Efficient Power Analysis using Synthetic Testcases

Robert H. Bell, Jr. †‡
†IBM Systems and Technology Division
Austin, Texas
robbell@us.ibm.com

Lizy K. John ‡
‡Department of Electrical and Computer Engineering
The University of Texas at Austin
ljohn@ece.utexas.edu

Abstract

Power dissipation has recently become an important consideration for processor designs. Assessing power using simulators is problematic given the long runtimes of real applications. Researchers have responded with techniques to reduce the total number of simulated instructions while still maintaining representative simulation behavior. Synthetic testcases have been shown to reduce the number of necessary instructions significantly while still achieving accurate performance results for many workload characteristics.

In this study, we show that the synthetic testcases can rapidly and accurately assess the dynamic power dissipation of real programs. Synthetic versions of the SPEC2000 and STREAM benchmarks can predict the total power per cycle to within 6.8% error on average, with a maximum of 15% error, and total power per instruction to within 4.4% error. In addition, for many design changes for which IPC and power change significantly, the synthetic testcases show small errors, many less than 5%. We also show that simulated power dissipation for both applications and synthetics correlates well with the IPCs of the real programs, often giving a correlation coefficient greater than 0.9.

1. Introduction

Power dissipation has recently become an important consideration in the design of processors [13]. As frequencies have passed into the multiple gigahertz range and the number of transistors integrated onto a single chip has surpassed 100 million [27], the maximum power dissipation for high-end processors has surpassed 100 watts [27][28]. The increases in power dissipation are of significant impact to chip reliability and mobile system battery life [4]. In an effort to study and alleviate increases in on-chip power dissipation early in the design process at the same time that design tradeoffs are being studied, researchers have integrated microarchitectural power estimators into processor simulation systems [7][21][16].

Architectural level simulators, whether executing traces in a trace-driven fashion [15], or executing binaries in an execution-driven simulation [8][7], can obtain accurate results in assessing computer performance and power dissipation. Some applications like SPEC [26] have become benchmarks of computer performance and power dissipation. However, long runtimes for the latest benchmarks such as SPEC2000 make full program simulation for early design studies or performance model validation impractical [6][12][24][3][30][25]. Hand-coded microbenchmarks or testcases [18][5][6][10] can quickly test specific features in a machine, but performance models validated to within 2% using microbenchmarks have been shown to still contain errors of 20% to 40% on actual programs [10]. Randomly-generated testcases [6] are inefficient at covering the spectrum of machine responses of real programs.

Sampling techniques such as SimPoint [24], SMARTS [30] and Intrinsic Checkpointing [22] can reduce simulation runtimes, but they still require the execution of tens of millions of instructions. Statistical simulation [9][19][1][11] creates representative synthetic traces of less than one million instructions and has successfully analyzed power-performance trade-offs in a trace-driven simulation system [12][11], but traces are not very portable to many modern design platforms including execution-driven simulators, RTL models, hardware emulators, and hardware itself [3]. Execution-driven simulators are useful for assessing the power dissipation of more accurate simulation systems including operating system effects [16]. RTL models and hardware emulators are useful for performance model validation [3], and performance monitor counters in hardware facilitate rapid power dissipation studies [4].

There have been few attempts to synthesize portable source code from traces. Sakamoto *et al.* combine a modified trace snippet with a memory image for execution on a specific machine and a logic simulator [23], but there is no attempt to reduce the total number of simulated instructions. In Hsieh and Pedram [14], assembly programs are generated that have power consumption signatures similar to applications. However, all workload characteristics are modeled as microarchitecture-dependent characteristics, so that work is not applicable to studies involving design trade-offs

[12]. Wong and Morris [29] investigate synthesis of the LRU hit function to reduce simulation time, but no method for simultaneously incorporating other workload characteristics is developed.

Bell and John [2][3] synthesize C-code programs from reduced synthetic traces generated from the workload characteristics of executing applications, as in statistical simulation. The low-level workload characteristics of the original application are retained by instantiating individual operations as volatile *asm* calls. The synthetic testcases execute orders of magnitude faster than the original workloads while retaining good accuracy. That work shows an average IPC within 2.4% of the IPC of the original applications, and prior work shows that IPC has a good correlation to average power dissipation [16][4]. Even though power is not considered in the synthesis process, since the synthetic benchmarks display good IPC correlation with actual programs, it is natural to expect that they can also be used to speed up power dissipation analysis for the applications with low errors. It should be noted that the synthetic workload cannot be used to compare total energy usage since the dynamic instruction count for the synthetic is significantly reduced from that of the original application.

In this work, we study the dynamic power dissipation characteristics of the synthetic benchmarks from [3]. We take the synthetics and execute them on the Wattch simulation framework [7]. We find that the synthetics correlate to absolute power dissipation to within 6.8% error on average, and relative power dissipation error for many design changes is less than 5%. We also show that the changes in power dissipation for design changes correlate well with those in the actual applications. The specific contributions of this work are:

i) We show that synthetic testcases generated strictly for performance purposes are also useful for analysis of dynamic power dissipation, giving reasonable errors while executing orders of magnitude faster.

ii) We show that the synthetics are also useful for the relative power analysis of design changes.

iii) We classify the results of the IPC and power dissipation design changes to facilitate analysis.

iv) We confirm prior results that demonstrate a good correlation between IPC and dynamic power dissipation, and we extend the results to design change correlations.

In the Section 2, we give an overview of our workload synthesis system. In Section 3 we give qualitative reasons why power dissipation is expected to correlate, and we describe the benefits for power model validation. In Section 4 we present the quantitative results of the power analyses. In the final Sections we give the conclusions and references.

2. Overview of Testcase Synthesis

The workload synthesis system in [3] is designed to speed up phase execution by focusing on the workload characteristics that contribute significantly to the performance of the machine while ignoring or deemphasizing those characteristics that do not. Only the instruction sequences that provoke machine responses

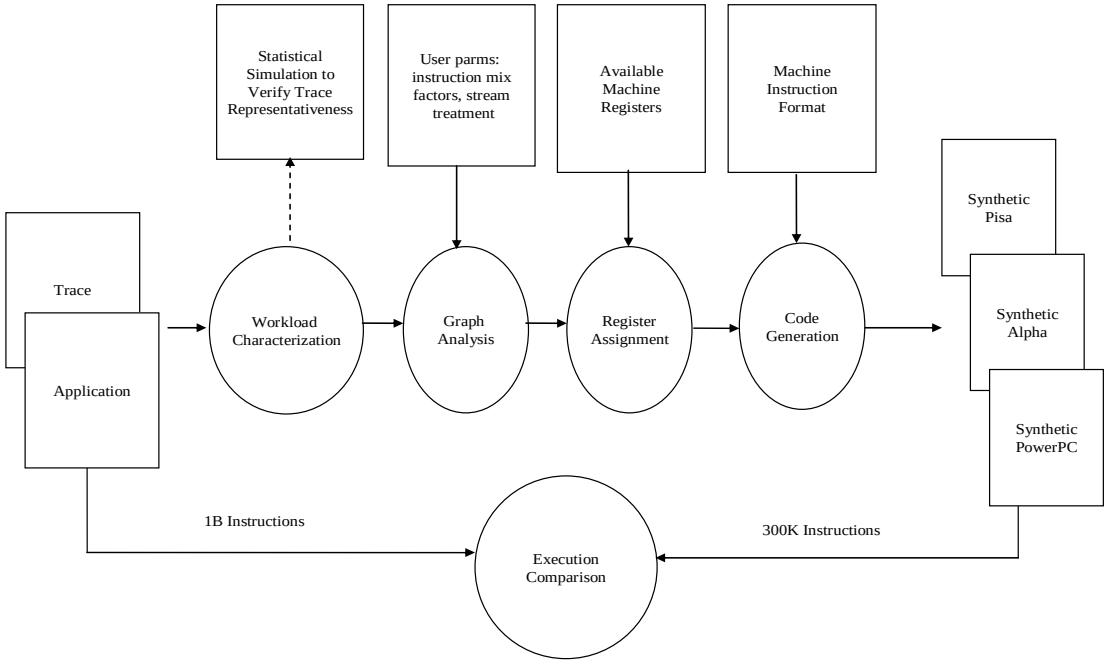

Figure 1: Synthesis and Simulation Overview

most similar to those in the original workload are retained in the synthetic. This focus ensures that the number of instructions that must be executed to obtain representative behavior can be significantly reduced. Detailed descriptions of the process can be found in [2][3]. Here we give an overview of the synthesis process.

Figure 1 gives a flow diagram of the synthesis process. The four phases of synthesis are workload characterization, graph analysis, register assignment and code generation. The system can take as input a trace or program binary. A trace is taken from a workload executing on hardware [15] and must be detailed enough to contain the information necessary for synthesis as described below. Note that most traces do not contain mispredicted instructions after a branch, only completed instructions, but prior work has achieved good results without using them [1][11][3]. A binary is created by simply compiling the application program.

Workload characterization analyzes the trace or executes the binary, collecting the same statistics necessary to achieve an accurate statistical simulation [1][11]. Both microarchitecture dependent and independent characteristics are collected into a statistical flow graph [1][11]: basic block sequences, instruction mix, dependency distances, cache miss rates and branch predictability. The instructions are abstracted into types that exercise the functional units in the same way as the original instructions. Figure 1 shows that a synthetic trace generated from walking the flow graph can be executed in statistical simulation to verify the representativeness of the workload characterization.

Figure 2 gives a flow diagram of the graph analysis phase. The diagram represents the analysis carried out during a pass through synthesis plus execution of the resulting synthetic benchmark. Each diamond block makes a decision regarding the quality of the synthesis pass for one workload characteristic. If the value of that characteristic is below a threshold relative to its value when executing the original program, each rectangular block perturbs the value of a synthesis parameter that putatively affects that characteristic. Synthesis and simulation are then restarted and the analysis is repeated. A particular parameter is designed to affect only one characteristic, but sometimes characteristics previously analyzed are adversely affected and must be revisited, and the process then takes longer to complete. Sometimes a threshold cannot be satisfied, in which case we simply retain the best-case synthesis result.

The details of the rectangular blocks in the flow diagram of Figure 2 are given in [3]. The initial number of basic blocks is based on a default I-cache size and associativity. The workload characterization specifies the initial values for the number of branches that are configured to branch conditionally and the stride table entries that will be accessed by each memory access counter. In each synthesis pass, the average basic block size and locality metrics for the synthetic are compared to those of the original workload.

Note that power dissipation is not one of the characteristics used to assess the synthesis quality. The reason is that power dissipation is an aggregate metric, like IPC. All other characteristics of the synthetic workload may impact power to some degree, so no small set of thresholds can be assigned in the analysis phase to accurately change only power.

Figure 1 shows that user parameters can influence the graph analysis phase. This demonstrates one of the benefits of synthesis using statistics, namely that particular statistics can be modified to synthesize variations on current workloads or even future workloads.

The register assignment phase of Figure 1 is separate from the code generation phase because it can often be performed without exact knowledge of the ISA. For example, experiments have shown that about twenty registers give good synthesis results for all three targets - Pisa, PowerPC, and Alpha.

The code generation phase, however, is strongly ISA dependent. The abstracted instruction operations in the basic blocks obtained during workload characterization are instantiated as volatile *asm* calls to instructions in the target ISA that exercise the proper functional units based on the abstracted instruction type.

3. Power Dissipation of Synthetic Testcases and Power Model Validation

Several studies have shown that the synthetic traces in statistical simulation exhibit power dissipation similar to cycle-accurate simulations [12][11]. The synthetic traces contain specific basic block sequences that represent the

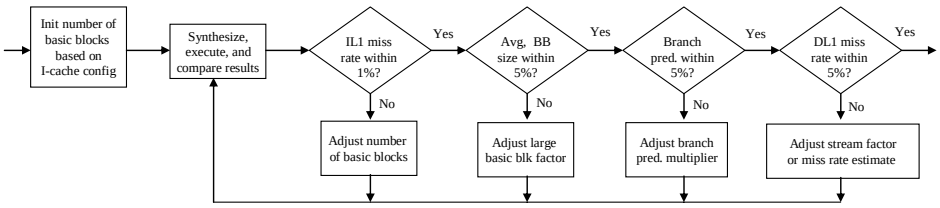

Figure 2: Flow Diagram of Graph Analysis Phase

major components of the performance of the workload. The number of instructions is reduced because instruction sequences that do not contribute to performance are not included. Since the number of instructions is reduced, the total energy for execution of the synthetic trace cannot be compared to that of the original application.

The synthetic traces in [1][11] provide accurate power analysis because the basic block sequences that provide accurate performance results must also exercise the machine such that the power dissipation is accurate. The synthetic traces exhibit dynamic workload features similar to those of the original applications, including instruction mix, number and type of operands, instruction-level parallelism, dependency distances, and memory access and branching behavior. The workload similarities imply that reorder buffer occupancies, pipeline throughput, cache hierarchy access and miss rates, pipeline stalls, and branch predictabilities will be similar. To a large extent, these machine features determine the dynamic power dissipation of the system [7][28][4]. As an example, the abstracted instruction types used for synthesis use one, two or three (for PowerPC) operands as in the original workload, so the proper power dissipation with respect to register port access is obtained.

The synthetic testcases described in [2][3] have dynamic workload characteristics similar to those of the synthetic traces, except that the locality models are less accurate [3]. Memory accesses are modeled as strides through uninitialized data structures in order to match miss rates for a default cache configuration. The overall miss rates of the original application are obtained, but particular miss rates at the granularity of individual loads and stores may be quite different from those of the original since integer stride values only generate particular miss rate quanta, rather than a continuous spectrum of miss rates [2]. Microarchitecture-independent memory access models would seek to model more closely the original workload access patterns [2][3].

The branching models are also not accurate at the granularity of individual branches. The overall application predictability is matched by configuring a subset of branches to jump past the next basic block 50% of the time [3]. The synthetic testcases would benefit from an exact analysis and configuration of particular branches in the workload [2][3]. Other anomalies in the synthetic testcases include the retargeting of integer instructions for data structure access and testcase looping, and register usage [3].

Table 1: Default Simulation Configuration, Alpha ISA

Instruction Size (bytes)	4
L1/L2 Line Size (bytes)	32/64
Machine Width	4
Dispatch Window/LSQ/IFQ	16/8/4
Memory System	16K 4-way L1 D, 16K 1-way L1 I, 256K 4-way unified L2
L1/L2/Memory Latency+transfer (cycles)	1/6/34
Functional Units	4 I-ALU, 1 I-MUL/DIV, 4 FP-ALU, 1 FP-MUL/DIV
Branch Predictor	Bimodal 2K table, 3 cycle mispredict penalty

All of the above inaccuracies imply that instruction dependencies, memory accesses and branch behavior are different from the original workload, that the machine responses to the workload will be correspondingly different, and that the power dissipation results, in turn, will contain inaccuracies. However, prior work has shown that the performance errors are relatively small or, if large, less relevant to the performance of the machine, so we expect that the power dissipation errors will be correspondingly low, as shown in the next section.

One of the benefits of the synthetic testcases is their portability to multiple platforms [3]. Combined with overall runtimes that are two or three orders of magnitude faster than those of the original applications, the synthetic testcases are ideal for performance model validations using combinations of detailed execution-driven simulators, system simulators, RTL model simulators, hardware emulation systems, and hardware itself [3].

Likewise, the synthetic testcases are useful for *power model validation*. Simulators that assess dynamic power can be validated against slow RTL and circuit simulators with greater assurance that the validated simulator will give more accurate performance and power results for longer runs.

The next section compares the absolute and relative accuracy of the power dissipation of the synthetic testcases to those of longer running programs.

4. Synthesis Results

In this section we present the power dissipation results for the benchmark synthesis system described in the last section.

4.1. Experimental Setup and Benchmarks

We start with the experimental system that exhibits good statistical simulation correlation against actual application simulations described in [3]. Our system is derived from the statistical simulation system HLS [19][20], which we updated with the statistical flow graph to improve correlation [1][11]. SimpleScalar 3.0 [8] was downloaded and *sim-cache* was modified to carry out the workload characterization. To *sim-outorder*, we added the event counters, the *power.h* includes, and *cacti* code from Wattch [7]. Wattch models the power dissipation of circuits and structures for a 0.35 micro, 600 MHz machine. We present results for an aggressive clock gating design with 10% leakage power [7].

Table 2: Average Power %Error, Synthetics vs. Applications

Metric/Structure	% Error	%Error SPECint	%Error fp/STREAM	Max. %Error
Power per Cycle	6.8	9.9	5.1	15.0 (mcf)
Power per Instruct	4.4	5.4	3.9	11.2 (twolf)
Rename	2.4	2.8	2.1	7.4 (mcf)
Branch Predictor	3.7	4.3	3.3	15.8 (apsi)
Dispatch Window	5.3	7.1	4.3	12.5 (wupw)
LSQ	2.8	2.5	3.0	15.2 (applu)
Register File	6.4	4.5	7.4	22.1 (wupw)
L1 I-cache	6.0	11.0	3.2	14.1 (mcf)
L1 D-cache	5.8	7.3	4.9	18.5 (applu)
L2 cache	2.3	2.6	2.2	11.8 (applu)
ALU	1.8	1.7	1.8	5.2 (facerec)
Result Bus	6.6	10.9	4.3	16.1 (gcc)
Global Clock	12.4	17.7	9.5	27.0 (mcf)
Fetch	3.4	6.0	2.1	9.7 (mcf)
Dispatch Logic	2.4	2.8	2.1	7.4 (mcf)
Issue Selection Lgc	3.6	4.9	2.9	8.2 (mcf)

The twelve SPECint2000 and fourteen SPECfp2000 *Alpha* binaries were executed in *sim-outorder* on the first reference dataset for the first billion instructions. Single-precision versions of eight STREAM and STREAM2 benchmarks [17] with a ten million-loop limit were also simulated. We use the default SimpleScalar configuration given in Table 1, as in [19][21][3]. SimpleScalar does not model an L3, but the memory latency used here estimates a fast L3. While the machine configuration is relatively small, in the experiments below we vary the window size and other machine parameters significantly and still obtain good power dissipation correlations as IPC increases. Another consideration is that this machine configuration is appropriate for use with the original Wattch model [13][7][12]. It is also still a useful model for smaller embedded or ASIC designs. The synthetic benchmarks were executed to completion on an IBM p270 (400 MHz).

The details of the number of instructions executed and runtime per benchmark are given in [3]. Typically the synthetics exhibit less than 300K dynamic instructions and execute two to three orders of magnitude faster than the actual programs.

4.2. Base Power Dissipation Results

Figure 3 shows the power dissipation per cycle in Watts for the actual programs and the synthetics. The

Table 3: Correlation Coefficients of Power Dissipation vs. IPC

Metric/Structure	Actual	Synthetic
Power per Cycle	0.94	0.96
Power per Instruction	-.84	-.84
Rename	0.99	0.99
Branch Predictor	0.65	0.62
Dispatch Window	0.96	0.97
LSQ	0.30	0.22
Register File	0.77	0.75
L1 I-cache	0.91	0.96
L1 D-cache	0.43	0.46
L2 cache	0.016	-0.033
ALU	0.83	0.81
Result Bus	0.90	0.94
Global Clock	0.91	0.95
Fetch	0.88	0.92
Dispatch Logic	0.99	0.99
Issue Selection Logic	0.90	0.90

average error is 6.8%, with a maximum error of 15% for *mcf*. The SPECint synthetics exhibit larger average errors than the SPECfp and STREAM, at 9.9% and 5.1% respectively, as shown in Table 2. Individual machine components generally show larger errors for SPECint, especially for the L1 I-cache and D-cache, the result bus, and the clock power. As mentioned in Section 3, the explanation for the cache errors is that the more complicated memory access behavior in SPECint is less likely to be well modeled by the simple synthetic streams, resulting in additional error in power dissipation in the I-cache and D-cache. This occurs in spite of the fact that IPC errors for synthetics with L1 cache miss rates above 1% are generally small [3] because the small miss rates have little impact on performance or are offset by errors in other parts of the memory subsystem [3]. However, these results indicate that the SPECint could especially benefit from more accurate microarchitecture-independent memory access models.

The synthetics also have relatively low clock power dissipation versus the applications. A partial explanation is the uniformly lower dispatch window occupancies for the synthetics, which exhibit an average decrease in occupancy error of 4.1% [3]. Additional errors for the SPECint again point to the memory access model as a major contributor to the overall clock and result bus power dissipation errors. The large errors in many features for *mcf* mirror its relatively large IPC error, 7.4%.

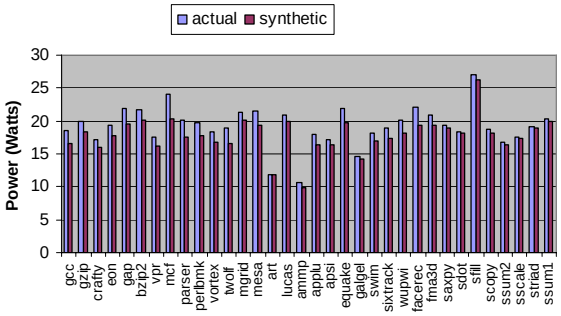

Figure 3: Power Dissipation per Cycle

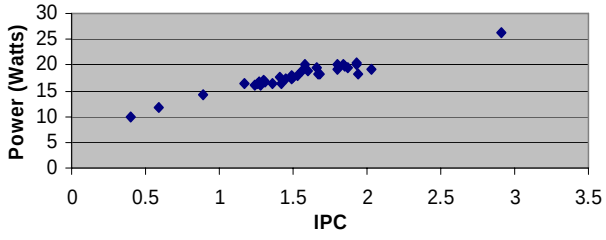

Figure 4: Power per Cycle vs. IPC for Synthetics

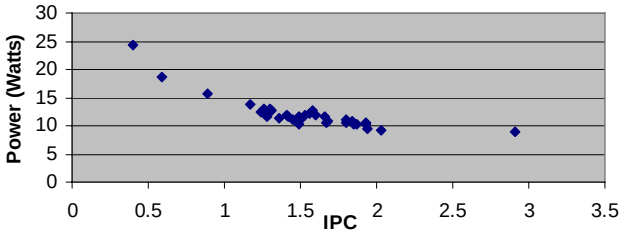

Figure 5: Power per Instruction vs. IPC for Synthetics

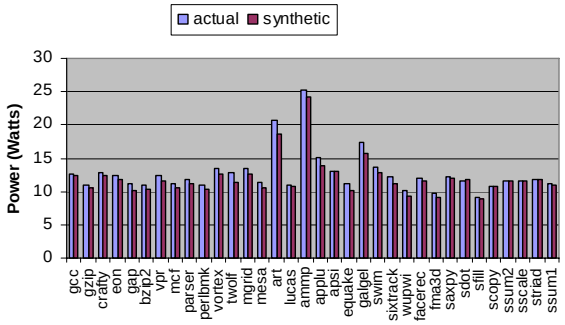

Figure 6: Power Dissipation per Instruction

Figure 4 is a scatter plot of the IPC versus the power per cycle for the synthetics. The correlation is quite good, with a Pearson correlation coefficient of 0.96. Table 3 gives correlation coefficients for the power dissipation metrics and the various machine features for both the actual programs and synthetics. The synthetic correlation coefficients generally follow those of the actual programs.

The results of Table 3 generally confirm the findings in prior microarchitectural power simulations [16][4]. For many features the correlation is greater than 0.90. However, there are some important exceptions. The LSQ, L1 D-cache, and L2 D-cache power dissipations appear not to be correlated with IPC, at least over the instructions executed. Also, the branch predictor, register file, and ALU power dissipations are only weakly correlated. IPC can generate a good rough estimate of overall power dissipation, especially for the pipeline structures, but additional analysis is necessary to estimate the power dissipation for many machine structures, especially the data cache hierarchy.

Figure 5 shows that the power dissipation per instruction decreases as IPC increases. The power dissipation of unused structures is 10% of maximum in the conditional clocking scheme we use in Wattch, and as IPC increases the unused structure overhead is amortized over more instructions per cycle, but it never reaches zero. In Table 3 we find a negative 0.84 correlation for both actual programs and synthetics because the curve is non-linear, and the minimum power appears to be asymptotic to a line below 9 Watts as IPC increases in Figure 5. This trend would extrapolate to a maximum power of about 36 Watts for this four-issue machine.

Figure 6 breaks down the per instruction power dissipation by benchmark. The average error from Table 2 is 4.4%, with a maximum error of 11.3% for *twolf*. At 5.4%, the error in power per instruction is more evenly distributed for SPECint than is power per cycle, while SPECfp and STREAM give 3.9% error.

4.3. Analysis of Design Changes

We now study the power dissipation for design changes using the same synthetic benchmarks; that is, we take the benchmarks described in the last section, change the machine configurations in Wattch, re-execute them, and compare the results to executions of the actual programs on the same configurations.

Table 4 gives information for the absolute and relative IPC prediction errors [11] and the absolute and relative power dissipation prediction errors when executing various design changes. We include here the IPC data from [3] to ease discussion. The only differences with the results in [3] are for the studies of machine width, issue and commit width and IFQ size. For each of those we changed the other parameters related to machine width to 16, instead of 4 in [3], to give larger changes in IPC as the parameter changes.

In the table, *Disp* is the dispatch window size. A change in *Machine Width* implies that the decode width, issue width and commit width all change by the same amount from the base configuration in Table 1. When the caches are increased or decreased by a factor, the number of sets for the L1 I-cache, D-cache and L2 cache are increased or decreased by that factor. Likewise, when the bimodal branch predictor is multiplied by a factor, the table size is multiplied by that factor from the default size. The *L1 D-cache 2x* and *L1 I-cache 2x* specify a doubling of the L1 D-cache (to 256 sets, 64B cache line, 8-way set associativity), and a doubling of the L1 I-cache configuration (to 1024 sets, 64B cache line, 2-way set associativity). The numbers here do not include *ammp* or *galgel*. Those benchmarks tend to be optimistic when the dispatch window changes because our small data footprint benchmarks do not model data-TLB misses, and the actual programs have large TLB miss rates, over 13% each.

We also studied a commit width change from 4 to 1, but the data for some actual programs was inconsistent, so it is not used. Speculation is that there is a power modeling error when the commit width is reduced to one. Evidence that the modeling is incorrect is given in Table 5, where most correlation coefficients for IPC and power dissipation are above 0.90, except when the commit width equals one. We do not examine the commit width equal one results further in this paper.

The other correlation coefficients in Table 5 indicate that the power dissipation achieved by the synthetics for a

design change follows fairly well the IPC change for that design change. Table 4 shows that the absolute and relative IPC errors and power dissipation errors *(%Error* and *Rel. %Error*) due to a design change for the synthetics versus the actual programs is often below 5% or 10%, except for the two cases in which the caches are reduced in size. The synthetics underestimate performance when the cache is significantly reduced due to capacity misses among the synthetic data access streams [2].

There are four classes of results in Table 4 that shed light on the quality of the power dissipation analysis using the synthetics and provide a starting point for discussion. We are concerned about whether the synthetics properly indicate power dissipation changes when the IPC changes significantly:

Class 1: The change in IPC and the change in power dissipation (*%Change* in Table 4) are greater than two times (or more) their absolute or relative errors for the design change.

Class 2: The change in IPC is greater than two times its absolute or relative error, but the change in power dissipation is not.

Class 3: The change in power dissipation is greater than two times its absolute or relative error, but the change in IPC is not.

Class 4: Neither the change in IPC nor the change in power dissipation is greater than two times its error.

The classes for each design change are given in Table 5. The choice of threshold equal to 2x is *ad hoc*, but it gives a good cushion between the average errors and the average change that is being indicated. Evidence that that is a good metric is given by the max absolute and relative error columns (*Max %Err* and *Max Rel. %Err*) for the design changes. Generally the max errors are less than or not too far removed from the *%Change*.

The *%Change* is the minimum change for either the actual or synthetic workload. For all cases the *%Change* for either is close to that of the other.

Most of the design changes are class 1 (or borderline class 3) and none of the design changes are class 2, which indicates that significant power dissipation changes can be assessed when the IPC changes significantly. The class 3 and borderline class 3 design changes indicate that the correct power dissipation changes are reflected even though the IPC changes may not be significant. In these cases, the correlation coefficients between IPC and power dissipation are lower.

Table 4: Average Absolute and Relative IPC and Power Dissipation Error

	IPC					Power Dissipation						
Design Change	%Error	Max %Err	Rel. %Err	Max Rel. %Err	%Change	%Error	Max %Err	Rel. %Err	Max Rel. %Err	%Change	Max Avg. Structure %Err (strc)	Max %Err Structure (synthetic)
Disp 8 LSQ 4	3.1	16.1	2.3	7.5	23.8	4.9	10.9	2.5	6.6	16.8	9.2(clk)	21.2 (gap)
Disp 32 LSQ 16	3.7	9.9	2.2	9.4	16.1	8.5	18.5	1.8	8.8	18.5	14.3(clk)	28.9 (mcf)
Disp 48 LSQ 24	5.0	12.3	3.9	14.3	24.6	9.2	17.9	2.9	10.9	31.7	15.4(clk)	32.4(applu)
Disp 64 LSQ 32	6.1	18.1	5.2	20.8	31.0	9.5	18.4	3.6	13.7	41.5	16.0(clk)	36.6(applu)
Disp 96 LSQ 48	8.3	29.3	7.6	32.2	43.2	9.9	22.4	4.8	18.9	63.4	17.1(clk)	41.4(applu)
Disp 128 LSQ 64	9.1	34.0	8.5	37.0	52.0	10.0	23.3	5.2	22.7	82.8	17.7(clk)	43.4(applu)
Issue Width 1	1.5	5.3	1.9	6.5	53.6	1.6	3.2	5.9	14.9	38.2	5.8(rbus)	16.9(fma3d)
Issue Width 8	3.0	9.4	1.6	9.2	7.0	6.3	13.1	1.0	7.3	11.3	11.4(clk)	26.7(wupw)
Commit width 8	2.9	9.1	1.4	8.9	7.0	5.4	11.9	2.3	11.4	42.5	9.9(clk)	24.3(wupw)
Machine Width 2	2.8	7.6	2.0	6.5	23.5	5.4	10.9	1.7	5.6	17.6	10.4(regf)	26.8(wupw)
Machine Width 6	3.2	9.8	1.0	7.6	5.6	6.9	15.6	0.68	4.4	9.23	12.5(clk)	16.9(mcf)
Machine Width 8	3.2	8.8	1.1	7.6	6.0	6.6	13.7	0.79	6.2	14.5	11.9(clk)	24.4(wupw)
Machine Width 10	3.1	9.2	1.5	9.1	6.5	6.0	12.5	1.2	8.1	20.1	11.1(clk)	24.7(wupw)
Machine Width 12	3.1	9.3	1.5	9.1	6.6	5.6	11.7	1.6	9.4	25.7	10.5(clk)	25.0(wupw)
Machine Width 14	3.1	9.4	1.5	9.2	6.6	5.2	11.6	2.0	10.8	30.2	9.8(clk)	25.3(wupw)
Machine Width 16	3.2	9.4	1.5	9.2	6.7	4.7	11.2	2.6	12.1	34.1	10.0(Icac)	26.7(wupw)
IFQ 8	3.0	9.3	1.2	7.7	5.4	4.6	10.8	2.6	11.5	33.2	8.9(clk)	25.5(wupw)
IFQ 16	3.2	9.4	1.5	9.2	6.7	4.8	11.2	2.5	12.1	34.2	10.0(regf)	26.7(wupw)
IFQ 32	3.3	9.0	1.3	8.9	7.4	5.0	11.3	2.2	11.8	34.9	9.7(Icac)	26.7(wupw)
Caches 0.25x	19.4	49.8	18.8	48.8	15.2	14.1	34.8	9.2	28.5	22.8	18.6(rbus)	45.0(wupw)
Caches 0.50x	23.9	46.7	23.1	47.0	7.0	17.6	33.2	11.9	26.7	13.2	23.0(wind)	45.5(fma3d)
Caches 2.0x	4.2	16.0	3.2	18.8	4.3	6.5	15.6	1.6	7.6	18.7	12.7(clk)	28.1(mcf)
Caches 4.0x	4.9	17.8	3.9	12.6	6.2	9.2	19.0	3.1	8.4	40.5	15.8(clk)	32.1(twolf)
L1 I-cache 2.0x	3.0	10.7	1.3	5.9	7.0	8.1	18.6	2.0	6.9	36.5	14.5(clk)	35.1(sfill)
L1 D-cache 2.0x	3.2	14.3	1.2	12.8	2.2	7.5	16.6	1.1	4.1	30.9	13.3(clk)	31.0(mcf)
L1 D-cache Lat 8	9.7	38.0	9.9	34.8	22.0	2.6	7.2	6.4	21.3	13.4	6.7(regf)	19.4(perlb)
BPred 0.25x	2.9	8.6	1.3	6.0	0.63	6.4	14.3	0.67	2.7	0.40	12.0(clk)	25.9(mcf)
BPred 0.50x	2.5	8.6	0.60	3.0	0.18	6.7	15.4	0.31	1.4	0.22	12.2(clk)	26.7(mcf)
BPred 2.0x	2.6	8.3	0.32	1.8	0.15	6.9	15.5	0.18	0.88	0.20	12.4(clk)	26.9(mcf)
BPred 4.0X	2.5	8.0	0.40	2.0	0.22	6.9	15.2	0.21	0.87	0.52	12.4(clk)	26.7(mcf)

The class 4 design changes indicate that the changes in IPC and power were not much different from the errors in the synthetics themselves. As mentioned, the reduced cache sizes put pressure on the memory access streams used in the synthetics, causing large percent errors. For the branch predictor studies, the percent errors are small, but the increase or decrease in the bimodal predictor table causes little change in performance (*%Change*), whether for the actual or synthetic workloads. A different predictor configuration is needed to assess the quality of predictor design changes for the synthetics.

Table 4 also gives the maximum average percent power dissipation error from among all the structures listed in Table 3 for each design change (*Max Avg. Structure %Err*). Similar to the absolute structure errors in the last section, the most prominent error is the global clock (*clk*), but the errors average only 12.5%, and for the strong class 1 design changes they are well below the average change in power dissipation.

The table also shows the maximum error found among all the synthetics for any particular structure for the design change (*Max %Err Structure Synthetic*). This is usually the global clock structure. Since these are much larger than the average structure errors (*Max Avg. Structure %Err*), these particular points are outliers. The results indicate that the synthetics for *mcf*, *applu*, and *wupwise* should be examined to understand why their predicted power dissipations are more variable than those of many others when the design changes. This analysis drives change back into the synthesis process itself.

Table 5: Correlation Coefficients of Power vs. IPC for Design Changes and Quality of Assessing Power Dissipation Changes

Design Change	Actual	Synthetic	Class
Disp 8 LSQ 4	0.95	0.97	1
Disp 32 LSQ 16	0.92	0.95	1
Disp 48 LSQ 24	0.91	0.94	1
Disp 64 LSQ 32	0.90	0.93	1
Disp 96 LSQ 48	0.92	0.93	1
Disp 128 LSQ 64	0.93	0.94	1
Issue Width 1	0.87	0.91	1
Issue Width 8	0.95	0.97	1
Commit Width 1*	-0.27	0.74	N/A
Commit width 8	0.95	0.95	1
Machine Width 2	0.92	0.94	1 (borderline 3)
Machine Width 6	0.95	0.97	1 (borderline 3)
Machine Width 8	0.95	0.97	1
Machine Width 10	0.95	0.97	1
Machine Width 12	0.95	0.96	1
Machine Width 14	0.95	0.96	1
Machine Width 16	0.95	0.95	1
IFQ 8	0.95	0.95	1 (borderline 3)
IFQ 16	0.95	0.95	1 (borderline 3)
IFQ 32	0.95	0.95	1 (borderline 3)
Caches 0.25x	0.98	0.99	4
Caches 0.50x	0.96	0.99	4
Caches 2.0x	0.92	0.96	3
Caches 4.0x	0.89	0.95	3
L1 I-cache 2.0x	0.94	0.98	1
L1 D-cache 2.0x	0.82	0.89	3
L1 D-cache Lat 8	0.97	0.97	1
BPred 0.25x	0.94	0.96	4
BPred 0.50x	0.94	0.96	4
BPred 2.0x	0.94	0.96	4
BPred 4.0X	0.94	0.96	4

5. Conclusions

In this study, we show that synthetic testcases can rapidly and accurately assess the power dissipation of real programs. Synthetic versions of the SPEC2000 and STREAM benchmarks can predict the total power per cycle to within 6.8% error on average, with a maximum of 15% error, and total power per instruction to within 4.4% error. Since the testcases execute orders of magnitude fewer instructions while maintaining accuracy, performance and power model validations using more realistic tests are feasible.

In addition, for many design changes for which IPC and power change significantly, the synthetic workloads show small errors, many less than 5%. We also show that simulated power dissipation for both applications and synthetics correlates well with the IPCs of real programs, often giving a correlation coefficient greater than 0.9. This confirms prior results that demonstrate a good correlation between IPC and power dissipation for simulated processors and hardware performance counters, and it verifies that the synthetic testcases produce similar results.

Aknowledgements

The authors would like to thank the anonymous reviewers for their detailed comments. This research is partially supported by the National Science Foundation under grant number 0429806, the IBM Center for Advanced Studies (CAS), an IBM SUR grant, the IBM Systems and Technology Division, and Advanced Micro Devices.

References

[1] R. H. Bell, Jr., L. Eeckhout, L. K. John and K. De Bosschere, "Deconstructing and Improving Statistical Simulation in HLS," Workshop on Debunking, Duplicating, and Deconstructing, in conjunction with ISCA '04, June 20, 2004.

[2] R. H. Bell, Jr. and L. K. John, "The Case for Automatic Synthesis of Miniature Benchmarks," Workshop on Modeling, Benchmarking, and Simulation, in conjunction with ISCA '05, June 4, 2005.

[3] R. H. Bell, Jr. and L. K. John, "Improved Automatic Testcase Synthesis for Performance Model Validation," International Conference on Supercomputing, June 20, 2005.

[4] W. L. Bircher, M. Valluri, J. Law and L. K. John, "Runtime Identification of Microprocessor Energy Saving Opportunities," International Symposium on Low Power Electronics and Design, August 2005.

[5] B. Black and J. P. Shen, “Calibration of Microprocessor Performance Models,” IEEE Computer, May 1998, pp. 59-65.

[6] P. Bose and T. M. Conte, “Performance Analysis and Its Impact on Design,” IEEE Computer, May 1998, pp. 41-49.

[7] D. Brooks, V. Tiwari and M. Martonosi, “Wattch: A Framework for Architectural-Level Power Analysis and Optimizations,” International Symposium on Computer Architecture, June 2000.

[8] D. C. Burger and T. M. Austin, "The SimpleScalar Toolset," Computer Architecture News, 1997.

[9] R. Carl and J. E. Smith, "Modeling Superscalar Processors Via Statistical Simulation," Workshop on Performance Analysis and Its Impact on Design, June 1998.

[10] R. Desikan, D. Burger and S. Keckler, “Measuring Experimental Error in Microprocessor Simulation,” International Symposium on Computer Architecture, 2001.

[11] L. Eeckhout, R. H. Bell, Jr., B. Stougie, L. K. John and K. De Bosschere, “Control Flow Modeling in Statistical Simulation for Accurate and Efficient Processor Design Studies,” International Symposium on Computer Architecture, June 2004.

[12] L. Eeckhout and K. De Bosschere, “Early design Phase Power/Performance Modeling Through Statistical Simulation,” International Symposium on Performance Analysis of Systems and Software, November 2001, pp. 10-17.

[13] M. K. Gowan, C. Polychronopoulos and G. Stamoulis, “Power Considerations in the Design of the Alpha 21264 Microprocessor,” Design Automation Conference, 1998, pp. 726-731.

[14] C. T. Hsieh and M. Pedram, "Microprocessor Power Estimation Using Profile-driven Program Synthesis," IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, Vol. 17, No. 11, November 1998, pp. 1080-1089.

[15] S. R. Kunkel, R. J. Eickemeyer, M. H. Lipasti, T. J. Mullins, B. O’Krafka, H. Rosenberg, S. P. VanderWiel, P. L. Vitale and L. D. Whitley, “A Performance Methodology for Commercial Servers,” IBM J. Res. Develp., Vol. 44 No. 6, November 2000.

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

[17] J. D. McCalpin, “Memory Bandwidth and Machine Balance in Current High Performance Computers,” IEEE Technical Committee on Computer Architecture newsletter, December 1995.

[18] L. McVoy, “lmbench: Portable Tools for Performance Analysis,” USENIX Technical Conference, Jan. 22-26, 1996, pp. 279-294.

[19] M. Oskin, F. T. Chong and M. Farrens, "HLS: Combining Statistical and Symbolic Simulation to Guide Microprocessor Design," Proceedings of the 27th Annual International Symposium on Computer Architecture, June 2000, pp. 71-82.

[20] http://www.cs.washington.edu/homes/oskin/tools.html

[21] R. Rao, M. Oskin, F. T. Chong, “HLSpower: Hybrid Statostical Modeling of the Superscalar Power-Performance Design Space,” International conference on High Performance computing (HiPC), December, 2002.

[22] J. Ringenberg, C. Pelosi, D. Oehmke and T. Mudge, “Intrinsic Checkpointing: A Methodology for Decreasing Simulation Time Through Binary Modification,” International Symposium on Performance and Simulation Systems, March 2005, pp. 78-88.

[23] M. Sakamoto, L. Brisson, A. Katsuno, A. Inoue and Y. Kimura, “Reverse Tracer: A Software Tool for Generating Realistic Performance Test Programs,” Symposium on High-Performance Computing,” 2002.

[24] T. Sherwood, E. Perleman, G. Hamerly and B. Calder, “Automatically characterizing large scale program behavior,” Proceedings of the International Conference on Architected Support for Programming Languages and Operating Systems, October 2002.

[25] R. Singhal, et al., “Performance Analysis and Validation of the Intel Pentium4 Processor on 90nm Technology,” Intel Tech. J., Vol. 8, No. 1, 2004.

[26] http://www.spec.org

[27] J. M. Tendler, J. S. Dodson, J. S. Fields, Jr., H. Le and B. Sinharoy, "POWER4 System Microarchitecture," IBM Journal of Research and Development, January 2002, pp. 5-25.

[28] M. Valluri and L. John, “Is Compiling for Performance == Compiling for Power?” Workshop on the Interaction Between Compilers and Computer Architectures (INTERACT-5), 2001.

[29] W. S. Wong and R. J. T. Morris, "Benchmark Synthesis Using the LRU Cache Hit Function," IEEE Transactions on Computers, Vol. 37, No. 6, June 1988, pp. 637-645.

[30] R. E. Wunderlich, T. F. Wenisch, B. Falsafi and J. C. Hoe, “SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling,” The International Symposium on Computer Architecture, June 2002.