A Study on the use of Performance Counters to Estimate Power in Microprocessors

Rance Rodrigues, *Member, IEEE,* Arunachalam Annamalai, *Member, IEEE,* Israel Koren, *Fellow, IEEE,* and Sandip Kundu, *Fellow, IEEE*

***Abstract*—We present a study on estimating the dynamic power consumption of a processor based on performance counters. Today's processors feature a large number of such counters to monitor various CPU and memory parameters such as utilization, occupancy, bandwidth and, page, cache and branch buffer hit rates. The use of various sets of performance counters to estimate the power consumed by the processor has been demonstrated in the past. Our goal is to find out whether there exists a subset of counters that can be used to estimate, with sufficient accuracy, the dynamic power consumption of processors with varying microarchitecture. To this end we consider two recent processor configurations representing two extremes of the performance spectrum, one targeting low power, while the other high performance. Our results indicate that only three counters measuring *(i)* the number of fetched instructions, *(ii)* level 1 cache hits and *(iii)* dispatch stalls, are sufficient to achieve adequate precision. These counters are shown to be effective in predicting the dynamic power consumption across processors of varying resource sizes achieving a prediction accuracy of 95%.**

***Index Terms*—Power estimation, performance counters, low power core (LP), high performance core (HPerf).**

I. Introduction

MONITORING power consumption of a microprocessor is important for power management to reduce power usage, improve battery life, and in meeting package thermal dissipation power limits. Power monitoring is also important in thread scheduling in Asymmetric Multicore Processors (AMPs) [1]. Dynamic power management requires online monitoring of the consumed power. Since direct online measurement of power at high frequencies is impractical, proxies must be used to estimate it [2], [3], [4].

Performance counters have been widely used as proxies to estimate processor power online [2], [3], [4]. Often, the selected counters and the expressions used for power estimation (based on these counters) differ from one architecture to another. There are two reasons for this: (i) differences in the accessibility and availability of counters in each architecture type [2], and (ii) a different set of counters may minimize power estimation error for a given architecture. We are interested in searching for a *universal* subset of performance counters that may be used to estimate power (with low error) on any architecture. The motivation behind this is the growing presence of AMPs [5] that consist of cores of different types. Having a single set of performance counters that can be used to estimate the power on each core would (i) greatly reduce the complexity of power estimation across architectures, (ii) guide chip designers in making the decision as to which performance counters should be accessible for accurate power estimation and (iii) help in making informed thread scheduling decisions.

For our study we consider two recent processor configurations at the opposite ends of the spectrum; a processor targeting low power (representative of an Intel Atom) and a high performance processor (representative of Intel Nehalem). Various performance counters are considered and shortlisted based on their correlation to power. Detailed analysis is presented for each core type on the possible choice of counters so that low (power) estimation error is achieved while using the smallest possible number of counters. Expressions for estimating the power are then derived and their sensitivity to the architecture type is analyzed. Our results indicate that although for different architectures, different sets of counters achieve minimum error (3%), there exists a subset of counters that can be used to estimate power for either architecture with adequate accuracy (average error of 5%). This error is considerably lower than that achievable by previously proposed schemes (9% [4] and 11.5% [3]). We also show that for small differences in architecture type, if the right counters are chosen, the expression obtained for one architecture may also be used to estimate power on the other with only a small increase in estimation error (∼3%).

II. Related Work

Most of the prior work on power estimation of processors is based on formulating expressions using performance counters [2]. This approach is easy to implement and requires no knowledge of the power consumption of the individual units, but the estimation accuracy greatly depends on the choice of the counters and the representative benchmarks used to obtain the expressions. Contreras *et al.* [2] employ hardware counters to estimate processor and memory power consumption. Instead of using a static power model for all components, Bansal *et al.* in [6] proposed a framework of heterogeneous power models. Li *et al.* [4] characterize the power behavior of a commercial Operating System (OS) for a wide range of applications using Instructions Per Cycle (IPC). Singh *et al.* [3] propose a power model using counters for a AMD Phenom processor. In [7], Bircher *et al.* explore the use of counters to predict power consumed by subsystems outside the processor. In all these schemes, the analysis is based on a single microarchitecture. No studies have been presented on using the same set of counters for different architectures or using expressions derived for one architecture, to estimate power on another. In this paper, we present such a study and compare the proposed scheme to prior work.

III. Methodology

To explore the possibility of microarchitecture-independent performance counters for estimating power, we consider two

The authors are with the Department of Electrical and Computer Engineering, University of Masschusetts, Amherst, MA, 01002 USA e-mail: {rodrigues, annamalai, koren, kundu}@ecs.umass.edu

This work was partially supported by an NSF-SRC grant number 1985.001.

very different cores. One core is suited for low power applications while the other for high performance applications, and we refer to these as LP core and HPerf core, respectively. These two cores are representative of Intel Atom and Intel Nehalem processors, respectively. Having two cores at opposite ends of the performance/power spectrum would clearly impact the accuracy of the scheme. The core parameters used for both core types were obtained from [8]. We used SESC as the architectural performance simulator [9] and employed Wattch [10] to calculate power. Since our experiments are run in a simulation environment, Wattch is used as the "golden" reference. We are aware that Wattch has an estimation error of almost 10% when compared to layout-level power estimation tools. Our focus here is on estimating instantaneous power and we are mainly interested in detecting changes in the power profile (which may trigger dynamic management schemes). Hence, comparing the estimated power to the power calculated by Wattch is satisfactory. For our experiments, we have selected a total of 38 benchmarks from SPEC [11], MiBench [12], and mediabench suites [13].

IV. Performance Counters

Performance monitoring counters reveal considerable amount of information about power consumption. These counters monitor different events that take place when a processor executes instructions (e.g., branch mispredictions, cache misses and TLB misses). We would like to confine our search to a small set of events and associated counters that would have the most impact on power consumption. In this study, we do not restrict ourselves to the counters that are available in current microprocessors. Our objective is to explore the possible choices, and if a high correlation to power is observed, we could make a strong proposition for including the corresponding counter. The counters studied by us include:

- Instructions per Cycle (IPC): Power consumption of a processor is dependent on its activity. If the IPC is high, the processor will very likely consume more power.
- Fetch counters: IPC considers only the retired instructions, but processors execute many instructions speculatively. These are flushed due to branch mispredictions but consume power. Hence, we keep track of *# Fetched instructions, Branch correct predictions (BCP)* and, *Branch mispredictions (BMP)*.
- Miss/Hit counters: Upon cache misses, the processor stalls. Thus, the events: *L1 hit, L1 miss, L2 hit, L2 miss, page hit and, TLB miss* may impact the power consumed.
- Retired instructions counters: Depending on the type of the retired instructions (Integer (INT), Floating-point (FP), Memory, Branch), different functional units are being exercised. If some of these are power-hungry (say FP), then, by monitoring the type of retired instructions, we would be able to estimate power more accurately.
- Stalls: Processors stall due to dependencies (data or resource conflicts). The Dispatch Stalls used by Singh *et al.* [3] (referred to as *Dispatch Stalls** in Fig. 1) includes stalls due to reservation stations, Reorder Buffer (ROB) and load/store queues (LSQ). In addition, we consider stalls due to register renaming and RAT (Register Alias Table). This counter is referred to as *Dispatch Stalls*.

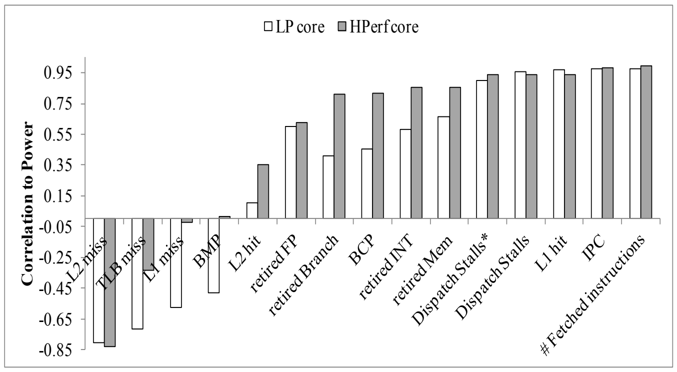

Fig. 1. Correlation of various performance counters to power consumed.

A. Power Modeling

Once the candidate counters are identified, the next step is to analyze the correlation of each one of them to power. To this end, we have selected the following 8 representative benchmarks (out of 38): INT intensive (*intStress,bitcount*), FP intensive (*fpStress,equake*), load/store intensive (*gcc*), have high IPC (*apsi*) and low IPC (*mcf,ammp*). These 8 benchmarks were run on both the LP and HPerf cores for 1 billion instructions, after skipping the initial 5 billion instructions to avoid simulating program initialization. The values of the counters were collected in intervals of 100K cycles along with power consumption from Wattch. We then computed the correlation between the normalized (with respect to the number of elapsed cycles) counter values and the power consumption (see Figure 1). The correlation was computed using the standard expression:

$$Correl(x,y) = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt[2]{\sum(x-\bar{x})^2\sum(y-\bar{y})^2}}$$

A high correlation is observed between the power and the *# Fetched instructions, L1 hits, IPC, Dispatch Stalls and retired memory instructions* counters. An estimation scheme involving a small number of counters is always desirable as it will save hardware and reduce the number of counters that have to be monitored simultaneously. In current processors the same counters are used for monitoring multiple events and it is not possible to simultaneously obtain the count for two different events from the same hardware counter [2]. To reduce the number of required counters, we investigated the mutual correlation between the performance counters. If two counters are highly correlated, only one of them is needed. For example *# Fetched instructions* is observed to have a high correlation with *Dispatch Stalls*, L2 misses, IPC, retired INT* and *BCP*. To select a specific counter from an equivalence set, a sensitivity analysis was performed by substituting one counter with another from the set and comparing the estimation accuracy. This process of elimination lead to a reduced set of just 3 counters: *# Fetched instructions, L1 hits* and *Dispatch Stalls*. This choice of counters is also intuitive. Processor power consumption depends on the activity in the L1 instruction/data caches and the stalls due to resource conflicts in the core. *Fetched instructions* indicates the amount of work done to get instructions from the L1 cache, the number of branch predictions performed etc. If there are misses in the L1 instruction cache, the processor will stall and power consumption will reduce. On the other

TABLE I
POWER MODELING PARAMETERS. IN THE TABLE, F- FETCHED INSTRUCTIONS, L1H - L1 HITS, L2M - L2 MISSES, BCP - CORRECTLY PREDICTED BRANCHES, IPC - INSTRUCTIONS PER CYCLE, FP - FLOATING-POINT INSTRUCTIONS, D* - DISPATCH STALLS*, D - DISPATCH STALLS.

#	Name	Expression for LP core: Power =	Expression for HPerf core: Power =
1	Ref.Exp1	8.07×IPC + 0.22	13.16×IPC + 1.57
2	Ref.Exp2	50.22×L2m + 10.68×IPC - 1.62×FP - 0.96×D* - 0.04	16.5×L2m + 23.3×IPC - 1.71×FP - 9.3×D* + 9.65
3	Exp1	8.53×F + 0.09	12.42×F + 1.49
4	Exp2	5.11×F + 4.6×L1h + 0.049	14.8×F - 2.51×L1h + 0.146
5	Exp3	5.22×F + 4.65×L1h - 0.207×D + 0.25	7.45×F + 6.45×L1h + 3.61×D - 3.19
6	Exp4	4.82×F + 5.07×L1h + 1.25×D + 38.3×L2m - 1.41	7.64×F + 6.28×L1h + 3.59×D + 17.6×L2m - 3.3
7	Exp5	6.13×F + 4.06×L1h - 0.47×D - 4.42×BCP + 0.5	9.19×F + 4.73×L1h + 3.02×D - 5.57×BCP - 2.7
8	Exp6	5.7×F + 4.5×L1h + 0.92×D + 35.3×L2m - 4.03×BCP - 1.06	9.53×F + 4.4×L1h + 2.98×D + 25.3×L2m - 5.81×BCP - 2.83

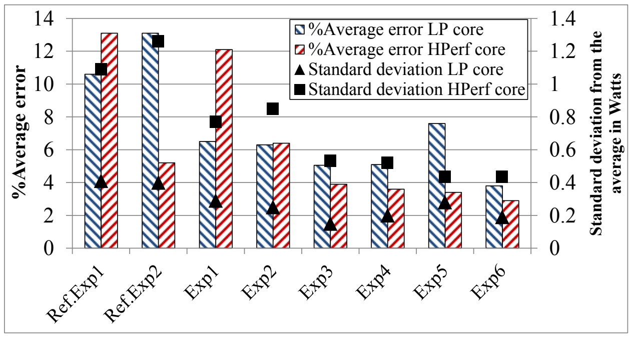

Fig. 2. Average error and standard deviation for all expressions.

hand, hits will result in increased activity and thus increased power. Similar behavior holds true for hits and misses in the L1 data cache. *Dispatch Stalls* measures the number of stalls in the processor due to resource conflicts. This counter thus measures the processor activity. The 3 counters selected are thus expected to yield expressions that provide fairly accurate power estimation. Additional counters may further increase the accuracy and they were considered in our study (see Table I). A multi-dimensional curve fitting and regression analysis was performed to obtain an expression for the estimated power for both the core types using various counters. This was conducted using the commercial tool Datafit 9.0 [14] that takes as input the counters and power values and outputs the coefficients of the expression used to estimate power. Curve fitting was done using the above mentioned 8 representative benchmarks. The expressions obtained for our scheme using each set of performance counters are referred to as Exp1 to Exp6. Similar expressions (Ref.Exp1 and Ref.Exp2) were derived for the reference schemes, proposed by Li *et al.* [4] (uses only IPC) and by Singh *et al.* [3] (4 counters used), respectively. The expressions obtained for our scheme and the reference schemes are shown in Table II.

V. RESULTS AND COMPARISON

In this section we evaluate the accuracy of each of the power expressions against the values obtained from Wattch. We first present the average error obtained for each expression over all the 38 benchmarks when run for 1 billion instructions. Detailed results including the frequency distribution of error are then presented for selected expressions. Since the effectiveness of the schemes used to manage power and temperature online rely on the accuracy of the instantaneous power estimated, we show the variance in error as a function of time. We conclude this section by evaluating the effect of changes in architecture type on the power estimation accuracy.

A. Average error over all workloads

The eight expressions were used to estimate the power of both core types for all the 38 workloads considered.

1) Average for all expressions: The average error (in %) and standard deviation for each expression is plotted in Figure 2. It can be seen that the average error for Ref.Exp1 is larger than 10% for both core types indicating that using IPC alone is insufficient. The error observed for Ref.Exp2 is high for LP (13%), but not for HPerf core (only 5.2%). This low error is due to the similarities between the HPerf core and the platform that Singh *et al.* [3] have used (an AMD Phenom core). Note that the error when using this expression to estimate power on the LP core is even higher than that when using the simple IPC metric (Ref.Exp1) demonstrating the platform dependence of the chosen counters. This error may also depend on the workloads used for training that expression. The workloads that will be executed on a core are generally unknown and hence, expressions based on training will always have limitations. When using the expressions studied by us (indicated by the Exp prefix in the figure), it can be seen that in general, an increase in the number of variables in the expression (indicated by the increasing postfix,1-6) results in an increasingly smaller power estimation error for both cores. An exception is the slight increase in error when using Exp5 to estimate power on the LP core. A probable reason is the fact that Exp5 is a function of correctly predicted branches which did not show high correlation to power on the LP core (see Figure 1). It can also be observed that if up to five counters are used (Exp6), a very low average error of 3% and 3.9% is observed for the HPerf and LP cores, respectively. However, going beyond Exp3, which is a function of three counters (*F, L1h* and *D*), does not yield notable benefits (4% for HPerf core and 6.1% for the LP core). A similar trend is observed with respect to the standard deviation. Hence, we conclude that Exp3 estimates power in both cores with a reasonably good accuracy.

2) Frequency distribution of error for selected expressions: In this subsection, we show the error frequency distribution for Exp3 as well as Exp6 when used to estimate power for the HPerf core. We also include Ref.Exp1 and Ref.Exp2 in the plot, for comparison purposes. Figure 3 shows that the error frequency distribution for Exp3, Exp6 and Ref.Exp2 is concentrated around zero while Ref.Exp1 has a longer tail. Comparing Exp3 and Exp6, we cannot justify the use of five counters as opposed to three. When considering the LP core, a similar trend was observed when evaluating Exp3 and Exp6. However, we have observed that Ref.Exp2 has a distribution that is worse than that of Ref.Exp1 for that core, which exhibits its platform dependence.

3) The confidence intervals of correlation coefficient: To test our confidence in the power estimation expressions that

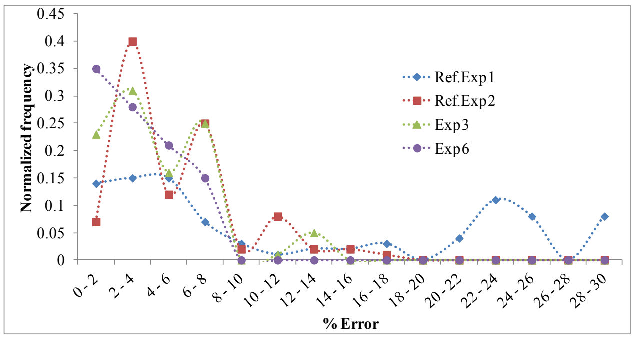

Fig. 3. Normalized frequency distribution of the error of selected expressions for the HPerf core. Points connected by dotted lines to show the trend.

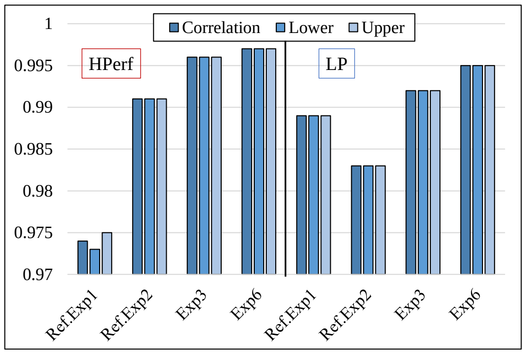

Fig. 4. The correlation and upper and lower levels of the confidence interval for a confidence level of 99%. Results are shown for selected expressions obtained for the HPerf and LP cores.

were deirved using curve fitting, we computed the confidence intervals of the correlation coefficient (*R*) [15] between the estimated and actual power (measured using Wattch) for selected expressions. The tool accepts as inputs the sample size and *R* and calculates the corresponding confidence interval. The results obtained for selected expressions on both the HPerf and LP cores for a confidence level of 99% are shown in Figure 4. Apart from Ref.Exp1 on the HPerf core, the lower and upper limits of the confidence interval for all other expressions were almost identical. This is due to the large sample size that we used in our experiments, e.g., 38K for the LP core and 7K for the HPerf core.

B. Error as a function of time

In addition to the average error, the instantaneous error of the estimated power is important as most online schemes rely on real-time information to make their decisions. Figure 5 shows the error in the estimated power of the LP core (vs. the power obtained from Wattch) as a function of time for Ref.Exp1, Ref.Exp2 and Exp3. It can be seen that the error does not show as much deviation from zero for Exp3 as that exhibited by the two reference expressions. Hence, we claim that Exp3 will provide better results when used to make decisions that depend on estimating instantaneous power. We have also observed that Ref.Exp2 shows a better time dependent behavior when estimating power for the HPerf core, compared to the Ref.Exp1, but do not show this figure due to space constraints.

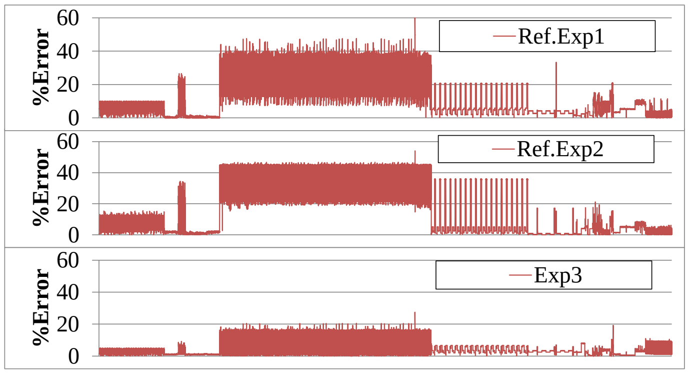

Fig. 5. Error as a function of time for various expressions evaluated on the LP core.

TABLE II
CONSIDERED VARIATIONS IN ARCHITECTURE OF THE HPERF CORE

Variation 1	Fetch width 2 and caches reduced by 50%
Variation 2	Fetch width 2, caches reduced by 50% and ROB, RAT and ISQ reduced by 25%
Variation 3	Fetch width 2, caches reduced by 50% and ROB, RAT and ISQ reduced by 50%

C. Evaluating the effect of changes in architecture on the power estimation error

We also conducted experiments to study the effect of changes in architecture type on the power estimation error when the expression derived for one architecture is used to estimate power on the other. Such an estimation may be useful in order to predict power in dynamically resizable architectures [16] or AMPs [5]. Our first step was to try the expression obtained for the LP core, to estimate the power on the HPerf core and vice-versa. We found this error to be very large. We then explored the impact on the error if there is only a limited difference between the two architectures. To that end, we considered three variations in the HPerf architecture, each increasingly different from the original architecture (Table I). Table III details the changes in architectural configuration of each variation considered. All the 38 workloads were run for 1 billion instructions on each variation of the HPerf core. The resulting errors between estimated and calculated power when using the expressions that were derived for the original HPerf core are shown in Figure 6. In general, using an expression derived on one architecture to measure power on another is expected to result in larger error. Still, some expressions are more sensitive to the change in architecture than others with Ref.Exp2 showing the highest sensitivity. Even for Variation 1, an error increase of 300% was observed. This increase was far larger than that observed for expressions using fewer counters for power estimation (Ref.Exp1, Exp1, Exp2, Exp3). The main reason for this is that not all the counters used in Ref.Exp2 were found to be architecturally independent in our analysis. From the various expressions proposed in this paper (Exp1 - Exp6), the error increases with an increasing variation in architecture, but it can be seen that for small to medium changes (Variation 1 and Variation 2), error increases of 3% and 8% were observed, respectively. This shows the relative architectural independence of the chosen performance counters. However, varying architecture even more (Variation 3) results in significant error (>20%) for all the proposed expressions.

In summary, these experiments show that for a small to medium variation in architecture, the expressions derived for one architecture may be directly used to estimate power on the

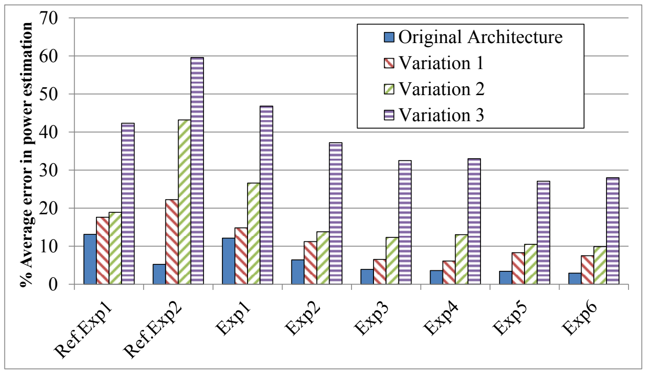

Fig. 6. Average error between estimated and calculated power (using Wattch) when the expression derived for the HPerf core is used to estimate power on a variation of that architecture. Variation 1, 2 and 3 are defined in Table III.

other, if the right counters are chosen. The power expression obtained for one architecture can thus be used as a first order estimate for the power on a small variation of that architecture.

D. Error as a function of window size

Choosing too small an interval may result in larger power estimation error as noisy behavior is expected in smaller intervals. Large intervals may lead to more accurate power estimation, but this will result in missing out on certain opportunities for energy or performance/power gains that may be made at smaller intervals [1]. Hence, it is important to arrive at a compromise between estimation error and interval length. It would be useful to also find an interval length that may be used across architectures. To this end, experiments were conducted on both HPerf and LP cores to determine a suitable interval length. Only Exp3 was considered in these experiments as it was observed that going beyond three counters does not yield notable benefits (see Figure 2). In these experiments, the expressions were recalculated for on each core type and every interval length. Each workload was then run on the two core types and the counters were sampled at intervals varying from 100K to 10M cycles. The average error obtained for each core type using various interval lengths is shown in Figure 7. It can be seen that increasing the interval length results in increasingly smaller error in power estimation for both the core types. However, for the HPerf core, going beyond a 100K-cycle interval does not result in appreciably smaller errors. For the LP core, this trend is observed at around 200K cycles interval length. We chose 100K cycle interval length since it seems to work reasonably well for both architectures. Even though the estimation error reduces when going from 100K to 200K cycles for the LP core, the reduction in error is only around 0.6%.

VI. Conclusions

We have presented a systematic study on the use of performance counters to estimate power online. We found a subset of counters (*# Fetched instructions, L1 hit and Dispatch Stalls*) that are suitable for estimating power across multiple architecture types within an error of 5%. This was illustrated using an OOO high performance (HPerf) and an in-order low power (LP) core. At 5% average error, the proposed estimator improves upon prior estimators with error of 9% [3] and 11.5% [4]. We have shown that for small to medium variations in architecture, the same power estimator may be used across architectures at a small loss of accuracy (3%). Such an estimator is useful for dynamic resource sizing of processors.

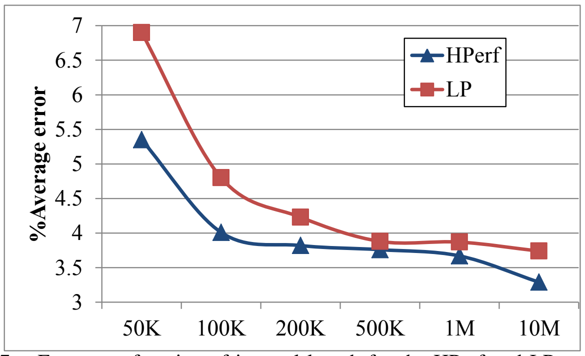

Fig. 7. Error as a function of interval length for the HPerf and LP cores.

References

[1] R. Rodrigues *et al.*, "Improving performance per watt of asymmetric multi-core processors via online program phase classification and adaptive core morphing," *ACM Trans. Des. Autom. Electron. Syst.*, Jan. 2013.
[2] G. Contreras and M. Martonosi, "Power prediction for intel xscale reg; processors using performance monitoring unit events," in *Low Power Electronics and Design, 2005. ISLPED '05.*, 2005.
[3] K. Singh *et al.*, "Real time power estimation and thread scheduling via performance counters," *SIGARCH Comput. Archit. News*, vol. 37, no. 2, Jul. 2009.
[4] T. Li and L. K. John, "Run-time modeling and estimation of operating system power consumption," in *Proc of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, ser. SIGMETRICS '03, 2003.
[5] P. Greenhalgh, "Big.little processing with arm cortex-a15 and cortex-a7," sep. 2011.
[6] N. Bansal *et al.*, "Power monitors: a framework for system-level power estimation using heterogeneous power models," in *VLSI Design, 2005. 18th International Conference on*, 2005.
[7] W. L. Bircher and L. K. John, "Complete system power estimation using processor performance events," *IEEE Trans. Comput.*, Apr. 2012.
[8] A. Annamalai *et al.*, "An opportunistic prediction-based thread scheduling to maximize throughput/watt in amps," in *Parallel Architectures and Compilation Techniques (PACT)*, 2013.
[9] J. Renau, "Sesc: Superescalar simulator," 2005.
[10] D. Brooks *et al.*, "Wattch: a framework for architectural-level power analysis and optimizations," in *Proc of the 27th annual international symposium on Computer architecture*, ser. ISCA '00, 2000.
[11] SPEC2000, "The standard performance evaluation corporation (spec cpi2000 suite)."
[12] M. Guthaus *et al.*, "Mibench: A free, commercially representative embedded benchmark suite," in *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*, dec. 2001.
[13] C. Lee *et al.*, "Mediabench: a tool for evaluating and synthesizing multimedia and communicatons systems," in *Proc of the 30th annual ACM/IEEE international symposium on Microarchitecture*, 1997.
[14] "Datafit 9.0 curve fitting software." [Online]. Available: http://www.oakdaleengr.com/
[15] Z. Lu, "Computation of correlation coefficient and its confidence interval in sas." [Online]. Available: http://www2.sas.com/proceedings/sugi31/170-31.pdf
[16] O. Khan and S. Kundu, "A model to exploit power-performance efficiency in superscalar processors via structure resizing," in *Proc of the 20th symposium on Great lakes symposium on VLSI*, 2010.