Optimizing Power of Many-Core Systems by Exploiting Dynamic Voltage, Frequency and Core Scaling

Bin Liu, Mohammad H. Foroozannejad, Soheil Ghiasi and Bevan M. Baas
Department of Electrical and Computer Engineering
University of California, Davis

***Abstract*—To address the well-known "power wall" issue, many-core processors with dynamic voltage and frequency scaling (DVFS) are widely investigated. To further improve the energy efficiency, DVFS with core scaling (DVFCS) has been proposed. In this paper, we address the problem of minimizing the power dissipation of many-core systems under performance constraints by choosing appropriate number of active cores and per-core voltage/frequency levels. A genetic algorithm based solution is proposed to solve the problem. Experiments with real applications show that (1) dynamically scaling the number of active cores can save up to 72% power compared with per-core DVFS; (2) the amount of extra power saving brought by core scaling is highly dependent on performance constraints.**

***Index Terms*—Many-core processors, dynamic voltage, frequency and core scaling (DVFCS), genetic algorithm (GA), globally asynchronous locally synchronous (GALS).**

I. Introduction

For the past half century, Moore's Law has been the fundamental driver of high-performance computing. The continued CMOS technology scaling doubles the transistor density of VLSI systems and also had provided a predictable linear performance improvement of single-core processors for every 18 to 24 months. However, as the threshold voltage stops scaling along with the lithographic dimensions of transistors, the era of scaling frequency and performance without increasing power density is over. Since 2005, the semiconductor industry shifted to multi-core and many-core processors to efficiently utilize the tremendous number of transistors. Many-core processors with network-on-chip interconnects have been demonstrated as promising architectures for high performance energy-efficient computing [1], [2]. As the technology shrinks, a single chip with 1000+ cores is expected to appear in the foreseeable future [3].

One of the critical challenges for many-core system design is energy efficiency. Processors with high energy efficiency not only save millions of dollars in energy bill for supercomputers and data centers, but also extend the battery life for mobile devices. Various low power techniques have been proposed and adopted to improve energy efficiency. Clock gating and power gating are two widely used techniques to reduce dynamic/leakage power, by shutting off unused components and cores from clock tree and power supply. Another approach to reduce power dissipation under performance constraints is to apply dynamic voltage and frequency scaling (DVFS). Many-core systems with per-core DVFS have been proved to be capable of reducing energy dissipation significantly by adapting both voltage and frequency according to the required performance and workload [4], [5].

To further improve the performance and energy efficiency for many-core processors, combination of DVFS and core scaling (DVFCS) has been proposed. Jian *et al.* addressed the problem of finding a chip-wide operating voltage and frequency setting as well as the number of active cores that minimizes the power consumption of a general-purpose chip multiprocessor under a performance constraint [6]. Lee *et al.* studied to improve the performance of power constrained GPUs by coordinating the number of active cores and chip-wide voltages/frequencies of both cores and caches [7]. All these work assumed that the DVFS is on the chip level, but not on the per-core level.

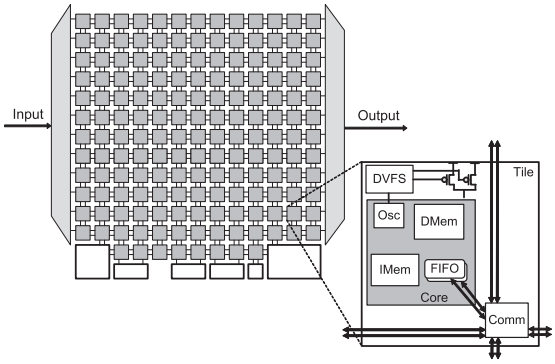


Fig. 1. Block Diagram of the targeted many-core system.

Compared to the previous work, this paper addresses the problem of minimizing the power dissipation of many-core systems under performance constraints by exploiting per-core DVFS with core scaling. The rest of the paper is organized as follows. Section II briefly describes the targeted many-core system. Section III formulates the problem of our study and the proposed algorithm. Section IV discusses the experimental results with real application benchmarks. Finally, Section V concludes the paper.

II. Targeted Many-Core Architecture

The targeted Asynchronous Array of Simple Processors (AsAP) architecture is an example of fine-grained many-core systems, supporting globally-asynchronous locally-synchronous (GALS) on-chip network and per-core DVFS [8].

Fig. 1 shows the block diagram of AsAP. The many-core system is composed of 164 small identical processors. All processors are clocked by local fully independent oscillators and are connected by a reconfigurable 2D-mesh network that supports both nearby and long distance communications. Each processor operates at a maximum clock frequency of 1.2 GHz at 1.3V. The AsAP many-core chip was fabricated in 65 nm CMOS technology [1].

In this paper, we assume that there are multiple discrete global voltage levels for the whole chip. The per-core DVFS is capable of controlling the oscillator of each core to run at the minimum frequency without violating performance constraints, and selecting the optimal voltage based on the frequency. We also assume that the number of cores on the platform can be scaled as necessary.

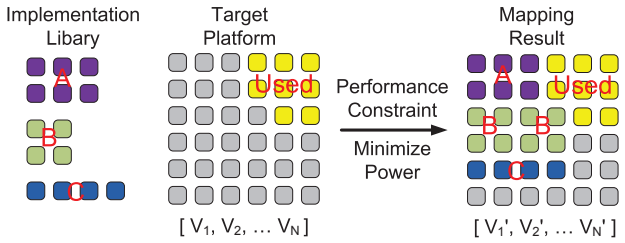


Fig. 2. Three implementations A, B, and C are mapped into the targeted many-core system based on the performance requirement. The global voltage levels are also altered to minimize the power dissipation of the chip.

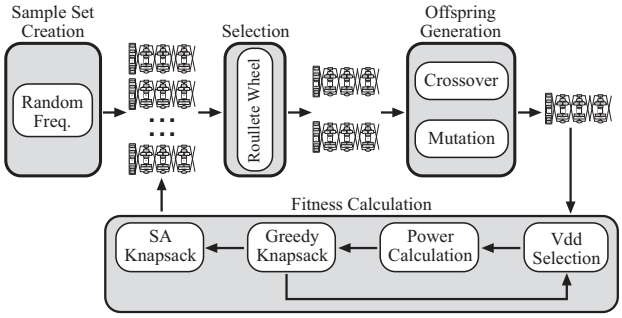


Fig. 3. Flow of the proposed algorithm.

III. Problem Statement and Proposed Algorithm

A. Problem Statement

An application is required to run at a throughput of TP_{REQ} on a many-core system which has $CORE_{AVL}$ cores available, and N gloabl voltage levels (V_1, V_2, ..., V_N) in ascending order. There are a set of implementations (A_1, A_2, ..., A_M) for the application. Each implementation requires $CORE_{A_i}$ cores. The throughput of each implementation is determined by the operating frequency F_{A_i} of its performance critical cores. The mapping result may include multiple instances for each implementation, and different instance could run at different F_{A_i}. The throughput of the mapping result TP_{MAP} is given by

$$TP_{MAP} = \sum_{n=1}^{N_1} TP_{A_1 n} + \sum_{n=1}^{N_2} TP_{A_2 n} ... + \sum_{n=1}^{N_M} TP_{A_M n} \quad (1)$$

where N_i is the number of copies of implementation A_i in the mapping result, while the throughput $TP_{A_i n}$ of the A_i's nth instance is obtained as

$$TP_{A_i n} = TPC_{A_i} \times F_{A_i n} \quad (2)$$

where TPC_{A_i} is the number of bits that can be processed by A_i per cycle. Similarly, the total power of the mapping result P_{MAP} is given by

$$P_{MAP} = \sum_{n=1}^{N_1} P_{A_1 n} + \sum_{n=1}^{N_2} P_{A_2 n} ... + \sum_{n=1}^{N_M} P_{A_M n} \quad (3)$$

where $P_{A_i n}$ is the power of the nth instance of A_i. For each A_i's instance, the power P_{A_i} is obtained as

$$P_{A_i} = \sum_{j=1}^{CORE_{A_i}} P_{CORE j}(F_{CORE j}, V_{CORE j}) \quad (4)$$

where $P_{CORE j}$ is the power consumption of each individual core in A_i, as a function of the frequency $F_{CORE j}$ and voltage $V_{CORE j}$. The performance critical cores are required to run at $F_{CORE j} = F_{A_i}$ to guarantee the throughput shown in Eq. 2. Other cores could decrease their frequency depends on their own workload. $V_{CORE j}$ is the minimum voltage level V_i from the global discrete voltages (V_1, V_2, ..., V_N), where $Freq(V_i) >= F_{CORE j}$. The $P_{CORE j}$ is approximated as follows:

$$\begin{aligned} P_{CORE j}(F_{CORE j}, V_{CORE j}) &= P_{DYN} + P_{LEAK} \\ &= C_{EFF} \cdot F_{CORE j} \cdot V_{CORE j}^2 + I_{LEAK}(V_{CORE j}) \cdot V_{CORE j} \end{aligned} \quad (5)$$

where C_{EFF} is the effective switching capacitance for each core, and $I_{LEAK}(V_{CORE j})$ is the leakage current that depends on $V_{CORE j}$.

The problem can be formulated as:

$$\begin{aligned} &minimize && P_{MAP} \\ &\text{subject to} && TP_{REQ} \le TP_{MAP} \\ & && CORE_{AVL} \ge \sum CORE_{A_i} \cdot N_i \end{aligned}$$

by exploring (1) the number of copies of each implementation (N_1, N_2, ..., N_M), (2) the throughput of each implementation instance $TP_{A_{in}}$, and (3) the global voltage levels (V_1, V_2, ..., V_N). Fig. 2 shows that three implementations are mapped on a many-core system based on the performance constraint. Additionally, the global voltage levels are altered to (V_1', V_2', ..., V_N') to minimize the power dissipation.

B. Proposed Algorithm

Due to the large search and optimization space, genetic algorithm (GA) is applied to solve the problem. GA is based on a mechanism of natural selection and evolutionary genetics. Each solution of the problem is represented as a *chromosome*. A *fitness* value is associated with each solution. The fitness value shows how close the solution is to the optimum. The process starts with an initial *population* of solutions created in some way, e.g. randomly. In each iteration, new offspring solutions are generated through *selection*, *crossover* and *mutation*. Then, some new solutions are added to, and some old solutions are removed from the *population* based on *fitness*. The evolution process stops when either the optimization goal or the maximum number of iterations is reached [9].

The block diagram of the proposed algorithm is shown in Fig. 3. At the beginning, a set of chromosomes are generated randomly to compose a population. Each chromosome is a sequence of genes. Each gene represents an implementation instance $A_i n$ with a particular $F_{A_i n}$ as shown in Fig. 4. The number of genes for each implementation A_i in one chromosome is the maximum number of A_i can fit into the current platform, *i.e.,* $CORE_{VAL}/CORE_i$. Based on the $F_{A_i n}$, the maximum voltage level V_N and throughput of each gene can be calculated. Other voltage levels (V_1, ..., V_{N-1}) for each gene are calculated to minimize power in a loop manner.

The power and throughput of each gene are sent to a greedy knapsack solver to decide which implementation instance (gene) should be included (turned "ON"), and which not (turned "OFF"). After the voltage levels are iteratively explored and sent to the greedy knapsack to obtain a solution, the final best solution is returned. The algorithm keeps the voltage levels of the returned solution and runs a simulated annealing knapsack to explore a better solution.

In each iteration of GA, two chromosomes are selected as parents for crossover. During selection, a roulette wheel strategy is applied, which means the higher the fitness value is, the better is the chance of the chromosome to be selected. The fitness value of a solution

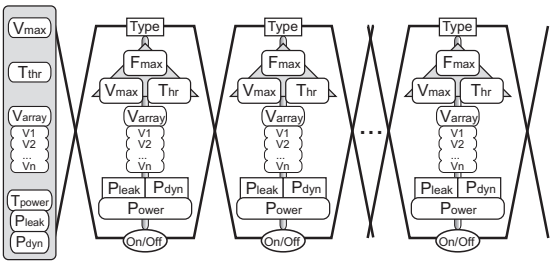


Fig. 4. Block diagram of the chromosome in GA.

TABLE I
NUMBER OF CORES AND THROUGHPUT OF DIFFERENT AES ENGINES [11]

AES Implementation	Cores Usage	1/Throughput (cycles/byte)
Small	8	167.375
One-task one-processor	9	223.875
Parallel-MixColumns	15	136.250
Parallel-SubBytes-MixColumns	18	84.375
Loop-unrolled Three Times	23	68.625
Loop-unrolled Nine Times	50	16.625
No-merge-parallelism	59	9.500
Full-parallelism	137	4.375

is inversely proportional to the power dissipation. During crossover, a random point γ is chosen to break each parent into two parts. Then the offspring is generated by copying the part before γ from ParentI and the part after γ from ParentII. After crossover, a mutation happens based on the probability assigned in the simulation. A random operation frequency is assigned to a randomly selected implementation instance. Finally, the offspring is sent to the fitness calculation process to generate a valid solution. If the power of the offspring is less than one of the chromosomes in the population set. The chromosome with weakest fitness (the solution with largest power) in the set is killed and the offspring stays in the set for the next iteration.

IV. EVALUATION

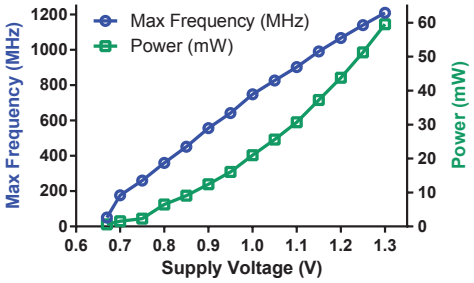
A. Power Model

Fig. 5(a) shows the maximum operation frequency and dynamic power dissipation of one AsAP core versus supply voltage. The data are measured from real-silicon chip, and used for the simulations in the next section.

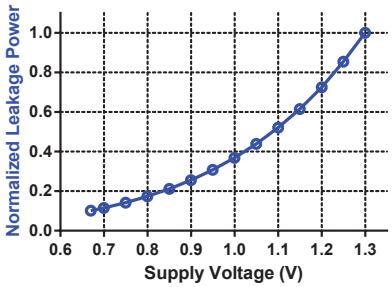
Since AsAP was fabricated with super low-power CMOS process, the leakage power from the chip measurement is negligible. To simulate processors with different leakage power ratio $LR = P_{LEAK}/P_{DYN}$ at $VDD = 1.3\ V$, the leakage power scaling factors for other voltage levels are generated from a dummy circuit with ST 65 nm CMOS technology. The dummy circuit is composed of a large number of NAND, NOR and INV gates. Each gate has different number of inputs (1 to 4) and the input states are randomly selected [10]. The normalized leakage power scaling factors are measured from the HSPICE simulation and shown in Fig. 5(b).

B. Benchmark – Eight AES Engines

An implementation library with eight AES engines is selected as the benchmark. Table I lists the number of cores and throughput for different AES implementations. The smallest implementation only occupies eight cores, while the largest one requires 137 cores. The throughput of the eight AES engines at 1.3 V and 1.2 GHz are ranged from 58 Mbps to 2.2 Gbps.



(a) Maximum Frequency and Dynamic Power



(b) Normalized Leakage Power

Fig. 5. (a) Maximum operation frequency and dynamic power of one core; (b) Normalized leakage power scaling factor.

C. Experiment Results

There are five different voltage/frequency/core scaling configurations are studied: (1) No DVFS; (2) per-core DVFS with one global *Vdd* (OV); (3) per-core DVFS with two global *Vdd*s (TV); (4) per-core DVFCS with one global *Vdd* (OVC); and (5) per-core DVFCS with two global *Vdd*s (TVC). The *NoDVFS* is selected as baseline to evaluate the efficiency of other configurations. One *No-merge-parallelism* engine is selected if core scaling is not applicable, since it is the most energy-efficient implementation compared with others [11]. The maximum throughput (Th_{MAX}) of one *No-merge-parallelism* engine is approximate to 1 Gbps.

Fig. 6 shows the power dissipation for each configuration, all normalized to *NoDVFS*. With the same number of global voltage levels, DVFCS outperforms DVFS for all different throughput requirements. DVFCS saves as much as 68% and 72% power compared with DVFS for systems with one and two global voltage levels, respectively. Moreover, OVC consumes less power than TV when the throughput requirement is less than 50 Mbps or greater than 700 Mbps, which implies that core scaling is more effective for reducing power dissipation than extra voltage levels when the system has either a loose or a tight performance constraint. As shown in Fig. 6, the optimal number of cores chosen by core scaling tends to increase when the throughput constraint is higher. Additionally, TVC tends to utilize less cores than OVC, which means that extra voltage levels could reduce the number of cores used in the optimal solution.

The extra power saving brought by core scaling shows a non-linear relationship with the throughput requirements. To understand the fundamental reasons of the non-linear relationship, we divide the simulation results into three categories, which are systems with (1) a loose ($\leq 20\%Th_{MAX}$), (2) an intermediate ($20\% \sim 80\%Th_{MAX}$), and (3) a tight ($\geq 80\%Th_{MAX}$) performance constraint. As shown in Table II, for all three categories of performance constraints, most of the power saving of OV and TV comes from dynamic power.

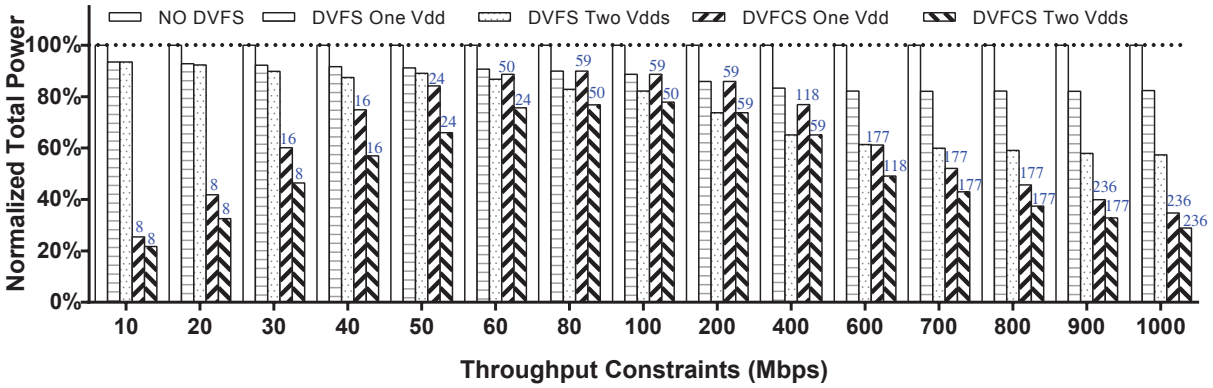


Fig. 6. Normalized power dissipation of different scaling configurations. The leakage power ratio $LR = 0.3$. *NoDVFS*, *DVFS One Vdd*, and *DVFS Two Vdds* use single *No-merge-parallelism* engine. The optimal number of cores chosen by *DVFCS One Vdd* and *DVFCS Two Vdds* are shown on the top of power bars.

TABLE II
AVERAGE POWER SAVINGS FOR DIFFERENT DVFCS CONFIGURATIONS AND PERFORMANCE CONSTRAINTS.

Configurations	Loose Perf. Target			Inter. Perf. Target			Tight Perf. Target		
	Dyn. Saving	Leak. Saving	Total Saving	Dyn. Saving	Leak. Saving	Total Saving	Dyn. Saving	Leak. Saving	Total Saving
DVFS with One Vdd	9.2%	0.0%	9.2%	17.1%	0.0%	17.1%	17.8%	0.0%	17.8%
DVFS with Two Vdds	11.6%	2.1%	13.6%	28.9%	7.6%	36.5%	31.7%	10.2%	41.9%
DVFCS with One Vdd	4.8%	25.4%	30.2%	35.7%	-3.9%	31.8%	53.5%	6.4%	59.9%
DVFCS with Two Vdds	5.9%	35.4%	41.5%	39.1%	4.5%	43.6%	56.7%	10.2%	66.9%

With a loose performance constraint, $\sim 75\%$ power dissipation is due to leakage, which limits the power saving of OV and TV that rely on dynamic power reduction. Compared to OV and TV, OVC and TVC saves extra 21% and 28% power by reducing leakage power significantly. The leakage power reduction is due to core scaling chooses implementations use small number of cores for loose performance constraints. As the performance constraints get into the intermediate range, the dynamic power starts to dominate in total power dissipation. As a result, the extra saving brought by core scaling is reduced to 14% and 7% for OVC and TVC, respectively. When the performance target is close to Th_{MAX}, core scaling starts to apply multiple instances for each implementation. Therefore, each instance requires a lower voltage/frequency level compared to OV and TV, which causes 42% and 25% less power dissipation.

V. CONCLUSION

In this paper, we address the problem of minimizing the power dissipation of many-core systems under performance constraints by exploiting per-core DVFS with core scaling. A GA-based algorithm is proposed to solve the problem. The experimental results demonstrate that choosing the number of operating cores and voltage/frequency levels appropriately can reduce power consumption by up to 72% compared with per-DVFS. We also demonstrate that the extra power saving brought by core scaling has a non-linear relationship with the performance constraints. The additional power saving brought by core scaling tends to increase when the performance constraint inclines to be either very loose or very tight.

VI. ACKNOWLEDGMENTS

The authors gratefully acknowledge support from NSF Grant 1018972 and 0903549, SRC GRC Grant 1598, 1971 and 2321, CSR Grant 1659, C2S2 Grant 2047.002.014, UC Micro, Intel, and Intellasys. The authors would like to thank STMicroelectronics for donating the chip fabrication.

REFERENCES

[1] D. N. Truong and W. H. Cheng *et al.*, "A 167-processor computational platform in 65 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 4, pp. 1130–1144, Apr. 2009.

[2] S. R. Vangal and J. Howard *et al.*, "An 80-tile sub-100-w teraflops processor in 65-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 29–41, Jan. 2008.

[3] S. Borkar, "Thousand core chips – a technology perspective," in *Design Automation Conference (DAC)*, June 2007, pp. 746–749.

[4] W. Kim and M. S. Gupta *et al.*, "System level analysis of fast, per-core DVFS using on-chip switching regulators," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb. 2008.

[5] B. Liu, B. Bohnenstiehl, and B. M. Baas, "Scalable hardware-based power management for many-core systems," in *IEEE Asilomar Conference on Signals, Systems and Computers (ACSSC)*, Nov. 2014.

[6] J. Li and J. F. Martinez, "Dynamic power-performance adaptation of parallel computation on chip multiprocessors," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2006.

[7] J. Lee and V. Sathisha *et al.*, "Improving throughput of power-constrained GPUs using dynamic voltage/frequency and core scaling," in *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, Oct 2011.

[8] D. Truong and W. Cheng *et al.*, "A 167-processor 65 nm computational platform with per-processor dynamic supply voltage and dynamic clock frequency scaling," in *VLSI Circuits, 2008 IEEE Symposium on*, June 2008.

[9] DE Goldberg, "Genetic algorithms in search, optimization, and machine learning," 1989.

[10] J. Lee and N. S. Kim, "Optimizing throughput of power- and thermal-constrained multicore processors using dvfs and per-core power-gating," in *Design Automation Conference (DAC)*, July 2009.

[11] B. Liu and B. M. Baas, "Parallel AES encryption engines for many-core processor arrays," *Computers, IEEE Transactions on*, vol. 62, no. 3, pp. 536–547, march 2013.