HaDeS: Architectural Synthesis for Heterogeneous Dark Silicon Chip Multi-processors

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

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

***Abstract*—In this paper, we propose an efficient iterative optimization based approach for architectural synthesis of dark silicon heterogeneous chip multi-processors (CMPs). The goal is to determine the optimal number of cores of each type to provision the CMP with, such that the area and power budgets are met and the application performance is maximized. We consider general-purpose multi-threaded applications with a varying degree of parallelism (DOP) that can be set at run-time, and propose an accurate analytical model to predict the execution time of such applications on heterogeneous CMPs. Our experimental results illustrate that the synthesized heterogeneous dark silicon CMPs provide between 19% to 60% performance improvements over conventional homogeneous designs for variable and fixed DOP scenarios, 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 [6] where, at any given point in time, only a percentage of transistors on the die are operational. Dark silicon chips are expected to be heterogeneous in nature, consisting of, for example, a multitude of dedicated hardware accelerators to assist the on-chip cores [7] or heterogeneous CMPs consisting of different types of general-purpose cores. In this paper, we address the optimal synthesis of heterogeneous dark silicon CMPs.

The problem of architectural synthesis of heterogeneous dark silicon CMPs can be defined as follows: given (i) a library of general-purpose cores, (ii) a set of multi-threaded benchmark applications, (iii) a peak power budget, and (iv) an area budget; determine the optimal number of cores of each type to provision the heterogeneous dark silicon CMP with, such that the average performance over all benchmarks applications is maximized. Compared to prior work on architectural synthesis for application-specific multi-processor systems [8], [3], architectural synthesis for dark silicon CMPs introduces a number of new challenges and metrics of interest. *First*, accurate analytical models for the execution time of multi-threaded benchmark applications running on heterogeneous cores are not readily available, although these are essential for optimization.

This is in contrast to the application specific-domain where the performance models are well defined, often using formal models of computation such as data-flow graphs [8]. *Second*, typical multi-threaded applications, for example the applications from the SPLASH-2 and PARSEC benchmark suites, can be executed with a variable degree of parallelism (DOP), *i.e.*, a varying number of parallel threads, at run-time. The architectural synthesis algorithm must be aware of the optimal DOP and optimal run-time scheduling of threads to cores for each application. In fact, we show that assuming a fixed DOP for the benchmark applications can result in sub-optimal architectures. *Finally*, from an empirical perspective, an important metric of interest is the performance benefit of heterogeneous versus homogeneous CMP architectures with increasing dark silicon area. With increasing dark silicon area, a greater number of cores specialized for each application can be included. Therefore, the performance benefits of heterogeneous CMPs over homogeneous CMPs should increase with increasing dark silicon – empirical validation of this trend is of immense interest to system designers.

In this paper, we propose Hades – a framework for architectural synthesis of heterogeneous dark-silicon CMPs. The Hades framework consists of the following novel features:

- A new analytical performance model for multi-threaded applications from the SPLASH-2 and PARSEC-2.1 benchmark suites. The proposed performance model is shown to be accurate for applications scheduled on both homogeneous and heterogeneous CMPs, and across a wide range of DOP values for each application.
- An efficient, iterative algorithm to determine the optimal number of cores of each type to provision the heterogeneous dark silicon CMP with. The algorithm takes into account run-time optimization of the DOP and mapping of application threads to cores.
- Comprehensive validation and evaluation of the proposed

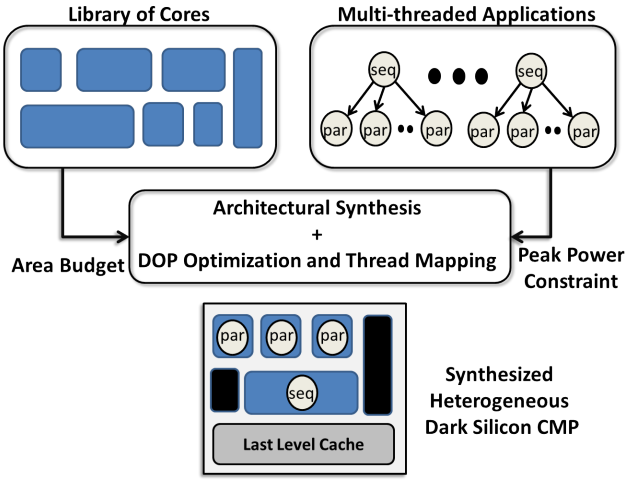

Fig. 1. Overview of the Hades framework. In the synthesized heterogeneous dark silicon CMP, a subset of cores are dark.

models and optimization technique on Sniper [5], a detailed software simulator for multi-core systems. Our results show that heterogeneous dark silicon CMPs provide up to 60% and 19% performance gains over homogeneous dark silicon CMPs with fixed and optimized DOPs, respectively.

To the best of our knowledge, Hades is the first architectural synthesis framework targeted at optimal synthesis of heterogeneous, dark silicon CMP architectures. Figure 1 provides an overview of the Hades framework.

II. Prior Work

General-purpose, heterogeneous CMPs were first proposed by Kumar et al. [9] in the context of single-threaded applications running on CMPs in a multi-programmed fashion. In [10], the authors propose hill-climbing based solutions to for heterogeneous CMP design, but only for multi-programmed worklaods and do not model or evaluate the impact of dark silicon. Lee et al. [11] have also performed similar studies.

The notion of dark silicon was recently introduced by Goulding et al. [7] and Esmailzadeh et al. [6]. Goulding et al. focus on provisioning the dark silicon area with application-specific accelerators while Esmailzadeh et al. focus on using only general-purpose cores. In their work, Esmailzadeh et al. use a highly simplistic performance model based on Amdahl's Law to select cores. In addition, a different architecture is synthesized for each application. In contrast, Hades (i) efficiently explores the entire design space of heterogeneous CMP architectures using a novel iterative optimization strategy, (ii) uses a more realistic performance model that is validated against detailed simulations on both homogeneous and heterogeneous CMPs, and (iii) optimizes over a set of benchmark applications. More recent work on architectural synthesis for dark silicon chips has focused on synthesizing and provisioning application specific accelerators [14] and general-purpose cores [2]. In the latter work, the authors focus only on multi-programmed workloads and use a simulated annealing based heuristic.

Finally, architectural synthesis has been widely studied in the application-specific domain for multi-processor systems-on-chip (MPSoC). Wolf et al. [15] provide a comprehensive account of work in this area. As mentioned before, the architectural synthesis problem for MPSoCs is significantly different because accurate performance models are readily available, the DOP of the application is typically predetermined, and the optimization objective is typically to meet hard or soft timing deadlines. In addition, this problem has not been explored in the dark silicon context.

III. Preliminaries and Assumptions

We begin by discussing the assumptions and mathematical notation relevant to our work.

1) Applications: We assume that we are given a set of N representative benchmark applications. Each application is multi-threaded and consists of both sequential and parallel phases. A single sequential thread executes in the sequential phase, while multiple parallel threads execute in the parallel phase.

The DOP of an application, *i.e.*, the number of threads in the parallel phase can be determined at compile time and is represented as D_i. Without loss of generality, we assume that $D_i \in \mathbb{N}$ and $1 \leq D_i \leq D_{max}$. Note that, in practice, the DOP for some applications can be restricted to a certain subset of values, values that are powers of two, for example.

In this paper, we assume that each multi-threaded application runs independently on the CMP. The run-time scheduler determines the optimal DOP and the optimal mapping of threads to cores to minimize the execution time within a peak power budget, P_{budget}. Each core executes only one application thread.

2) Core Library: We assume a library of M different, general-purpose cores — in other words, each core is capable of executing each application. Each core consists of the micro-architectural pipeline and private instruction and data caches. In the experimental results section, we discuss the micro-architectural parameters, *e.g.*, issue width, cache size *etc.*, that we vary to generate a library of cores.

The peak power dissipation of core j ($j \in [1, M]$) executing one of the parallel threads of application i ($i \in [1, N]$) is denoted by P_{ij}, and includes both the peak dynamic and leakage power components. When a core is idle, *i.e.*, no thread is mapped to it, its leakage power dissipation is denoted by P_j^{idle}. The area of core of type j is A_j, including the pipeline and private caches. Since our goal is to maximize performance, we assume that each core runs at at its highest voltage and frequency level.

3) Uncore Components: In this paper, we focus only on optimizing the number of cores (including their private caches) of each type. We assume that all cores share a single, global last-level cache (LLC) with a uniform access penalty. For fairness, the LLC size is kept constant over all experiments. By the same token, the configuration of other uncore components,

for example, the number and bandwidth of the off-chip DRAM memory controllers, is fixed.

IV. Proposed Hades Framework

Hades enables the efficient exploration of the vast design space of heterogeneous dark silicon chip multi-processor architectures to pick the optimal design that maximizes application performance within area and power budgets. In addition, the optimization is inherently aware of and accounts for run-time decisions including the optimal DOP for each application and mapping of threads to cores — *i.e.*, Hades performs joint design-time and run-time optimization.

We now discuss the two components of the Hades framework: we first discuss the proposed application performance model, and then the iterative optimization used to determine the optimal heterogeneous dark silicon CMP architecture.

A. Application Performance Model

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.

1) Homogeneous CMPs: As the DOP of the application is increased, the execution latency of the parallel phase decreases and is ideally inversely dependent on the DOP. However, due to increased contention for shared hardware and software resources, the benefits of increasing the DOP begin to saturate and, for some applications, the execution time of the parallel phase might actually *increase* beyond a certain DOP. This can be observed in Figure 2(a).

Based on this observation, we propose the following performance model for multi-threaded applications executing on homogeneous cores, and then generalize the model for heterogeneous CMPs. The execution time E_{ij} of an application i ($i \in [1, N]$) running on a homogeneous CMP with cores of type j ($j \in [1, M]$) is expressed as:

$$E_{ij} = t^s_{ij} + \frac{t^p_{ij}}{D_i} + D_i K_{ij} \quad i \in [1, N],\ j \in [1, M] \qquad (1)$$

In this equation, t^s_{ij} is the execution time of the sequential phase, t^p_{ij} is the execution time of the parallelizable part of the parallel phase, and K_{ij} represents the increase in execution latency in the parallel phase because of resource contention.

The model parameters of Equation 1 are learned by executing each application with different DOP values on homogeneous CMPs and using standard regression techniques to minimize the error between the execution times obtained from simulation and from analytical prediction. Therefore, at most $N \times M \times D_{max}$ simulations are required to learn the model parameters. Figure 2(a) shows that the proposed model is, in fact, able to provide very accurate estimates of the actual execution time for varying DOPs.

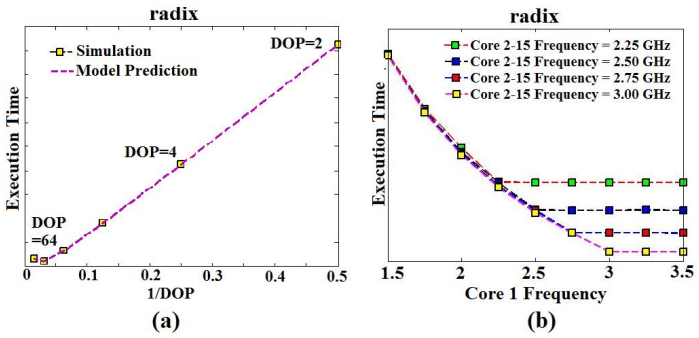

Fig. 2. (a) Execution time for the *radix* benchmark on a homogeneous 64 core CMP with varying DOP. (b) Execution time for the *radix* benchmark with DOP=16 on a heterogeneous CMP. In the experiment, heterogeneity is introduced by varying the core frequency.

2) Heterogeneous CMPs: In a heterogeneous CMP, the execution time of each parallel thread is different. In the parallel phase of execution, threads typically synchronize on a barrier, *i.e.*, all threads must finish execution before the application can proceed to the next phase. Therefore, the latency of a parallel phase is determined by the worst case execution latency across all parallel threads.

This observation is empirically validated in Figure 2(b), again using the *radix* benchmark. Although the application is executed on a homogeneous CMP, heterogeneity is introduced by varying the frequency of one of the parallel cores. Observe that the slowest core does indeed determine the application execution time.

Based on this observation and Equation 1, we propose the following model for the execution time of an application executing on a heterogeneous CMP:

$$E_i = t^s_{im^s} + \max_{v \in [1, D_i]} \left(\frac{t^p_{im^p(v)}}{D_i} + D_i K_{im^p(v)} \right) \qquad (2)$$

In this equation, m^s represents the type of core that the sequential thread executes on, while $m^p(v)$ represents the type of core that the v^{th} parallel thread executes on.

B. Architectural Synthesis

We now formulate the architectural synthesis problem as an integer program. As we will show, the direct formulation that utilizes Equation 2 results in a non-linear, integer program that can be converted, in polynomial time, to an integer linear program (ILP) without loss of optimality. Although powerful commercial-off-the-shelf ILP solvers exist, we observe empirically that for even reasonable problem instances, the computational cost of using an ILP solver is significant. To address this concern, we then propose an iterative optimization procedure that provides orders-of-magnitude speed-up without significant loss in optimality.

1) Non-linear Integer Programming Formulation: We begin by discussing the performance maximization objective function and then incorporate the constraints.

Objective Function: The goal is to minimize the weighted sum of execution time for each benchmark application,

$$\min\left(\sum_{i=1}^{N}\rho_i(l_i^s+l_i^p)\right) \tag{3}$$

where l_i^s and l_i^p are the sequential and parallel execution times for each benchmark. The weight ρ_i is a designer specified constant.

Sequential Execution Time: Let $s_{ij} \in \{0,1\}$ be an indicator variable that is 1 if the sequential thread of application i executes on a core of type j. The sequential execution time of application i can be written as:

$$l_i^s = \sum_{j=1}^{M} s_{ij}t_{ij}^s \quad \forall i \in [1,N] \tag{4}$$

In addition, each application is allowed to use exactly one sequential core. Therefore:

$$\sum_{j=1}^{M} s_{ij} = 1 \quad \forall i \in [1,N] \tag{5}$$

Parallel Execution Time: Let $b_{ij} \in \{0,1\}$ be an indicator variable that is 1 if at least one parallel thread of application i executes on a core of type j. The execution time of the parallel phase is determined by the slowest parallel thread. Therefore:

$$l_i^p \geq b_{ij}\left(\frac{t_{ij}^p}{D_i}+K_{ij}D_i\right) \quad \forall i \in [1,N], j \in [1,M] \tag{6}$$

Note that since the DOP values, D_i, are also variables in the formulation, Equation 6 is a *non-linear* constraint.

Let $r_{ij} \in \mathbb{N}$ be the number of cores of type j used by parallel threads of application i. The DOP of the application must be equal to the total number of parallel cores utilized:

$$\sum_{j=1}^{M} r_{ij} = D_i \quad \forall i \in [1,N] \tag{7}$$

and must be less than the maximum value:

$$1 \leq D_i \leq D_{max} \quad \forall i \in [1,N] \tag{8}$$

Finally, r_{ij} should be zero when b_{ij} is zero and at most D_{max} when $b_{ij} = 1$:

$$r_{ij} - D_{max}b_{ij} \leq 0 \quad \forall i \in [1,N], j \in [1,M] \tag{9}$$

Number of Cores of Each Type: The design vector $Q \in \mathbb{Z}^M$ represents the number of cores of each type: $Q = \{Q_1, Q_2, \ldots, Q_M\}$, and is the ultimate objective of the architectural synthesis problem.

The number of cores of type j should be at least larger than the number of cores of that type used by the sequential and parallel threads of any application. Therefore:

$$Q_j \geq s_{ij} + r_{ij} \quad \forall i \in [1,N] \tag{10}$$

Peak Power Constraint: The peak power dissipation of the dark silicon heterogeneous CMP must be below the peak power budget for each application:

$$\sum_{j=1}^{M} r_{ij}P_{ij} + (Q_j - r_{ij})\,P_j^{idle} \leq P_{budget} \quad \forall i \in [1,N] \tag{11}$$

Dark Silicon Area Constraint: All cores must fit in the prescribed area budget.

$$\sum_{j=1}^{M} Q_j A_j \leq A_{budget} \quad \forall i \in [1,N] \tag{12}$$

The objective function in Equation 3 and the constraints in Equations 4 to 12 represent a non-linear integer optimization problem which we refer to as **NILP-OPT**. Solving the problem yields the optimal number of cores of each type, *i.e.*, the vector Q, *and* the optimal DOP for each application, $D_i, \forall i \in [1,N]$.

2) ILP Formulation: We now show that the non-linear constraint in NILP-OPT, *i.e.*, Equation 6, can be readily linearized to result in a standard ILP problem.

We introduce an indicator variable, $y_{iw} \in \{0,1\}$ ($i \in [1,N], w \in [1,D_{max}]$) that is 1 if and only if application i has an optimal DOP of w. Equation 6 can be re-written as:

$$l_i^p \geq \sum_{w=1}^{D_{max}} b_{ij}y_{iw}\left(\frac{t_{ij}^p}{w}+K_{ij}w\right) = \sum_{w=1}^{D_{max}} m_{ijw}\left(\frac{t_{ij}^p}{w}+K_{ij}w\right) \tag{13}$$

where $m_{ijw} = b_{ij}y_{iw} = \min(b_{ij}, y_{iw})$ and $m_{ijw} \in \{0,1\}$. The following three linear equations express the relationship between m_{ijw}, b_{ij} and y_{iw}:

$$m_{ijw} \leq b_{ij} \;\; \forall i \in [1,N], j \in [1,M], w \in [1,D_{max}] \tag{14}$$

$$m_{ijw} \leq y_{iw} \;\; \forall i \in [1,N], j \in [1,M], w \in [1,D_{max}] \tag{15}$$

$$m_{ijw} \geq b_{ij} + y_{iw} - 1 \;\; \forall i \in [1,N], j \in [1,M], w \in [1,D_{max}] \tag{16}$$

Equations 13 to 16 are the linearized versions of Equation 6. This completes the ILP formulation. We refer to this formulation as **ILP-OPT**.

3) Iterative Optimization: Although ILP-OPT guarantees optimality, we find that empirically the computational time of running the ILP optimization to completion can be significant. To address this issue, we propose an iterative optimization approach, **ITER-OPT**, that separates the architectural optimization from DOP optimization.

ITER-OPT operates as follows: it starts with an initial guess for the optimal architectural design vector Q^*, determines the optimal DOP for each application for this heterogeneous design, re-synthesizes the optimal heterogeneous architecture for this *fixed* DOP, and iterates till convergence, *i.e.*, till no further improvement in performance is observed. The solution to which ITER-OPT converges represents a *local* optima in the design space. Figure 3 as an illustration of how the ITER-OPT algorithm works. We now discuss the two primary components of the ITER-OPT algorithm: (i) the architecture

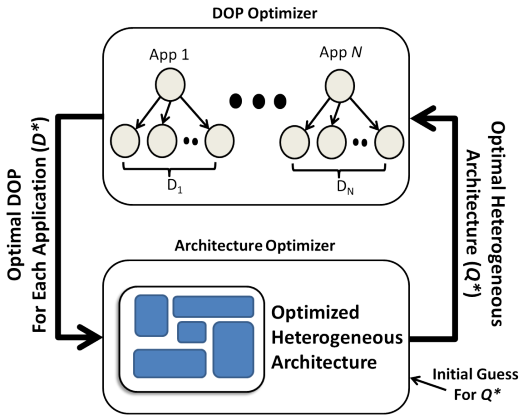

Fig. 3. Illustration of the **ITER-OPT** algorithm.

optimizer and (ii) the DOP optimizer.

Architecture Optimizer: Note that if the DOP of each application is fixed, then Equation 6 is, in fact, linear. Thus NILP-OPT is converted to an ILP problem with only $\mathcal{O}(MN)$ variables, and can therefore be solved significantly faster than ILP-OPT. The architecture optimizer takes a fixed DOP, D^*, vector as input and outputs the optimal heterogeneous architecture Q^* for these input DOPs.

DOP Optimizer: The DOP optimizer (described in Algorithm 1 of Appendix) determines, in polynomial time, the optimal DOP for each application, given a heterogeneous architecture, Q^*, *i.e.*, the number of cores of each type. Intuitively, Algorithm 1 is based on the fact that the execution time of any scheduling of threads to cores depends only on the performance of two cores — the sequential core and the slowest parallel core. Thus, it is sufficient to search over all possible pairs of cores types in the heterogeneous architecture, which is polynomial in the number of core types [13].

V. Experimental Methodology

All our experiments are run on the Sniper [5] multi-core simulator that provides the ability to model heterogeneous core configurations and detailed models for the memory hierarchy.

A library of 108 cores was generated by varying a number of key micro-architectural parameters as shown in TABLE I, yielding a rich design space of area, power, and performance values. These are obtained from the McPAT tool [12] for a 22 nm technology node and a 1.0V nominal supply voltage.

The last-level L3 cache size is set to 32 MB and has a uniform access latency. We model a 1 GB DRAM main memory that is accessed through a DRAM memory controller with aggregate bandwidth of 7.6 GBps.

We experiment with five multi-threaded applications from the SPLASH-2 [16] and PARSEC [4] benchmarks suites — *blackscholes*, *fft*, *fluidanimate*, *radix*, and *swaptions*. The maximum DOP is set to 32 for each application. Simulating these applications on the entire core library reveals that only 15 cores

TABLE I
Micro-architectural Parameters of Core Library

Core Parameter	Values
Dispatch Width	{1,2,4}
ROB Window Size	{8,16,32,64}
L1-I/D Cache size	{64, 128, 256} KB
L2 cache (private)	256 KB
Frequency	{2.5, 3.5, 4.5} GHz

are Pareto optimal in terms of area, power or performance for at least one application. These 15 cores are retained for further experimentation. We have used the Gurobi ILP tool-box [1] to implement both ILP-OPT and ITER-OPT.

VI. Experimental Results

We conduct our first set of experimental results with an area budget $A_{budget} = 180mm^2$ and for different peak power budgets: $P_{budget} = \{40W, 60W, 80W\}$.

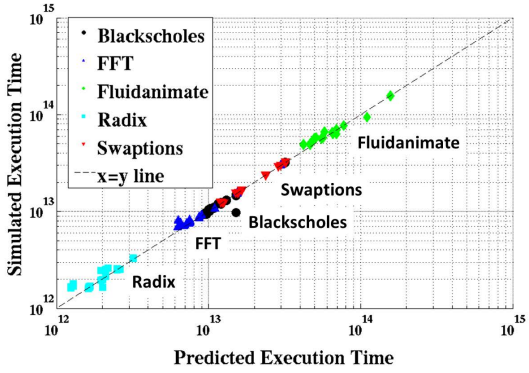

Fig. 4. Validation of proposed analytical performance model against Sniper simulations for both homogeneous and heterogeneous dark silicon CMPs.

A. ITER-OPT vs. ILP-OPT

We begin by noting that the proposed iterative optimization scheme (ITER-OPT) compares favorably with the optimal ILP (ILP-OPT) solution. For $A_{budget} = 180mm^2$ and $P_{budget} = 60W$, ITER-OPT took only 17 seconds to provide a solution within 2.5% of the solution provided by ILP-OPT in 2 hours. This represents a $430\times$ reduction in run-time with only marginal loss in optimality.

B. Performance Model Validation

Figure 4 shows the scatter plot of predicted performance using the proposed model and the simulated performance for a variety of homogeneous and heterogeneous dark silicon CMP architectures that we experimented with. As it can be seen, the performance model agrees very well with simulated values and provides 5.2% error on average over 180 experiments.

C. Heterogeneous Vs. Homogeneous

We synthesized heterogeneous dark silicon CMPs using the Hades framework for the three power budgets mentioned above. These are compared with: (i) a homogeneous dark

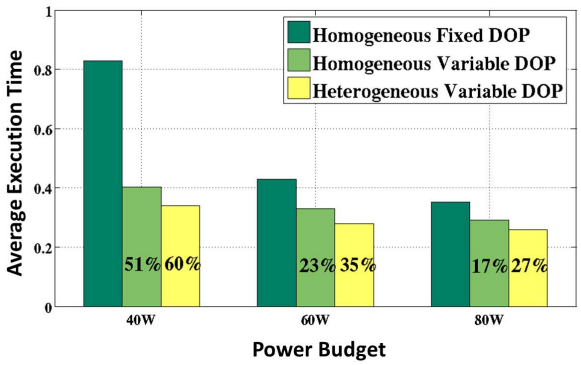

Fig. 5. Average execution time over all benchmarks for different architectural configurations and power budgets.

silicon CMP where every application has a fixed DOP=16 and (ii) a homogeneous dark silicon CMP where the DOP for every application was optimized for the homogeneous architecture. The heterogeneous CMP always uses DOPs optimized for its architecture.

For power budgets of 40W, 60W and 80W, the heterogeneous CMP has 60%, 35% and 27% higher performance than a homogeneous CMP with fixed DOPs, respectively. Assuming a homogeneous CMP with variable DOPs, the heterogeneous CMP still has 19%, 16% and 12% higher performance. Low power budgets correspond to more dark silicon, and we observe that the benefits of heterogeneity are, as expected, greater with increasing dark silicon.

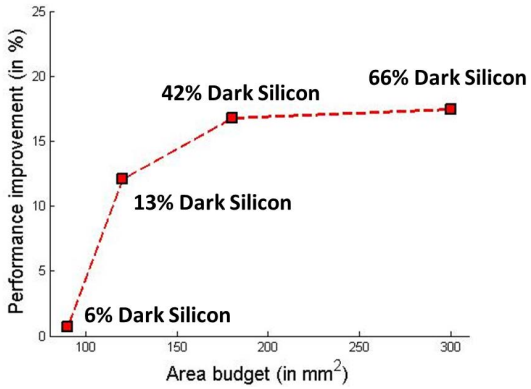

Fig. 6. Performance improvement of a heterogeneous dark silicon CMP with increasing area budgets and increasing proportion of dark silicon transistors.

D. Performance Benefits with Increasing Dark Silicon

We plot in Figure 6 the performance improvement of heterogeneous versus homogeneous for increasing area budgets, from $90mm^2$ to $300mm^2$. Also shown is the $\%$ dark silicon for each area budget. It is evident that the benefits of heterogeneity saturate after a certain point.

VII. Conclusion

In this paper we have proposed Hades, a framework for optimal architectural synthesis of heterogeneous dark silicon CMPs. As part of Hades, we propose a new, analytical performance model for general-purpose multi-threaded applications running on heterogeneous platforms, and an iterative optimization algorithm, ITER-OPT, that determines the optimal number of cores of each type to provision the heterogeneous dark silicon CMP with. ITER-OPT takes into account the optimal DOP of each application while synthesizing the heterogeneous architecture. As future work, we plan to incorporate application specific accelerators in the Hades framework.

References

[1] Gurobi optimizer (www.gurobi.com).

[2] J. Allred, S. Roy, and K. Chakraborty. Designing for dark silicon: a methodological perspective on energy efficient systems. In *Proceedings of the 2012 ACM/IEEE ISLPED*, 2012.

[3] F. Angiolini, J. Ceng, R. Leupers, F. Ferrari, C. Ferri, and L. Benini. An integrated open framework for heterogeneous mpsoc design space exploration. In *DATE'06. Proceedings*, 2006.

[4] 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*, 2008.

[5] 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.

[6] H. Esmaeilzadeh, E. Blem, R.S. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*, pages 365–376. IEEE, 2011.

[7] 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.

[8] A. Kumar, B. Mesman, B. Theelen, H. Corporaal, and Y. Ha. Analyzing composability of applications on mpsoc platforms. *Journal of Systems Architecture*, pages 369–383, 2008.

[9] R. Kumar, K.I. Farkas, N.P. Jouppi, P. Ranganathan, and D.M. Tullsen. Single-isa heterogeneous multi-core architectures: The potential for processor power reduction. In *Microarchitecture, 2003. MICRO-36. Proceedings.* IEEE, 2003.

[10] R. Kumar, D.M. Tullsen, and N.P. Jouppi. Core architecture optimization for heterogeneous chip multiprocessors. In *Proceedings of the 15th international conference on Parallel architectures and compilation techniques*, pages 23–32. ACM, 2006.

[11] B.C. Lee and D.M. Brooks. Illustrative design space studies with microarchitectural regression models. In *High Performance Computer Architecture, 2007. HPCA 2007. IEEE 13th International Symposium on*, pages 340–351. IEEE, 2007.

[12] 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*. IEEE, 2009.

[13] B. Raghunathan, Y. Turakhia, S. Garg, and D. Marculescu. Cherry-picking: Exploiting process variations in dark-silicon homogeneous chip multi-processors. In *Proceedings of the Design Automation and Test in Europe Conference*. EDAA, 2013.

[14] G. Venkatesh, J. Sampson, N. Goulding-Hotta, S.K. Venkata, M.B. Taylor, and S. Swanson. Qscores: trading dark silicon for scalable energy efficiency with quasi-specific cores. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 163–174. ACM, 2011.

[15] W. Wolf, A.A. Jerraya, and G. Martin. Multiprocessor system-on-chip (mpsoc) technology. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, pages 1701–1713, 2008.

[16] 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*, pages 24–36, 1995.

APPENDIX

We note that ITER-OPT is guaranteed to converge since: (a) the objective function, *i.e.*, the application execution time, always decreases in each iteration; and (b) the globally optimal execution time is bounded. As an initial guess for Q^*, the available area budget is divided equally between all core types.

Algorithm 1 Optimal DOP for an application $i \in [1, N]$ given heterogeneous architecture Q^*

1: $L \leftarrow$ List of cores in ascending order of power dissipation
2: $t^* \leftarrow \infty \;\; D_i^* \leftarrow 1$
3: **for** $d \in [1, D_{max}]$ **do**
4: **for** $j \in [1, M]$ **do**
5: $Q^{used} \leftarrow Q^*$
6: **if** $Q_j^{used} \geq 1$ **then**
7: $t^{seq} \leftarrow t_{ij}^s$
8: $Q_j^{used} \leftarrow Q_j^{used} - 1$
9: **else**
10: $t^{seq} \leftarrow \infty$
11: **end if**
12: **for** $k \in [1, M]$ **do**
13: **if** $Q_k^{used} \geq 1$ **then**
14: $t^{par} \leftarrow \frac{t_{ik}^p}{d} + K_{ik}d$
15: $Q_k^{used} \leftarrow Q_k^{used} - 1$
16: $P_{used} \leftarrow P_{used} + P_{ik}$
17: **else**
18: $t^{par} \leftarrow \infty$
19: **end if**
20: $req \leftarrow d - 1$
21: **for** $l \in [1, M]$ **do**
22: $c \leftarrow L_l$ /* l^{th} element of list L */
23: **if** $\frac{t_{il}^p}{d} + K_{il}d \leq \frac{t_{ik}^p}{d} + K_{ik}d$ **then**
24: $add \leftarrow \min(Q_c^{used}, req)$
25: $req \leftarrow req - add$
26: $P_{used} \leftarrow P_{used} + add \times P_c^{ik}$
27: **if** $P_{used} > P_{budget}$ **or** $req = 0$ **then**
28: Go to line 37
29: **end if**
30: **end if**
31: **end for**
32: **if** $req = 0$ **then**
33: $t_{jk} \leftarrow t^{seq} + t^{par}$
34: **end if**
35: **end for**
36: **end for**
37: **if** $\min_{jk}(t_{jk}) \leq t^*$ **then**
38: $t^* \leftarrow \min_{jk}(t_{jk}) \;\; D_i^* \leftarrow d$
39: **end if**
40: **end for**
41: **return** D_i^*