A Power Modelling Approach for Many-core Architectures

Zhiquan Lai[#], King Tin Lam[*], Cho-Li Wang[*], Jinshu Su[#]

[#]*National Key Laboratory of Parallel and Distributed Processing (PDL), Changsha, China*
[#]*College of Computer, National University of Defense Technology, Changsha, China*
[*]*Department of Computer Science, The University of Hong Kong, Hong Kong, China*

{zqlai, sjs}@nudt.edu.cn, {ktlam, clwang}@cs.hku.hk

***Abstract*—Many-core architectures are playing an important role in the HPC systems. But they are giving high performance at the cost of a great electrical power consumption. On Tianhe-2 supercomputer, the Xeon Phi many-core processors contribute nearly 80% of the system power. Power models are important to guide the design of dynamic power management (DPM) algorithms by predicting the power consumption with respect to power states and program execution patterns. However, the complexity of many-core hardware design makes power modelling be a challenging work. These concerns lead us to try a power modelling approach for many-core architectures based on the performance monitoring counters (PMC). The key insight is based on a large number of microbenchmarks on a real many-core platform, where we find some essential rules determining the chip power. Following the modelling approach, we develop an accurate chip power model for the Intel SCC many-core chip. Experimental comparison shows that our model is much more accurate than others.**

Keywords—many-core; power management; power modelling; model

I. INTRODUCTION

Energy efficiency is one of the most challenging problems in the academia and industry of high performance computing (HPC). Many-core architectures are playing an important role in HPC systems, but their delivering of the high performance is also consuming a great portion of electricity in the meantime. On Tianhe-2 supercomputer, the fastest supercomputer in the latest Top500 list [1], the Xeon Phi many-core processors contribute nearly 80% of the system power. Power management of many-core chips becomes the urgency to achieve energy efficiency of the whole HPC systems.

Power models are important to guide the design of dynamic power management (DPM) algorithms by predicting the power consumption with respect to the power states (i.e. voltage and frequency settings) and program execution patterns. There are many prior works to model the power for single core or multi-core chips [2, 3]. However, on many-core architectures, the complexity of hardware designs makes power modelling be a challenging work. Apart from the a large number of CPU cores, many-core chips usually have complex network on chip (NoC), multiple memory controllers, on-chip caches and some other units [4]. These all increase the complexity of the modelling work.

There are few of work modelling the power of many-core chips. Bartolini *et al.* [5] evaluated the impact of DVFS on the performance and power consumption of MPI applications. However, their study only presented the power/performance results in different power settings rather than proposing any power model. Sadri *et al.* [6] proposed a power model for thermal management on the Intel SCC, but they did not take account of the voltage. The power model proposed by Cichowski *et al.* [7] did not consider the impact of application pattern to the power consumption of the chip. However, as we will reveal, the voltage and the program pattern are both the key factors determining the chip power.

These concerns lead us to develop a power modelling approach for many-core architectures with consideration of more factors as possible. The key insight is based on a large number of microbenchmarks on the Intel SCC platform which we design to test the chip power of the many-core architectures. The main contributions of this work are as follows:

- We develop a power modelling approach for many-core architectures based on the performance monitoring counters (PMC). Following this approach, we derive a chip model of the Intel SCC many-core chip, with comprehensive consideration of voltage/frequency settings and the program execution patterns.
- We conduct experiments on the Intel SCC to compare our model with others. The results show that our model is more accurate than other models to predict the chip power.

The remainder of this paper is organized as follows. Section II describes our many-core chip power model and the process to specify to parameters of the model for the Intel SCC. Section III presents the experimental evaluation, including the results and analysis. Finally, we conclude the paper in Section IV.

II. POWER MODELLING

We will describe our chip power model of many-core architectures. Then show how we specify the parameters of the model on the Intel SCC platform.

A. Power Model

The many-core architectures usually have much more complex hardware design than single core or multi-core chips. Apart from the a large number of CPU cores, many-core chips have network on chip (NoC), multiple memory controllers, on-chip caches and some other units. Thus, the power of chip is made up with the power of CPU cores, the on-chip mesh network (NoC) etc.

First of all, Let us consider the main power contributor, the CPU cores. The power of a single core consists of two parts, namely *static power* and *dynamic power*. Static power depends on the voltage and the CPU's thermal design power (TDP). As described in (1), dynamic core power ($CorePower_{dyn}(v,f)$) has a proportional relationship with frequency (f), squared supply voltage (v), and the *activity factor (A)*. The activity factor is not usually a constant coefficient, but has a functional relationship with the program patterns. To simplify the power model, we assume the static core power, denoted by $CorePower_{sta}$ is a constant.

$$\begin{aligned} &CorePower = CorePower_{dyn}(v,f) + CorePower_{sta} \\ &CorePower_{dyn}(v,f) \propto A \cdot f \cdot v^2 \end{aligned} \tag{1}$$

For a certain program, we assume that the activity factor A is also a constant. So, we can estimate the power of core when it is used to run a program (or a phase of a program) in any voltage/frequency settings.

For many-core chips, the chip power is made up with the power of cores and the NoC etc. As in current many-core design, dynamic power state tuning of the NoC mesh network is not supported by the current SCC, we assume the power of the mesh network and other units except CPU cores, denoted by *NocPower*, is a constant. The formula of total chip power can be expressed as

$$\begin{aligned} &ChipPower \\ &= \sum_{i=0}^{N_{core}} (CorePower^i) + NocPower \\ &= \sum_{i=0}^{N_{core}} (CorePower^i_{dyn}(v,f)) + \sum_{i=0}^{N_{core}} (CorePower^i_{sta}) + NocPower \\ &= \sum_{i=0}^{N_{core}} (A_i \cdot f_i \cdot v_i^2) + (\sum_{i=0}^{N_{core}} (CorePower^i_{sta}) + NocPower). \end{aligned} \tag{2}$$

Let $Power_{sta} = \sum_{i=0}^{N_{core}} (CorePower^i_{sta}) + NocPower$, the above formula can be expressed as

$$ChipPower = \sum_{i=1}^{N_{core}} (A_i \cdot f_i \cdot v_i^2) + Power_{sta}, \tag{3}$$

where the A_i denotes the activity factor of the program on core *i*. As some prior works revealed, power dissipation is strongly correlated to Instructions per Cycle (IPC) [6, 8, 9]. Thus we assume the activity factor of core *i* A_i is a function of IPC at a specific power setting (assuming it's 800MHz), and name the function as $g(IPC_i)$.

The next work of power modelling, which is the important work, is to specify the parameters of A_i and $Power_{sta}$.

B. Parameter Specifying

We are going to describe how to specify the parameters of the model on the Intel SCC platform. The Intel SCC is an 48 cores experimental many-core chip with fine-grained DVFS mechanism [10]. The IPC pattern of each core can be derived by reading the PMC counters provided by the CPU core hardware. The real chip power also can be measured using the mechanism provided by the Intel SCC platform [11].

In order to derive $Power_{sta}$ and the function of $g(IPC_i)$, we conducted some micro-benchmarking by designing two programs, INT and FP. INT performs arithmetical computing on an array of integer variables while FP does so on an array of float-point variables. The sizes of the two arrays can be configured easily. We use these two different programs with different problem sizes to produce different IPC patterns. They are launched using 15 V/F settings and seven problem sizes as shown in TABLE I . All these programs with different settings are launched on the 48 cores of the Intel SCC (all the cores perform the same computation). Thus, we have 2×15×7=210 experimental results plotted as Fig. 1 .

TABLE I BENCHMARK SETTING TO DETERMINE POWER MODEL

Program						
INT				FP		
Problem Size *(# of 4B Integers or 8B Doubles)*						
4	1K	4K	16K	64K	256K	1024K
Frequency/Voltage *(MHz/V)*						
800/1.1	533/0.9	400/0.8	320/0.8	267/0.8		
229/0.8	200/0.8	178/0.8	160/0.8	145/0.8		
133/0.8	123/0.8	114/0.8	107/0.8	100/0.8		

We make use of a trick here to simplify the chip power function for specifying the parameters of $g(IPC_i)$ and $Power_{sta}$. As we cannot measure the power of each core directly, we launch the same program on all the 48 cores of the SCC, and set the V/F levels of all cores to be the same. Thus, the activity factors of each program on all the cores are the same, denoted by *g(IPC)*. So the *ChipPower* can be formulated as

$$ChipPower = N_{core} \cdot g(IPC) \cdot f \cdot v^2 + Power_{sta}. \tag{4}$$

Fig. 1 presents the results of these microbenchmarks. The *x*-axis is the value of $f \cdot v^2$ with unit of $MHz \cdot V^2$, and the *y*-axis shows the power of the chip (*ChipPower*). The results of each program with different problem sizes are plotted in different colors. From the results, we observe that the power consumption of the chip is linear with respect to $f \cdot v^2$. We use a linear function $Y = \alpha X + \beta$ to fit the resulting data of each program using different problem sizes. In other words,

for each program, we can get a linear function $Y = \alpha X + \beta$, where X denotes $f \cdot v^2$, Y denotes *ChipPower*, α denotes the fitting value of $N_{core} \cdot g(IPC)$ *and* β denotes the fitting value of $Power_{sta}$. The regression analysis results are showed in TABLE II .

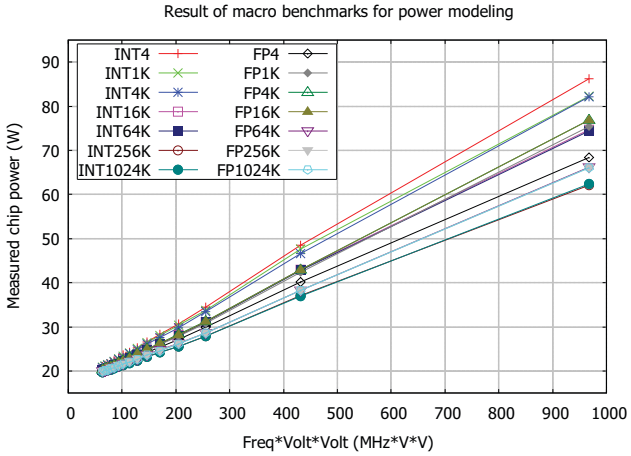

Fig. 1 Benchmark results to specify the parameters of the power model

TABLE II RESULT OF REGRESSION ANALYSIS

Program	IPC800M	α	β
INT4	0.9130	0.072	16.14
INT1K	0.7335	0.068	16.39
INT4K	0.7281	0.068	16.17
INT16K	0.4310	0.060	16.07
INT64K	0.4222	0.060	16.09
INT256K	0.0642	0.047	16.45
INT1024K	0.0640	0.047	16.41
FP4	0.1239	0.053	16.37
FP1K	0.5883	0.061	15.86
FP4K	0.2382	0.062	15.91
FP16K	0.2380	0.062	15.91
FP64K	0.0309	0.051	16.30
FP256K	0.0309	0.051	16.33
FP1024K	0.0310	0.051	16.19
Average			16.185

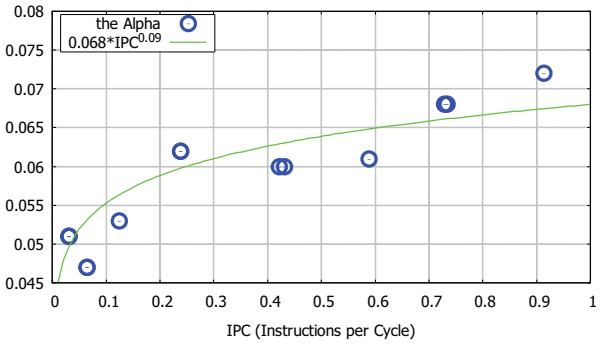

Fig. 2 Fitting analysis of *active factor* A_i

We can see from TABLE II , for each program, the power is linearly related to $f \cdot v^2$ with almost the same $Power_{sta}$ (i.e. the β) except that the coefficients are slightly different. Thus we take the average value of coefficient β as the static chip power as

$$Power_{sta} \approx 16.185. \quad (5)$$

On the other hand, to get the specific active factor A_i, i.e. *g(IPC)*, we use a function to fit the values of the coefficient α towards *IPC* (the estimated values of $N_{core} \cdot g(IPC)$ in our model) . The α values with respect to IPC are plotted in Fig. 2 . Eventually, using least squares regression algorithm we get the specified function of $N_{core} \cdot g(IPC)$ with R^2 coefficient = 0.812. It's described in (6).

$$N_{core} \cdot g(IPC) = 0.068 \cdot IPC^{0.09} \quad (6)$$

Thus, since N_{core} equals to 48, we can derive the activity factor $A_i = g(IPC_i)$ for each CPU core *i*, as described in (7).

$$A_i = g(IPC_i) = 1.417e^{-3} \cdot IPC_i^{0.09} \quad (7)$$

In (7), the IPC equals to the instructions per clock at a profiling frequency of 800MHz. At last, we replace the parameters of A_i and $Power_{sta}$ into (3), and get the chip power model of the Intel SCC as follows:

$$ChipPower = \sum_{i=0}^{N_{core}} 1.417e^{-3} \cdot IPC_i^{0.09} \cdot f_i \cdot v_i^2 + 16.185. \quad (8)$$

The units of the variables are: watts (W) for *ChipPower*, MHz for f_i, and volts (V) for v_i. With this chip power model of the Intel SCC, we can predict the chip power under any power states or program execution patterns, as shown in Fig. 3 . Assuming we always supply the least support voltage for a certain frequency, we can use the frequency to denote the power state. As we never see a IPC larger than 2 instructions per cycle, we present the IPC from 0 to 2 in the figure. Moreover, we will evaluate this power model in Section III.

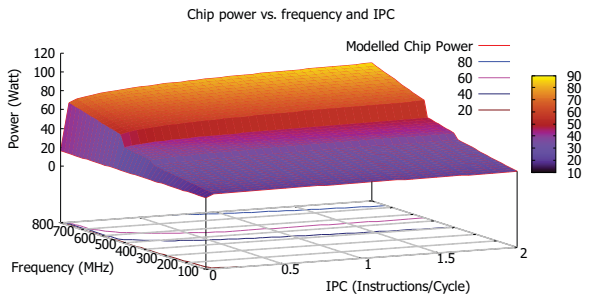

Fig. 3 The estimated chip power of the Intel SCC with repect to frequency settings (i.e. power states) and IPC (i.e. program execution patterns)

III. Evaluation of Power Model

In order to evaluate our power model proposed in Section II, we compare the estimated chip power using our model with that using power model proposed by Sadri *et al.* [6]. Sadri *et al.* proposed a single core power modelling approach for the Intel SCC. In their model, they also considered the CPU frequency and program execution pattern, except the supplied voltage. Their modelling includes three main steps.

- First, model the core power consumption by splitting it into idle power and active power, $P_{core} = P_{idle} + P_{active}$, and propose the fitting functions each one.
- Second, measure the power of single CPU core at different frequency and program pattern settings. The authors use CPI (Clock per Instruction), which is the multiplicative inverse of IPC used in our model, to represent the program pattern.
- Third, use a least square optimization algorithm to find the coefficients of the fitting power function.

The fitting functions for both idle power and active power are shown as (9) and (10), where p, q, a, b, c, d, a', b' and c' are constant coefficients. After these coefficients are determined in the third step, we get the core power model as shown in (11). The unit of f_{core} is GHz.

$$P_{idle} = p + q \cdot f_{core} \tag{9}$$

$$P_{active} = (a + b \times CPI^{c}) \times f_{core} + (a' + b' \times CPI^{c'}) \times f_{core} + d \tag{10}$$

$$P_{core} = (0.38 - 0.24 \cdot CPI^{0.085} + 0.22 \cdot CPI^{0.02}) \cdot f_{core} + 0.41 \tag{11}$$

According to their modelling approach, we can suppose that the rest chip power apart from the CPU cores equal to the chip power when no core is active. By the measurement in their paper, we find out the rest power is about 34.0 Watt. So the chip power can be estimated by adding the power of all the cores and this 34.0 Watt, as shown in (12).

$$P_{chip} = \sum_{i=0}^{N_{core}} P_{core}^{i} + 34.0 \tag{12}$$

There might be some problems in Sadri's model due to the power measurement of single core and the model builded based on this. As most of the many-core chips like the Intel SCC, do not provide direct power measurement mechanism (for example, power senses) for each single core. We usually only get the power of the whole chip, include the power of CPU cores, Network on chip and other on-chip units. In order to measure the power of single CPU core, Sadri *et al.* increase the number of active cores, and measurement the increase of the chip power, then the increased chip power is taken as the power of activated core. But actually this increased chip power is just the increased dynamic power as the static power, only affected by the supplied voltage, would not be changed after the activation. However, the problem is the authors take the increased power as the sum of idle power and active power, which we think mean static power and dynamic power respectively.

To compare our power model with Sadri's model, we conduct an experiment on 48 cores at 800MHz/1.1V using Graph500 benchmark [12]. As we can only get the real power of the whole chip, we estimate the chip power for comparison using both models. We estimated the chip power as follows:

First, conduct Graph500 on 48 cores at static 800MHz/1.1V. At the meantime, we profile the PMC counters of each CPU core for CPI or IPC of the Graph 500 execution. The real power is monitored using the mechanism provided by the Intel SCC platform [11]. As there exits some latency in reading the real-time power of the chip, we can only achieve the power values with a rate of 3.3 samples per second. But this is enough for us to compare it with the modelled power.

Second, based on the profiled CPI or IPC pattern of the Graph 500 execution, we estimate the chip power using both the chip power models, our model as (8) and Sadri's model as (12).

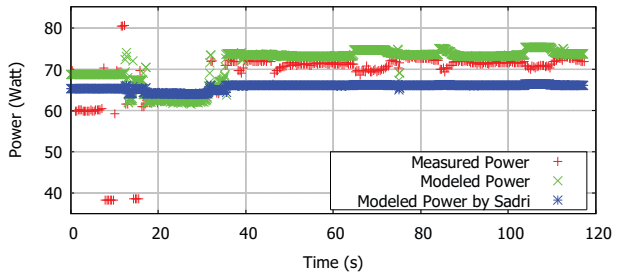

Fig. 4 Estimated and measured chip power of Graph 500 running on 48 cores at static CPU frequency of 800MHz

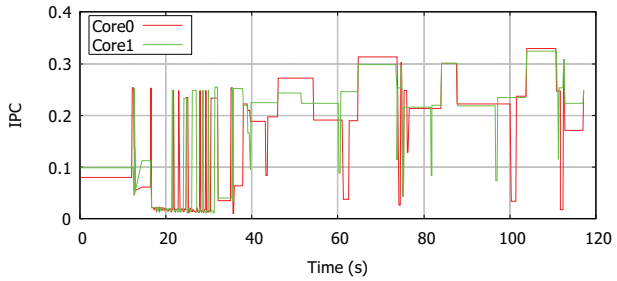

Fig. 5 The IPC of core0 and core1 during Graph 500 execution

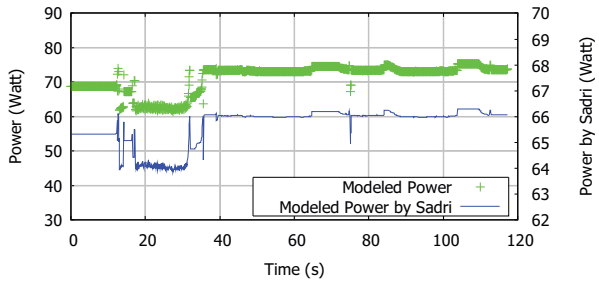

Fig. 6 Power comaprison by zooming in Sadri's modelled power

The measured chip power and the modelled power are shown in Fig. 4 . Comparing with the measured power, it's obviously observed that our modelled power is closer to it than Sadri's modelled power. During Graph 500 execution, the chip power changes due to different execution patterns. But the estimated power by Sadri's model seems being more stable without the awareness of the program execution pattern.

However, our model is not perfect yet. At the beginning 12 seconds of the execution, for both modelled powers, we found the estimated power has much error. This is because we take this part of program as a single phase, although the patterns at different time are different. After time of 17s, the estimated power is more accurate. We observe four places (around 50s, 70s, 85s and 90s) that the errors of estimated power have a different trend with the measured power. By the analysis of the IPC pattern of the execution (as shown in Fig. 5), we find that the IPCs of both master and slave are relatively high at these four places. These phenomena suggest that our power model is much more accurate when IPC is low, whereas when IPC is high (larger than 0.3 instructions per second) our power model is not so accurate.

As mentioned before, Sadri's modelled power is too stable. But when we zoom in it, we find something much more interesting. As shown in Fig. 6 , the green points in the figure are our modelled power with left y-tics, the blue lines represent the Sadri's modelled power with right y-tics. We can find that the right y-tics is zoomed in. Then, we can see that the changing trends of the modelled powers are much similar and nearly fit the measured power. The only difference is the granularities. This observation implies two concerns. On one hand, it's much necessary to take account of the program pattern. We can find that at the same power setting, the chip power changes in a large range. And the models considering the IPC metric fit the changing trend very well. This implies that IPC is a good metric to represent the program execution pattern. On the other hand, our power modelling approach is more accurate to model the chip power of many-core chips. Measured as Sadri's modelling approach, the single core power does not include the static power. But they still take the measured single core powers as the sum of the static power and dynamic power (idle power and active power). This may be the reason why their model is not accurate enough.

IV. Conclusion

In this paper, we propose power modelling approach for many-core architectures. With this approach, we achieve the power model of the Intel SCC many-core chip with considering both of power states and program execution patterns. Then we conduct the experimental comparison with other many-core chip power models. The result verifies that the model derived by the modelling approach is more accurate. As the future work, we plan to continue to improve our model and evaluate it by applying it in real-time power management systems.

Acknowledgment

The work of this paper is supported by Hong Kong RGC grant HKU 716712E, Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT, No. IRT1012) and Aid Program for Science and Technology Innovative Research Team in Higher Educational Institutions of Hunan Province (No. 11JJ7003). Special thanks go to Intel China Center of Parallel Computing (ICCPC) in Wuxi, China for providing us the SCC hardware platform to carry out this research work.

References

[1] *Top500 List - June 2014*. Available: http://www.top500.org/lists/2014/06/

[2] S. Song, "Power, Performance and Energy Models and Systems for Emergent Architectures," Doctor of Philosophy, the Virginia Polytechnic Institute and State University, 2013.

[3] J. W. Choi, D. Bedard, R. Fowler, and R. Vuduc, "A Roofline Model of Energy," in *Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing*, 2013, pp. 661-672.

[4] J. L. Manferdelli, N. K. Govindaraju, and C. Crall, "Challenges and Opportunities in Many-Core Computing," *Proceedings of the IEEE,* vol. 96, pp. 808-815, 2008.

[5] A. Bartolini, M. Sadri, J.-N. Furst, A. K. Coskun, and L. Benini, "Quantifying the Impact of Frequency Scaling on the Energy Efficiency of the Single-Chip Cloud Computer," presented at Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, 2012.

[6] M. Sadri, A. Bartolini, and L. Benini, "Single-Chip Cloud Computer Thermal Model," presented at the 17th International Workshop on Thermal Investigations of ICs and Systems (THERMINIC'11), Paris, France, 2011.

[7] P. Cichowski, J. Keller, and C. Kessler, "Modelling Power Consumption of the Intel SCC," presented at Many-core Applications Research Community Symposium (MARC), 2012.

[8] T. Li and L. K. John, "Run-time modeling and estimation of operating system power consumption," in *Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, San Diego, CA, USA, 2003, pp. 160-171.

[9] W. Bircher, J. Law, M. Valluri, and L. K. John, "Effective use of performance monitoring counters for run-time prediction of power," TR-041104-01, Electrical and Computer Engineering Department, University of Texas, 2004.

[10] J. Howard, S. Dighe, S. Vangal, G. Ruhl, N. Borkar, S. Jain*, et al.*, "A 48-Core IA-32 Message-passing Processor in 45nm CMOS Using on-die Message Passing and DVFS for Performance and Power Scaling," *IEEE Journal of Solid-State Circuits,* vol. 46, pp. 173-183, 2011.

[11] "SCC External Architecture Specification (EAS) (Revision 0.94)," Intel Labs, 2010.

[12] Graph500. *The Graph 500 Benchmark*. Available: http://www.graph500.org