System-level Max Power (SYMPO) - A Systematic Approach for Escalating System-level Power Consumption using Synthetic Benchmarks

Karthik Ganesan, Jungho Jo, W. Lloyd Bircher,
Dimitris Kaseridis, Zhibin Yu and Lizy K John
ECE Department, University of Texas at Austin, TX, USA
{karthik,jungho.jo,kaseridis}@mail.utexas.edu, lloyd.bircher@amd.com,
yuzhibin@mail.hust.edu.cn and ljohn@ece.utexas.edu

ABSTRACT

To effectively design a computer system for the worst case power consumption scenario, system architects often use hand-crafted maximum power consuming benchmarks at the assembly language level. These stressmarks, also called power viruses, are very tedious to generate and require significant domain knowledge. In this paper, we propose SYMPO, an automatic SYstem level Max POwer virus generation framework, which maximizes the power consumption of the CPU and the memory system using genetic algorithm and an abstract workload generation framework. For a set of three ISAs, we show the efficacy of the power viruses generated using SYMPO by comparing the power consumption with that of *MPrime* torture test, which is widely used by industry to test system stability. Our results show that the usage of SYMPO results in the generation of power viruses that consume 14-41% more power compared to *MPrime* on SPARC ISA. The genetic algorithm achieved this result in about 70 to 90 generations in 11 to 15 hours when using a full system simulator. We also show that the power viruses generated in the Alpha ISA consume 9-24% more power compared to the previous approach of stressmark generation. We measure and provide the power consumption of these benchmarks on hardware by instrumenting a quad-core AMD Phenom II X4 system. The SYMPO power virus consumes more power compared to various industry grade power viruses on x86 hardware. We also provide a microarchitecture independent characterization of various industry standard power viruses.

Categories and Subject Descriptors

C.0 [**Computer Systems Organization**]: GENERAL;
B.8.2 [**Hardware**]: Performance and Reliability—*Performance Analysis and Design Aids*

General Terms

Design, performance, measurement

Keywords

System-level power virus, synthetic benchmark, thermal design point

1. INTRODUCTION

Excessive power consumption and heat dissipation have been a critical problem faced by computer designers in the past decade. The recent trend towards consolidation in server systems (e.g. blade servers), has resulted in an explosion in power density leading to high costs related to electricity and cooling system. The 'power dissipation per square foot' of recent server systems is estimated to be 160 Watts per square foot. Data center energy costs are starting to exceed hardware costs and it was estimated that in 2010, the power a server burns over its lifetime will cost more than the server itself. This problem has driven data center based companies to set up sites near power stations and design some of them to be wind-cooled naturally to save on cooling costs.

Designing processors for a given power budget and the design of various power management features of computer systems demand an intimate understanding of the power characteristics of the system under design. Specifically, knowing the realistically attainable worst case power behavior of the system is critical in designing most of the above mentioned power related utilities. Estimating this maximum power dissipation of a processor is needed for designing heat sink/cooling for the chip/system. Overheating of chips can result in timing errors and even failure of the chip/system. For a variety of such reasons, chip designers, computer architects and system designers need to understand maximum power characteristics of designs. This worst-case power consumption has a direct impact on attainable microprocessor performance and implementation cost. Current generation multi-core performance is almost universally limited by power delivery, cooling and reliability rather than critical path delay. In order to limit the cost of power delivery and cooling, manufacturers artificially limit the peak frequency of multi-core processors. This eliminates the need to design costly voltage regulators and cooling solutions to handle worst-case power consumption. It is crucial that an accurate attainable worst-case power is known for a given processor. If this value is estimated too conservatively (i.e. overestimate), performance is unnecessarily limited. If it is estimated too aggressively, the processor will suffer problems with reliability. Identifying the attainable worst-case power in current generation microprocessors is a challenging task that will only become more daunting in the future. As additional system components are integrated into single packages, it becomes increasingly difficult to predict aggregate worst-case power. Existing designs integrate multiple cores and memory controllers on a single die. The trend for future designs is to include a wider array of components including graphics processors and IO bus controllers [1] [27].

Chip designers have often written hand crafted power viruses for this purpose [4]. But, it is very tedious to manually write a power virus for a given architecture. But perhaps more importantly, one cannot be sure about the fact that the generated power virus is actually a maximum case. There are also torture tests such as *MPrime* [5] and *CPUburn-in* [6]. But, industry designers often worry that their max power benchmarks (or stressmarks as they are often called) are inadequate [3] . We automatically generate stressmarks by using a machine learning based search through a workload space constructed with microarchitecture independent characteristics broadly classified into instruction mix, instruction level parallelism, control flow behavior, memory access patterns and memory level parallelism. Joshi et al. [15] also presented an automatic approach to the generation of power viruses. But, their work has many limitations, i) their study was limited to the power consumption of the CPU and did not include the DRAM ii) the framework was tested only by comparing with SPEC workloads and not with industry standard hand crafted power viruses iii) the results were only based on a simulator and was not validated on real hardware iv) they had not modeled the burstiness of memory accesses or the Memory Level Parallelism (MLP) of the workloads v) it was done only for the Alpha ISA

This paper overcomes the aforementioned limitations and generates system level power stressmarks. Though the processor consumes the maximum power among the various subsystems of a system, recent trends have shown that the power consumption of the DRAM subsystem is significantly high [19] [17] and is predicted to increase in the future. Thus, it is important to characterize the power consumption of the entire system rather than just the processor while constructing max-power viruses. Our metrics include the burstiness of accesses to DRAM by characterizing memory level parallelism. The granularity of the instruction mix in the generated synthetic is very important to generate good power viruses and our synthetic benchmark generation is more fine grained in terms of the number of instruction types generated than compared to Joshi et al's work [15]. The significant contributions of this paper are: i) propose the usage of SYMPO, which is an abstract workload generator integrated with an industry-grade Genetic Algorithm (GA) tool set, the IBM SNAP [2] [26], for generating system level power viruses in Sparc, Alpha, and x86 ISAs. ii) validation of the efficacy of the generated power viruses on a full system simulator by comparison with that of the popular stressmark *MPrime* (torture test) and show that our methodology results in power viruses that consume 16-40% more power than *MPrime*. iii) validation of the generated power virus on real hardware using an instrumented AMD quad-core Phenom II system. We show that our power virus has a power consumption higher than most of the other state of the art hand crafted power viruses. iv) comparison of the efficacy of the power viruses generated in Alpha ISA to that of the previous approach by Joshi et. al [15], showing that SYMPO results in consumption of 9-24% more power than the previous approach.

It is to be noted that the worst-case power of a system is not simply the sum of the maximum power of each component. Due to underutilization of resources and contention for shared resources, such as caches or memory ports, the aggregate worst-case is significantly less than the sum. The aim of this work is to find a reasonable worst case power virus that is present in the real world workload space.

The rest of the paper is organized as follows. In section 2, we provide a survey of the various industry standard power viruses along with their power consumption on real hardware. Section 3 introduces SYMPO, our power virus generation framework and section 4 elaborates on the experimental setup using full-system, processor simulators and results showing the effectiveness of the generated power viruses in comparison to that of the *MPrime* based torture test. Section 5 provides a microarchitecture independent characterization of the industry grade power viruses. We provide the related work in section 6 and summarize in Section 7.

Power Virus	Optimized to stress	Language
MPrime	All CPUs, all ISAs	C
CPUburn-in	X86 machines	x86 assembly
BurnP5	Intel Pentium w&w/o MMX processors	x86 assembly
BurnP6	Intel PentiumPro, Pentium II, Pentium III and Celeron CPUs	x86 assembly
BurnK6	AMD K6 processors	x86 assembly
BurnK7	AMD Athlon/Duron processors	x86 assembly
BurnMMX	cache/memory interfaces on all CPUs with MMX	x86 assembly

Figure 1: Power viruses widely used in the industry

2. STATE OF THE ART POWER VIRUSES

There have been many industry efforts towards writing power viruses and stress benchmarks. Among them, *MPrime* [5], *CPUburn-in* [6], *CPUburn* [4] are the most popular benchmarks. We first give a brief description of these power viruses and then characterize them based on microarchitecture independent metrics.

MPrime [5] is a BSD software application that searches for a Mersenne prime number using an efficient Fast Fourier Transform (FFT) algorithm. For the past few years, *MPrime* has been popularly called the torture test and has been used for testing the stability of a computer system by overclockers, PC enthusiasts and the processor design industry. This is because of the fact that this program is designed to subject the processor and memory to an incredibly intense workload resulting in errors. The amount of time a processor remains successfully stable while executing this workload is used as a measure of that system's stability by a typical overclocker. *MPrime* has been used in testing the CPU, memory, L1 and L2 caches, CPU cooling, and case cooling efficiencies.

CPUburn-in [6] is advertised as an ultimate stability testing tool written by Michal Mienik, which is also written for overclockers. This program attempts to heat up any x86 processor to the maximum possible operating temperature. It allows the user to adjust the CPU frequency to the practical maximum while still being sure that stability is achieved even under the most stressful conditions. The program continuously monitors for erroneous calculations ensuring the CPU does not generate errors during calculations. It employs FPU intensive functions to heat up the CPU.

CPUburn [4] is a power virus suite written in assembly language, copyrighted but freely licensed under the GNU Public License by Robert Redelmeier. The purpose of these programs is also to heat up x86 CPUs as much as possible. Unlike *CPUburn-in*, they are specifically optimized for different processors. FPU and ALU instructions are coded at the assembly level into an infinite loop. The goal has been to maximize CPU temperature, stressing the cooling system, motherboard and power supply. The programs are *BurnP5, BurnP6, BurnK6, BurnK7, BurnMMX*. The description of each of the power viruses are given in Figure 1.

Cores	4
Clock	3.0 GHz
Technology	45 nm
IL1, DL1 Cache	2-way, 64 B line, 64 KB per core
L2 Cache	16-way, 64B line, 512 KB per core
L3 Cache	32-way, 64B line, 6MB shared
Fetch size	32 bytes
Branch Predictor	12 global history, 16 K bi-mod predictor 2 K target buffer
ROB	72 entries in 3 micro-op groups
ALU	3 symmetry Int ALUs, 3 FP ALUs

Figure 2: Machine configuration of AMD Phenom II

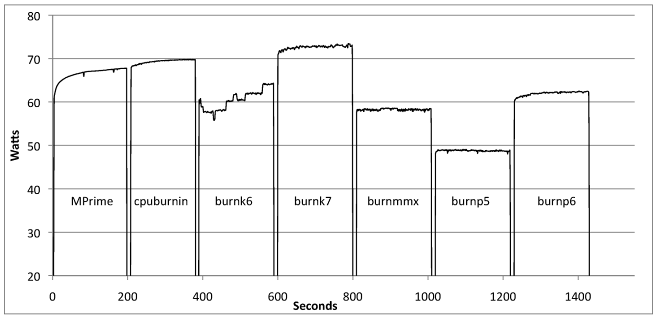


Figure 3: Power measurement on quad-core AMD Phenom II

2.1 Power Measurement Using Instrumented Hardware

To see the effectiveness of these power viruses on real hardware, we measure their power and thermal characteristics on the AMD Phenom II X4 (K10) Processor Model 945 system. Figure 2 shows the configuration of this system. The CPU core power of this system is measured using in-system instrumentation. A specialized AMD-designed system board is used which provides fine-grain power instrumentation for all power rails, including CPU core. Each high-power rail, such as CPU core, contains a Hall-Effect current sensor connected at its origin. The sensor provides a 0-5V signal that is linearly proportional to the power flowing into the rail. The voltage signal is measured by a National Instruments PCI-6255 data logger. The data logger attaches to the current sensor through a small twisted pair conductor. The data logger samples current and voltage applied to each rail at a rate of 10KHz. Since the voltage cannot be assumed to be a constant due to droops, spikes and drifts, we measure both voltage and current to calculate power. Using the data logs, application power is calculated off-line with post-processing software.

The measured power of the various power viruses are shown in Figure 3. Four copies of these benchmarks were run on the quad-core hardware until the power consumption reached a stable state, which was around 200 seconds. It can be noted that *BurnK7* consumes the maximum power on this hardware of 72.1 Watts after reaching a steady state. Even the highest power consuming two SPEC CPU2006 workloads, 416.gamess and 453.povray consume only 63.1 and 59.6 Watts respectively. *BurnK7* consuming maximum power on this hardware can be attributed to the fact that the machine configurations of the AMD Phenom II (K10) and K7 are to some extent similar to each other. It can be observed that the power viruses generated for other machines like the *BurnP5*, *BurnP6* do not consume as much power as *BurnK7*, again showing the importance of developing a specialized power virus for each of the microarchitectures. In the next two sections, we describe our power virus generation framework and the validation by comparison with the industry grade *MPrime* torture test.

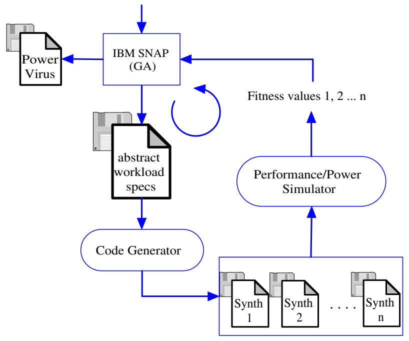


Figure 4: Overall methodology

No.	Metric	Range	Category
1	Number of basic blocks	10 , 30, 50, 100, 200	Control flow predictability
2	Average basic block size	10, 30, 50, 100	
3	Average overall branch predictability	0.8, 0.86, 0.92, 0.96, 0.98, 0.99, 1.0	
4	Integer ALU instruction weight	0 - 4	Instruction mix
5	Integer mul instruction weight	0 - 4	
6	Integer div instruction weight	0 - 4	
7	FP add instruction weight	0 - 4	
8	FP mul instruction weight	0 - 4	
9	FP div instruction weight	0 - 4	
10	FP mov instruction weight	0 - 4	
11	FP sqrt instruction weight	0 - 4	
12	Load instruction weight	0 - 4	
13	Store instruction weight	0 - 4	
14	Register dependency distance distribution in terms of number of instructions	1, 2, 4, 8, 12, 16, 20, 32, 48, 64	Instruction level parallelism
15	Local stride value per static load/store. 10 buckets each with equal probability	0, 4, 8, 12, 16, 32, 64 in each bucket	Data locality & memory level parallelism
16	Data footprint (number of iterations before reset to beginning of the array)	1, 10, 20, 40, 100, 200	
17	Mean of memory level parallelism	1, 2 , 3, 6	

Figure 5: Search space and parameter settings

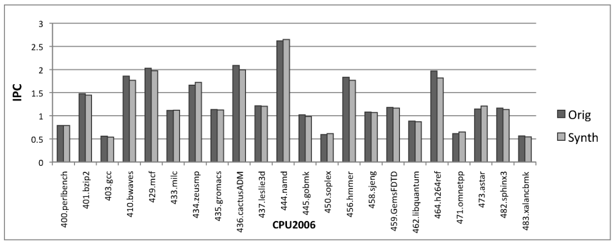
3. SYMPO FRAMEWORK

The overall power virus generation methodology is given in Figure 4. Our framework consists of 3 important components as described below.

3.1 Abstract Workload Model and Code Generator

Our workload space consists of a set of 17 dimensions falling under the categories of control flow predictability, instruction mix, instruction level parallelism, data locality and memory level parallelism as shown in Figure 5. An abstract workload synthesizer was developed to be able to generate a C file with embedded assembly instructions following the characteristics of a given workload specification. These embedded assembly instructions are packed into a loop and this loop is iterated until the performance characteristics converge. The workload specification consists of the following,

Instruction mix: Specifies the frequency of each type of instruction in the program. Each of the instruction type in the abstract workload model has a weight associated with it ranging from 0 to 4. The proportion of this instruction type in the generated synthetic is not only governed by this weight, but also based on the weights associated with the remaining instruction types as they are correlated with each other. As different instruction types have different latencies and power consumption, the instruction mix has a major effect on the overall power consumption of a workload. Since the


(a) IPC

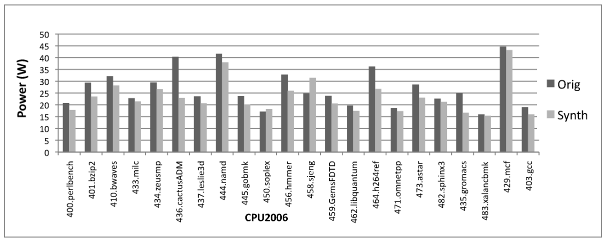

(b) Power

Figure 6: Comparison of IPC and power between the synthetic projection and the original workloads

code generator generates embedded assembly, we have direct control over the instruction mix of the generated workload. Based on a static analysis of the power consumption, the typical power consumption of integer and floating point ALUs for an out-of-order superscalar processor is around 4%-6% and 6%-12% respectively.

Number of basic blocks: The number of basic blocks in the program combined with the basic block size determines the instruction footprint of the application. The number of basic blocks present in the program has a significant impact on the usage of the instruction cache and its power consumption.

Basic block size: The basic block size of a program determines the frequency of branches and hence affects the control flow predictability of the workload. The percentage of correctly predicted branches in the generated code dictates the activity level of the branch predictor. The power consumption of a typical branch predictor is usually around 4%-5% of the overall processor power.

Register dependency distance distribution: The register dependency distance distribution has a significant impact on the overall ILP of the workload. This parameter affects the throughput of the pipeline directly. The overall throughput of the pipeline is related to the power consumed by the instruction window, clock and rename logic. The instruction window and the clock power are significant contributors to the power consumption of a processor ranging around 8% and 20% respectively.

Branch predictability: The branch predictability of a workload is an important characteristic that also affects the overall throughput of the pipeline. When a branch is mispredicted, the pipeline has to be flushed and this results in a reduced activity in the pipeline.

Local strides: The memory access pattern of the workload is modeled using the stride based model for each of the static load and stores in the generated code. Each of the static load and store in the workload walk a bounded array of memory references with a constant stride. This parameter along with the data footprint of the application directly controls the power consumption in various levels of caches and the pipeline throughput due to cache misses.

Data footprint: The data footprint of the application controls the number of cache lines that will be touched by the different static loads and stores. Also, it has a direct impact on the power consumption of the data caches.

Memory level parallelism: The average number of long-latency loads (loads with big strides that can miss in the last level of the cache) outstanding when there is at least one long-latency load outstanding is called as the Memory Level Parallelism (MLP) present in a workload. The memory level parallelism of a workload also affects the power consumption due to its impact on the DRAM power and also the pipeline throughput.

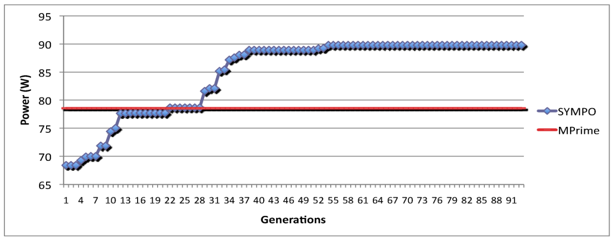
The code generator goes through 4 steps in the process of generating the synthetic, viz., i) generating the skeleton of the code using the number of basic blocks, basic block size and the instruction mix information ii) configure the branches to a particular predictability iii) configure the memory access instructions to follow the specified memory access model iv) assign the source and destination registers to each instruction based on the specified dependency distance distribution. The memory access model used in this paper is governed by 3 parameters in our model i) a constant stride based walk of each static load/store through a declared memory array as detailed by Ganesan et al [11]). ii) resetting the memory pointers to the starting address of the array to make the walk a bounded memory stream equal to the desired data footprint iii) the load-load dependency distance is varied to control the memory level parallelism in the workload. To achieve a given branch predictability for the workload, each of the branches in the code are classified into two categories, viz., branches which have a high transition rate between taken and not taken paths and branches which have very low transition rate. Branches are made to transition between the taken and the not taken paths based on a modulo operation. By having the right mix of these two kinds of branches, we can achieve almost any given branch predictability.

3.2 Validation of the Search Space

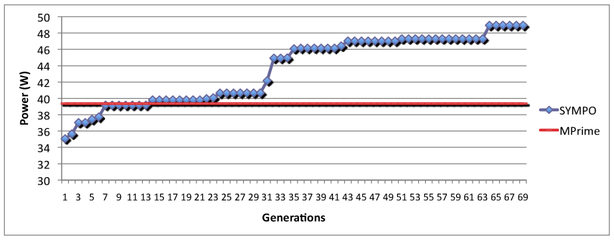
After setting up the abstract workload model that needs to be searched through to find a power virus, the next step is to validate the fact that this workload space is representative of the real world applications. This is an important step in this study, because our aim in this work is not to find the theoretically max-power consuming virus, rather to generate a virus that can be possibly written by a normal user. Since our workload model is completely independent of the underlying microarchitecture, the model being detailed enough is very important to enable the machine learning algorithm to be able to stress fine grain processor components. To validate our workload model, we have projected a set of the SPEC CPU2006 workloads onto our abstract workload space and measured the similarity in the performance and power characteristics between the projection (miniature synthetic) and the original workload. Figures 6(a) and 6(b) show the errors in IPC and power metrics for this projection between the original and the synthetic. The error in IPC averages around 2.8% and that in power averages around 14%. The higher error in power consumption can be attributed to the fact the generated synthetics almost always underestimate the power consumption to some extent than the original workloads.

3.3 Power Estimation Using Simulators

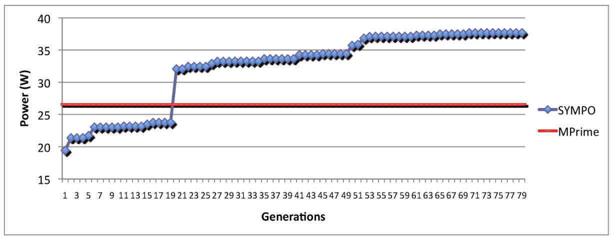
In our framework, we use two simulator tool sets to evaluate the power consumption of a workload on a given machine configuration. For the SPARC ISA, we use the Simics full system simulation environment with the Multifacet GEMS [22] tool set with an integrated DRAMsim [29] simulator to find the system level power consumption. We use the cycle accurate out-of-order processor simulator Opal and the detailed memory simulator Ruby, both in-



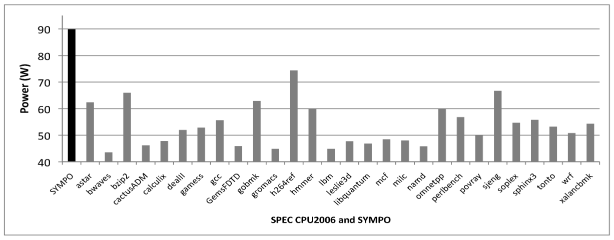
(a) Config 1



(b) Config 2



(c) Config 3



(d) Comparison with SPEC CPU2006

Figure 7: Power consumption of the best individual of the different generations of GA execution for the SPARC ISA using the Simics/GEMS environment

cluded in GEMS for all our experiments. Opal integrates power models from Wattch [23] and can produce both power and performance estimates. The DRAMsim is integrated with Ruby to accurately simulate the memory controller and a DDR2 DRAM. We compile all our workloads using GCC 3.4 compiler on the Solaris 10 operating system for the SPARC V9 architecture. This infrastructure is used to evaluate the power consumption of the system including the processor core, caches and the main memory for a given workload and a machine configuration. For the experiments using the Alpha ISA, we use the Wattch simulator built on the SimpleScalar [7] simulation framework to evaluate the CPU power consumption. We compile our workloads on an alpha machine running the Tru64 UNIX operating system using gcc 4.2. We use the most aggressive clock gating model cc3 provided by Wattch for both the SPARC and the Alpha ISAs.

3.4 IBM SNAP - The GA Tool Set

Among the various machine learning techniques, Genetic Algorithm (GA) [9] is known to be very effective with respect to global optimization problems. GA is a particular class of search heuristics that use techniques like mutation, crossover, inheritance and selection to solve optimization problems. A population in the genetic space is referred to as the set of potential candidate solutions. Initially, GA starts with a population of a set of random individuals or chromosomes. In each generation, the fitness of each individual in the population is evaluated and based on these fitness values, multiple individuals are selected from the current population. The individuals are crossed over or randomly mutated to form a new population for the next generation. The above process of evolution happens until the desired fitness is achieved. The GA tool set, IBM SNAP [2] [26] takes in the description of the search space in terms of the bounds for the various parameters in the abstract workload model given as input by the user. SNAP initializes the individuals of the first population with random workload characteristics. The fitness of each of the individuals is evaluated, which in our case is the power consumption on the microarchitecture under investigation using the simulation infrastructure as described above. Based on the fitness of each of the different individuals, the next generation of individuals is generated using the genetic operations of copy,

	Config 1	Config 2	Config 3
ALU	4 Int, 4 FP	3 Int, 2 FP	1 Int, 1 FP
L1 I- & D-Cache	64 KB, 256 MSHR	32 KB, 128 MSHR	16 KB, 64 MSHR
L1 Cache Latency	3 cycles	2 cycles	1 cycles
L2 Cache	4 MB, 128 MSHR	2 MB, 64 MSHR	512 KB, 32 MSHR
DRAM	8 GB	4 GB	2 GB
ROB	256 entries	128 entries	32 entries
Machine Width	8	4	2

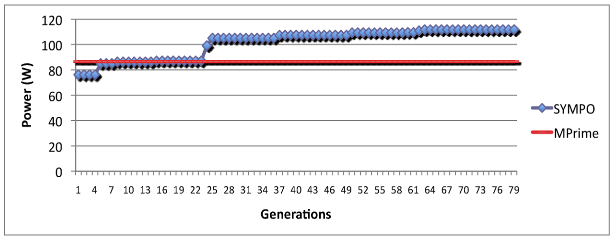
Figure 8: Machine configurations used to evaluate SYMPO on SPARC ISA

crossover and mutation. SNAP provides the following parameters to control how the individuals are chosen for the next generation, i) Mutation rate: number of individuals that should be probabilistically chosen to mutate ii) Reproduction rate: number of individuals that should be probabilistically chosen to copy into new population iii) Elite reproduction rate: number of fittest individual of previous generations that should be copied into new generation iv) Crossover rate: number of individuals probabilistically chosen to serve as parents for point crossover, where a crossover point within a parent is selected and then interchange the two parent chromosomes at this point to produce two new offsprings. v) Uniform crossover rate: number of individuals probabilistically chosen to serve as parents for uniform crossover. Uniform crossover is the process in which individual bits in the chromosome are compared between two parents and are swapped with a fixed probability of 0.5.

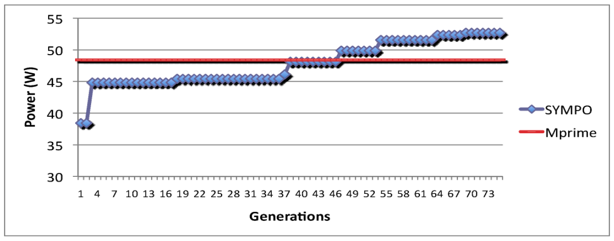
After the workload parameters of the individuals for the next generation are constructed, they are fed to the code generator to generate the synthetic clone. This synthetic clone is automatically compiled and run on the corresponding processor/full-system simulator to evaluate the power consumption, which is used as a feedback to generate the next generation of individuals. SNAP continues to explore the space until the required fitness is reached or until there is no forward progress for a given number of generations.

4. RESULTS AND ANALYSIS

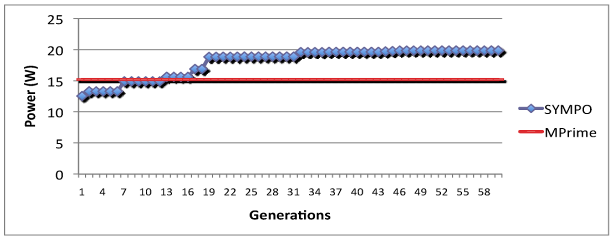
The effectiveness of the power viruses generated by SYMPO are validated on the SPARC and the Alpha ISAs by comparing with the industry grade power virus *MPrime* torture test along with mea-



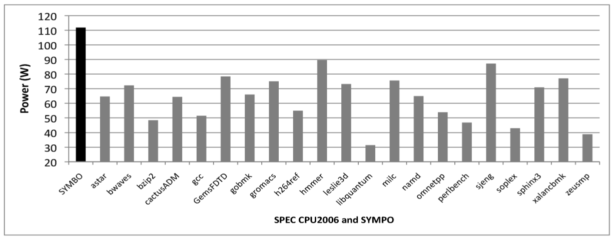
(a)



(b)



(c)



(d) Comparison with SPEC CPU2006

Figure 9: Comparison of the power consumption of the generated power virus to that of *MPrime* on Alpha ISA using SimpleScalar

	Config 1	Config 2	Config 3
ALU	8 Int, 4 FP	4 Int, 2 FP	2 Int, 1 FP
L1 Cache	64 KB, 4-way	32 KB, 4-way	16 KB, 2-way
L2 Cache	4 MB, 8-way	4 MB, 8-way	256 KB, 4-way
ROB / LSQ	256/128	128/64	16/8
Machine Width	8	4	2
Branch predictor	Hybrid 4 KB	Hybrid 4 KB	2-level
Mem access time	150 cycles	150 cycles	40 cycles

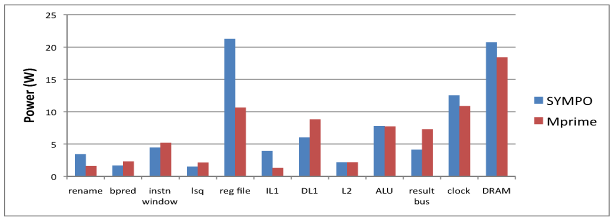
Figure 10: Machine configurations used to validate the power viruses on SimpleScalar

surements on the instrumented quadcore AMD system. Among the different industry grade power viruses discussed in section 3, *MPrime* is the only benchmark for which the source code is available. Most of the other power viruses discussed were handcrafted using x86 assembly and can only be used on x86 machines. Due to this limitation, we compare the power consumption of the SYMPO viruses only with that of *MPrime* on SPARC and Alpha ISAs. But, on x86 ISA, we use all the industry grade power viruses for comparison.

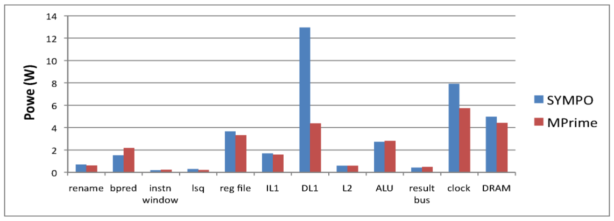
4.1 Results on SPARC ISA

To see the efficacy of using SYMPO to find power viruses, we compare the power consumption of the industry grade *MPrime* torture test with that of the individuals chosen by SYMPO for a set of 3 entirely different microarchitecture configurations using the GEMS full system processor simulator. The 3 microarchitecture configurations used are given in Figure 8. The microarchitectures differ in terms of number of functional units, cache sizes, instruction window size, DRAM size and the machine width. Figures 7(a), 7(b) and 7(c) show the increase in the power consumption of the best power virus as SYMPO progresses with each generation for each of the 3 microarchitectures respectively. The same figures also show the power consumption of *MPrime* torture test for comparison. The power viruses generated by SYMPO consume 14%, 24% and 41% more power than *MPrime* for microarchitectures 1, 2 and 3 respectively. For the above results, GA was seeded with random workloads and run for 91, 69 and 79 generations for each of the microarchitectures resulting in 728, 552 and 632 simulations. The number of dynamic instructions in the power viruses were set to be 10 million. It is to be noted that the caches get warmed up in just a few thousand instructions in the synthetic and the power consumption converges to steady state in not more than 10 million dynamic instructions. Since these simulations were done on a Xeon parallel machine, the fitness evaluation for the individuals in a generation were let to run in parallel resulting in an efficient exploration consuming a total simulation time of 15 hours, 11 hours and 13 hours for SYMPO to generate the viruses for the machine configurations 1, 2 and 3 respectively. The GA parameters that were used and found to be well suited to explore this particular search problem are a mutation rate of 0.03, reproduction rate of 0.01, elite reproduction rate of 0.125, crossover rate of 0.825 and a uniform crossover rate of 0.01. Since many parameters in our search space are correlated with each other, having a higher non-disruptive point crossover rate performs better than having higher disruptive uniform crossover rate.

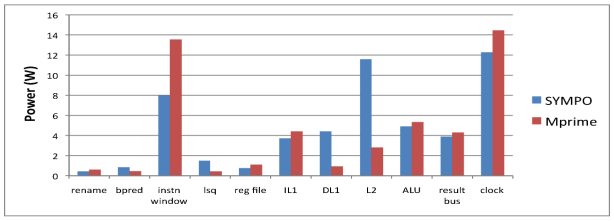
To further compare the power consumption of the generated power virus to that of the real world applications, the SPEC CPU2006 workloads were simulated on our full system simulation infrastructure for 1 billion dynamic instructions after fast forwarding for 2 billion instructions on machine configuration 1. Figure 7(d) shows the power consumption of the SYMPO virus compared to real world SPEC workloads. The SPEC workloads have an average power consumption of 53.4 Watts compared to 89.8 Watts consumed by the SYMPO virus. It is to be noted that the power viruses generated for each of the configurations are different. For instance the characteristics of the power virus generated for machine configuration 1 are a basic block size of 10 instructions, 200 static basic blocks, the memory pointers are reset to beginning every 200 iterations, a branch transition rate of 0.98, 10% of the memory access instructions having a stride of 12 and 90% of the memory access instructions have a stride of zero, a memory level parallelism of 1. The instruction mix of this power virus was int ALU-19.5%, int mul-6.5%, int div-19.5%, FP mov-19.5%, load-6.5%, store-19.5% and branch-10%. The characteristics of the virus generated for machine configuration 3 was significantly different in its instruction mix compared to that generated for machine configuration 1. The instruction mix of the power virus generated for machine configuration 3 was int ALU-18%, load-36.3%, store-36.3% and branch-10%. It is very hard to make general inferences about the im-



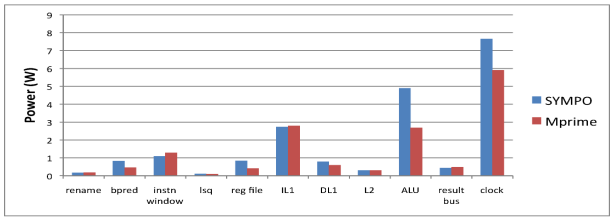
(a) SPARC config 1



(b) SPARC config 3



(c) Alpha config 2



(d) Alpha config 3

Figure 11: Comparison of the power consumption of individual components of the generated power virus to that of *MPrime* on SPARC ISA using Simics/GEMS and Alpha ISA using SimpleScalar

portance of the characteristics of the synthetics for the various hot cases as they vary extensively based on the targeted machine configuration. The main aim of using micro-architecture independent characteristics along with machine learning for this problem is to be able to have a black box approach towards the generation of the power virus and avoid making models/inferences about how the power virus should be designed given a machine configuration.

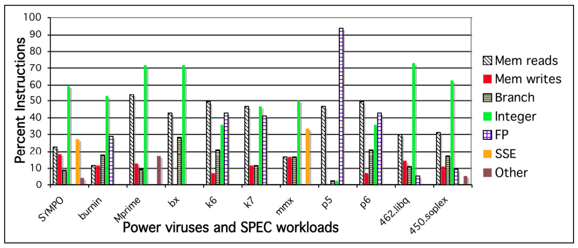
4.2 Results on Alpha ISA

Figures 9(a) 9(b) 9(c) show the results of using SYMPO for generating power viruses in the Alpha ISA to maximize the power consumption in the processor core for configurations 1, 2 and 3 as given in Figure 10. The machine configurations used for the experiments on the Alpha ISA are the same as used in the previous work by Joshi et. al [15] to enable us to do a direct comparison of the power consumption of the generated viruses. The power virus generated using SYMPO consumes 30%, 7.5% and 29% more power in the processor core than *MPrime* torture test on Alpha ISA. To be able to make a fair comparison to the stressmarks of the previous approach [15] we compare only the CPU power for all the experiments in the Alpha ISA. It is to be noted that the power viruses generated using SYMPO consume 15%, 9%, and 24% more power than the stressmarks generated for the same set of machine configurations using the Wattch simulator by the previous approach. This improvement in the power consumption is attributed to the fact that we model the instruction mix at a finer granularity and we also model the memory level parallelism in the synthetic. In our previous work [11] aimed at generating miniaturized clones for long running workloads, we show that the memory level parallelism of a workload is a very significant factor when modeling the performance and power consumption of a workload even at the core level. We show an improvement of 12.5% in the accuracy of the workload model when including memory level parallelism. It should also be noted that our GA framework (SNAP) is more sophisticated, enabling us to explore a larger search space than what was used in [15].

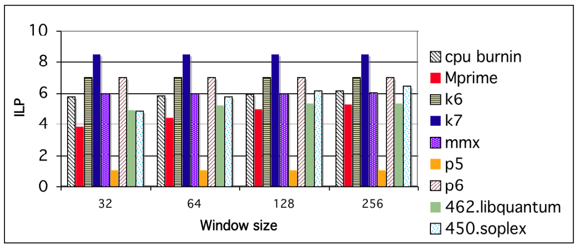
Figure 9(d) shows the power consumption of the SYMPO virus compared to real world SPEC workloads. The SPEC workloads have an average power consumption of 63.22 Watts compared to 111.79 Watts consumed by the SYMPO virus. The sum of the power consumption numbers of all the units present in a machine defines the 'theoretical maximum' for this max-power search problem. Since all the units of a machine cannot be kept busy all the time by any practical real world workload, reaching this theoretical maximum is almost an impossible event. For example, the theoretical maximum for the machine configuration 3 is 220 Watts and the power virus generated for this configuration consumes a sustainable average power of 112 Watts. Designing a system with a worst case power behavior equal to that of the theoretical maximum can result in highly wasteful over provisioning. This further motivates the necessity towards using an automatic search to be able to design a system for a reasonable worst case behavior.

4.3 Suitability of Genetic Algorithm (GA) for SYMPO

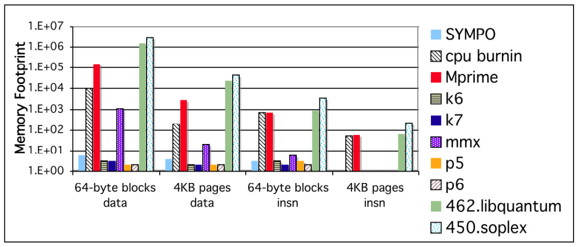
Figures 11(a) and 11(b) show the break down of the power consumption in each individual component of the system including the DRAM for the SPARC configurations 1 and 3 respectively. From this breakdown, It can be observed that SYMPO leverages the power consumption in the DRAM to maximize the overall system level power consumption. This shows that a power virus generated specifically for the CPU may not be the best power virus at the system level. It is to be noted that the virus generated by SYMPO does not always consume more power than *MPrime* in various components. This is due to the reason that the fitness function that the search algorithm targets to maximize is the total power consumption of the system. The same framework can be used to generate different types of stressmarks as in [15] by changing the fitness function evaluation. For the Alpha ISA, Figures 11(c) and 11(d) show the components of the power consumption in the various parts of the CPU. Figure 9 shows that each of the synthetic workloads have interacted with the different microarchitectures in a unique fashion. This shows that how non-trivial it is to hand craft a max-power virus by speculating about this complex interaction, thus emphasizing the need for an automatic search methodology. Machine learning based approaches are considered to be more efficient than most of the brute force searches and GA has proven to be a promising solution to this problem. In the machine configurations, SPARC config-3 has a lower access latency for L1 cache than config-1 and one can observe that SYMPO is aware of this and generates a power virus that stresses the L1 cache effectively.



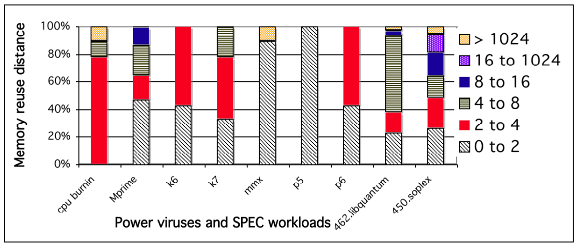
(a) Instruction mix



(b) Instruction level parallelism



(c) Memory footprint



(d) Memory reuse distribution

Figure 12: The instruction mix, instruction level parallelism, memory footprint and memory reuse distance of the industry standard power viruses

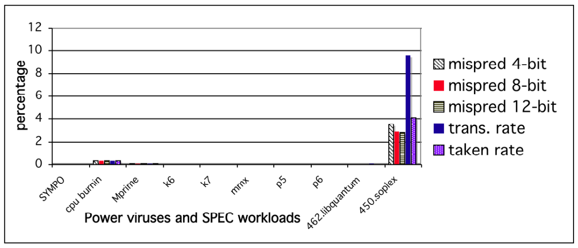
4.4 Validation of SYMPO on Real Hardware

Having validated the power consumption of the viruses generated using SYMPO on simulators using two different ISAs, the next step is to measure the power consumption of a the viruses on real hardware. Since our code generator was not equipped to generate code using CISC ISAs, we constructed a microarchitecturally equivalent system for the instrumented AMD Phenom II system on GEMS full system infrastructure and first generated power viruses in SPARC ISA. These viruses were ported to x86 ISA with the help of the LLVM [16] compiler infrastructure and the power consumption was measured on real hardware. These indirectly generated virus has a power consumption of 72.5 Watts on the cores, which is more than all the other hand crafted power viruses that were executed on this hardware, viz., *CPUburn-in*-68.42 W, *MPrime*-68.1 W, *BurnK6*-64.2 W, *BurnK7*-72.1 W, *BurnMMX*-58.4 W, *BurnP5*-48.7 W and *BurnP6*-62.4 W. The difference in the way the power viruses were generated for the SPARC/Alpha ISAs and the x86 ISA is that the automatic feed back loop was complete in the case of SPARC/Alpha ISAs due to the usage of the cycle accurate simulators, but there was no automatic feedback to SYMPO in the case of x86. The hardware power readings were manually fed to GA and since this process was too tedious, GA was run only for 3 generations. If the feedback loop could have been automated in x86, the generated power virus might have consumed even higher amounts of power. This shows the importance of automating the process of a max-power virus generation as compared to the usage of enormous human effort.

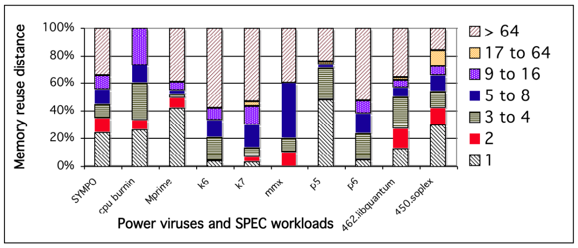
5. CHARACTERISTICS OF THE INDUSTRY GRADE POWER VIRUSES

In this section, we analyze the characteristics of the industry grade power viruses and compare them to the power virus generated by SYMPO using microarchitecture independent metrics. To understand the functional behavior of the industry grade hand crafted power viruses, we use the tool MICA [13] to record and analyze the microarchitecture independent characteristics viz., the instruction mix, Instruction Level Parallelism (ILP) , memory footprint, memory reuse distance, branch predictability, register reuse distance and stride based memory access patterns. We also characterize two arbitrarily chosen SPEC CPU2006 benchmarks 450.soplex and 462.libquantum to see how different the power viruses are from normal workloads. Figure 12(a) shows the instruction mix of these different power viruses. It can be found that most of the workloads have a high percentage of integer operations. This can be attributed to the fact that the integer operations are relatively short latency compared to floating point operations and have the ability to keep the activity factor of the pipeline by keeping all stages of the pipeline occupied. Almost all the power viruses also have moderately high percentage of memory reads compared to other operations and with the help of a Pin [21] tool, we observed that these power viruses tend to always hit in the first level data cache. This can be attributed to the fact that caches in a processor are known to consume relatively more power than other components. Though the presence of short latency integer operations are known to increase power due to high activity, the floating point ALUs consume more power than the integer ALUs. This fact is evident from the presence of a substantial number of floating point operations in the power viruses written for AMD K7, AMD K6, Intel P5 and Intel P6. The instructions classified under 'other' in Figure 12(a) include semaphore, rotate, conditional move and interrupts.

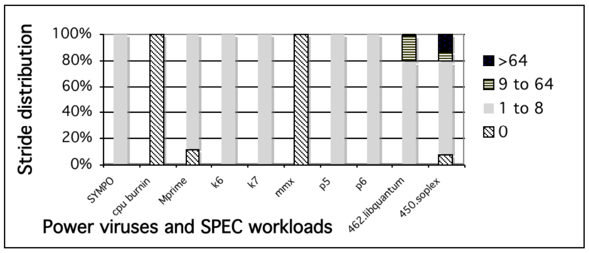
Figure 12(b) shows a measure of the Instruction Level Parallelism (ILP) of the different power viruses. In this case, instruction level parallelism is measured as the number of instructions that can be executed per cycle assuming a perfect cache, perfect branch predictor and all instruction latencies to be one cycle. It is found that most of these applications have higher ILP compared to the SPEC workloads. It is to be noted that *BurnK7* has a very high ILP measure of about 8.5. Figure 12(c) shows the memory footprint of these different applications compared to the SPEC workloads in a logarithmic scale. It can be noted that the most of these power viruses have a data foot print which is many orders less than the real world SPEC workloads. Figure 12(d) shows the memory reuse distance distribution for the various power viruses in terms of 64 byte blocks. Memory reuse distance is an important metric used to characterize the cache access behavior (locality) of a workload.



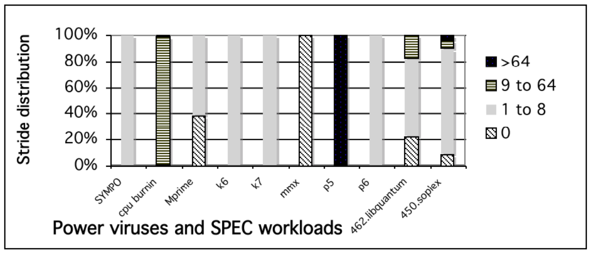
(a) Branch predictability



(b) Register reuse distance distribution



(c) Stride distribution of memory reads



(d) Stride distribution of memory writes

Figure 13: The branch predictability, register reuse distance and stride distribution of the industry standard power viruses

For each of the memory accesses, the cache block being accessed is determined. The number of unique cache blocks accessed since the last time it was referenced is recorded using an LRU stack. Figure 12(d) shows that most of the cache blocks being accessed by the power viruses are reused within 2 to 8 of other cache blocks being accessed. It is evident that these power viruses have a lot of memory reuse compared to the real world SPEC workloads to enable a quicker execution leading to higher power consumption. A Pin tool was used to simulate a DL1 cache size of 64 KB and the miss rates of these power viruses were studied. It was observed that *MPrime* had a miss rate of 3.35%, *CPUburn-in* had a miss rate of 12.5% and all the other benchmarks had a hit rate greater than 99.9%.

Figure 13(a) shows the percentage of branches that are mispredicted for the power viruses and the SPEC workloads. It can be observed that these power viruses have highly predictable branches compared to normal applications. High activity factor and high ILP is required to keep power high. Higher mispredictions will drain the pipeline and reduce chances of the pipeline being full/active. Most of these power viruses have high register reuse distances as given in Figure 13(b) showing the presence of high amounts of ILP in these workloads. Figures 13(c) and 13(d) show the distribution of a stride based memory access pattern, which is another metric to measure the data locality of the workloads under study. We observe that most of these power viruses have very small strides compared to real world applications showing the presence of high amounts of data locality in these workloads. *CPUburn-in* is the only application that has a dominant higher stride of 64 for the memory writes.

6. RELATED WORK

Joshi et al. [15] introduced the idea of automatic stressmark generation using an abstract workload generator. Joshi et al. also show that the characteristics of stressmarks significantly vary across microarchitecture designs, emphasizing the fact that separate custom stressmarks should be developed for different microarchitectures. In the VLSI community, there has been a lot of research to estimate the power consumption of a given CMOS circuit [8] [20]. To maximize the switching activity in these circuits, test vector patterns are generated using heuristics and statistical methods. Our approach and goals in this paper are similar to these previous research, except the fact that we generate embedded assembly instructions that can be compiled into a legal program instead of the generation of test vectors. The advantage of using legal programs to search for a stressmark is that it guarantees that the maximum power consumption is achieved within the normal operating constraints. Industry has developed hand-crafted power viruses [10] [12] [28] [14] [24] to estimate the maximum power dissipation and thermal characteristics of their microprocessors. Hand-crafted benchmarks are also used in generating temperature differentials across microarchitecture units [18]. Synthetic benchmark generation has been investigated in the past in [11] [25] for speeding up processor performance estimation. Ganesan et. al [11] show the importance of characterizing the memory level parallelism of applications to accurately clone modern workloads.

7. SUMMARY

In this paper we have proposed the usage of SYMPO, a framework to automatically generate system level max-power viruses for a given machine configuration. We have shown that with the proposed workload space along the machine learning based search, we can automatically generate reasonably good power viruses for any given microarchitecture within a few hours. We have shown the efficacy of the power viruses by comparing their power consumption with that of *MPrime* for various microarchitectures in the SPARC, Alpha and x86 ISAs. A summary of these results are shown in Figure 14. These results clearly show that SYMPO is very effective in leveraging the power consumption on the SPARC, Alpha ISAs compared to the x86 ISA. If the feedback loop could have been completed in x86, the generated power virus might have consumed much high amounts of power. This shows the importance of automating the process of a max-power virus generation as compared to the usage of time consuming human effort.

We also show that the power viruses generated by SYMPO are superior compared to the automatically generated power viruses using the previously proposed methodology as in [15]. We measure the power consumption of the various industry grade hand crafted power viruses on an instrumented AMD Phenom II system and

	Sparc			Alpha			X86
	Config 1	Config 2	Config 3	Config 1	Config 2	Config 3	
SYMPO	89.8 W	48.95 W	37.68 W	19.86 W	52.70 W	111.8 W	72.5 W
MPrime	78.56 W	39.36 W	26.58 W	15.20 W	48.41 W	86.58 W	68.1 W
% increase	14.3 %	24.36 %	41.78 %	30.6 %	8.80%	29.1 %	2.3 %

Figure 14: Comparison of the power consumption of the virus generated by SYMPO to *MPrime* on Alpha, SPARC and x86 ISAs

compare it with that of an x86 power virus generated by SYMPO. We also provide a detailed analysis of these various industry grade hand crafted power viruses and the x86 virus generated by SYMPO based on a set of microarchitecture independent characteristics.

8. ACKNOWLEDGMENTS

This work has been supported and partially funded by SRC under Task ID 1797.001, National Science Foundation under grant numbers 0702694, 0751112, 0750847, 0750851, 0750852, 0750860, 0750868, 0750884 and 0751091, Lockheed Martin, Sun Microsystems, IBM and NSF China under grant number 60973036. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or other sponsors.

9. REFERENCES

[1] The industry changing impact of accelerated computing `http://sites.amd.com/us/Documents/AMD_fusion_Whitepaper.eps`.

[2] Private Communication with Jason F Cantin, IBM.

[3] Private Communication with Leslie Barnes, Advanced Micro Devices.

[4] `http://pages.sbcglobal.net/redelm`.

[5] `http://www.mersenne.org/freesoft`.

[6] `http://www.softpedia.com/get/System/Benchmarks/CPU-Burnin.shtml`.

[7] D. C. Burger and T. M. Austin. The simplescalar tool set, version 2.0. Technical Report CS-TR-97-1342. *University of Wisconsin, Madison*, June 1997.

[8] T. Chou and K. Roy. Accurate power estimation of cmos sequential circuits. *IEEE Transactions on VLSI Systems,*, 1996.

[9] L. D. Davis and M. Mitchel. Handbook of genetic algorithms. *Van Nostrand Reinhold*, 1991.

[10] W. Felter and T. Keller. Power measurement on the apple power mac g5. *IBM Tech Report RC23276*, 2004.

[11] K. Ganesan, J. Jo, and L. K. John. Synthesizing Memory-Level Parallelism Aware Miniature Clones for SPEC CPU2006 and ImplantBench Workloads. *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, March 2010.

[12] M. Gowan, L. Biro, and D. Jackson. Power considerations in the design of the alpha 21264 microprocessor. *Design Automation Conference*, 1998.

[13] K. Hoste and L. Eeckhout. Microarchitecture-independent workload characterization. *IEEE Micro Hot Tutorials, (Vol. 27, No. 3) pp. 63-72*, May/June 2007.

[14] R. Joseph, D. Brooks, and M. Martonosi. Control techniques to eliminate voltage emergencies in high performance processors. *High Performance Computer Architectures*, 2003.

[15] A. Joshi, L. Eeckhout, L. K. John, and C. Isen. Automated microprocessor stressmark generation. *The 14th International Symposium on High Performance Computer Architecture (HPCA)*, February 2008.

[16] C. Lattner and V. Adve. Llvm: A compilation framework for lifelong program analysis & transformation. *Proc. of the 2004 International Symposium on Code Generation and Optimization (CGO'04), Palo Alto, California*, March 2004.

[17] J. Laudon. UltraSPARC T1: Architecture and Physical Design of a 32-threaded General Purpose CPU. *Proceedings of the ISSCC Multi-Core Architectures, Designs, and Implementation Challenges Forum*, 2006.

[18] K. Lee, K. Skadron, and W. Huang. Analytical model for sensor placement on microprocessors. *ICCD*, 2005.

[19] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. Keller. Energy management for commercial servers. *IEEE Computer, 36(12):39âĂŞ48*, December 2003.

[20] C. Lim, W. Daasch, and G. Cai. A thermal-aware superscalar microprocessor. *ISQED*, 2002.

[21] C. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, W. S. a. V. J. R. Lowney, G., and K. Hazelwood. Pin: building customized program analysis tools with dynamic instrumentation.

[22] M. M. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, , and D. A. Wood. Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset. *Computer Architecture News (CAN),*, September 2005.

[23] M. Martonosi, V. Tiwari, and D. Brooks. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. *isca, pp.83, 27th Annual International Symposium on Computer Architecture (ISCA 2000).*

[24] F. Najm, S. Goel, and I. Hajj. Power estimation in sequential circuits. *Design Automation Conference*, 1995.

[25] J. Robert H. Bell, R. R. Bhatia, L. K. John, J. Stuecheli, J. Griswell, P. Tu, L. Capps, A. Blanchard, and R. Thai. Automatic Testcase Synthesis and Performance Model Validation for High Performance PowerPC Processors. *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS 2006)*, March 2006.

[26] S. Sharkawi, D. Desota, R. P, R. Indukuru, S. Stevens, and V. Taylor. Performance Projection of HPC Applications Using SPEC CFP2006 Benchmarks. *IEEE International Parallel & Distributed Processing Symposium*, May 2009.

[27] S. L. Smith. Intel roadmap overview. intel developer's forum 2009 san francisco, ca. `http://download.intel.com/pressroom/kits/events/idffall_2009/pdfs/IDF_SSmith_Briefing.eps`. September 2009.

[28] R. Vishwanath, V. Wakharkar, A. Watwe, and V.Lebonheur. Thermal performance challenges from silicon to systems. *Intel Technology Journal*, 2000.

[29] D. Wang, B. Ganesh, N. Tuaycharoen, K. Baynes, A. Jaleel, and B. Jacob. DRAMsim: A memory-system simulator. *Computer Arch. News, vol. 33, no. 4, pp. 100-107*, Sep 2005.