Power-Efficient Load-Balancing on Heterogeneous Computing Platforms

Muhammad Usman Karim Khan[1], Muhammad Shafique[2], Apratim Gupta[3], Thomas Schumann[3], Jörg Henkel[2]
[1]IBM Research and Development Lab, Böblingen, Germany, (mkhan@de.ibm.com)
[2]Chair for Embedded Systems, Karlsruhe Institute of Technology, Germany, ({muhammad.shafique, henkel}@kit.edu)
[3]University of Applied Sciences, Darmstadt, Germany, (apratimgupta@gmail.com, thomas.schumann@h-da.de)

***Abstract*— In order to address the throughput constraints of the system at minimal power consumption, the workload of computing nodes should be balanced. This requires accounting for the underlying hardware characteristics and throughput sustainable by these nodes. This work provides a workload distribution and balancing methodology of a divisible load under a throughput constraint, on heterogeneous nodes. The power efficiency of each node is considered during load distribution. For load balancing, the frequency of the node is determined which just fulfills the timing requirement. Compared to a state-of-the-art-scheme, our scheme results in up to 64% performance improvement for the benchmarks evaluated in this paper.**

***Keywords*— Heterogeneous Computing, Power-Efficiency, Load Balancing, Hardware Accelerator, DVFS.**

I. Introduction and Related Work

High throughput demands of compute intensive applications must be met within a tight quality-of-service constraints. For example, video processors usually have a frames processed per second (fps) requirement. These requirements can be sustained by employing high-power compute devices. However, challenges like power-wall [1] limit the applicability of these systems, as high power consumption gives rise to multiple issues related to temperature and reliability. Therefore, power-efficiency is one of the prime design metrics for modern on-chip systems. Furthermore, the computational load of an application must be balanced among the computing nodes such that these nodes are fully utilized [2]. This results in maximizing the application's throughput, or in other words, maximizing the power-efficiency. In short, the performance (or throughput)-per-watt metric needs to be maximized.

In order to maximally utilize the hardware, the workload of the application must be equally and fairly distributed among the nodes. Every computing node should ideally finish its assigned workload at the same time (i.e., the workload needs to be balanced). For homogeneous system with many nodes, if the number of tasks is considerably larger than the number of nodes, it is easier to balance the workload [3] as more freedom is available for task assignment. If not, then the voltage-frequency levels of the nodes are adjusted to balance the execution time. Now, heterogeneous computing platforms have gained interest both in industry and academia (e.g., ARM's big.LITTLE, GreenDroid, Invasive Computing [4]), as they can provide higher energy efficiency and tackle power-wall related issues [5]. However, for heterogeneous platforms, some nodes are faster and more power-hungry than the others. Therefore, the hardware of the faster nodes might be underutilized, resulting in a reduced throughput-per-watt.

The entire work presented in this paper was conducted when the first author (M. U. K. Khan) was still with the Chair for Embedded System (CES) at Karlsruhe Institute of Technology (KIT).

For power efficiency, either the load of an application can be distributed on a given platform (load balancing [6]), or, a platform can be synthesized for the given load (load driven synthesis [7]) under throughput and/or power constraints. Load balancing techniques usually target homogenous systems under constant total load and do not consider load variation at runtime [8]. Research has focused on combining the distribution and balancing of load, and Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Manage (DPM) of the cores [9]. For example, [10] independently determine the frequency of each core while distributing and balancing application load.

To increase the throughput-per-watt under modern system design challenges, heterogeneous multi-/many-core systems are becoming popular [11]. Some examples of heterogeneous systems are current microprocessor working in conjunction with GPUs [12], hardware accelerators [13] etc. In [11], authors target energy efficient workload allocation and voltage-frequency tuning of the underlying single-ISA processing elements. The goal is to minimize energy/power of the system. However, their approach does not consider the case if the throughput of the application(s) cannot be met. In [14], authors propose to identify the program's and cores' characteristics and then appropriately match them for scheduling. Ref. [15] studies parallelized database on heterogeneous, single-ISA architectures. These approaches usually neglect the load distribution and assume that tasks with particular attributes are available to their approaches. Further, most of these approaches are concerned with either maximizing the throughput or reducing the power consumption, and usually do not consider the resource allocation along with power-efficiency.

Thus, an apt load distribution and balancing scheme is required to address the challenges related to heterogeneous systems. Furthermore, the varying application characteristics and complexity variation of multiple threads of the same application can also imbalance the workload. Similarly, the distribution of load among compute nodes needs to consider the throughput sustainable by the compute nodes. For example, a hardware accelerator might provide high power efficiency and thus, more workload needs to be allocated to the hardware accelerator than the nodes processing purely the software code. Moreover, the maximum supportable frequencies of the nodes, voltage-frequency settings and varying thread workload increases the complexity of the load- balancing problem.

A. Our Novel Contributions

This work presents *a load distributing and balancing methodology on computing nodes of heterogeneous systems, such that the power-efficiency of the system is maximized*. The proposed methodology considers the supportable frequencies, compute characteristics and power consumption of the nodes. This methodology considers the throughput requirement of the application by accounting for the given processing deadlines. Our scheme distributes an application's load such that the

number of utilized nodes is minimized, and highest efficiency node gets the largest share of the load. Three different efficiency-metrics are tested, based upon which the tasks are distributed among the nodes and their voltage-frequency levels are adapted. A power optimization problem is derived, and a heuristic method is provided for solving the problem. In summary, we propose a workload distribution and balancing methodology for heterogeneous systems, comprising of compute nodes with varying power and compute characteristics.

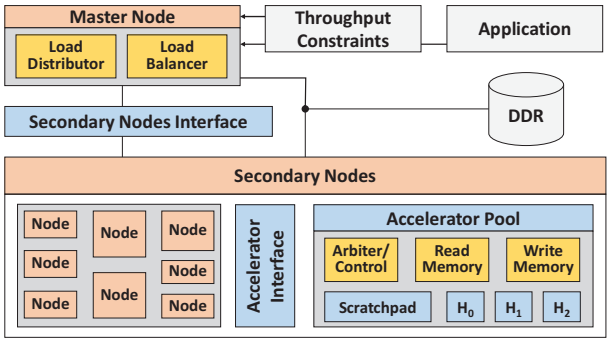

Fig. 1: Overview of our novel contributions.

The overview of our proposed system is shown in Fig. 1. As seen, a node called Master Node is responsible for distributing the workload among secondary nodes. The secondary nodes comprise of different computing nodes (even hardware accelerators). The hardware accelerator consists of different custom logic elements, like Read/Write Memory modules for communicating with the external memory. Further, in the paper, a *scalar* is denoted by a small italic letter, while a ***vector*** is given as a bold italic letter.

II. Load-Balancing on Heterogeneous System

A. System Model

Consider an application that is scheduled to run on a heterogeneous system with multiple nodes. The application consists of multiple independent jobs that can be subdivided and executed in parallel. The number of jobs that must be processed within a deadline (t_{max}) equals n_{tot}.

The heterogeneous compute system consists of r_{tot} total nodes. Our scheme selects only k_{tot} nodes for sustaining the throughput of the application ($k_{tot} \leq r_{tot}$), and selects their voltage and frequencies (f_i for node i), which determines the power consumed by these nodes (p_i). Basically, the frequency of the node is a function of the number of jobs allocated to the node (n_i) and the cycles consumed to process a job on node i (c_i) as will be discussed later in Section II.B. Moreover, the compute nodes can either be soft-cores, GPUs, accelerators etc. We make no assumption about the sustainable throughput and power consumed by these nodes. Each node can sustain a different throughput then the rest, and can consume different power for equivalent throughput.

In this paper, we target to minimize the total power consumed by the heterogeneous system (p_{tot}) while meeting a certain throughput constraint. The **optimization goal** can, therefore, be written as shown in Equation 1. This problem suggests that a node can either be power gated (with $f_i = 0$) or its frequency range can be between permissible limits. However, each parallel computing node must finish the tasks within t_{max}, while the cumulative number of tasks processed should equal the total number of tasks.

$$\min\left(p_{tot} = \sum_{i=0}^{rtot-1} p_i(f_i)\right) \qquad s.t.$$
$$f_i \in \left\{f \mid 0 \cup \left\{f_{\min} \leq f \leq f_{\max}\right\}\right\} \quad \forall i \in \{0,\cdots,r_{tot}\} \qquad (1)$$
$$t_i \leq t_{\max} \qquad \forall i \in \{0,\cdots,k_{tot}\}$$
$$\sum_{i=0}^{ktot-1} n_i = n_{tot}$$

In order to solve the proposed load distribution and balancing problem, some issues need to be addressed. First, determining the k_{tot} (the actual number of nodes used for processing) is not known beforehand. Secondly, a node frequency is permissible to have values within two distinct ranges (i.e., either it can be 0 or between f_{min} and f_{max}). Moreover, the variables used in Equation 1 need to be derived in terms of tunable system parameters for optimization, which might be cumbersome. Thus, to efficiency solve these issues, we use a heuristic, as explained below.

B. Load Balancing Algorithm

The load balancing algorithm is initiated in the master node of the heterogeneous system (see Fig. 1). This algorithm is basically consists of two major steps:

1- Gathering the compute and power profiles of all nodes before starting the distribution of jobs among the nodes.
2- Actual load distribution, which distributes the job among the nodes, depending upon the metrics gathered by the first step. The compute and power profiles are also used for selecting the appropriate running frequency of the nodes.

For the first step, the power profiles (power vs. frequency of the node, $\boldsymbol{p}(\boldsymbol{f})$) of all the nodes is collected. This can be done using online power measurement (e.g., using Intel Power Gadget and reading MSR registers), or, by exploring the data-sheets of the nodes. Furthermore, the average power over the available frequency range or operation modes ($\boldsymbol{p_a}$) is also computed. Also, the maximum average power for all nodes ($p_{a,max}$) is determined via:

$$p_{a,\max} = \max\{\boldsymbol{p_a}\}, \quad \forall r_{tot} \text{ nodes} \qquad (2)$$

The cycles consumed to process a job on a node i (given by c_i) is collected for all the nodes either offline (regression analysis) or online [16]. Similarly, the maximum cycles consumed by a node to process a job (c_{max}) is also determined. Using these metrics, the *efficiency index* for a node i (μ_i) is computed for the use cases shown by the following equations:

$$\mu_{i,1} = \frac{c_{\max} p_{a,\max}}{c_i p_{i,a}} \qquad (3)$$

$$\mu_{i,2} = \frac{c_{\max}}{c_i} \qquad (4)$$

$$\mu_{i,3} = \frac{p_{a,\max}}{p_{i,a}} \qquad (5)$$

Index 1 shows that the node which takes the least cycles and power to process a job has higher computation efficiency (throughput-per-watt). Therefore, it must be preferred for use if

LoadBalancing ( ):
Input: Frame processing deadline t_{max}, Number of subtasks per frame n_{tot}; Number of available nodes r_{tot}; Power-frequency/configuration relationship of all nodes $p(f)$; Efficiency indices of all nodes $\boldsymbol{\mu}$; Cycles for a subtask per node $\boldsymbol{c}$;
Output: Total number of used nodes k_{tot}; Frequency of nodes $\boldsymbol{f}$;

```
1.  g_μ ← SortNodesDesc(μ);              // Sort nodes according to μ
2.  k_tot ← 1;i ← 0;
3.  LoopIf(k_tot ≤ r_tot){
4.    ∀i∈{0 to k_tot-1 nodes}{
5.      f_i ← f_i,min;                   // Start with min. frequency
6.      n_i ← DistLoad(n_tot, μ);}       // Equation 7
7.    ThrptMet ← true;                   // Assume throughput is met
8.    ∀i∈{0 to k_tot-1 nodes}{
9.      t_i ← TimeOnNode(c_i, n_i, f_i); // Equation 6
10.     NodeThrptMet ← true; // Assume node throughput met
11.     if(t_i > t_max){                 // Throughput not satisfied
12.       NodeThrptMet ← false;
13.       ∀fj∈{fi,min to fi,max}{
14.         t_i ← TimeOnNode(c_i, n_i, f_j); // Equation 6
15.         if(t_i > t_max){NodeThrptMet ← true;break;}}
16.       if(NodeThrptMet = false){ThrptMet ← false;}}
17.   } // All k_tot nodes tested with every possible frequency
18.   if(ThrptMet = false){              // Throughput not satisfied
19.     k_tot ← IncrementNode(g_μ);      // Introduce additional node
20.   else{break;}                       // Configuration met, break
21. }
```

Fig. 2: Determining number of nodes (k_{tot}) and their frequencies ($\boldsymbol{f}$)

a higher throughput-per-watt metric is desired. Similar argumentation can be made for Indices 2 and 3. Note that the efficiency index can also be changed to account for other metrics (like network hops among the nodes etc.).

For the second step, the actual load distribution algorithm takes place as shown in Fig. 2. Basically, nodes are adaptively introduced to share the load of the application, and then their frequencies are tuned such that the system consumes minimum amount of power. At start, the nodes are ordered in a list ($\boldsymbol{g_\mu}$) with descending efficiency indices (line 1). Every time a new node is required to distribute the jobs, the next node from this list is considered (line 19). We start with only one node (having the highest efficiency index) and with its minimum frequency setting ($f_i = f_{i,min}$, line 5). That is, the entire load is allocated to this node (i.e., $n_i = n_{tot}$). Then, the time consumed by a node to process the workload is given by the formula:

$$t_i = \frac{c_i n_i}{f_i} \tag{6}$$

In case the timing constraints are not met ($t_i > t_{max}$), the frequency of this node is increased by a single step until $f_i = f_{i,max}$. If $f = f_{max}$, then frequency cannot be increased further, and more nodes are introduced ($k_{tot} > 1$). The next node is fetched from the efficiency index array $\boldsymbol{g_\mu}$. Now, the algorithm starts again with minimum power configuration ($f_i = f_{i,min}$) for all k_{tot} nodes. The load is distributed among the nodes using:

$$n_i = \frac{n_{tot}\mu_i}{\sum_{j=0}^{ktot-1} \mu_j} \tag{7}$$

That is, the node with the highest efficiency factor gets more load. Once again, the time that will be consumed by a node while processing n_i jobs can be estimated using Equation 6. In case a node is unable to process the allocated number of jobs within t_{max}, its frequency is increased. If any node is at its maximum power capacity and the system still cannot sustain the workload, another computation node is introduced and the process is repeated. If all the nodes meet their deadlines, the algorithm terminates and the configuration which can sustain the application's load is found. Further, we define the maximum time taken by any node as the time to process a frame t_f:

$$t_f = \max\{t_i\} \qquad \forall i \in \{0, \cdots, k_{tot} - 1\} \tag{8}$$

Table I: Attributes of cores and benchmarks, used for evaluation.

Core	Attributes	Area [mm²]	Average cycles per operation		
			DCT	Quant	HEVC
Tiny	$L2=64 Dispatch=1	86.76	4594	3416	100×10^6
Medium	$L2=256 Dispatch=2	89.99	2378	1658	59.0×10^6
Large	$L2=512 Dispatch=4	95.26	1687	971	45.3×10^6

III. Experimental Evaluation

A. Experimental Setup

For accessing the performance of the proposed load distribution and balancing approach on a heterogeneous system, we use different compute cores and benchmarks, as given in Table I. These numbers are obtained for 45nm x86 cores, with 32 KB L1 data and instruction caches via Sniper simulator [17]. The average cycles per operation are obtained by running the benchmarks for the set of frequencies given in Fig. 3. The variance of the cycles per benchmark is negligible, and therefore, the average numbers are reported and used for evaluation. Fig. 3 also provides the power profiles of the cores used in this work. For the "Quant" benchmark, although the number of cycles consumed by the "Large" core is ~3.5× lesser than the "Tiny" core, the power consumption of the "Large" core at the lowest frequency (i.e., 1000 MHz) is greater than the power of the "Tiny" core at maximum frequency (3400 MHz).

B. Results and Discussion

For evaluation, a multi-core heterogeneous system with four cores (i.e., $r_{tot} = 4$ with two "Tiny", one "Medium" and one "Large" cores) is tested. A throughput constraint of t_{max} = 30msec is set. The quality metric is defined as throughput-per-watt and is given by the relation $1/(t_f \times p_{tot})$. The average throughput-per-watt for multiple runs of the benchmarks, with varying n_{tot} is given in Fig. 4. As noticed, for these benchmarks, $\mu_{i,1}$ and $\mu_{i,2}$ outperform the state-of-the-art load balancing scheme [11] which uses a bin-packing heuristic to distribute the load among cores. However, the efficiency index $\mu_{i,3}$ does not perform as well as the other indices. Hence, this shows that only power-aware load distribution will not result in high performance.

For the "Quant" benchmark, Fig. 5 shows the power, time (i.e., throughput) and number of cores actually used for processing (k_{tot}). The number of quantization operations performed for different image resolutions are also presented in this figure. As seen, for efficiency index $\mu_{i,1}$ and specially for $\mu_{i,2}$, the load distribution is not distributed only based upon the power of the node. Index $\mu_{i,1}$ uses the combination of power and cycles consumed in processing the load, while $\mu_{i,2}$ uses only the

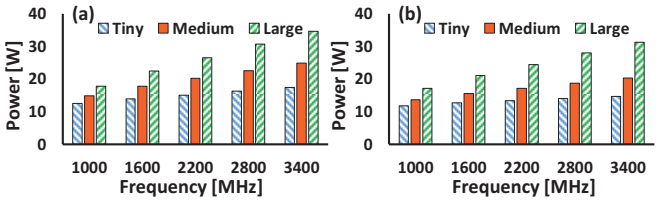

Fig. 3: Power consumption profiles of different cores given in Table I for (a) DCT and (b) "Quant" benchmarks. The power consumed by the cores is almost identical for both benchmarks and dependent upon the frequency.

number of cycles. Therefore, these two indices and [11] always result in throughput being satisfied ($t_i \leq t_{max}$ = 30 msec) as shown in Fig. 5 (b). However, $\mu_{i,3}$ starts to miss the deadline once the load on the system increases considerably. Moreover, as shown in Fig. 5 (c), the number of cores used for processing by $\mu_{i,1}$ and $\mu_{i,2}$ are also lesser than $\mu_{i,3}$ and [11].

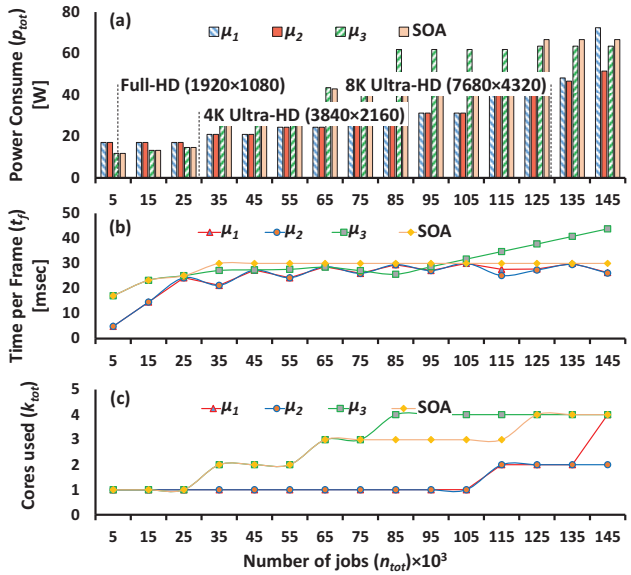

Fig. 5: For the "Quant" benchmark, (a) power consumption, (b) time per frame and (c) number of cores used is given for different n_{tot}, using our proposed efficiency indices and State-of-the-Art (SOA) [11] approach.

IV. Conclusion

This work proposes a methodology for load balancing among heterogeneous nodes, by evaluating different efficiency indices pertaining to the combination of power consumed and cycles per operation. For this system, we derive an optimization goal in order to increase the throughput per unit of power, and solve it heuristically. Our approach not only selects the cores to use, but also determines their voltage-frequency levels to sustain the throughput requirement. Compared to the state-of-the-art load balancing approach [11], our approach results in up to 64% increase in performance for the presented benchmarks.

Acknowledgement

This work was partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre "Invasive Computing" (SFB/TR 89); http://invasic.de

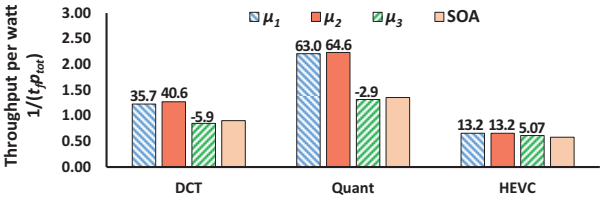

Fig. 4: Average throughput per watt $1/(t_f \times p_{tot})$ for different efficiency indices and State-of-the-Art (SOA) [10]. The percentage improvement against SOA is written on top of the bars.

References

[1] H. Esmaeilzadeh, E. Blem, R. Amant, K. Sankaralingam and D. Burger, "Dark silicon and the end of multicore scaling," in *International Symposium on Computer Architecture*, 2011.

[2] I. Ahmad and A. Ghafoor, "Semi-distributed load balancing for massively parallel multicomputer systems," *IEEE Transactions on Software Engineering,* vol. 17, no. 10, pp. 987-1004, 1991.

[3] E. Cesar, A. Moreno, J. Sorribes and E. Luque, "Modeling Master/Worker applications for automatic performance tuning," *Parallel Computing,* vol. 32, no. 7, pp. 568-589, 2006.

[4] J. Henkel, A. Herkersdorf, L. Bauer, T. Wild, M. Hubner, R. Pujari, A. Grudnitsky, J. Heisswolf, A. Zaib, B. Vogel, V. Lari and S. Kobbe, "Invasive manycore architectures," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2012.

[5] K. Van Craeynest and L. Eeckhout, "Understanding Fundamental Design Choices in single-ISA Heterogeneous Multicore Architectures," *ACM Transactions on Architecture and Code Optimization (TACO),* vol. 9, no. 4, pp. 1-23, 2013.

[6] T. G. Robertazzi, "Ten Reasons to Use Divisible Load Theory," *Computer,* vol. 36, no. 5, pp. 63-68, 2003.

[7] Y. Turakhia, B. Raghunathan, S. Garg and D. Marculescu, "HaDeS: Architectural synthesis for heterogeneous dark silicon chip multi-processors," in *Design Automation Conference (DAC)*, 2013.

[8] S. Shah and R. Kothari, "Convergence of the dynamic load balancing problem to Nash equilibrium using distributed local interactions," *Information Sciences,* vol. 221, pp. 297-305, 2013.

[9] J. Kim, S. Yoo and C.-M. Kyung, "Program Phase-Aware Dynamic Voltage Scaling Under Variable Computational Workload and Memory Stall Environment," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 30, no. 1, pp. 110-123, 2011.

[10] M. U. K. Khan, M. Shafique and J. Henkel, "Software architecture of High Efficiency Video Coding for many-core systems with power-efficient workload balancing," in *Design, Automation and Test in Europe*, 2014.

[11] A. Colin, A. Kandhalu and R. Rajkumar, "Energy-Efficient Allocation of Real-Time Applications onto Single-ISA Heterogeneous Multi-Core Processors," *Journal of Signal Processing Systems,* pp. 1-20, 2015.

[12] W. Xiao, B. Li, J. Xu, G. Shi and F. Wu, "HEVC Encoding Optimization Using Multi-core CPUs and GPUs," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT),* vol. pp, no. 99, pp. 1-14, 2015.

[13] M. U. K. Khan, M. Shafique and J. Henkel, "Power-efficient Accelerator Allocation in Adaptive Dark Silicon Many-core Systems," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015.

[14] C. Jian and L. John, "Efficient program scheduling for heterogeneous multi-core processors," in *Design Automation Conference (DAC)*, 2009.

[15] T. Mühlbauer, W. Rödiger, R. Seilbeck, A. Kemper and T. Neumann, "Heterogeneity-conscious Parallel Query Execution: Getting a Better Mileage While Driving Faster!," in *International Workshop on Data Management on New Hardware*, 2014.

[16] M. U. K. Khan, M. Shafique and J. Henkel, "Power-Efficient Workload Balancing of Video Applications," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI),* 2016.

[17] T. Carlson, W. Heirman 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*, 2011.

[18] S. Li, J. H. Ahn, R. Strong, J. Brockman, D. Tullsen and N. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Microarchitecture*, 2009.