Runtime Identification of Microprocessor Energy Saving Opportunities

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

ABSTRACT
High power consumption and low energy efficiency have become significant impediments to future performance improvements in modern microprocessors. This paper contributes to the solution of these problems by presenting: linear regression models for power consumption and a detailed study of energy efficiency in a modern out-of-order superscalar microprocessor. These simple (2-input) yet accurate (2.6% error) models provide a valuable tool for identifying opportunities to apply power saving techniques such as clock throttling and dynamic voltage scaling (DVS). Also, future work in improving energy efficiency is motivated by a detailed analysis of SPEC CPU 2000 workloads. The vast majority of workloads are found to yield very low energy efficiency due to the frequency of level two (L2) cache misses and misspeculated instructions.

Categories and Subject Descriptors
C.0 [**Computer Systems Organization**]: General

General Terms
Measurement, Performance and Experimentation.

Keywords
power, modeling, energy efficiency, speculative microprocessors

1. INTRODUCTION
Microprocessor performance has been increasing exponentially for the last three decades. Unfortunately, this impressive rate of improvement seems to be slowing. Excessive power consumption is reducing the rate of performance improvement. The four primary issues are excessive cooling costs, low energy efficiency, power supply limitations and reduced reliability. This paper makes several important contributions to alleviate the power bottleneck. We present a detailed characterization of the Pentium 4 processor, a representative modern superscalar processor. This characterization is important because it sheds light on the areas of the microprocessor cores that require further optimization for power consumption. We then propose lightweight, linear regression-based software power/energy models to predict run-time power and energy efficiency. We first identify the most representative performance metrics and show that an accurate power model (97.5%) can be constructed using only two metrics. Our models can be used to enable effective application of power saving techniques. For example, power/energy models can be used for program phase detection. Techniques such as clock throttling could be applied in high power phases to limit peak power consumption and consequently peak case temperature. This allows for use of lower cost/complexity cooling mechanisms and power supply circuits. Similarly, detecting phases of low energy efficiency allows for the application of more aggressive power saving techniques such as DVS.

Another important contribution of this paper is a detailed analysis of processor energy efficiency for the SPEC CPU 2000 benchmark suite. We motivate the development and application of further improvements in energy efficiency by showing that the majority of workloads operate at a less than 50% of maximum efficiency. We find efficiency to range from 5.04 to 133.55 Muops/Joule (Muops/Joule ≈ MIPS/Watt). We provide insight into this observation by showing that like performance (MIPS), energy efficiency (MIPS/Watt) is dominated by the effectiveness of the L2 cache and the branch predictor.

In section 2 we compare and contrast related work. Section 3 describes our experimental methodology including power and performance monitoring counter sampling and linear regression modeling. Results for our power models are presented in Section 4. Section 5 presents a characterization of energy efficiency for the SPEC 2000 suite. In section 6 we conclude by summarizing our findings. Section 7 suggests future work.

2. RELATED WORK
Previous research suggests that microprocessor power consumption is primarily determined by the number of instructions retired per cycle (IPC). Li et al [8] present a simple linear model for power consumption by operating system services. The resultant models are a function of only IPC. Their modeling only considers operating system routines and requires a separate model for each operating system routine. Most importantly, their model is simulation-based and consequently not representative of real systems. In contrast, our model is based on real measurement and shows that power depends more on fetched uop/cycle rather than IPC. In addition, while their model was dominated by the IPC component, ours is dominated by a constant component. We show that our real processor has a minimum or constant power consumption of about 36 Watts. Bellosa [2] uses synthetic workloads to demonstrate a correlation between observable performance events and power consumption. He shows that a

correlation exists for: uops/sec, fuops/sec, L2 accesses/sec and memory accesses/sec. Since only synthetic workloads are characterized, these results do not contain the effect of complex interactions between performance events in real workloads. Isci et al [5] build a comprehensive power model based on utilization factors of the various components of the processor. Using 22 performance monitoring counters they are able to model average power consumption of SPEC2000 workloads within 5%. Our model yields similar accuracy, yet with only 2 PMC metrics. Also, instead of explicitly modeling each component of the processor, our model combines the contributions of multiple components into just two combined with fixed minimum power consumption.

Lee et al [6] uses a similar detailed model of power and extends it to predict processor case temperature. Valluri et al [14] consider the effect of compiler optimization on power and energy consumption. The analysis is performed using the Wattch [3] architectural simulator. They found that power consumption was related to IPC while energy consumption is dominated by program execution time. Seng et al [12] validate Valluri's findings on real hardware. They found that the energy consumption is determined by program execution time. While these studies do offer a comparative study of energy efficiency in real user workloads, they do not suggest how efficient the processor is with respect to the maximum.

3. METHODOLOGY

In this section, we describe our experimental approach including power sampling, performance monitoring counter sampling and issues surrounding construction of a linear-regression power model.

3.1 Power Sampling

For this study instantaneous processor power consumption was measured using a clamp-on current probe. The probe, an Agilent 1146A, reports the current passing through its sensor by detecting the magnitude and polarity of the electromagnetic field produced by the sampled current. This type of measurement simplifies instrumentation since the observed conductors do not have to be cut to insert current sensing resistors. The drawback of this approach is that only wire-type conductors can be sampled. It is not possible to sample conductor embedded in the printed circuit board. For our target system this restricts power measurement to the input conductors of the processor voltage regulator module (VRM). As a result, a portion of the reported power consumption is actually attributed to the inefficiency of the VRM. These modules have an efficiency of 85%-90%. The reader should consider the 10%-15% loss when comparing results to manufacturer reported power consumption. The voltage provided by the current probe is sampled at 10 KHz by a National Instruments AT-MIO-16E-2 data acquisition card. The LabVIEW software tool can interpret the voltage trace or as in our case it is written to a binary file for offline processing.

3.2 On-chip Counters

The second source of data is the on-chip performance monitoring counters (PMC) provided by the Pentium 4 processor. These counters provide a non-intrusive mechanism for observing a comprehensive set of metrics. Compared to the previous generation PMCs that had a similar number of observable metrics, these PMCs allow the concurrent observation of up to 18 distinct metrics. 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. 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 [13]. This toolset simplifies PMC configuration and data acquisition. For our experiments, the selected PMCs are sampled and cleared at a rate of 100Hz and recorded by Brink/Abyss in ASCII file format. The reader should note that the performance metric "uop" or micro-op is used throughout the paper. For the Pentium 4 processor this is similar an instruction. We will describe some of the finer distinctions in section 4.1.1.

3.3 Linear Regression Model

When using a simple linear model such as ours it is important to select representative inputs. Since we cannot directly represent complex behaviors it is critical to select input data that does not only represent a subset of all behavior. When combining the effects of numerous complex events, care must be taken in ensuring an equal representation of all events in the training data. To that end, we use a clustering approach demonstrated by Phansalkar et al [11]. They show that SPEC 2000 benchmarks can be clustered into groups of ten discernable behaviors. Programs with like behaviors are clustered together. Ideally, the cluster member with the behavior most representative of the other cluster members could safely be used for exploring the behavior space of its cluster members. Our approach was to use one member of each cluster for model validation. The remaining cluster members were used for creation of the linear model. Table 1 lists the validation workloads in bold and the model input workloads in normal text. Since clusters 2, 3 and 8 have only one member, those benchmarks were used in creating and validating the models. Additionally, two synthetic benchmarks were used in building the model to explore the minimum (lowipc) and maximum (maxipc) power consumption behavior. The resultant model power model takes the simple slope-intercept form. We describe the components of this model in the form power= $\alpha_0 + \alpha_1 \cdot \text{metric}_1 + \ldots + \alpha_n \cdot \text{metric}_n$

Table 1. CPU 2000 Clustering

Cluster	Name
1	**galgel**
2	**twolf**
3	**lucas**
4	**applu**, mgrid
5	**gcc**, crafty, parser, vortex, vpr
6	**gzip**, bzip2
7	**equake**, fma3d, art
8	**mcf**
9	**apsi**, swim, wupwise, ammp
10	**eon**, mesa

4. POWER MODELS

4.1 Model Input Selection

While past research [8][2] and intuition suggest that IPC should be a component of any power model, we chose to consider a larger array of metrics for building our models. Correlation coefficients were calculated for all twenty-three observed PMCs. Initially, we attempted to find correlation across multiple sample points in a single workload trace. However, we found that minor discrepancies in alignment of the power trace to the PMC trace could cause large variations in correlation. Since we had such a large set of workloads we chose to use each workload as a single data point in our correlation calculation. For each metric we found its average rate across each workload. For most, the metrics were converted to event/cycle form, but a few were in other forms such as hit rates. Additional derived metrics were added such as completed uops/cycle (retired + cancelled uops). A subset of the correlation results can be seen in Tables 2 and 3.

Table 2. High Correlation (>0.32)

Metric	Correlation
Spec Del/Cycle	0.89
Fetched Uop/Cycle	0.84
Instr Total/Cycle	0.84
Completed Uop/Cycle	0.83
Load/Cycle	0.80
Uop/Cycle	0.79
Branch/Cycle	0.78
Stores/Cycle	0.64
Mispred Branch/Cycle	0.41
L2 Miss/Cycle	-0.33
Cancelled Uop/Cycle	0.33

Table 3. Low Correlation (<0.32)

Metric	Correlation
L2 Hit/Cycle	0.31
Bus Access/Cycle	-0.31
TC Del/Cycle	0.32
Bus Util	-0.31
Fp Op/Uop	-0.22
Prefetch Rate	0.17
TC Build/Cycle	-0.15
ITLB Hit/Cycle	-0.09
TC Miss/Cycle	-0.09
ITLB Miss/Cycle	-0.04
L2 Hits/Cycle	0.03
L2 Access/Cycle	-0.02

As expected the IPC-related metrics show strong correlation. One of the more unexpected findings is the weak negative correlation of floating instruction density (ratio of all dynamic instructions). This is in contrast to past findings [2] that show a strong correlation between floating point operations per second and power. Later in section 4.1.2 we will give an explanation. Another unexpected result was the lack of correlation to data prefetch rate.

Our research shows that rather than considering just IPC, a more accurate model can be constructed using a metric that encompasses power consumed due to speculation. Figure 1 shows the average number of uops for the SPEC 2000 benchmarks that are fetched, completed and retired in each cycle. Table 4 shows what portions of fetched uops complete or retire, for each of the twenty-four benchmarks. The first bar in Figure 1 "Fetch" shows the number of uops that are fetched from the Trace Cache in each cycle. The second bar "Complete" shows the sum of uops that are either retired or cancelled each cycle. Cancelled uops are due to branch misprediction. The third bar, "Retire", shows only uops that update the architectural state. This figure shows that the processor fetches 21.9% more uops than are used in performing useful work. Therefore, a more accurate power model should use the number of uops fetched per cycle instead of the number retired. Table 5 provides a comparison of linear regression power models based on the three previously mentioned metrics.

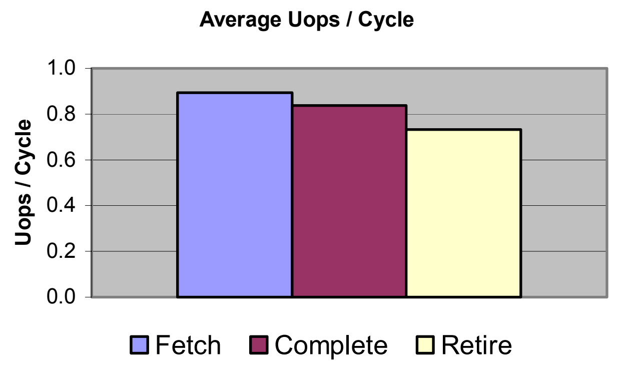

Figure 1. Throughput Metrics

Table 4. Percent of Fetched Uops

Name	%Complete	%Retire	Name	%Complete	%Retire
gzip	92.7	69.8	swim	99.9	99.7
vpr	85.3	60.0	mgrid	99.1	98.6
gcc	94.2	77.7	applu	98.7	96.6
mcf	63.0	31.5	equake	96.8	93.5
crafty	94.6	78.4	sixtrack	99.2	97.8
bzip2	92.0	72.1	mesa	92.1	75.2
vortex	98.0	95.0	art	84.9	77.5
gap	92.8	73.5	facerec	95.5	90.5
eon	91.7	81.5	ammp	94.8	88.5
parser	90.1	69.0	fma3d	97.0	94.3
twolf	85.2	55.2	lucas	99.9	95.9
wupwise	97.0	91.0	apsi	97.1	93.6

4.2 IPC Related Power Models

Twenty-three processor performance metrics were examined for their correlation to power consumption. The most correlated metrics were all similar to instructions per cycle (IPC). Using this finding as a guide we constructed numerous linear models using regression techniques. Power is calculated as the sum of a positive constant α_0 and the product of another positive constant

α_1 and a performance metric $metric_1$. Results for seven of the best models are listed below in Tables 5,6 and 8. Tables 5 and 6 support our hypothesis regarding fetched uops as being the most representative of IPC type metrics. The worst of these metrics was the familiar Instructions Per Cycle. This is caused by the lack of a one-to-one mapping of instructions to uops. Many x86 instructions map to a sequence of uops. For example, a single ADD instruction that uses memory as its source and destination is actually composed of three uops. The first uop loads a value from memory, the second adds a register or immediate to the value from memory and the third stores the result back to memory. Alternatively, an ADD instruction that does not use memory as an operand has a one-to-one mapping of instruction to uop. Assuming all uops consume the same amount of power, the instruction that uses memory would consume three times as much power.

Of the uop-based models fetched uops is the most representative metric for power consumption. This suggests that uops that do not update the architected state of the machine still consume a significant amount of power. For the case of cancelled uops, this is not to surprising since these uops did complete execution but were not retired. So, they would have traversed nearly the entire processor pipeline consuming a similar power as retired uops. More surprising is the effect of fetched uops on the power model. Fetched uops includes retired and cancelled. It also includes the remaining uops that were cancelled before completing execution. Since fetched uops provides the most accurate model, cancelled uops must be consuming a significant amount of power.

Table 5. Uop Linear Regression Model Comparison

	Retire uop/cyc		Complete uop/cyc		Fetch uop/cyc	
Coefficients	α_0	α_1	α_0	α_1	α_0	α_1
	36.3	4.37	35.8	4.44	35.7	4.31
Avg Error	3.26%		2.8%		2.6%	
Coefficient of Determinatio n	0.696		0.735		0.737	

Table 6. Instruction Linear Regression Model Comparison

	Retire instr /cyc		Complete instr/cyc	
Coefficients	α_0	α_1	α_0	α_1
	36.8	5.28	36.3	5.52
Avg Error	5.45%		4.92%	
Coefficient of Determination	0.679		0.745	

These models generate minimum and maximum power values (36W – 47W) similar to what [2] found on a Pentium 3 (31W-48W) with similar uop/cycle ranges (0 – 2.6). The stated average error values were found using the validation set described in Table 1.

4.3 Micro ROM Related Power Models

The power models in Tables 5 and 6 perform best when applied to workloads mostly composed of integer-type instructions (SPEC-INT). However, larger errors result for workloads with high rates of floating point instructions (SPEC-FP). Isci et al [5] demonstrates that FP workloads such as equake use complex microcode ROM delivered uops. While the complex instructions execute, microcode ROM power consumption is high, but total power consumption is reduced slightly. In order to determine if this was the case for our traces, we wrote several synthetic workloads dynamically composed almost entirely of complex instructions. Each of the programs was composed of a single very large loop that was repeated for approximately ten seconds. The loop body was composed of numerous instances (30+) of only one instruction. Since more than 90% of executed instructions were identical, average power due an individual instruction can be estimated.

Table 7. Instruction Power Consumption

	Power (Watts)	Latency (Cycles)	Throughput (Cycles)
fcos	30	180-280	130
fsin	31	160-200	130
fpta	25	240-300	170
imul	28	15-18	5
idiv	32	66-80	30

Table 7 [4] shows that high latency instructions such as floating point type, consume less power than the 36W minimum predicted by our models. One possible cause is greater opportunity for clock gating. Since these instructions are guaranteed to take a very long time to complete, more aggressive power saving techniques may be performed. Further investigation will be required to validate this hypothesis. Since Table 7 supports our conclusion that high latency instructions consume less power, we then proposed to improve our power model by including this behavior. Our approach was to note that most high latency instructions are composed of relatively long uop sequences provided by the microcode ROM. Microcode ROM events can be observed using the trace cache metric microrom uops. This metric counts the number of uops delivered from the microcode ROM. The resultant models are given in Table 8. As expected from our observations about power consumption of microcode ROM delivered instructions, the model's microcode ROM component is negative. This small correction allows our power model to extend below 36W for workloads with high instances of complex microcode ROM instructions.

Table 8. Uop Linear Regression Model Comparison

	Deliver, Microrom			Deliver, Microrom, Build			
Coefficients	α_0	α_1	α_2	α_0	α_1	α_2	α_3
	36.7	4.24	-11.8	36.7	4.24	-14.6	5.74
Avg Error	2.50%			2.55%			
Coefficient of Determination	0.844			0.850			

5. ENERGY EFFICIENCY ANALYSIS

The power models of the previous section suggest that the majority of power consumption is not directly related to the rate at which useful work is performed. Energy consumption due to useful work is a relatively small portion of total power. This agrees with the findings of [10] in which they find that for the Alpha 21264 only 26% percent of energy goes to useful work.

Similarly, given our power model with 36W consumption under the minimum case of IPC = 0 and 47W for the maximum case of IPC = 3, at most only 30.5% of power is directly attributable to useful work. In order to better understand the cause of this low efficiency operation, we now examine the dominant metrics that effect efficiency. Note that energy efficiency results in this section are presented in terms of Muops/Joule. This metric is similar to the popular MIPS/Watt metric.

5.1 Effect of L2 Cache Misses

Like performance (MIPS), efficiency is greatly affected by high latency cache misses. High latency misses, such as those in the L2 cache, are often too long for the processor to hide using out-of-order issue. Instruction streams with too little ILP and an instruction window that is too small to identify additional parallelism cause this [14]. Figure 2 shows the relation between efficiency and L2 miss frequency (misses/uop). L2 miss frequency is defined as the number of L2 cache misses per dynamic uop. The right side of the graph represents workloads that are more memory bound. The point in the upper left represents the synthetic workload written to explore the range of high processor utilization (upc = 2.25) not visible in the SPEC 2000 workloads. As predicted, the highest levels of efficiency are obtained when processor utilization is maximized.

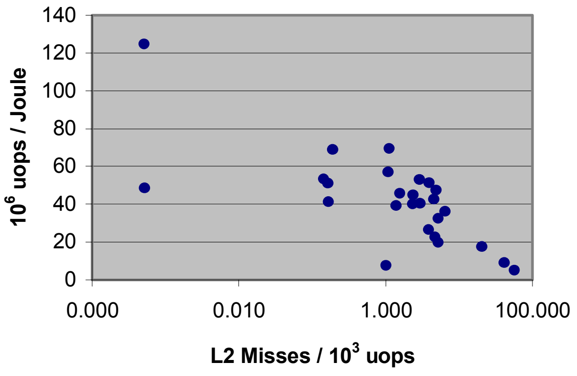

Figure 2. Efficiency vs. L2 Miss Density

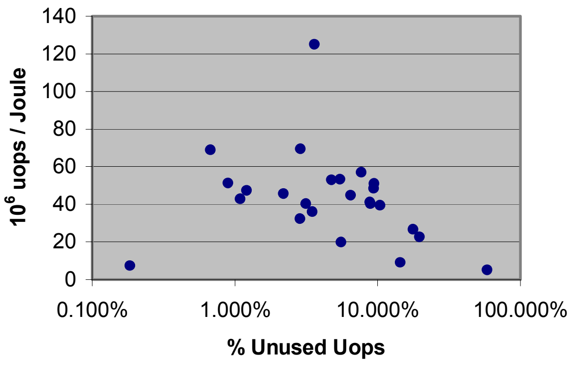

Figure 3. Efficiency vs. Unused Uop Rate

5.2 Effect of Speculative Execution

Another contributor to low efficiency is cancelled uops due to branch speculation. As we showed in Figure 1, 20.9% of fetched uops do not result in useful work. These unused uops consume energy (fetch/cycle) without contributing to useful work (uop/cycle). Figure 3 shows the effect these unused uops have on energy efficiency. The outlying data points are synthetic workloads: bottom left (lowipc) and upper middle (maxipc). Similar to the relation of L2 misses to efficiency, branch mispredictions have a clear linear relation to efficiency.

5.3 What are the Bounds of Efficiency?

Given the significant minimum power consumption predicted by our model, maximum efficiency is coincident with maximum power consumption. This is obvious since fetch rate and retire rate (uop/cycle) are approximately equal and the α_0 term has a smaller effect as these rates increase. Using the maximum sustained throughput of the Pentium 4 (3 uops/cycle) as an upper bound, we project maximum efficiency to be 133.55 Muop/Joule for our 2.2GHz processor. Minimum efficiency was assumed the smallest observed efficiency for any workload. We found this value to be 5.04 Muop/Joule for mcf. Given these bounds, we explored the range of energy efficiencies for the SPEC CPU 2000 workloads. We found that the majority of workloads operate at less than half of projected maximum efficiency. The average efficiency was found to be 42.1 Muop/Joule. Twenty of the workloads fell within one standard deviation of the mean. The only workload with efficiency significantly higher than the mean was the synthetic maxipc. These results are provided in Figure 4.

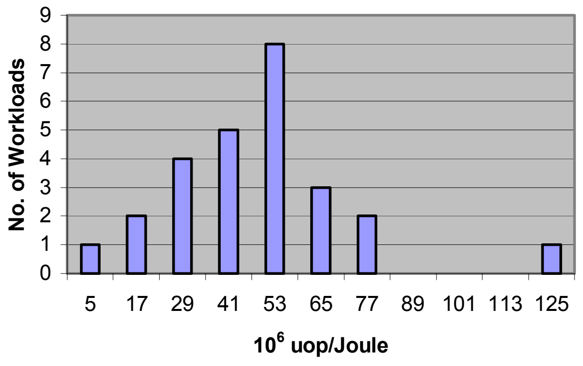

Figure 4. Efficiency Metric W/E for SPEC 2000 workloads

5.4 How Efficiency can be Improved

Given the previous analysis of efficiency how can these results be used to design and/or utilize microprocessors in a more efficient manner? First, since minimum power consumption is so large compared to scalable power consumption (uop/cycle dependent) improving processor performance using techniques that don't increase incorrect speculation offers the best improvement. This agrees with the findings of [14], which showed that efficiency increases with increased micro-architecture utilization. In those cases, improvements were made through utilizing improved static optimizations (compile time).

Another approach would be to identify the inefficient phases of programs and reduce the effect of the α_0 term. Current microprocessors already have facilities for scaling frequency and voltage [9][1] in order to save power. Unfortunately, existing power scaling policies guarantee safe operating conditions, not optimum efficiency. Their approach is to keep case temperature within a safe range. The result is application of voltage and frequency scaling at the worst time from an energy efficiency perspective. According to our power models, maximum power and therefore maximum case temperature are coincident with maximum processor utilization. Applying scaling when $metric_1 \bullet \alpha_1$ is maximum reduces efficiency. A better approach would be to apply scaling when $metric_1 \bullet \alpha_1$ is at a minimum. According to Figures 2 and 3 scaling should be applied during program phases with high frequencies of branch mispredictions and L2 misses. Li et al [7] attempted a similar approach using L2 misses and available ILP. Their approach used simulation for verification and was not applicable to existing processors. Since it is possible that efficiency is being affected by a combination of L2 miss rate, branch misprediction rate and other more minor events, better approach is to consider the resultant overall efficiency. Rather than attempting to find the exact contribution of numerous metrics, phases of low efficiency could be identified directly using our model of efficiency. This software-only approach requires no microarchitecture change and is not computationally demanding. Unlike scaling only when the processor exceeds a specified temperature, this approach improves energy efficiency and reduces the likelihood of temperature emergencies. Since power consumption is reduced during low efficiency phases, the case temperature is lower when the processor enters high power phases. Therefore, the processor is less likely to exceed thermal limits.

6. CONCLUSION

In this paper we presented simple power/energy models that could be used to address the critical issues of high power consumption and low energy efficiency. We also motivated further work in this area through a detailed analysis of energy efficiency for SPEC CPU 2000 workloads. We showed that an accurate power prediction could be made using a simple, PMC-based linear regression model. IPC-related metrics were shown to be the most representative. Of these metrics, uops fetched per cycle was the most accurate. Accuracy was improved further by utilizing a two-input, trace cache-based model that considered the effect of microcode ROM delivered uops. Additionally, we presented an efficiency analysis of the SPEC CPU 2000 benchmark suite. The majority of workloads yielded low energy efficiency. The cause of the poor efficiency was attributed to level 2 cache misses and branch mispredictions. We also compared our finding to existing PMC-based power modeling research. Our results showed that the simulation-based model of [8] was not representative of real systems. In addition, an accurate power model can be created using considerably fewer PMC metrics than [5].

7. FUTURE WORK

Our primary objective for future work is to apply our runtime efficiency model to detecting opportunities for energy savings on a real system. We plan to detect efficiency phases using a periodic system service and make voltage and frequency scaling decisions based on the results. In order to make the most effective use of scaling, a characterization of typical efficiency phase duration will be required. Finally, our models could be improved by controlling for processor die temperature during the training workloads to reduce underestimation of lengthy workloads.

8. REFERENCES

[1] AMD Corporation. *PowerNOW! Technology White Paper*, November 2000.

[2] Bellosa, F. The Benefits of Event-Driven Energy Accounting in Power-Sensitive Systems. *Proceedings of 9th ACM SIGOPS European Workshop*, (September 17-20, 2000).

[3] Brooks, D., Tiwari, V., and Martonosi, M. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. *International Symposium on Computer Architecture*, (June 12-14, 2000).

[4] Intel Corporation. *IA-32 Architecture Optimization Reference Manual*. 2004.

[5] Isci, C. and Martonosi, M. Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data. *International Symposium on Microarchitecture,* (December 3-5, 2003).

[6] Lee, K. and Skadron, K. Using Performance Counters for Runtime Temperature Sensing in High-Performance Processors. *High-Performance, Power-Aware Computing*, (April 4, 2005).

[7] Li, H., Cher, C., Vijaykumar, T. N., and Roy, K. VSV: L2-Miss-Driven Variable Supply-Voltage Scaling for Low Power. *International Symposium on Microarchitecture*, (December 3-5, 2003), 19-28.

[8] Li, T. and John, L. Run-Time Modeling and Estimation of Operating System Power Consumption. *Conference on Measurement and Modeling of Computer Systems*, (June 10-14, 2003).

[9] Meijer, M., Pessolano, F., and Gyvez, J. Technology exploration for adaptive power and frequency scaling in 90nm CMOS. *International Symposium on Low Power Electronics and Design* (August 9-11, 2004).

[10] Natarajan, K., Hanson, H., Keckler, S., Moore, C. and Burger, D. Microprocessor Pipeline Energy Analysis. *International Symposium on Low Power Electronics and Design*. (August 25-27, 2003).

[11] Phansalkar, A., Joshi, A., Eeckhout, L., and John, L. Measuring Program Similarity. *Proceedings of the Intl Symposium on Performance of Systems and Software*, (March 20-22, 2005).

[12] Seng, J. and Tullsen, D. The Effect of Compiler Optimizations on Pentium 4 Power Consumption. *Workshop on Interaction Between Compilers and Computer Architectures,* (February 8, 2003).

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

[14] Valluri, M. and John, L. Is Compiling for Performance == Compiling for Power? *Proceedings of the 5th Annual Workshop on Interaction between Compilers and Computer Architectures*. (January 21, 2001).