Effective Use of Performance Monitoring Counters for Run-Time Prediction of Power

W. L. Bircher, J. Law, M. Valluri, and L. K. John
Laboratory for Computer Architecture
Department of Electrical and Computer Engineering
The University of Texas at Austin
{bircher, law, valluri, ljohn}@ece.utexas.edu

ABSTRACT

Power dissipation and energy consumption have recently become first-order design constraints for microprocessors. In this paper we present the measured power dissipation of several SPEC benchmarks executing on a Pentium 4 processor. Using on-chip performance monitoring counters, we dynamically correlate relevant performance metrics such as instructions per cycle, second-level cache hits, and bogus/non-bogus instructions retired to the measured power. This quantitative analysis lends insight into selecting metrics to be used for predicting power. We found that incorrectly predicted branches had the strongest correlation, of 0.8979, to power dissipation, while execution time had the highest correlation to total energy consumption, 0.99.

1. INTRODUCTION

Processor design trends continue to pursue a performance-centric approach resulting in increasing microarchitectural complexities, clock frequencies and die sizes, all of which are pushing power dissipation and energy consumption close to the tolerance limits. Power dissipation impacts circuit reliability and packaging costs, while total energy consumption is a crucial metric in battery-limited systems such as laptops and other mobile devices.

Recently, several microarchitectural techniques such as dynamic adaptation of processor resources [1][2][3], dynamic voltage and frequency scaling [4][5], etc have been proposed to alleviate the power dissipation and energy consumption problems. Typically, these techniques estimate dynamic processor power dissipation by relying on information that can be obtained from the processor's performance monitoring counters. For example, the scheme proposed in [1] uses the IPC (instructions per cycle) of the executing application to manipulate the size of the issue queue. In this paper, we perform a detailed and quantitative analysis on the suitability of different performance counters for estimating processor power dynamically. We examine the correlation of power dissipation and energy consumption with performance metrics such as instructions retired per cycle, number of instructions squashed, L2 hits, L2 misses, number of branch mispredictions.

Our experimental setup includes: a clamp on current probe that measures the energy delivered to the voltage regulator modeul (VRM) and thus to the processor, the Brink/Abyss toolset for capturing data from performance monitoring counters, and tools we wrote to analyze the collected data.

Our evaluation using 6 SPEC CPU2000 benchmarks shows that:

- IPC alone is not an accurate indicator of power dissipation for all workloads.
- Any measure of IPC should include speculative instructions.
- When IPC is above 0.5 it can generate a good rough estimate of power dissipation, below 0.5 other metrics become more important.

It has been shown that compiler optimizations [6][7][16] can have a dramatic effect on total energy. We further show that for different compiler optimizations the correlation between any given metric and power varies significantly.

The rest of the paper is organized as follows. Section 2 describes prior research that is related to this work. The measurement methodology and experimental setup is explained in Section 3. We present our results in Section 4, followed by our final conclusions in Section 5.

2. RELATED WORK

In this section we present some of the prior research that is related to our work.

Valluri et al. [6] study the effect of compiler optimizations on power dissipation using the SimpleScalar simulator coupled with the Wattch architecture level model. However, this approach is based purely on simulation. Extending this work to the physical world, Seng and Tullsen [7] performed a similar analysis using a Pentium 4 system instrumented for power measurement. They use two series resistors in Vcc supply traces to measure the current delivered to the processor. However, they only present the average values of power dissipation for the complete execution of each benchmark. Furthermore, they only recorded the total number of micro-ops retired, instead of other interesting architectural metrics.

As in the Valluri paper, Li and John examine power dissipation using simulation in [8]. Their experiments showed that operating system power dissipation is strongly correlated to the easily observable metric, IPC. They also showed that a model composed of a few metrics obtained from performance monitoring counters estimates energy to within 1% of the simulated quantity. Kandemir et al examined the effect of several compiler optimizations on the energy consumption of multimedia benchmarks in [15]. Their study looked at energy consumed in both the processing core and in the memory subsystem as modeled by the SimplePower toolset.

Isci and Martonosi [9][10] present a runtime power modeling methodology based on using hardware performance counters to estimate component power breakdowns for the Intel Pentium 4 processor. Their approach involves measuring and using multiple metrics to perform intensive calculations to accurately produce live total and subunit breakdowns of Pentium 4 power dissipation.

Bellosa measured total energy consumption and qualitatively compared that to data gathered using performance monitoring counters in [11]. Their approach examined micro-ops executed, floating-point operations, L2 references and L2 misses, but neglected other relevant events. Their comparisons are also limited only to total energy consumption, not dynamic power dissipation. A strong correlation was noticed between total energy consumption and the measured events, but this correlation is not quantified. Also, as shown in [7] total energy is most directly related to execution time.

Unlike previous studies, which have only considered average values of power and IPC for an entire workload, we consider the correlation with a finer granularity. Additionally, we quantify our results using correlation coefficients. We present the correlation of average power during a benchmark's execution to several interesting metrics across multiple benchmarks and compiler optimizations, and the correlation of a trace of the power dissipation to a trace of the metric as the benchmark executes. Based on this data we can provide insight into the selection of easily observable metrics to predict power dissipation.

3. METHODOLOGY

This section explains the measurement methodology we utilized. Section 3.1 describes the power and energy measurement setup and Section 3.2 describes the performance measurement setup. Section 3.3 describes the benchmarks and the compiler options we used to create our workload.

3.1 Power and Energy Measurement

The two primary sources of experimental data required for our analysis are: processor power dissipation and easily observable processor performance metrics. Processor power dissipation refers to the rate of energy delivered to the Pentium 4 Xeon processor and no other support circuitry. For example, system memory, i/o bridges and disk drive power dissipation are not included here. On-chip level one and two caches are considered part of processor power. Isolation of processor power is easily performed due to the power supply implementation used on our target system. The conductors that supply current to the processor voltage regulator modules (VRM), supply current to no other components. No other devices obtain energy through these conductors. A diagram of the power measurement setup can be seen in Figure 1.

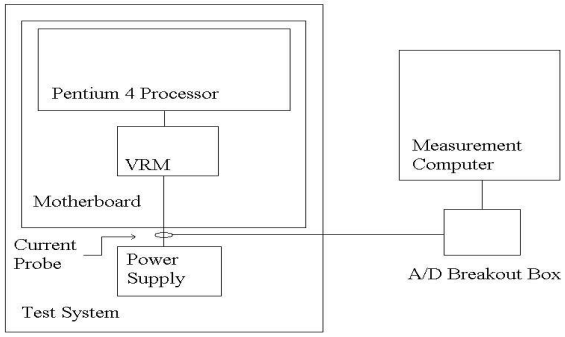

Figure 1: Power Measurement Setup

The power measurement shown is at the two power supply conductors supplying 12VDC to the processor VRM. An Agilent 1146A current probe reports the sum of current in these two conductors as a voltage (100mV/A). This probe detects current in a conductor by observing the magnetic field produced by the current. The observed conductors do not have to be cut/spliced, nor are shunt resistors needed. One issue of note regarding the location of the current probe is that a portion of the energy reported as being consumed by the processor is consumed by the VRM. For these modules the efficiency is on the order of 85%-90%. The reader should consider the 10%-15% loss when comparing results to manufacturer reported power dissipation. The voltage provided by the current probe is sampled at 10KHz by a National Instruments AT-MIO-16E-2 data acquisition card. The voltage trace can be interpreted by the LabVIEW software tool or as in our case it is written to a binary file.

3.2 On-Chip Counters

The second source of data is the on-chip performance monitoring counters (PMC) provided by the Pentium 4 Xeon processor. These counters provide a non-intrusive mechanism for observing a comprehensive set of metrics. Compared to the previous generation PMCs which had a similar number of observable metrics, these PMCs allow the concurrent observation of up to 18 distinct metrics [14]. All of the events used in this analysis were of the aggregate type. They report the aggregate count of the requested event between the assertion and deassertion of a software controlled enable flag. The event-based counters were not used here. Since configuration of the PMCs is restricted to operating system or privileged processes, a device driver is required for access by user-mode applications. The device driver used in our experiments is provided with the Brink/Abyss toolset [12]. This toolset runs under the Linux operating system and provides tools for simplifying PMC configuration and data acquisition. For our experiments the selected PMC are sampled and cleared at a rate of 50Hz and recorded by Brink/Abyss in ASCII type files.

3.3 Benchmarks and Compiler Optimizations

We chose several benchmarks from the SPEC CPU2000 suite that have been shown to have dissimilar execution characteristics [17]. Integer benchmarks included: gzip, eon, equake, and mcf. We also examined the floating-point benchmarks: mesa and vortex.

To compile the benchmarks we used *gcc* version 2.96 from a standard Red Hat 7.1 installation. We used similar optimization options as in [6]. Here is a brief description:

-O0 No optimizations performed.

-O1 The compiler turns on many local and a few global optimizations such as common subexpression elimination, copy propagation, code motion, and some minimal code scheduling.

***-**O2* Turns on all optional optimizations except for loop unrolling, function inlining, and register renaming, but will not perform optimizations that involve a space-speed tradeoff.

-O3 Optimize yet more. -O3 turns on all optimizations specified by -O2 and also turns on loop unrolling, function inlining, and register renaming.

-finline-functions Integrate all simple functions into their callers. The compiler heuristically decides which functions are simple enough to be worth integrating in this way.

-fschedule-insns If supported for the target machine, attempt to reorder instructions to eliminate execution stalls due to required data being unavailable.

-fschedule-insns2 Similar to `-fschedule-insns', but requests an additional pass of instruction scheduling after register allocation has been done. This is especially useful on machines with a relatively small number of registers and where memory load instructions take more than one cycle.

-funroll-loops Perform the optimization of loop unrolling. This is only done for loops who's number of iterations can be determined at compile time or run time.

All benchmarks are written in C except eon, which is written in C++ [13]. The specific compile options where not invoked when we compiled eon so they do not appear. Also, *-fschedule-insns* did not generate a working binary for mesa, so it does not appear.

4. RESULTS

Section 4.1 and 4.2 show the two methods we used to validate our measurement techniques. Section 4.3 presents results for interesting metrics observed using the PMCs and shows how they are correlated to the average power dissipation for each benchmark. Section 4.4 looks in depth at a single metric, IPC, presenting the IPC trace to instantaneous power trace correlation. The last section presents insights helpful in developing a more accurate understanding of the Pentium 4 Xeon power characteristics.

4.1 Basic Power Characterization

To validate our measurement techniques we first measured the idle, minimum, and maximum power of the processor and compared our measurements with known quantities from [7]. Idle power was found using two techniques. The first was to observe power dissipation with no user-mode applications active. Operating system processes were still active. A second approach was used to validate this information. A small application was written which issued the Unix sleep() function for several seconds. Similar results were found for both cases and average power was approximately 10W. To determine minimum power we wrote a small program that had an IPC of 0.09. While executing this program the processor was forced out of any power savings mode. The average measured power was roughly 30W. Determination of maximum power was more challenging since none of the normal workloads tested utilized the processor sufficiently to reach maximum power. Our approach was to write a small, highly customized code sequence which scheduled instructions to optimally utilize the Pentium 4 resources. The best case found yielded an IPC of 2.25 and power dissipation of 50W. All of these measurements are inline with expected power dissipation for the 2.2GHz Pentium 4 Xeon processor.

4.2 Compiler Optimizations

To further validate our method of measuring the power dissipation by non-invasively measuring the current supplied to the VRM and thus the power dissipated by the processor we reproduced the results of [7]

Figure 2 presents the reduction in power as different compiler optimizations are invoked. The columns show normalized power. These results are typical of all results we found for integer and floating-point workloads. Power dissipation is largely insensitive to compiler optimizations. Observed variations showed a maximum of 5% difference in average power dissipation. Average power dissipation is not significantly affected by any of the compile options we examined. However, we found that energy consumption is clearly dependent on the use of any compiler optimizations. This is due to a combination of the major effect optimizations have on application execution time and the minor effect they have on power dissipation.

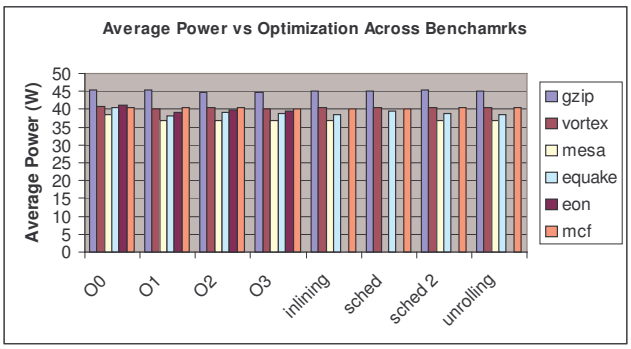

Figure 2: Average power of benchmarks by optimizations

Largely these results support the findings in [6] which use a simulated processor power model, and duplicate the measured results of [7]. The primary deviation is magnitude of the effect compiler optimizations have on power dissipation. For similar workloads (integer compression) the simulated model predicts average power reductions of about 10%. The experimental approach showed reductions closer to 1%. Not surprisingly, execution time reductions were consistent.

4.3 Interesting Metrics

Using the PMCs we measured event counts such as instructions retired, micro-ops retired (uop), L2 hits and misses, branch mispredictions, types of branches retired, loads and stores retired, and trace cache activity. Using this data, metrics such as IPC, uPC, L1 miss rate, L2 miss rate, and branch mispredictions per

instruction can be developed. While we were gathering the PMC data we also recorded the average power dissipation. Presented below in Figure 3 is the correlation of the average power dissipation for each benchmark to each metric across all of the benchmarks we examined.

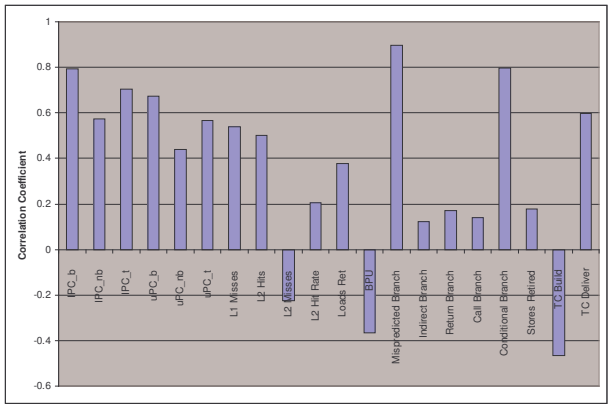

Figure 3: Correlation coefficient of performance metrics to average power

We expected IPC to have the strongest correlation to power of all metrics that we observed, so we considered three cases of measured IPC. The first version of IPC (IPC_nb) counts only non-bogus instructions, or the instructions that complete to retirement without being squashed. The second version of IPC (IPC_b), counts just the bogus instructions that existed in the pipeline at some time. The last version of IPC (IPC_t) counts all instructions, whether they are bogus or not. We expected IPC_t and IPC_nb to have the strongest relationship to power, but we found IPC_b to have the strongest relationship. The correlation coefficient of IPC_t is 0.7052, of IPC_nb is 0.5735, and of IPC_b is 0.7929.

We also measured uPC or micro-ops retired per cycle. The same trend occurs with uPC as does with IPC. The correlation coefficients to the average power are: uPC_t = 0.5654, uPC_nb = 0.4397, and uPC_b = 0.6741. It is surprising that uPC as a metric is not more closely related to power than IPC. This leads us to recommend using IPC instead of uPC as a performance metric for estimating power dissipation.

The strong correlations to power of mispredicted branches per instruction (0.8979) and conditional branches per instruction (0.7985), reinforces the importance of a highly accurate branch predictor.

Another important metric we observed is L1 misses per cycle. We found this to be more closely related to power than the L2 hit rate. This shows that the penalty for missing the L1 is important in both performance and power dissipation, i.e. good L2 cache performance is necessary and a metric to capture this is important in predicting power. We present a further proof of this in Section 4.5.

4.4 Correlating Power Traces to IPC Traces

In this section we will show that an easily observable metric such as Instructions Per Cycle (IPC) can be used to model microprocessor power dissipation. For a non-speculative processor the selection of IPC is intuitive. Performing logical work in the functional units at a given rate should require energy consumed at a directly proportional rate. Unfortunately, this intuitive model doesn't really hold for a speculative processor. The metric IPC doesn't represent the quantity of instructions which were fully or partially executed, but were later cancelled due to a branch misprediction. For a combination of processor and an application that attains a very high prediction accuracy it is reasonable to neglect power due to incorrectly speculated instructions. Fortunately, in this study, we were able to improve accuracy of the power-IPC model by improving the assumption about mis-speculated instructions. Because the Pentium 4 provides visibility to the number of mis-speculated instructions as well as completed instructions, we were able to capture the effect of the mis-speculated instructions. Our model assumes that cancelled instructions are detected late enough in the pipeline that they consume a quantity of energy similar to that of a committed instruction. This seems reasonable considering the large amount of logic in the front-end of x86 processors required for decoding.

To further solidify our understanding of the relationship of power to IPC we correlated the power trace to the IPC trace. It is important to note that this is different than correlating average power to average IPC as presented above. Table 1 shows this correlation coefficient for each version of the benchmarks we examined.

Table 1. Power Trace to IPC Trace Correlation Coefficients

	gzip	eon	equake	mcf	mesa	vortex
O0	0.734572	0.756487	0.897633	0.621187	0.691433	0.751876
O1	0.288616	0.621848	0.615983	0.577581	-0.17307	0.536567
O2	0.617906	0.553813	0.597922	0.575241	-0.00519	0.117081
O3	0.60198	0.771191	0.810057	0.251721	0.075371	0.643934
inlining	0.568419	n/a	0.758938	0.315535	0.084092	0.67633
sched	0.567632	n/a	0.382067	0.586439	n/a	0.707083
sched 2	0.32482	n/a	0.825607	0.594623	0.07331	0.611635
unrolling	0.563678	n/a	0.644375	0.585459	0.064383	0.673443

We expected the trace-to-trace correlation to be higher for all benchmarks than the average correlation. This was not the case. Table 1 shows that looking at just a single case for each benchmark would not give the full picture. For example, looking simply at the O0 optimization point, IPC alone would appear to be a useful metric for predicting power dissipation. However, any of the other optimization points show that IPC alone does not consistently correlate to power.

To help understand visually what Table 1 is describing, the power and IPC traces for the integer benchmark Vortex are given in Figure 4. The samples shown are normalized to one for clear comparison. The graph shows a distinct phase behavior, which is very similar between the two traces. One of the more significant differences is the range of values taken by the traces. The IPC trace varies greatly, swinging from the unnormalized peak of almost one to almost zero. Conversely, the power trace rarely drops below 60% of its peak value. Using the results for idle power in Section 4.1, it is clear that the minimum steady state power dissipation is lower than the instantaneous values observed in 4.3. The likely difference in the two measurements is due to the presence of capacitance in the power supply and processor transistors. When processor utilization changes rapidly, short deficits or surpluses of energy are provided or stored by the various capacitances in the system. Therefore, the VRM is not

required, nor able, to provide or absorb the extra energy instantaneously. The very brief minimum and maximum currents are being filtered from our view.

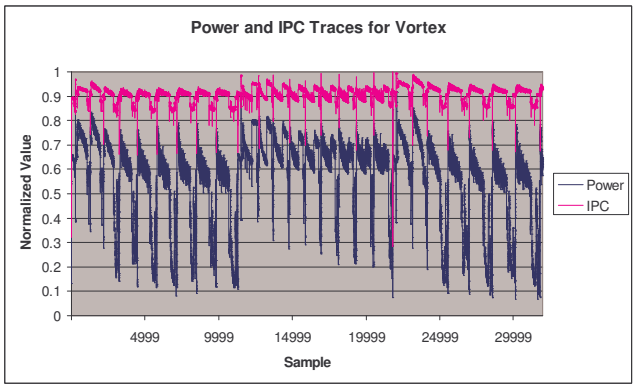

Figure 4: A sample of power and IPC traces for Vortex Correlation coefficient = 0.7518

Figure 5 also depicts a relationship between IPC and power dissipation. In this chart normalized power is plotted on the vertical axis while normalized IPC is on the horizontal axis. For IPC above 0.5 there is a clear positive correlation (0.697). However, IPC values less than 0.5 show almost no relation to power.

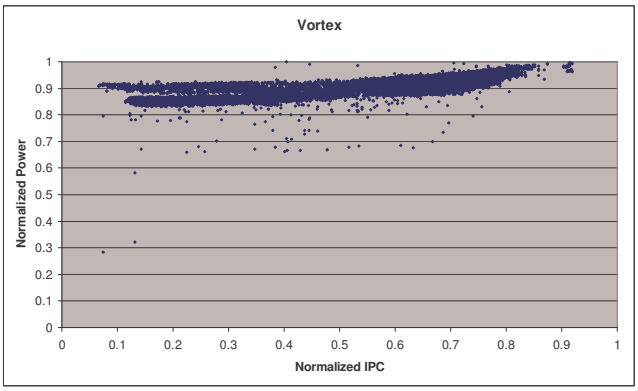

Figure 5: A sample graph of power compared to IPC

It is interesting to note that though the average correlations presented above in Section 4.3 show a strong relationship between IPC and power several of the benchmarks show little correlation when compared trace to trace. In fact, mesa seems to be almost completely uncorrelated. In Figure 6, looking more closely at a portion of the mesa traces representing just a few seconds of the entire benchmark's execution, we can see why the trace-to-trace correlations for mesa are so low. When optimized, mesa's power and IPC traces are very flat, and the variations from the mean are relatively small. To the eye it appears that there is a pattern in both traces and further examination using a FFT might be appropriate. Mesa is the only benchmark we noted that had this characteristic, so we did not look into this further.

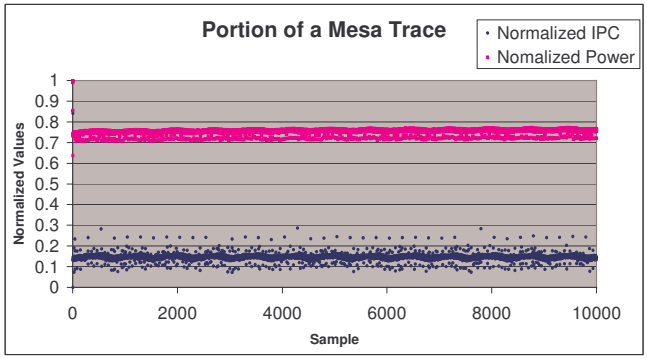

Figure 6: Portion of a Mesa trace showing small variations Correlation coefficient = 0.0754

4.5 Other Observations

When more in depth measurements were taken on the different optimization levels we found that when the largest improvements were made in execution time and hence total energy, the branch prediction accuracy, the L2 hit rate, and the percentage of issued instructions that were not squashed all increased. In other words, the speculation performed was more accurate.

In order to gain a lower CPI the amount of speculative instructions that end up being squashed must be reduced. These wasted instructions incur a performance penalty as well as an energy and power penalty. Improving branch prediction accuracy through compiler optimizations helps reduce incorrect speculation.

For example, for the equake benchmark compiled with –O3, the number of bogus instructions that were squashed is less than half that of the O0 case. Both equake and gzip compiled with –O3 show the largest energy/performance improvement, i.e. the total energy and the execution time is half that for the O0 case. For both benchmarks, the number of mispredicted branches is reduced by over half. For mesa and vortex, benchmarks which show roughly a 20 percent performance improvement, the number of mispredicted branches is reduced by 10 percent or less.

We found that the branch predictor is almost always better than 90 percent accurate. For the floating-point benchmarks and gzip the accuracy hardly varies even from the O0 case, though the energy shows a dramatic improvement beyond O0. Even these small variations are highly related to the average power. We observed correlations as high as 0.8979. A single branch misprediction leads to several speculative instructions that only waste energy.

Figure 7 below shows the average power compared to average IPC. Note that mcf stands out from the other benchmarks in this graph even though it does not in Table 1, and mesa blends in with the rest of the benchmarks here even though it stands out in Table 1. Examining the execution characteristics of mcf in more depth we found that the low IPC is primarily caused by a high L1 miss rate and poor L2 performance. In fact, the L2 hit rate for mcf was the lowest of all observed at 61.7%. Equake had an L2 hit rate of 74.5% but the L1 hit rate was much higher than mcf's. This reinforces the fact that L1 miss rate is a metric that should be included in a power model. All other benchmarks had an L2 hit

rate of over 90%. With mcf included in the correlation of average power to IPC over all of the benchmarks the result is 0.72. Removing mcf improves the correlation coefficient to 0.91. This shows the danger of looking at only a limited set or workloads.

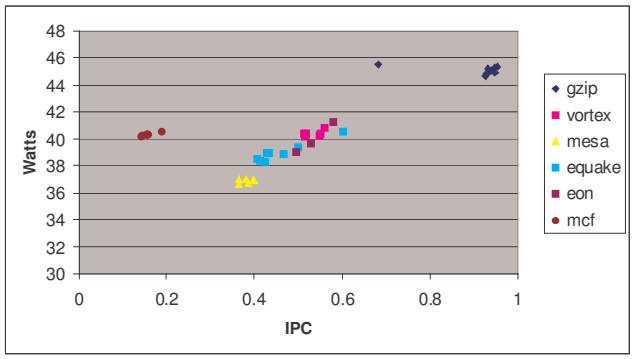

Figure 7: Average power compared to IPC_t

5. CONCLUSION

We have shown that microprocessor power dissipation can be predicted at a fine resolution using an easily observable metric such as IPC, given that the IPC is higher than 0.5. Also, it is important that any IPC measure include speculative instructions. Using additional PMC information such as the L1 miss rate and the branch misprediction rate, we were able to extend the accuracy of the power estimation. Finally, we provide insight into selecting easily observable metrics for estimating power or developing a power model.

6. REFERENCES

[1] D.Folegnani and A.Gonzalez. Energy-effective issue logic. In *28th International Symposium on Computer Architecture*, Jun. 2001.

[2] A.Buyuktosunoglu, S.Schuster, D.Brooks, P.Bose, P.Cook, and D.Albonesi. An adaptive issue queue for reduced power at high performance. In *Workshop on Power-Aware Computers Systems*, held in conjunction with ASPLOS, Nov 2000.

[3] F. Bellosa. The benefits of event-driven energy accounting in power-sensitive systems. In *Proceedings of 9th ACM SICOPS European Workshop*, September 2000.

[4] Alexander Klaiber, The Technology Behind Crusoe Processors, Transmeta Corporation, Jan 2001.

[5] Intel StrongARM SA-1100 Microprocessor Developer's Manual, Intel Corporation, April 1999.

[6] M. Valluri and L. John. Is Compiling for Performance == Compiling for Power? In *Proceedings of the 5th Annual Workshop on Interaction between Compilers and Computer Architectures*, Jan. 2001.

[7] J. S. Seng and D. M. Tullsen. The Effect of Compiler Optimizations on Pentium 4 Power Consumption. In *Proceedings of the 7th Annual Workshop on Interaction between Compilers and Computer Architectures*, Feb. 2003.

[8] Tao Li and Lizy John. "Run-Time Modeling and Estimation of Operating System Power Consumption", Sigmetrics '03, June 10-14, 2003.

[9] C. Isci and M. Martonosi. Identifying Program Power Phase Behavior Using Power Vectors. In *Proceedings of the 6th Annual IEEE International Workshop on Workload Characterization*, Oct. 2003.

[10] C. Isci and M. Martonosi. Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data. In *Proceedings of the 36th International Symposium on Microarchitecture*, Dec 2003.

[11] F. Bellosa. The benefits of event-driven energy accounting in power-sensitive systems. In *Proceedings of 9th ACM SICOPS European Workshop*, September 2000.

[12] B. Sprunt. *Brink and Abyss: Pentium 4 Performance Counter Tools for Linux*, Oct. 2003. http://www.eg.bucknell.edu/~bsprunt/emon/brink_abyss/brink_abyss.shtm

[13] The Standard Performance Evaluation Corporation. *SPEC CPU2000 Suite*, Sept. 2003. http://www.specbench.org/osg/cpu2000/.

[14] B. Sprunt. Pentium 4 performance-monitoring features. *IEEE Micro*, 22(4):72-82, Jul/Aug 2002.

[15] M. Kandemir, N.Vijaykrishnan, M. Irwin, and W. Ye. Influence of compiler optimizations on system power. In *Design Automation Conference*, July 2000.

[16] L. Chakrapani, P. Korkmaz, V. Mooney, K. Palem, K. Puttaswamy, and W. Wong. The emerging power crises in embedded processors: What can a (poor) compiler do? In *CASES 2001*, Nov. 2001.

[17] A. Phansalkar and L. K. John. Analyzing Program Behavior of SPECint2000 Benchmark Suite using Principal Components Analysis. Tech. Report TR-040122-01, Laboratory for Computer Architecture, The University of Texas at Austin, January 2004