frequency of a digital circuit under the impact of process variations can be accurately modeled as the worst-case delay of N_{cp} identical critical paths. Assuming that all the gates in a critical path lie entirely within a grid cell and critical paths are uniformly distributed over the core area, we can write the maximum frequency of core $i(i \in [1, N_{core}])$ as [8]:

$$f_i^{MAX} = K' \min_{k,l \in S_{CP,i}} (\frac{1}{p_{kl}}), \qquad (2)$$

where K' is a technology dependent constant and $S_{CP,i}$ represents the set of N_{CP} grid cells in core i that contain critical paths.

The power consumption of core i depends on its dynamic and leakage power components. Although the direct impact of process variations on dynamic power consumption is negligible, the leakage power consumption depends exponentially on process parameters such as effective gate length. The total power consumption of core i is written as

$$P_i = \sum_{k,l \in S_i} C_{kl}^{sw} V_{DD}^2 f_i + V_{DD} I_{kl}^S e^{-K'' p_{kl}} \quad (3)$$

$$= P_D f_i + P_{L,i} \quad (4)$$

where S_i represents the set of all the grid cells in core i, C_{kl}^{sw} represents the average switched capacitance of a grid cell, V_{DD} is the chip supply voltage, I_{kl}^S is the nominal sub-threshold leakage current for the gates in a grid cell, and K'' is a technology dependent constant. The values of the technology dependent constants can be derived by curve fitting circuit simulation outputs.

V. Performance Modeling for Multi-threaded Applications

We focus on multi-threaded applications from the scientific computing domain, such as those found in the SPLASH-2 and PARSEC benchmark suites. These applications consist of two phases of execution — a sequential phase, which consists of a single thread of execution; and a parallel phase in which multiple threads process data in parallel. The parallel threads of execution in a parallel phase typically synchronize on a barrier, in other words, all threads must finish execution before the application can proceed to the next phase. Therefore, the latency of a parallel phase is dominated by the worst case execution latency across all parallel threads, and the total execution latency is the sum of the latencies of the sequential and parallel phases. Based on this observation, we model the execution time of an application, E, as

$$E = \frac{W_{seq}}{f_{seq}} + \frac{W_{par}}{M \times \min_{i \in [1,M]}(f_{par,i})}, \quad (5)$$

where W_{seq} is the percentage of program execution spent in the sequential phase and W_{par} is the percentage of program execution spent in the parallel phase for an implementation with only one parallel thread. M is the number of parallel threads in the application. Furthermore, f_{seq} and $f_{par,i}$ refer to the frequency at which the sequential and the i^{th} parallel thread are executed, respectively. These values depend on the scheduling of threads to cores in the CMP system, *and* the frequency assigned to each core. Determining the optimal scheduling of threads to cores and the core frequency values is the primary goal of this work. To this end, we assume that each application is pre-characterized in terms of its W_{seq} and W_{par} values.

Figure 2 shows simulation data for the radix and fluid-animate benchmarks that validates the proposed model on a 16-core CMP. In both examples, the sequential thread is mapped to Core 0 and the parallel threads are mapped to Cores 1-15. In each experiment, the frequency of Cores 2-15 is kept constant and the frequency of Core 1 is varied. We observe that when Core 1 is slower than Cores 2-15, it dominates the application execution time. However, when the frequency of Cores 2-15 is slower than Core 1, the application execution time no longer changes with increasing Core 1 frequency. Additional data validating the proposed model will be presented in Section VIII where we discuss our experimental results.

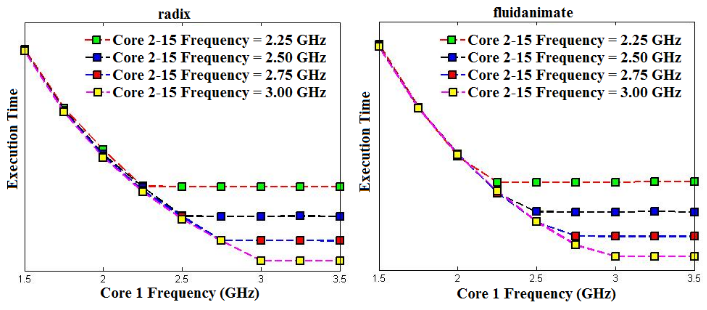

Fig. 2. Execution time for different core frequency values for the radix and fluid-animate benchmarks. The sequential thread is mapped to Core 0 and the parallel threads are mapped to Cores 1 to 15.

VI. Cherry-Picking: Variation-Aware Core Selection and Scheduling

As mentioned before, dark silicon chips consist of more cores than can be simultaneously powered on at any given point in time because of a peak power constraint. Assume that the peak power constraint is given by a designer specified value P_{max}. We will now propose an online optimization strategy to optimally determine for a homogeneous CMP affected by process variations: (a) *which* cores are turned on and which cores are left dark, (b) which core to run the sequential thread and parallel threads on, and (c) what frequency to run every core at. The goal of the online optimization is to maximize performance (minimize application execution latency) under the peak power constraint.

Let $x_{seq}(x_{seq} \in [1, N_{core}])$ represent the core on which the sequential thread executes and, correspondingly, $x_{par,i}(x_{par,i} \in [1, N_{core}])$ be the core on which the i^{th} parallel thread executes. Furthermore, $f_i(i \in [1, N_{cores}])$ represents the frequency at which each core in the CMP executes. To maximize performance within a power budget, we need to solve the following optimization problem.

$$\min_{x,f,M} \frac{W_{seq}}{f_{x_{seq}}} + \frac{W_{par}}{M \times \min_{i \in [1,M]}(f_{x_{par,i}})}, \quad (6)$$

subject to:

$$\sum_{i \in [1,M]} P_{x_{par,i}} = \sum_{i \in [1,M]} P_D f_{x_{par,i}} + P_{L,x_{par,i}} \leq P_{max} \quad (7)$$

$$f_i \leq f_i^{MAX} \, \forall i \in [1, N_{cores}] \quad (8)$$

$$x_{seq} \neq x_{par,i} \, \forall i \in [1, N_{cores}]^{1} \quad (9)$$

[1]special case for parsec benchmark

$$x_{par,i} \neq x_{par,j}\ \forall i,j \in [1, N_{cores}], i \neq j \qquad (10)$$

$$x_{seq} \in [1, N_{cores}] \qquad (11)$$

$$x_{par,i} \in [1, N_{cores}] \qquad (12)$$

Equation 7 represents the dark silicon constraint, i.e., the peak power dissipation during the parallel phase must be less than the maximum power budget. In this formulation, we have assumed for mathematical simplicity that the cores that are off do not dissipate any power (perfect power gating), although a more realistic scenario in which off cores dissipate a fraction of leakage power is modeled in our experimental results. Note that in the dark silicon constraint, the leakage power dissipation differs from one core to another due to process variations. Equation 8 represents the process variation constrained maximum operating frequency of each core. If required, cores can be assigned frequencies lower than their respective maximum frequency constraint to conserve power and meet the maximum power budget. Equation 9 and 10 ensure that only one thread is mapped to each core. The optimization formulation presented here represents a mixed-integer non-linear program (MINLP). We note that Teodorescu et al. [12] have previously proposed a simpler integer linear programming (ILP) based variation-aware scheduling solution. However, their formulation only address multi-programmed and not multi-threaded workloads and does not take into account dark silicon constraints, i.e., they assume that the number of threads is always equal to the number of cores.

We now propose a polynomial time solution to the optimization problem formulated above. Assume that x^* and f^* are the optimal solutions for this problem.

Now, assume that the sequential thread is scheduled for execution on core i, and the *slowest* parallel thread is scheduled on core j. Let the set $Q^{i,j}$ represent the set of cores on the chip (not including i and j) with frequency *greater* than f_j^{max}. The elements of the set Q^{ij} are ordered in ascending order of their leakage power dissipation. With the help of this definition, we can now compute the optimal scheduling and frequency assignment for the remaining $M-1$ parallel threads.

Algorithm 1 is a formal description of the proposed scheduling and frequency assignment algorithm.

Algorithm 1 Optimal core selection, frequency scheduling and frequency assignment

```
1: $E^* \leftarrow \infty$
2: for $i \in [1, N_{cores}]$ do
3:   for $j \in [1, N_{cores}]\ j \neq i$ do
4:     if $|Q^{ij}| \geq M-1$ then
5:       $\Delta \leftarrow P_{max} - MP_D f_j^{max} - L_j - \sum_{k=1}^{M-1} L_{Q_k^{ij}}$
6:       if $\Delta \geq 0$ then
7:         $\theta \leftarrow 1$
8:         $E_{new} \leftarrow \frac{w_{seq}}{f_i} + \frac{w_{par}}{M f_j^{MAX}}$
9:       else
10:        $\theta \leftarrow \frac{\Delta}{MP_D}$
11:        $E_{new} \leftarrow \frac{w_{seq}}{f_i} + \frac{w_{par}}{M\theta f_j^{MAX}}$
12:      end if
13:      if $E_{new} < E^*$ then
14:        $E^* \leftarrow E_{new}$
15:        $x_{seq}^* \leftarrow i$
16:        $x_{par,1}^* \leftarrow j$
17:        $x_{par,k}^* \leftarrow Q_{k-1}^{ij} \forall k \in [2, M]$
18:        $f_k^* \leftarrow 0\ \forall k \in [1, N_{cores}]$
19:        $f_i^* \leftarrow f_i^{MAX}$
20:        $f_j^* \leftarrow \theta f_j^{MAX}$
21:        if $\theta < 1$ then
22:          $f_{Q_k^{ij}}^* \leftarrow \theta f_j^{MAX}\ \forall k \in [1, M-1]$
23:        else
24:          $f_{Q_k^{ij}}^* \leftarrow f_j^{MAX}\ \forall k \in [1, M-1]$
25:        end if
26:      end if
27:    end if
28:  end for
29: end for
30: return $\{x^*, f^*\}$
```

VII. Experimental Methodology

In this paper, we evaluate the proposed ideas for the 22 nm technology node with a nominal $V_{DD} = 1.0V$.

A. Process Variation Parameters

The variability parameters are set as follows. We divide the chip surface into a 100×100 grid and model variations in effective gate length on this grid. The standard deviation of process variations is set to be 10% of the nominal process parameters ($\frac{\sigma_p}{\mu_p} = 0.1$). The spatial correlation parameter α is set such that spatial correlations become negligible ($< 1\%$) at a distance of half die length [15]. The technology specific parameters that model the relationship between the process parameter and delay/leakage (K' and K'') are set based on curve fitting SPICE simulations of ring oscillators in a 22 nm predictive technology model (PTM).

B. Application Evaluated and Performance Modeling

We experiment with five multi-threaded applications from the SPLASH-2 [14] and PARSEC [1] benchmarks suites as shown in Table I. The application parameters W_{seq} and W_{par} are extracted by demarcating the beginning and end of the sequential and parallel sections and measuring their respective execution times on the Sniper full-system multi-core simulator.

C. Micro-architectural Parameters

We model homogeneous CMPs of varying size with $N_{core} = \{16, 24, 32\}$ and in all cases, we set the dark silicon peak power constraint to be $110W$.

Table II shows the architectural configurations of a single tile (core+L2) in the homogeneous CMP architectures that we model. Experiments are run on the **Sniper** multi-core simulator

Application	Description	$W_{par} : W_{seq}$
blackscholes	Financial option pricing	5.59 : 1
fft	Fast Fourier transform	2.90 : 1
fluid-animate	Fluid dynamics	1.18 : 1
radix	Image processing	37 : 1
swaptions	Portfolio pricing	13.36 : 1

TABLE I
APPLICATION DETAILS

Core Parameters	Value
Nominal Frequency	3.0 GHz
Area	10.3 mm^2
Peak Dynamic Power	4.08W
Peak Leakage Power	2.1W
L2 Cache Parameters (per core)	**Value**
Size	2 MB
Area	11.2 mm^2
Peak Dynamic Power	0.76W
Peak Leakage Power	1.56W

TABLE II
MICRO-ARCHITECTURAL DETAILS

[3] which accurately models the core, network and memory sub-system components of a CMP using a trace-driven timing model. The simulator is extended to allow for online binding of threads to cores, in order to validate the proposed scheduling algorithms. Finally, McPAT [10] is used to estimate the area and power consumption of the on-chip components.

Note that the 16-core CMP has a nominal peak power dissipation of $108W$ in the absence of process variations, which is just within the power budget of $110W$. Thus, the 16-core CMP serves as the baseline design in our experiments since it does not have any dark silicon. On the other hand, the 24-core and 32-core CMPs have 33% and 50% dark silicon, respectively.

VIII. EXPERIMENTAL RESULTS

A. Performance Model Validation

We begin by validating the proposed performance model against full-system Sniper simulations which , does not take into account leakage. To do so, we ran 30 simulations with randomly selected frequency values and thread to core mappings for the 16-core, 24-core and 32-core CMPs. The number of parallel threads in the application was set to $M = 16$. Figure 3 shows the scatter plot of predicted versus measured performance for each CMP architecture. The average error over all Monte Carlo runs and across all five applications is only 4.3% while the root mean square (RMS) error is 7.2%.

B. Performance Improvements

We performed experiments using the proposed core selection, thread scheduling and frequency scaling algorithms for a power budget of $110W$ for three different CMP architectures: $N_{core} = 16$ (baseline design, no dark silicon), $N_{core} = 24$ (33% dark silicon) and $N_{core} = 32$ (50% dark silicon). The process variation parameters used are the ones described in the previous section. Note that since in all cases the power budget

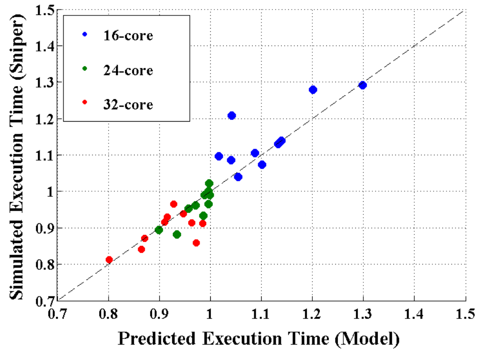

Fig. 3. Accuracy of proposed performance model. Ideally, all points should lie along the dotted line.

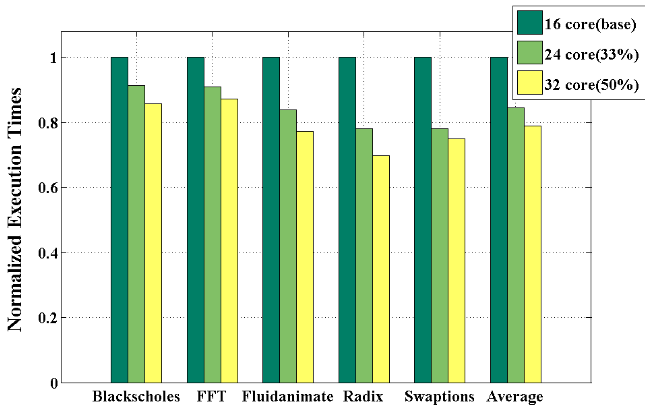

Fig. 4. Average execution time for the 16-core baseline, 24-core and 32-core chips over all runs of Monte Carlo simulations.

remains the same, all benefits in performance come from the ability of dark silicon chips to exploit process variations. For each experiment, we performed 5 runs of Monte Carlo simulation.

As shown in Figure 4, the average performance improvement of the 32-core CMP (50% dark silicon) is 21% and that of the 24-core CMP (33% dark silicon) is 16%. The maximum performance improvement with respect to the 16-core baseline chip is 30% for the 32-core chip (on the radix benchmark) and 22% for the 24-core chip (on the swaptions benchmark). Note that in all these cases, the performance improvement is a result of the greater ability to exploit process variations in the 24- and 32-core chips and not a result of increasing parallelism. The results also that, as expected, indicate that cherry-picking provides more benefits for applications with higher percentage parallelism.

As expected, the execution latency decreases in all cases with increasing power budgets, since higher leakage cores can be selected for execution. It is observed that cherry-picking consistently provides performance benefits over the 16-core baseline over a range of power budget values.

To understand the reason for the performance benefits that arise from cherry-picking cores in dark silicon homogeneous

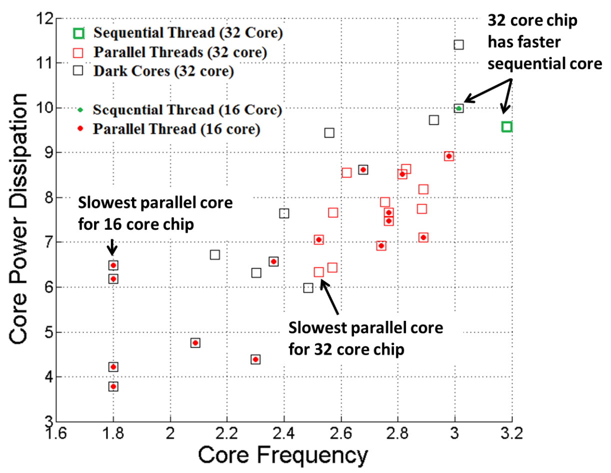

Fig. 5. Power and frequency scatter plots for a 16 core and a 32 core CMP. Also shown are the cores on which the sequential and parallel threads are scheduled and the dark cores for the 32 core CMP case. Note: best viewed in color.

CMPs, we show in Figure 5 a scatter plot of frequency and power values for a 16 core and a 32 core homogeneous CMP. Also shown in the figure are the cores that are picked for sequential and parallel execution and the cores left dark (only for the 32 core CMP) when scheduling blackscholes on these platforms. It can be seen that because there are more cores to pick from and more core-to-core variability in the 32 core CMP case (i) the sequential core picked on the 32 core CMP is 6.4% faster than the sequential core picked on the 16 core CMP; and (ii) the slowest parallel core on the 32 core CMP is 33% faster than the slowest parallel core on the 16 core CMP.

C. *Overhead of Scheduling Algorithm*

Although the proposed core selection, thread scheduling and frequency assignment algorithm is polynomial time, it is still important to characterize the overhead associated with executing this algorithm before application execution. The average execution time (averaged over all experiments) for the 16, 24 and 32 core CMPs is 3.3 , 11.1 and 21.7 milliseconds respctively. Note that the algorithm is executed only *once* before the application is executed and therefore the overhead is negligible if the application execution time is large compared to the time the core selection, thread scheduling and frequency assignment algorithm.

IX. Conclusion and Future Work

In this paper we challenge the conventional wisdom that dark silicon chips must be heterogeneous in nature, and build a case for *homogeneous* CMPs in the dark silicon era. In particular, we observe that process variations introduce core-to-core variations in the power and frequency of cores in homogeneous CMPs and these variations can be exploited to (i) cherry-pick the best subset of cores, and (ii) to optimally map threads to the selected cores, so as to maximize performance under a power budget. To evaluate the benefits of cherry-picking in homogeneous dark silicon CMPs, we have proposed an accurate analytical performance model for barrier synchronization based multi-threaded applications, and a polynomial time algorithm for the joint core selection, thread mapping and frequency assignment problem. Our experimental results using Sniper, detailed multi-core simulator, on applications from the SPLASH-2 and PARSEC benchmark suites show that up to 30% improvement in performance can be obtained for a homogeneous CMP with 50% dark silicon. Our future work will focus on optimal mapping for multiple multi-threaded applications.

References

[1] C. Bienia, S. Kumar, J.P. Singh, and K. Li. The parsec benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 72–81. ACM, 2008.

[2] K.A. Bowman, S.G. Duvall, and J.D. Meindl. Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. *Solid-State Circuits, IEEE Journal of*, 2002.

[3] T.E. Carlson, W. Heirmant, and L. Eeckhout. Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, pages 1–12. IEEE, 2011.

[4] S. Dighe, S.R. Vangal, P. Aseron, S. Kumar, T. Jacob, K.A. Bowman, J. Howard, J. Tschanz, V. Erraguntla, N. Borkar, et al. Within-die variation-aware dynamic-voltage-frequency-scaling with optimal core allocation and thread hopping for the 80-core teraflops processor. *Solid-State Circuits, IEEE Journal of*, 46(1):184–193, 2011.

[5] H. Esmaeilzadeh, E. Blem, R. St Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *Proceeding of the 38th annual international symposium on Computer architecture*, pages 365–376. ACM, 2011.

[6] N. Goulding, J. Sampson, G. Venkatesh, S. Garcia, J. Auricchio, J. Babb, M.B. Taylor, and S. Swanson. Greendroid: A mobile application processor for a future of dark silicon. In *Hot Chips*, 2010.

[7] S. Herbert, S. Garg, and D. Marculescu. Exploiting process variability in voltage/frequency control.

[8] S. Herbert and D. Marculescu. Characterizing chip-multiprocessor variability-tolerance. In *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*, pages 313–318. IEEE, 2008.

[9] U.R. Karpuzcu, B. Greskamp, and J. Torrellas. The bubblewrap many-core: popping cores for sequential acceleration. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 447–458. IEEE, 2009.

[10] S. Li, J.H. Ahn, R.D. Strong, J.B. Brockman, D.M. Tullsen, and N.P. Jouppi. Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 469–480. IEEE, 2009.

[11] D. Shelepov, J.C. Saez Alcaide, S. Jeffery, A. Fedorova, N. Perez, Z.F. Huang, S. Blagodurov, and V. Kumar. Hass: a scheduler for heterogeneous multicore systems. *ACM SIGOPS Operating Systems Review*, 43(2):66–75, 2009.

[12] R. Teodorescu and J. Torrellas. Variation-aware application scheduling and power management for chip multiprocessors. *ACM SIGARCH Computer Architecture News*, 36(3):363–374, 2008.

[13] J.A. Winter and D.H. Albonesi. Scheduling algorithms for unpredictably heterogeneous cmp architectures. In *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*, pages 42–51. IEEE, 2008.

[14] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta. The splash-2 programs: Characterization and methodological considerations. In *ACM SIGARCH Computer Architecture News*, volume 23, pages 24–36. Acm, 1995.

[15] J. Xiong, V. Zolotov, and L. He. Robust extraction of spatial correlation. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(4):619–631, 2007.