FEATURE SELECTION FOR SLICE-BASED WORKLOAD CHARACTERIZATION AND POWER ESTIMATION

THESIS

Presented to the Graduate Council of
Texas State University-San Marcos
in Partial Fulfillment
of the Requirements

for the Degree

Master of SCIENCE

by

Matthew V. Brock, B.S.

San Marcos, Texas
May 2010

FEATURE SELECTION FOR SLICE-BASED WORKLOAD CHARACTERIZATION AND POWER ESTIMATION

Committee Members Approved:

_____

Dan Tamir, Chair

_____

Anne Ngu

_____

Khosrow Kaikhah

_____

James Holt

Approved:

_____

J. Michael Willoughby
Dean of the Graduate College

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Dan Tamir, for his time and energy spent working with me for the past three years. I would like to give the biggest thanks to my wife, Brianna, for being both supportive and patient despite my frustrations and anxieties. I would finally like to thank both of my parents for always supporting me in all of my endeavors and always encouraging me to excel just a little bit beyond where I think I'm capable of.

This manuscript submitted on April 30th, 2010.

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS....iv

TABLE OF CONTENTS....v

LIST OF TABLES....viii

LIST OF FIGURES....ix

ABSTRACT....xii

CHAPTER

1. INTRODUCTION....1

2. BACKGROUND....7

- 2.1 Workload Characterization....7
- 2.2 Microprocessor Modeling and Simulation....9
- 2.3 Architecture and System Level Model Simulations....11
- 2.4 Partitional Cluster Analysis....16
  - 2.4.1 K-Means....18
  - 2.4.2 ISODATA....18
  - 2.4.3 Cluster Dispersion Metrics....20
- 2.5 Feature Selection....21
  - 2.5.1 Hill Climbing....24
  - 2.5.2 Genetic Algorithm....24
- 2.6 Known Power-Affecting Features....27

3. RELEVANT WORK....29

4. METHODOLOGY....35

4.1 Benchmark Applications....36
4.2 Architecture Simulator....37
4.3 Feature Extraction....38
4.4 Feature Selection....38
4.5 Cluster Analysis....39
4.6 Power Estimation....39

5. EXPERIMENTS....41

5.1 Algorithm Calibration and Evaluation....42
5.1.1 ISODATA Calibration....43
5.1.2 Feature Selection Calibration....47
5.1.3 Feature Selection Algorithm Evaluation....48
5.1.3.1 The Exhaustive Search....48
5.1.3.2 Genetic Algorithm....50
5.1.3.3 Hill Climbing....51
5.1.3.4 Small Feature Subset Sizes....52
5.1.3.5 Exhaustive Search Versus Genetic Algorithm Search....55
5.2 Affecting Factors of Power Estimation Accuracy....59
5.2.1 ASTD Slice Size....59
5.2.2 ISODATA Slice Selection....61
5.2.3 Cluster Quality....65
5.3 Verification of Hypotheses....68
5.3.1 ASTD Compression....69
5.3.2 Genetic Algorithm Feature Selection....74
5.3.3 Known Power-Affecting Features....78

6. ANALYSIS OF RESULTS....84

6.1 Algorithm Calibration and Evaluation....84
6.1.1 Feature Selection Algorithm Evaluation....84
6.1.2 Exhaustive Search Versus Genetic Algorithm Search....85
6.2 Affecting Factors of Power Estimation Accuracy....86
6.2.1 ASTD Slice Size....86
6.2.2 ISODATA Slice Selection....86
6.2.3 Cluster Quality....86
6.3 Verification of Hypotheses....87
6.3.1 ASTD Compression....87
6.3.2 Genetic Algorithm Feature Selection....87
6.3.3 Known Power-Affecting Features....87

7. CONCLUSIONS....89

7.1 Future Work....91

BIBLIOGRAPHY.....xciii

LIST OF TABLES

Table **Page**

1: ANOVA Table of Combined ISODATA Experiments.....45

2: ANOVA Table of Averaged ISODATA Experiments.....46

3: ANOVA Table of Averaged Genetic Algorithm Feature Selection Experiments.....47

LIST OF FIGURES

Figure **Page**

1: Instruction Stream, Trace Data, and System Level Output....12

2: The Foundation of the Methodology....15

3: A Clustered Set of Elements....17

4: The Filter Approach to Feature Selection....22

5: The Wrapper Approach to Feature Selection....23

6: A Generalized Example of a Genetic Algorithm Iteration....26

7: The Components of the Methodology....35

8: Exhaustive Search of Feature Subsets....49

9: Genetic Algorithm Identifying Optimal Feature Subsets....50

10: Hill Climbing Search of Feature Subsets....51

11: Subset Size and Cluster Quality, Slice Size 1000....53

12: Subset Size and Cluster Quality, Slice Size 2000....53

13: Subset Size and Cluster Quality, Slice Size 5000....53

14: Subset Size and Cluster Quality, Slice Size 10000....53

15: Subset Size and Power Estimation Accuracy, Slice Size 1000....54

16: Subset Size and Power Estimation Accuracy, Slice Size 2000....54

17: Subset Size and Power Estimation Accuracy, Slice Size 5000.....54

18: Subset Size and Power Estimation Accuracy, Slice Size 10000.....54

19: Exhaustive vs. GA Search, Basic Math Benchmark, k-Max: 10.....57

20: Exhaustive vs. GA Search, Fast Fourier Transform Benchmark, k-Max: 10.....58

21: Effect of Slice Size: Basic Math Benchmark.....60

22: Effect of Slice Size: Q-Sort Benchmark.....61

23: Effectiveness of ISODATA, Basic Math Benchmark.....63

24: Effectiveness of ISODATA, Fast Fourier Transform Benchmark.....64

25: Effect of Cluster Quality, Basic Math Benchmark.....66

26: Effect of Cluster Quality, Dijkstra Benchmark.....68

27: Compression Ratio, Basic Math Benchmark.....71

28: Compression Ratio, Basic Math Benchmark, Slice Size 1000.....72

29: Compression Ratio, Basic Math Benchmark, Slice Size 2000.....72

30: Compression Ratio, Basic Math Benchmark, Slice Size 5000.....72

31: Compression Ratio, Basic Math Benchmark, Slice Size 10000.....72

32: Compression Ratio, Q-Sort Benchmark.....73

33: Compression Ratio, Q-Sort Benchmark, Slice Size 1000.....74

34: Compression Ratio, Q-Sort Benchmark, Slice Size 2000.....74

35: Compression Ratio, Q-Sort Benchmark, Slice Size 5000.....74

36: Compression Ratio, Q-Sort Benchmark, Slice Size 10000.....74

37: Effect of GA Feature Selection on Basic Math Benchmark.....76

38: Effect of GA Feature Selection on Fast Fourier Transform Benchmark.....77

39: Effect of GA Feature Selection on Q-Sort Benchmark.....78

40: Effect of Power Affecting Features, Slice Size 1000.....80

41: Effect of Power Affecting Features, Slice Size 2000.....81

42: Effect of Power Affecting Features, Slice Size 5000.....82

43: Effect of Power Affecting Features, Slice Size 10000.....83

ABSTRACT

FEATURE SELECTION FOR SLICE-BASED WORKLOAD CHARACTERIZATION AND POWER ESTIMATION

by

Matthew V. Brock, B.S.

Texas State University-San Marcos

May 2010

SUPERVISING PROFESSOR: DAN TAMIR

Modeling a microprocessor results in defining its architecture (instruction set, registers, memory hierarchy, etc.), its micro-architecture (pipelines, branch prediction, etc.), and its hardware (gate level logic, cycle-accurate timing, circuitry, etc.). Several software-based modeling tools exist for describing and facilitating simulation of the microprocessor model. Simulation at the hardware or system level in order to estimate power consumption is time consuming due to the complexity and size of the system level

model. Simulation at the architecture level is significantly faster, but less detailed. If the simulation trace data generated by the architecture simulation are highly correlated with the operations of system level simulation, only part of the benchmark test may be required for simulation at the system level. In other words, if a number of trace sections, or slices, are identified as similar, then the corresponding instruction stream slices will yield similar power estimation results when simulated at the system level. Thus, identifying similar slices of the architectural simulation trace data may reduce the amount of benchmark testing required at the system level. Cluster analysis is used to identify similar slices of the architectural simulation trace data, resulting in clusters of slices. Feature selection is used to eliminate less relevant features within the trace data, allowing for better clustering. Finally, domain-specific knowledge is applied by prioritizing feature selection towards known power-affecting features. Experimentation demonstrates the effectiveness of the cluster analysis, feature selection, and use of power-affecting features in order to achieve accurate power estimation while reducing the amount of benchmark testing required at the system level of simulation.

1. INTRODUCTION

Microprocessor design involves several steps that occur before fabrication, one of which is workload characterization. Workload characterization is the process of determining a microprocessor's capabilities by executing a set of benchmark applications on the microprocessor and measuring certain aspects of the microprocessor during execution, such as cache and memory access, power consumption, cycles per instruction, and other important characteristics. To perform benchmark testing on a microprocessor which has not been fabricated, a virtual model of the microprocessor is implemented and the benchmark testing is simulated using a set of software tools.

Microprocessor models are generally abstracted into three levels: the architecture, micro-architecture, and the system levels. Software tools are capable of modeling and simulating the microprocessor model at these levels of abstraction (Armstrong & Woodruff, 1977). The system level is the lowest level of microprocessor model abstraction, and is thus the most accurate in terms of determining certain characteristics of the microprocessor, such as power consumption. However, estimation of power consumption at the system level of simulation is not always feasible. Due to the complexity and size of the system level microprocessor model, system level simulation is

considerably slower than higher levels of modeling and simulation, such as the architecture level.

Several methods exist for reducing the amount of time involved in power estimation. One method is identifying similar architectural event sequences, or slices, found in the architectural simulation trace data (ASTD). The assumption is that if architectural events are significantly correlated with system level events, then the architectural events can be used as classifying data for the system level of simulation (Hsieh, Chen, & Pedram, 2001). By identifying similar slices of ASTD, the corresponding instruction stream slices will return approximately the same power estimations to one another when simulated at the system level of abstraction. In order to identify these similar slices of ASTD, the slices are represented as elements in a multi-dimensional feature space, where each architectural event type (cache hit, instruction execution, etc.) represents a dimension, or feature, within the space.

Partitional cluster analysis enables the identification of clusters from the slices in the feature space (Jain, Murty, & Flynn, 1999). The Iterative Self-Organizing Data Analysis Technique (ISODATA) algorithm, a generalization of the *k*-means algorithm, enables flexible and adaptive partitional clustering of large data sets, in this case ASTD slices. A cluster of ASTD slices represents a set of similar architectural event sequences. The slice that most accurately represents the entire cluster, i.e. is closest to the cluster center, is used to determine the corresponding slice of the instruction stream to run via system level

simulation.

N = Number of representative slices
R = Set of cluster sizes for each representative slice, $\{R_0, R_1, .., R_N\}$
P = Set of power measurements for each representative slice, $\{P_0, P_1, .., P_N\}$
P_c = Power measurement of complete benchmark test
Relationship of power estimation:

$$P_c \approx \sum_{i=0}^{N} R_i * P_i \qquad \text{(EQ-1)}$$

The number of slices in the cluster, or cluster size, is used as a weight to multiply the power estimation value returned from the system level simulation of that representative slice. Thus an approximation of the total power estimation is provided for all of the instruction stream slices corresponding to the clustered ASTD slices. The first hypothesis of this paper is that the use of the ISODATA algorithm to cluster ASTD slices may result in accurate power estimation while reducing data throughput in system level simulation.

Depending on the ISODATA parameters, varying levels of power estimation accuracy may be achievable. However, some of the features, or architectural event types, may be less relevant. Feature selection enables the elimination of less relevant features by searching through the feature subset space, i.e. evaluating feature subsets and choosing the one resulting in highest cluster quality (Guyon & Elisseeff, 2003). Eliminating less relevant features results in greater clustering optimization, which in turn provides a stronger correlation between the architectural and system levels of simulation. The strong correlation may increase power estimation accuracy, as certain features, such as cache activity, are known to have a significant effect on power consumption. The second

hypothesis of this paper is that feature selection may increase power estimation accuracy by providing an optimal subset of features for ISODATA to use when clustering the ASTD.

As certain features are known to have a significant effect on power consumption, such as cache activity, use of these features in cluster analysis should result in increased power estimation accuracy. These features establish a minimal subset from which additional features can be selected. These additional features should contribute further to the discovery of a clustering configuration which results in higher power estimation accuracy. The third hypothesis of this paper is that adding the known power-affecting features to the selected subset of features may result in increased power estimation accuracy.

Our methodology is thus based on and evaluated according to three hypotheses:

1. The ISODATA algorithm may be effective in clustering ASTD slices, resulting in accurate power estimation while reducing data throughput in system level simulation.
2. Feature selection may result in increased power estimation accuracy by selecting an optimal subset of features for the ISODATA algorithm to use in clustering the ASTD slices.
3. Adding known power-affecting features to the optimal subset after feature selection may result in increased power estimation accuracy.

The first category of experiments in this paper are designed to provide auxiliary support to the hypotheses. The first experiment in this category calibrates the ISODATA and feature selection algorithms. These algorithms are tested with synthetic data to determine how to apply them to the experiment data. The next experiment evaluates exhaustive, genetic, and hill climbing feature selection algorithms using synthetic data in order to determine the best algorithm to apply to the experiment data. An additional experiment is performed with experiment data to demonstrate a lack of correlation between cluster quality and power estimation accuracy for very small feature subset sizes.

The second category of experiments explores various conditions which affect power estimation accuracy. The first experiment in this category demonstrates the effect of the ASTD slice size on power estimation. The second experiment compares the ISODATA algorithm to randomly selecting representative ASTD slices. The final experiment in this category observes the effect of cluster quality on power estimation accuracy.

The third category of experiments in this paper are designed to verify the hypotheses. The first experiment in this category demonstrates the trade-offs between ASTD compression and power estimation accuracy. The second experiment demonstrates the effect of feature selection on power estimation accuracy. The third experiment demonstrates the effect that known-power affecting features have on power estimation accuracy.

The results achieved by experimentation align with expectations, particularly those experiments which are designed to verify the hypotheses. Using ISODATA to cluster similar ASTD slices resulted in effective trade-offs between power estimation accuracy and instruction stream slice compression. The genetic algorithm feature selection yielded further gains in power estimation accuracy for most of the benchmarks. Selecting those features which are known to affect power yielded even further gains in power estimation accuracy for large window sizes. These three experiments, alongside the preceding foundational experiments, demonstrate that the methodology proposed in this thesis paper can be efficiently and effectively be utilized for slice-based characterization.

2. BACKGROUND

The following sections describe the background material required for understanding the basis of this thesis paper.

2.1 Workload Characterization

The topic of workload characterization is spawned from the complexity of predicting performance in computer systems (Downey & Feitelson, 1999). A microprocessor's workload represents the capabilities of the microprocessor to achieve various tasks. The complete characterization of a microprocessor's workload implies that every aspect of its performance is known. In other words, no program would produce unexpected results during execution. Workload characterization is an important aspect of microprocessor design, especially if it is accomplished before resources are invested into fabrication. For this paper specifically, power analysis will be the primary performance metric of interest with respect to workload characterization.

Currently, power consumption is estimated by several different methods such as instruction-level modeling, characterization-based macro-modeling, architectural event

sequence slicing, and phase-accurate system level simulation (Li & John, 2003). Instruction-level modeling is performed by associating a power consumption value with each instruction, factoring in energy dissipation from the circuit switching that occurs between instructions, and also taking into consideration other power-consuming activities that occur due to inter-instruction effects such as cache misses and write buffer stalls (Li & John, 2003). In characterization-based macro-modeling, a function or subroutine is modeled with a set of characteristics which correlate to various power consumption metrics. Regression analysis is applied to adjust macro-modeling coefficients based on a set of known function inputs and power consumption outputs, allowing the entire execution stream to be characterized for power consumption (Li & John, 2003). Architectural event sequence slicing extracts segments, or slices, of the data stream returned from an architecture level simulation of a benchmark test on a microprocessor model. These slices can be used for the purposes of sampling-based reduction or for identifying slices which approximate the entire benchmark test. Power analysis can then be focused on the samples or representative slices, reducing the data throughput required for system level simulation.

Finally, end to end system level simulation is the most accurate means for power consumption estimation, as it is the most detailed representation of the microprocessor's functionality. A system level simulator takes a model of a microprocessor and a benchmark test, and, using the specifications defined by the model, executes the entire benchmark test and returns some output to indicate properties of the microprocessor

model during execution.

2.2 Microprocessor Modeling and Simulation

Microprocessor design involves many layers of abstraction (Sherwood, 1977). The instruction set, memory cache hierarchy, pipelines, and other fundamental aspects of a microprocessor's design exist within one or more of these layers of abstraction. Generally, computer architecture is divided into three subcategories, or layers of abstraction (Hennessy & Patterson, 2003):

1. Architecture or Instruction Set Architecture (ISA)
2. Micro-architecture
3. System Design

The ISA is comprised of the instruction set, CPU registers, and memory addressing. The micro-architecture is comprised of the memory hierarchy, pipelining, branch prediction, instruction-level parallelism, out-of-order execution, multi-processing/threading and several other features that are considered below the ISA layer. System level design is the actual description of the hardware components such as buses, switches, gates, and memory controllers.

Microprocessor design makes use of various software simulation and modeling tools for the three subcategories of computer architecture described above. The advantages of

software simulation and modeling include (Sherwood, 1977):

1. Enabling design verification to take place before hardware prototypes are manufactured.
2. Correcting design flaws without physical modification.
3. Optimizing trade-offs between speed and cost.
4. Collecting more data on the given state of a microprocessor.
5. Studying physical anomalies and environmental conditions without the need to investigate individual signals.

Typically, these software tools involve some formal language definition for describing the microprocessor, such as the hardware description language Verilog. Specific uses for microprocessor simulation include formal hardware verification (Beatty & Bryant, 1944), power analysis (Hsieh et al., 2001) , and fault diagnosis (Armstrong & Woodruff, 1977).

One drawback to simulation, however, is the speed at which simulation takes place. Some aspects of microprocessor design, such as power analysis, can only take place at the system level of abstraction in order to be very accurate. Due to the complexity of modern microprocessors, however, simulation, specifically at such a low level of abstraction, is costly in terms of time. For this reason, system level simulation is often impractical when estimating a performance metric such as power consumption (Brandolese, Fomacian, & Salice, 2000).

2.3 Architecture and System Level Model Simulations

As the size and complexity of the system level model increases, the amount of time required to simulate the model also increases. As mentioned previously, this increase in size and complexity often occurs to the extent that simulating a benchmark on a system level model becomes impractical. At the architecture level, however, the simulation is faster and more practical.

Figure 1 describes the relationship between the benchmark's instruction stream, the ASTD from the architecture model simulation, and the output from the system level model simulation.

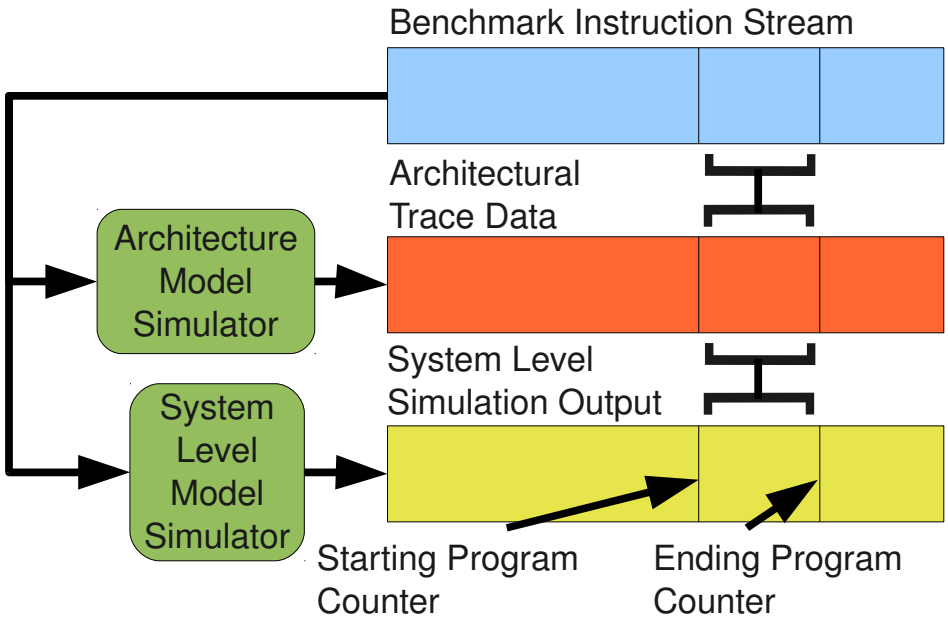

Figure 1: Instruction Stream, Trace Data, and System Level Output

The figure shows how the benchmark's instruction stream is executed via the architecture

and system level simulators to generate architectural trace data and system level simulation output, respectively. The benchmark's instruction stream, the architectural trace data, and the system level simulation output are synchronized via program counters. A slice of the architectural simulation data thus corresponds to the instruction stream and the system level simulation output via a starting and ending program counter. This enables the architectural trace data slices, which represent the entire benchmark test, to correspond to instruction stream slices. The instruction stream slices can then be simulated at the system level, given the corresponding architecture and system level models.

Simulating a benchmark with an architecture model results in architectural simulation trace data (ASTD). The ASTD describes a trace through the model, where various architectural features such as task level parallelism, cache access, register modification, and hardware instructions are represented (Kahne, 2006). Although ASTD does not contain enough information to be as accurate as the output of a system level model simulation, it does contain information to be correlated with the output of a system level model simulation. A section of output from the system level model simulation can thus be approximately characterized by the corresponding section of ASTD from the architecture model simulation. Program counters are used to track the interrelationships between the benchmark's instruction stream, ASTD, and system level model simulation output sections.

Generally, tracing the architecture model during the execution of the benchmark results in a significant amount of similar events, due to looping, recursion, and common paths of execution. If several slices of ASTD are identified as similar, the corresponding instruction stream slices should return similar power estimation results when run through the system level simulator. Identifying these slices can be accomplished by clustering the slices and identifying representative slices.

These slices can be clustered using a number of techniques. Regardless of technique, however, the events described in the architecture model simulation should correspond to the desired microprocessor characteristic returned from system level simulation. For instance, if the number of integer instructions occurring in a slice affects the amount of power consumption determined by the system level model simulator, then integer instructions should be taken into consideration when clustering the slices. A large number of event types, or features, can be extracted from the architecture model simulation. As such, selecting an optimal subset of features can improve the quality of clustering and reduce clustering time (Zhao, Wang, Wu, & Tang, 2002).

Figure 2 describes the concept of reducing the amount of instruction stream data to run through the system level model simulator.

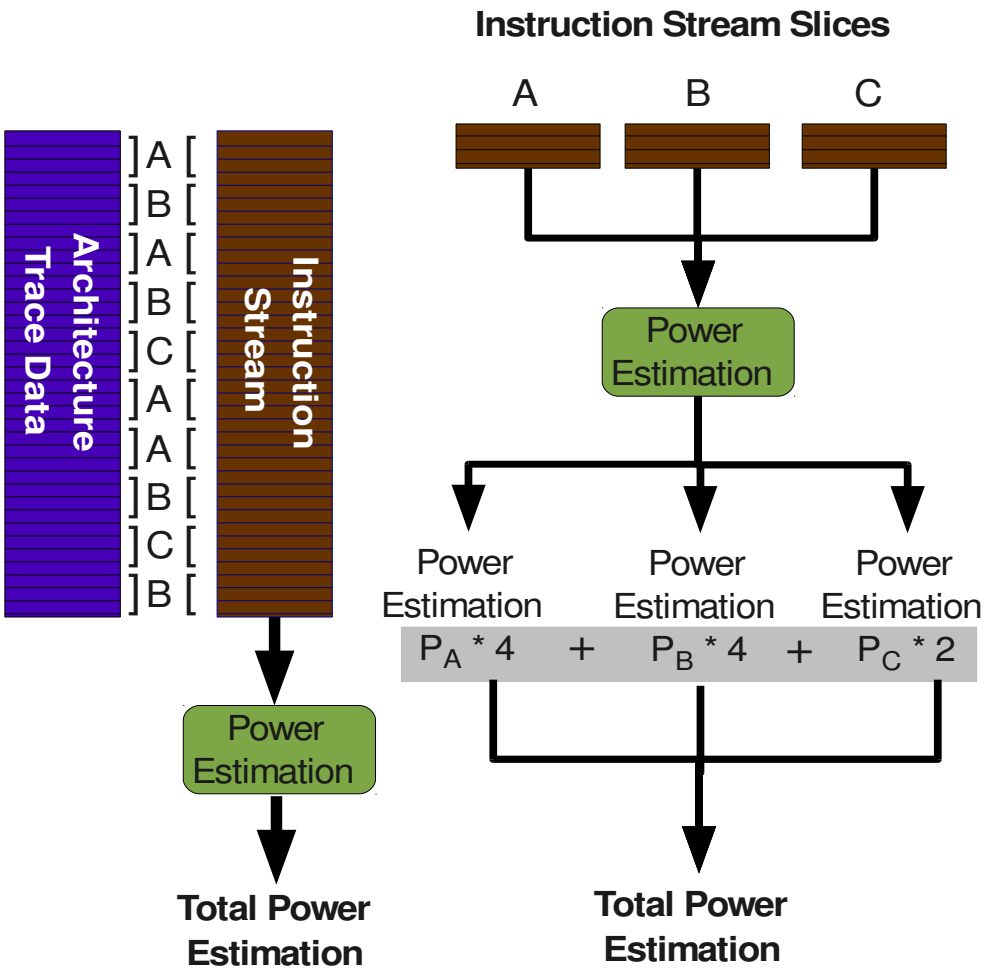

Figure 2: The Foundation of the Methodology

As shown in the figure, by determining similar ASTD slices, the corresponding instruction stream slices can be run through the system level simulator. The power estimation returned from simulating these instruction stream slices is multiplied by the number of occurrences for each slice and summed (see equation 1).

The scope of this thesis, however, is limited to architecture level simulation, and has substituted the system level model simulator with power estimation software that takes

architecture trace data and estimates power consumption. As such, the figures and results described in this thesis do not correspond to hardware-accurate power estimation achievable through system level simulation. The verification of our hypotheses is strictly limited to heuristic power estimation achieved from the architecture level features of the microprocessor model.

2.4 Partitional Cluster Analysis

Partitional cluster analysis consists of unsupervised classification grouping of data elements. These elements are represented as points in a multi-dimensional feature space, and the clusters are determined by a distance measure between the points (Jain et al., 1999). Evaluation of these clusters is calculated by the distance of the points within each of the respective clusters from the center of the cluster, as well as the distance between all of the cluster centers. The distance of the points within each of the respective clusters from the center of the cluster represents a measure of similarity for all the points within that cluster. The distance of the points within each of the respective clusters from the center of the cluster represents a measure of similarity for all the points within that cluster. The distance between cluster centers represents a measure of uniqueness for each of the clusters. These two factors provide a means for evaluating clusters in such a way that cohesive and identifiable clusters are discovered.

Figure 3 describes a set of elements in two-dimensional feature space that have been classified into clusters based on proximity.

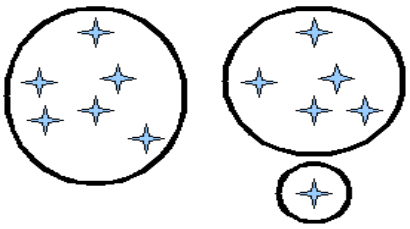

Figure 3: A Clustered Set of Elements

Partitional cluster analysis is useful with large datasets where element similarity is not easily distinguishable (Jain et al., 1999), as is the case for the data sets generated by the architecture model simulation. This classification process is visualized most easily when the elements are able to be plotted as points in two-dimensional space. In two dimensions, the clusters are often observed by humans. However, as the dataset becomes larger and more dimensions are involved, the ability to manually classify the data becomes exceedingly difficult. Hence, several formal methods for cluster analysis have been devised.

There are several other categories of clustering analysis methods, such as hierarchical, nearest-neighbor, artificial neural networks, and evolutionary (Jain et al., 1999). The partitional methods, specifically ISODATA, are best suited for clustering architecture simulation trace data slices due to the size and high dimensionality of the data sets.

2.4.1 K-Means

K-Means is a partitional cluster analysis algorithm which refines a set of *k* means until they have been shifted, over the course of several iterations, to the centers of *k* identified clusters of elements in the feature space. One problem with the k-means algorithm,

however, is that the selected *k* value is fixed. Additionally, there are no fine-grain controls for the cluster centers and the resulting clusters that are discovered as a result of these cluster centers.

2.4.2 ISODATA

The Iterative Self-Organizing Data Analysis Technique (ISODATA) algorithm is a refinement of the k-means concept with added parameters for additional customization of cluster count, size, inter-distance, and concentration.

The algorithm parameters are:

- Minimum *k* – The smallest number of *k* clusters the algorithm is allowed to identify.
- Maximum *k* – The largest number of *k* clusters the algorithm is allowed to identify.
- Minimum Cluster Size – The minimum size of any cluster identified by the algorithm.
- Merge Condition – A threshold for distance between any two clusters. Exceeding this threshold results in a merger between the two clusters.
- Split Condition – A threshold for cluster dispersion. Exceeding this threshold results in the cluster splitting into two clusters.

The algorithm steps:

1. Represent each element as a vector in space.
2. These elements define a subspace. Randomly place $(k_a + k_b)/2$ vectors in that subspace, where k_a and k_b represent the minimum and maximum number of allowable cluster centers, respectively.
3. Associate each element with the nearest cluster center.
4. Find the average value of all the elements associated with a particular cluster center. Shift that cluster center to the average value. Repeat for all cluster centers and their associated elements.
5. If a cluster has a sufficiently large standard deviation (split condition), place two cluster centers at the original point, plus and minus the standard deviation. Remove the original cluster center.
6. If two cluster centers are sufficiently close together (merge condition), place a single cluster center in between the two cluster centers. Remove the two original cluster centers.
7. If the number of elements associated with a cluster is less than the minimum cluster size, remove the cluster center.
8. If the number of clusters exceeds k_b or falls short of k_a, then remove or add a cluster center at random, respectively. The addition or removal may also be based on some criteria (e.g. removing the cluster center with the fewest associated elements).
9. If the average cluster quality is sufficiently small, then stop and report the clusters.

Otherwise, disassociate the elements from their cluster centers and go to step 3.

Determining how to set some of the parameters in ISODATA is difficult if nothing is known about the dataset ISODATA is being run on. In the case of our methodology, a small number of clusters is desired, resulting in less representative slices to be simulated. This affects the entire set of parameters of the ISODATA and is further discussed in section 5.1.1.

2.4.3 Cluster Dispersion Metrics

The evaluation technique used to generate the data for all of the experiments is based on the concept of measuring the dispersion between the elements in a single cluster and the dispersion between all of the cluster centers within the feature space. Within-cluster dispersion is the minimum mean square distance of the data elements in a cluster from the cluster center. As formalized in the following equation, the between-cluster dispersion metric is the minimum mean square distance of the cluster centers from the center of all cluster centers.

$n =$ number of elements in cluster i
$m =$ number of clusters
$p =$ set of elements in cluster $i, \{p_0, p_1, ..., p_n\}$
$c =$ set of cluster centers in cluster configuration, $\{c_0, c_1, ..., c_m\}$
$\bar{p} =$ center of the cluster elements
$\bar{c} =$ center of the cluster centers

$$\text{within-cluster dispersion for cluster } i = \frac{1}{n} * \sum_{j=0}^{n} \| p_j - \bar{p} \|$$

$$\text{between-cluster dispersion} = \frac{1}{m} * \sum_{j=0}^{m} \| c_i - \bar{c} \|$$

A cluster configuration is considered optimal when the minimum mean square distance of the elements from the cluster center is as small as possible, and when the minimum mean square distance between clusters is as large as possible. In other words, low dispersion between cluster elements indicates high cluster coherency, while high dispersion between clusters indicates high independence. The scalar value returned as the evaluation of a particular cluster configuration is the within-cluster dispersion measure over the between-cluster dispersion measure. To evaluate an entire cluster configuration, or the set of clusters determined by ISODATA, the evaluations for each cluster are averaged to return the overall configuration dispersion.

2.5 Feature Selection

Generally, not all of the features in a dataset are useful for classification. Some features may only add noise to the dataset by not providing any additional information that a rule or pattern may be induced from, resulting in the over-fitting of the classification process (Guyon & Elisseef, 2003). Feature selection attempts to eliminate irrelevant features before the data is applied to the classification system. In the context of this paper, feature selection is useful for determining which features from the architecture simulation trace data are most relevant.

Feature selection involves one of the two methods for evaluation of the data: filters or wrappers. Figure 4 describes the filter approach to feature selection.

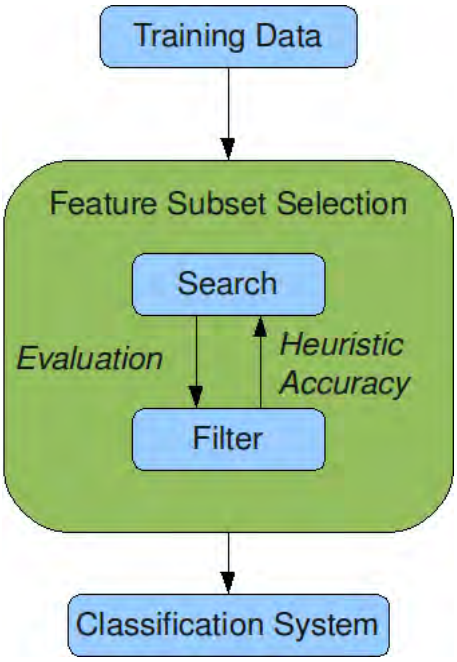

Figure 4: The Filter Approach to Feature Selection

In the filter method, the feature selection is performed purely as a preprocessing step, without any knowledge or application of the classification system.

Figure 5 describes the wrapper approach to feature selection.

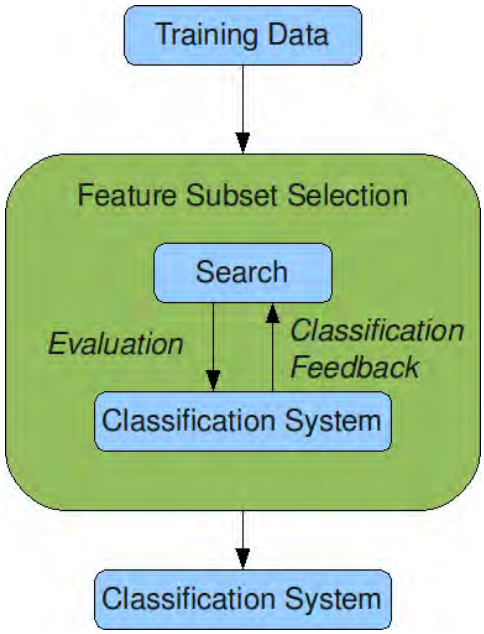

Figure 5: The Wrapper Approach to Feature Selection

The wrapper method uses the classification system as a "black box" mechanism from which it evaluates and selects particular features (John, Kohavi, & Pfleger, 1994). In effect, the feature selection method and the classification system become tightly integrated. Due to the configurability of the ISODATA algorithm as the classification system, the feature selection methods explored in this thesis uses the wrapper model for selection. The following sections describe these feature selection methods.

2.5.1 Hill Climbing

Hill climbing is a greedy search strategy. The algorithm is initialized with a single empty subset. In the first iteration, a single feature is tentatively added to the subset, and the subset is evaluated. This occurs for every existing feature. The feature that contributes to the best evaluation of the subset is added to the subset, which means there is now one

feature in the subset. In the subsequent iterations of this method, single features that have not been selected yet are tentatively added, and those features contributing to the best evaluation of the subset are added. The iteration ends when the subset reaches a particular evaluation criteria, such as dispersion, or a predetermined number. The evaluation is performed by the classification system (Guyon & Elisseeff, 2003), in this case ISODATA.

2.5.2 Genetic Algorithm

A genetic algorithm is initialized with a randomly generated population of elements which are evaluated using the classification system. A percentage of the population with the lowest evaluation is eliminated, while the remaining population is used for generating a new population using cross over and mutation operations. The crossover process occurs by merging two elements of the old population, forming a member for the new population. Mutation is applied to the new population to prevent premature convergence on some local optima. After cross over and mutation, the new population is evaluated so that the next iteration of the algorithm can continue. The algorithm ends when a termination condition is reached.

The algorithm parameters:

- Population Size
- Mutation Coefficient
- Survival Percentage

- Feature Count Range
- Termination Condition

The algorithm:

1. Randomly generate the initial population.
2. Evaluate all of the members of the population according to the classification system.
3. Choose a percentage of the highest-evaluated members to contribute to the next generation. These members are referred to as the elite group.
4. Perform a cross over on randomly selected members of the elite group to create the next generation.
5. Apply mutation based on the mutation coefficient to avoid premature convergence on some local optima.
6. If the highest evaluated member of the next generation does not meet the termination condition, repeat steps 2 through 5, otherwise the highest evaluated member is result returned from the algorithm.

Figure 6 describes a generalized iteration of the genetic algorithm.

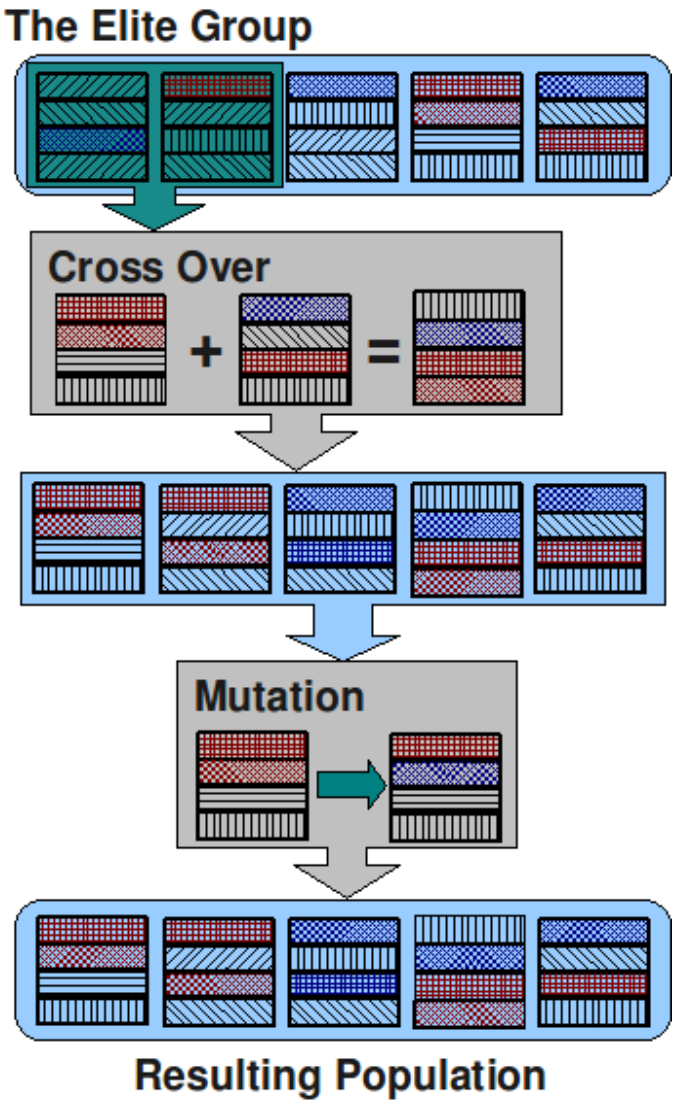

Figure 6: A Generalized Example of a Genetic Algorithm Iteration

The figure demonstrates one iteration of the genetic algorithm. The iteration begins with a population of elements, some of which belong to the elite group. The cross over operation is applied to the elite group, resulting in a new set of elements. Mutation is applied to those new elements to produce the population for the next iteration.

In the context of feature subset selection, the members of the population refer to feature

subsets. The population is initialized with a number of randomly selected feature subsets from the set of features. The subsets are evaluated according to their clustering quality. The remaining feature subsets are used to create the next generation using mutation and cross over operations. Factorial design is applied in calibrating the genetic algorithm in 5.1.2.

2.6 Known Power-Affecting Features

The features are those that are known to affect power are cache and register-related:

- Cache Hits
- Cache Misses
- Cache Eviction
- Cache Reads
- Cache Writes
- Cache Flushes
- Cache Touches
- Cache Allocations
- Cache Sets
- Cache Ways

These features are known to affect power consumption, and should have the greatest

influence on power estimation accuracy. By intentionally including these features after feature selection has applied, the power estimation accuracy should increase. In other words, feature selection is an information-theoretic method, which can only make use of the data provided to the algorithm. Including known power-affecting features is an application of domain-knowledge to the clustering process.

3. RELEVANT WORK

Brandolese et al. (2003) attempt to generalize instruction execution across microprocessor models which have only been partially characterized with respect to power consumption. By abstracting the architecture level further into functionalities involved in instruction execution, a model is created based on knowledge of functionalities when applied to particular microprocessor models. While the methodology described in this paper does make use of instruction simulation, the purpose of doing so is not directly related to power estimation. Instead of using instructions for power estimation, our methodology employs instructions as a pre-characterization step.

Xi, Huang, and Zhong (2005) model sets of instructions as macro-operations based on a high-level language, like C or C++. By representing an entire benchmark in macro-operations, power consumption is estimated for each macro-operation, and thus for the entire benchmark, without the need for execution of the entire benchmark. Using SystemC as the simulation environment, average error of 6.7% was maintained while gaining an average 241.84X speedup.

An even higher level approach to energy macro-modeling is to group instructions based

on functions or subroutines within the benchmark application. Tan et al. (2001) present two approaches to function-level macro-modeling: one which uses the function parameters as macro-modeling parameters for estimating power consumption, and the other which profiles the function's execution, and uses the internal statistics generated from profiling as macro-modeling parameters. Using these two approaches, power estimation has averaged within 95%, with a speedup of over five orders of magnitude over instruction-level and other architectural-level power estimation techniques.

While these methodologies are similar to our methodology, they are different in that they use a linear fitting method that is based on a set of known inputs and outputs from the macro-operation or function to define a model for the benchmark. Our methodology statically produces dynamic execution slices by dividing the architecture simulation trace data into slices of fixed size, and instead of statistically characterizing each slice, slices are clustered together and a number of representative slices are chosen to represent the entire stream of execution.

Slice-based characterization relies on dividing the benchmark's instruction stream, or corresponding streams such as the ASTD, from the simulator into a set of slices. One particular approach to slice-based characterization is using the subset of slices as a sample of the entire instruction stream. The SMARTS approach (Wunderlich et al., 2003), achieves an average error of less than 1% in estimating clock cycles and energy per instruction by simulating only 50M out of 2 to 5B instructions in a set of benchmarks.

SMARTS systematically selects samples at certain intervals and simulates the micro-architectural aspects of the model, while only simulating the architectural aspects of the model for non-sampled data in order to maintain an approximate continuation of the micro-architectural state. Our methodology uses cluster analysis and feature selection to select a sample of slices from the entire stream of ASTD, rather than selecting samples at certain intervals.

Joshi, Luo, and John (2007) describe a system, DynaSim, that is specifically targeted at OnLine Transaction Processing (OLTP), which results in simulation time improvements over the SMARTS approach. DynaSim takes advantage of the continuous and statistically phase-less nature of OLTP simulations. This enables DynaSim to simulate continuously until some desired confidence interval is met, in some cases without having to resort to intermediate architectural simulation. Our methodology does not use continuous simulation in order to meet a desired confidence interval, but instead uses data from an initial architectural simulation to determine which slices should be executed in a follow-up micro-architectural simulation.

In addition to sampling instruction stream slices, a single slice can also be determined as representative, or prototypical, of a number of other slices. Sherwood, Perelman, Hamerly, and Calder (2002) approach the task of determining representative slices by using random linear projection to reduce data dimensionality and cluster analysis to obtain representative slices. Each slice contains counts of execution for basic blocks

within the benchmark code (a section of code with one exit and one entry point). The methodology described in this thesis is very similar to the methodology described by Sherwood et al. (2002). However, they did not make use of feature selection. Additionally, they use block executions counts per slice. In our research, architectural events per slice are counted.

The methodology used by Todi (2001) is also similar to the methodology of this thesis paper in that architectural events are sampled at regular intervals throughout execution and then clustered using k-means. To reduce data dimensionality, however, Principle Component Analysis is used instead of feature selection as is done in our thesis. Also, only a single slice size of 1M instructions is used whereas multiple slice sizes, ranging from 1K to 10K, are used in our research.

Luo, Joshi, Phansalkar, John, and Gosh (2008) compare a number of clustering methods as well as introducing a new feature for locating phases within a benchmark's instruction stream. The benchmark is divided into several single-exit single-entry code slices. The paper demonstrated the effectiveness of using the CLARANS (Clustering Large Applications based on RAN-domized Search) algorithm over k-means. Additionally, it introduced a feature, as alternative to using basic blocks, to measure data locality within slices as an indicator of program phases. Our method differs, as it does with Sherwood's (2002) use of basic block vectors, in that slices are measured based on architectural events.

Phansalkar and John (2006) attempt to use program similarity to predict the cache-miss rate of a benchmark with unknown performance characteristics by using a group of benchmarks with known performance characteristics. They propose two methods: 1) assigning weight to each of the benchmark programs with known performance characteristics and applying the weighted mean to the new benchmark to predict performance; 2) clustering the similar benchmarks with known performance characteristics, and then use the representative benchmark from the cluster containing the new benchmark.

Another benefit of determining program similarity is the ability to select a subset of benchmarks out of an entire benchmark suite for simulation, rather than simulating the entire benchmark suite. The goal is to select a subset of benchmarks that are distinct from one another and are also representative of the entire benchmark suite. Joshi, Phansalkar, Eeckhout, and John (2006) make use of microarchitecture-independent characteristics such as instruction mix, control flow behavior, instruction level parallelism, data locality, and instruction locality to characterize each benchmark in order to determine similarity. The methodology described in this thesis does not consider similarities between two benchmarks in order to characterize a microprocessor's workload. However, further research could apply our use of cluster analysis and feature selection to identify similarities between two benchmarks using ASTD.

4. METHODOLOGY

The methodology for reducing the amount of time required to estimate power consumption is a sequential process involving the benchmark applications, simulator software capable of performing ISA-level simulations and power estimations given a program counter range, a means to extract data from the ISA-level simulations, and a means to perform feature selection and cluster analysis on the extracted data. Figure 7 describes the process.

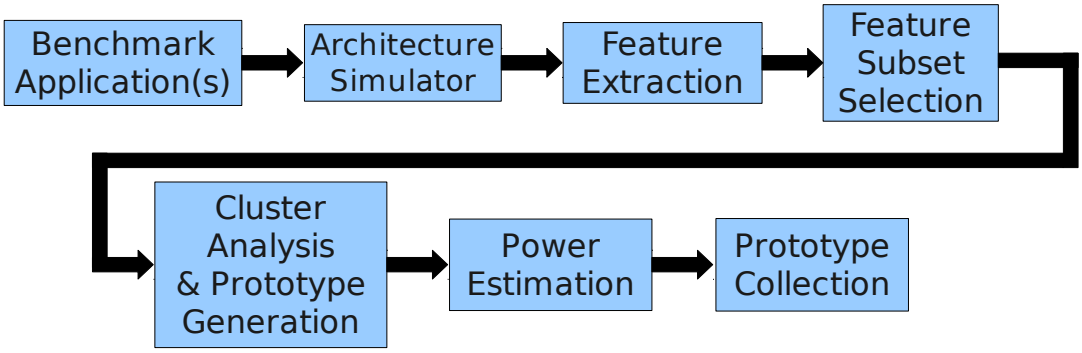

Figure 7: The Components of the Methodology

This figure describes the sequence of power estimation using the slice-based methodology. The benchmark applications are first run through the architecture simulator to generate the ASTD. Following the architecture simulation, the features are extracted from the ASTD. Feature selection is applied, resulting in an optimal subset of features.

Cluster analysis is performed to determine which ASTD slices most accurately represent the benchmark test as representative slices. The corresponding instruction stream slices are run using the system level simulator, which returns power estimation for each slice. The original benchmark test is then expressed in terms of the representative instruction stream slices, reducing data throughput for system level simulation.

4.1 Benchmark Applications

The choice of benchmark tests are ultimately up to the designers of the microprocessor. The choice of benchmark tests does not affect the our methodology in the same manner that it would affect workload characterization, as it is not dependent on the type or quality of the benchmark itself. The microprocessor designer should only be aware of the instructions used in the benchmark application, as the model may not be designed for or capable of handling the specific aspects of the benchmark application. The execution of the benchmark, especially if created by a third party, may result in the unintended behavior of the model, skewing the performance results.

For the purposes of this paper, the following benchmark applications are used:

1. *Basic Mathematical Operations* – Runs basic mathematical operations on random sets of data.
2. *Fast Fourier Transform Algorithm* – Runs the FFT algorithm on random sets of data.
3. *Dijkstra's Shortest Path Algorithm* – Runs Dijkstra's SP algorithm on random sets

of data.

4. *Q-Sort Algorithm* – Runs the Q-Sort algorithm on random sets of data.

4.2 Architecture Simulator

The architecture simulator being used for this paper is Freescale's Architecture Description Language (ADL) simulator. The ADL simulator takes an Executable and Linking Format (ELF) binary and simulates its execution on a particular ADL model. The output generated from the simulator is a trace through the model as the ELF machine code is executed. This ASTD describes (Kahne, 2006):

1. The start of an instruction sequence
2. Information about the instruction itself
3. Resource update events
4. Breakpoint occurrences
5. Cache access
6. Memory data access
7. The occurrence of an exception
8. Register modification
9. MMU/TLB access
10. Watchpoint occurrence

4.3 Feature Extraction

A simple Python script has been written for the purposes of extracting the ASTD output from the ADL Simulator. The script extracts information related to the execution of instructions, memory and cache operations, and register modification. The script also makes use of several subcategories for the various types of instructions (load, store, mathematical, logical, etc.), each of which represent a single feature. The output from the script is a file of comma-separated values, where each row represents a slice of the ASTD, and each column represents a feature. The feature selection and cluster analysis processes are then read in each row as a data element, where columns may be discarded as the feature selection process determines.

4.4 Feature Selection

A library of cluster analysis and feature selection methods have been written specifically for this thesis in C++.

1. Feature Selection
    1. Exhaustive Search
    2. Hill Climbing Algorithm
    3. Genetic Algorithm
2. Cluster Analysis
    1. ISODATA

The library reads the input from the feature extraction script and returns the best feature subset, which the clustering algorithms use to return the high cluster quality. These clusters form the representative slices, representative of the entire ASTD file.

4.5 Cluster Analysis

Each cluster identified by the ISODATA algorithm is associated with a representative ASTD slice. Since each data element being clustered corresponds to a ASTD slice from the simulation, the data element closest to the center of the cluster is regarded as the representative slice. This representative slice is used in place of the other slices that correspond to the data elements belonging to that particular cluster. The program counter ranges of the representative ASTD slices are used to extract slices of the instruction stream, which are run on the system level simulator. The system level simulator returns power estimation for each slice, thus indicating how accurate the representative slices are at representing the entire benchmark test in terms of power consumption.

4.6 Power Estimation

Freescale maintains an open source software suite for modeling a processor's architecture using an architecture description language (Freescale Semiconductor, 2010). Models designed using ADL can be simulated at the architecture level. Freescale has also provided a power estimation script which uses a heuristic equation for power estimation based on a power estimation model designed by Sunwoo, Al-Sukhni, and Holt (2007). The simulation software is able to apply the power estimation script during the simulation process to return the estimated power consumption for a given instruction stream. Instead

of using system level simulation software to estimate power consumption, the architecture level simulation software is used in conjunction with power estimation script. This enables a greater range of experimentation due to the speed of architecture level simulation software. However, the power estimation accuracy demonstrated within the experiments is not necessarily as accurate as it would be if system level simulation software were used.

5. EXPERIMENTS

The following experiments are designed around the three hypotheses of this thesis:

1. The ISODATA algorithm may be effective in clustering ASTD slices, resulting in accurate power estimation while reducing data throughput in system level simulation.
2. Feature selection may result in increased power estimation accuracy by selecting an optimal subset of features for the ISODATA algorithm to use in clustering the ASTD slices.
3. Selecting the known power-affecting features may result in increased power estimation accuracy.

The experiments fall into three categories:

1. Calibrating and evaluating the cluster analysis and feature selection algorithms. The following is a list of experiments for this category:
    1. ISODATA Calibration
    2. Feature Selection Calibration
    3. Feature Selection Algorithm Evaluations

1. Exhaustive Search
2. Genetic Algorithm
3. Hill Climbing
4. Small Feature Subset Sizes
5. Exhaustive Search vs. Genetic Algorithm

2. Demonstrating the affecting factors of power estimation accuracy. The following is a list of experiments for this category:
    1. ASTD Slice Size
    2. ISODATA Slice Selection
    3. Cluster Quality
3. Evaluating the effectiveness of the methodology based on the three hypotheses. The following is a list of experiments for this category:
    1. ASTD Compression
    2. Genetic Algorithm Feature Selection
    3. Known Power-Affecting Features

5.1 Algorithm Calibration and Evaluation

The following experiments describe the calibration of the ISODATA algorithm and the evaluation of feature selection methods.

5.1.1 ISODATA Calibration

In an effort to maximize the effectiveness of the ISODATA in other experiments, the algorithm is calibrated using a set of training data. A one-way ANalysis Of Variance (ANOVA) test determines the statistical significance of the relationships between the independent and dependent variables involved in the ISODATA algorithm. One-way ANOVA produces an F-test statistic, which is the ratio of the variance between the means of the data to the variance within the data samples. The F value determines the significance of the independent variables with respect to the dependent variables.

In ISODATA, the independent variables, or parameters, are:

1. Maximum *k* (maximal number of clusters possible)
2. Minimum Cluster Size
3. Merge Threshold
4. Split Threshold

The dependent variables, or results, are:

1. Evaluation
2. Distortion Delta
3. Elimination Count
4. Merge Count
5. Split Count

There are three sets of experimental data by which to characterize the ISODATA

algorithm. Two are comprised of randomly generated points around eight cluster centers, while the third is comprised of uniformly generated points around eight cluster centers. Factorial design is applied to determine the ISODATA parameters or independent variables to use. The design uses every combination of the following independent variable values:

1. Maximum k [10, 100, 1000]
2. Minimum Cluster Size [10, 100, 1000]
3. Merge Threshold [0.5, 1.0, 2.0]
4. Split Threshold [0.5, 1.0, 2.0]

The merge and split thresholds are actually coefficients applied to a default threshold that is calculated under the assumption that the cluster centers are uniformly placed.

From the factorial design, ISODATA is run 81 times on each data set, or 3^4; every combination of the independent variable values. From each ISODATA run, the following results are extracted as the dependent variables for the statistical analysis: evaluation, distortion delta, elimination count, and merge count. Using one-way ANOVA the significance of the independent variables with respect to the dependent variables (namely the Evaluation), is established.

Table 1 represents the results from running ANOVA over all four of the calibration datasets.

Table 1: ANOVA Table of Combined ISODATA Experiments

Independent Variables	**Degrees of Freedom**	**Sum of Squares**	**Mean Squares**	**F Value**	**Prob. > F**	**Significance**
(1) Max. K	1	154.997	154.997	423.033	0.000	99.90%
(2) Min. Cluster Size	1	39.607	39.607	108.112	0.000	99.90%
(3) Merge Threshold	1	3.013	3.013	8.225	0.004	95.00%
(4) Split Threshold	1	0.065	0.065	0.117	0.674	< 90%
1, 2	1	8.045	8.045	21.961	0.000	99.90%
1, 3	1	1.206	1.206	3.291	0.071	< 90%
1, 4	1	0.037	0.037	0.101	0.751	< 90%
2, 3	1	0.030	0.030	0.081	0.776	< 90%
2, 4	1	0.004	0.004	0.012	0.914	< 90%
3, 4	1	0.137	0.137	0.375	0.541	< 90%
1, 2, 3	1	0.000	0.000	0.001	0.981	< 90%
1, 2, 4	1	0.010	0.010	0.027	0.870	< 90%
1, 3, 4	1	0.008	0.008	0.023	0.879	< 90%
2, 3, 4	1	0.019	0.019	0.052	0.820	< 90%
1, 2, 3, 4	1	0.006	0.006	0.018	0.894	< 90%

The two independent variables that stand out as being the most significant are the maximum *k* value and the minimum cluster size.

Table 2 represents the results from running ANOVA on the average of all four of the calibration datasets.

Table 2: ANOVA Table of Averaged ISODATA Experiments

Independent Variables	**Degrees of Freedom**	**Sum of Squares**	**Mean Squares**	**F Value**	**Prob. > F**	**Significance**
(1) Max. K	1	44.722	44.722	142.935	0.000	99.90%
(2) Min. Cluster Size	1	11.149	11.149	35.632	0.000	99.90%
(3) Merge Threshold	1	0.951	0.951	3.041	0.086	< 90%
(4) Split Threshold	1	0.019	0.019	0.061	0.806	< 90%
1, 2	1	2.353	2.353	7.519	0.008	99.00%
1, 3	1	0.398	0.398	1.273	0.263	< 90%
1, 4	1	0.007	0.007	0.022	0.882	< 90%
2, 3	1	0.005	0.005	0.017	0.897	< 90%
2, 4	1	0.002	0.002	0.005	0.945	< 90%
3, 4	1	0.036	0.036	0.116	0.735	< 90%
1, 2, 3	1	0.001	0.001	0.002	0.968	< 90%
1, 2, 4	1	0.006	0.006	0.020	0.888	< 90%
1, 3, 4	1	0.003	0.003	0.011	0.917	< 90%
2, 3, 4	1	0.007	0.007	0.022	0.884	< 90%
1, 2, 3, 4	1	0.004	0.004	0.011	0.915	< 90%

The two independent variables that stand out as being the most significant are the maximum *k* value and the minimum cluster size. These two variables have the most bearing on the evaluation of the clusters generated by the ISODATA algorithm. The maximum *k* constrains the number clusters that the ISODATA algorithm can generate.

5.1.2 Feature Selection Calibration

Only the mutation probability and survival percentage parameters of the genetic algorithm require calibration using a set of training data. These parameters are calibrated

using ANOVA. Table 3 represents the results from running ANOVA on all four of the feature selection calibration datasets averaged:

Table 3: ANOVA Table of Averaged Genetic Algorithm Feature Selection Experiments

Independent Variables	Degrees of Freedom	Sum of Squares	Mean Squares	F Value	Prob. > F	Significance
(1) Mutation Probability	1	0.001	0.001	1.1365	0.335	< 90%
(2) Survival Probability	1	0.001	0.001	1.3374	0.300	< 90%
1,2	1	0.000	0.000	0.335	0.588	< 90%

According to the calibration data, neither the mutation or survival probability have any significant effect in determining the local optima returned from the search.

5.1.3 Feature Selection Algorithm Evaluation

These experiments demonstrate the effectiveness of the feature selection algorithms on a set of calibration data. While some feature selection algorithms may identify optimal feature subsets, they may be too slow or unpredictable.

The experiments are run using the following parameters (explained in “2.4.2 ISODATA“):

1. ISODATA and Genetic Algorithm parameters:

1. *ISODATA k-Max Values*: 10, 20, 50, 100, 200, 500, 1000
2. *ISODATA Termination Condition*: 0.3
3. *ISODATA Max Iterations*: 50
4. *ISODATA Merge Condition*: 1.0
5. *ISODATA Split Condition*: 1.0
6. *Feature Selection Termination Condition*: 0.3
7. *Genetic Algorithm Max Iterations*: 50

5.1.3.1 The Exhaustive Search

A comparison of the feature selection methods is put into perspective by comparing them to an exhaustive search of all the feature subsets for a given subset size. Unfortunately, the feasibility of an exhaustive search is not achievable for all subset sizes, given the combinatorial effect. Figure 8 describes an exhaustive search through every possible combination of feature subsets for subset sizes 1, 2, 3, 4, 5, 18, 19, 20, 21, and 22. Additional graph points have been added to indicate the predicted values for the best and worst evaluations returned from the exhaustive search.

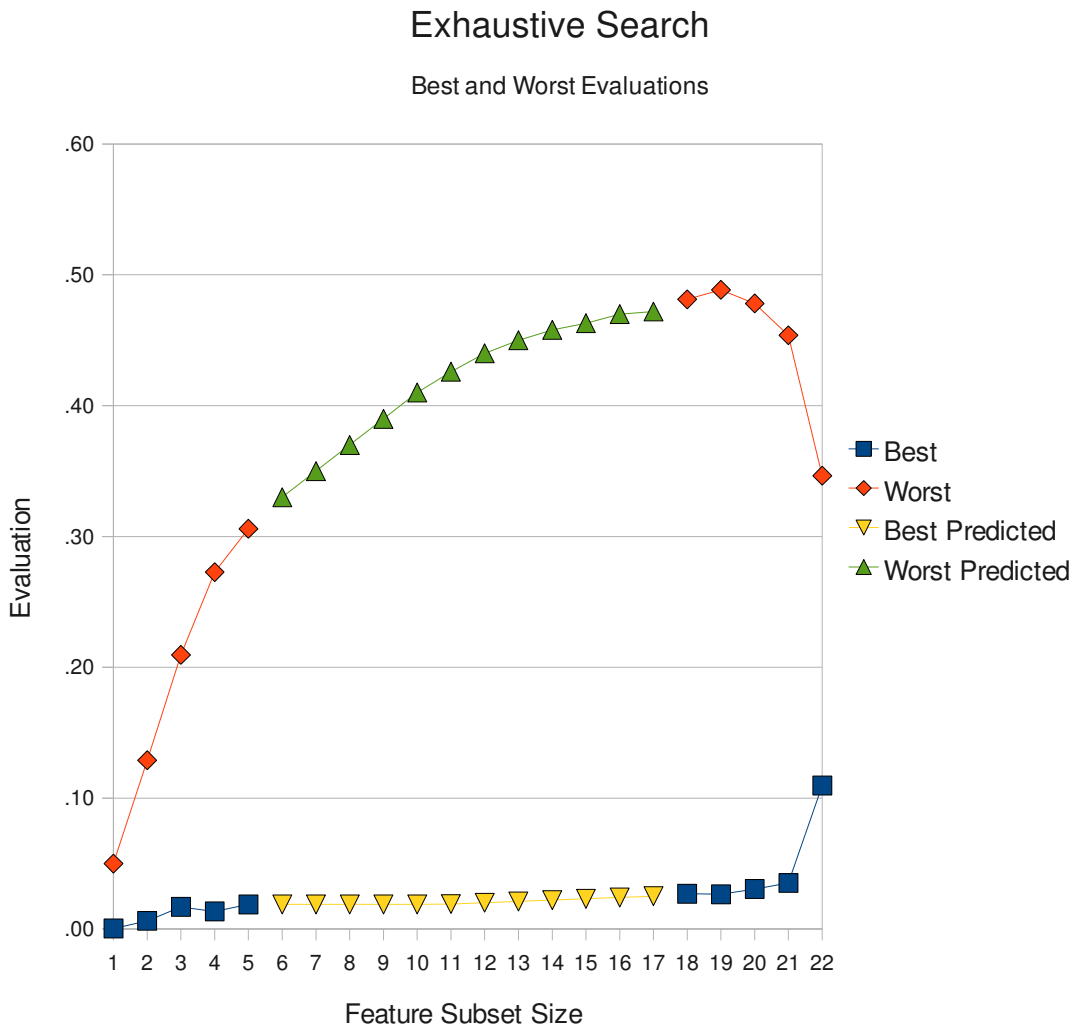

Figure 8: Exhaustive Search of Feature Subsets

The figure shows that the gap between the best and worst evaluations for the exhaustive search is very distinct. The gap also grows very rapidly in the first few feature subset sizes, while not necessarily shrinking symmetrically at the tail end of the feature subset sizes.

5.1.3.2 Genetic Algorithm

Figure 9 represents the genetic algorithm being run for all possible feature subset sizes on

a single benchmark.

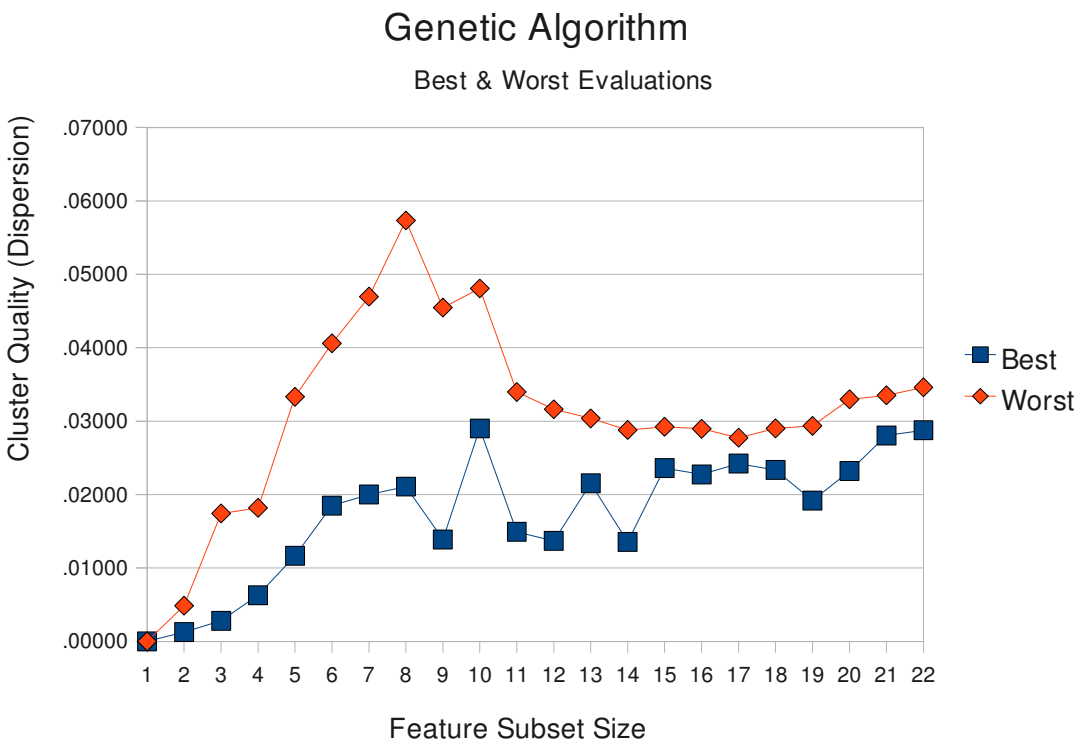

Figure 9: Genetic Algorithm Identifying Optimal Feature Subsets

A noticeable effect within this figure is the stabilization of the worst evaluations at subset size 11, and also the stabilization of the best evaluations at subset size 5. The difference between the best and worst cluster quality for each feature subset size is considerably smaller than demonstrated with the exhaustive search.

5.1.3.3 Hill Climbing

Figure 10 describes the cluster quality returned from the hill climbing algorithm on all possible feature subset sizes.

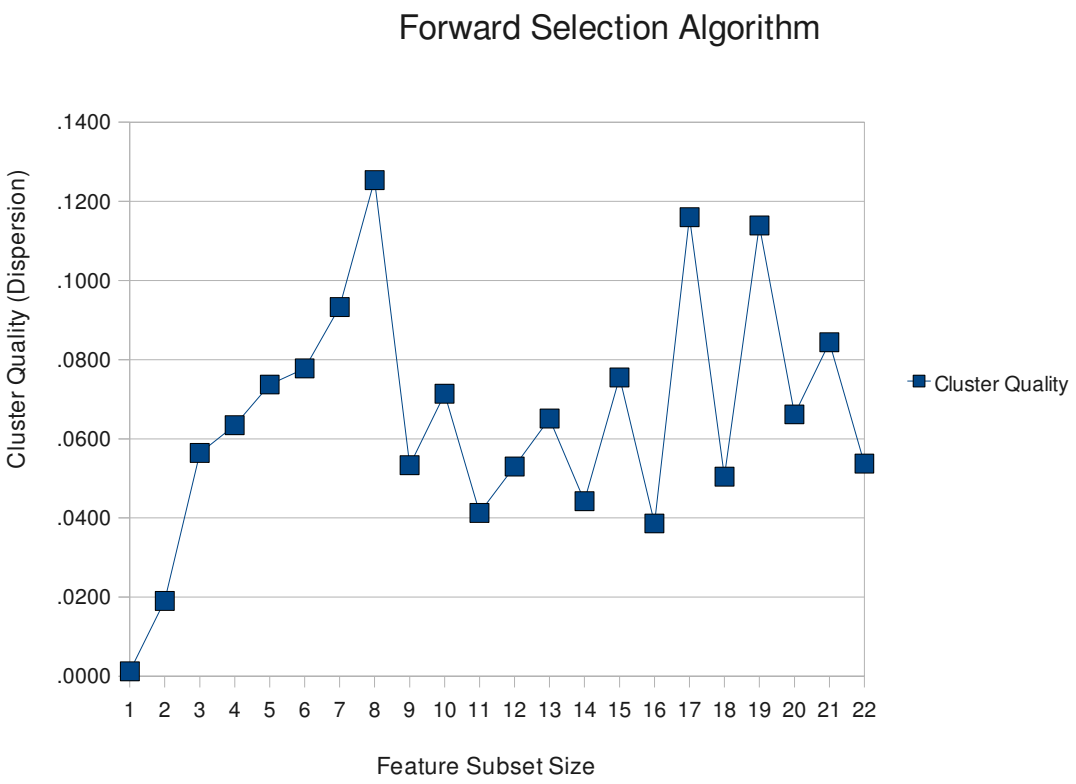

Figure 10: Hill Climbing Search of Feature Subsets

In the case of this figure, only one iteration of the algorithm is performed for each feature subset size. The cluster quality rapidly increases starting at the feature subset size of 1 to the feature subset size of 8.

5.1.3.4 Small Feature Subset Sizes

This experiment is designed to indicate the lack of correlation between cluster quality and power estimation accuracy for very low feature subset sizes.

The experiments are run using the following parameters:

1. ISODATA and Genetic Algorithm parameters:
    1. *ISODATA k-Max Values* (minimum *k* is always 1): 20
    2. *ISODATA Termination Condition*: 0.3
    3. *ISODATA Max Iterations*: 50
    4. *ISODATA Merge Condition*: 1.0
    5. *ISODATA Split Condition*: 1.0
    6. *Genetic Algorithm Termination Condition*: 0.3
    7. *Genetic Algorithm Max Iterations: 50*
    8. *Percent of Features Selected*: 75%
2. Experiment parameters:
    1. *Execution Trace Slice Sizes*: 1000, 2000, 5000, 10000
    2. *Benchmarks*: Basic Math
    3. *Feature Selection*: Genetic Algorithm
    4. *Known Power-Affecting Features*: With, Without

Figures 11 through 14 demonstrates the cluster qualities achieved for each feature subset size.

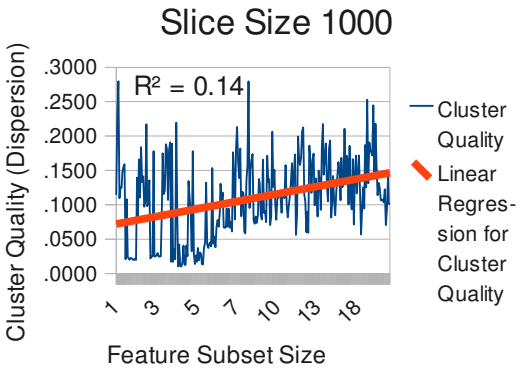

Figure 11: Subset Size and Cluster Quality, Slice Size 1000

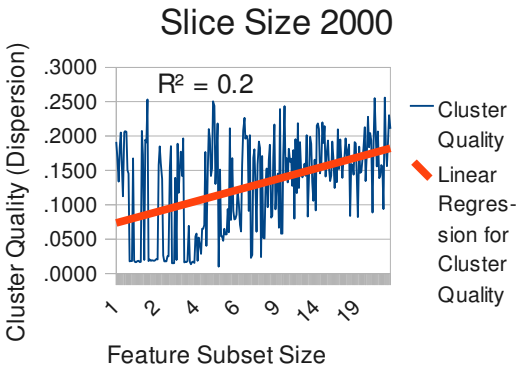

Figure 12: Subset Size and Cluster Quality, Slice Size 2000

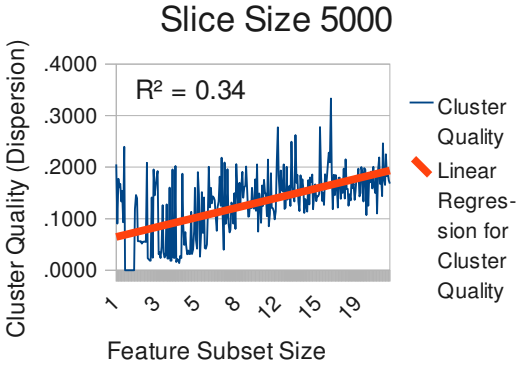

Figure 13: Subset Size and Cluster Quality, Slice Size 5000

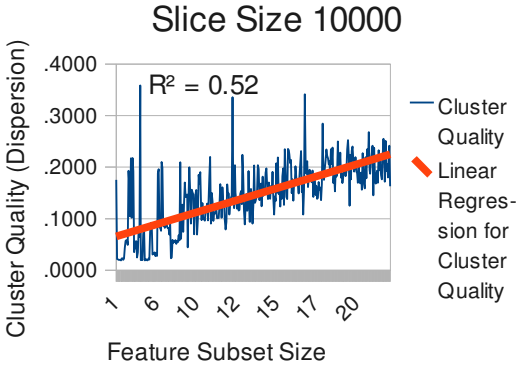

Figure 14: Subset Size and Cluster Quality, Slice Size 10000

The cluster quality in the figures decreases as the feature subset size increases. Another interesting trend is the small decrease in cluster quality variance as the feature subset size increases. Cluster quality variance also decreases as the slice size increases.

Figures 15 through 18 demonstrate power estimation resulting from the feature selection and subsequent cluster analysis on the data set demonstrated in figures 11 through 14.

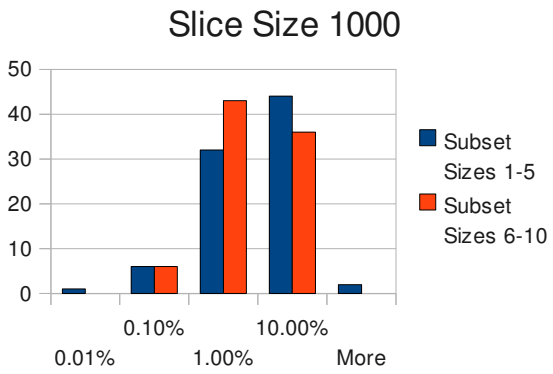

Figure 15: Subset Size and Power Estimation Accuracy, Slice Size 1000

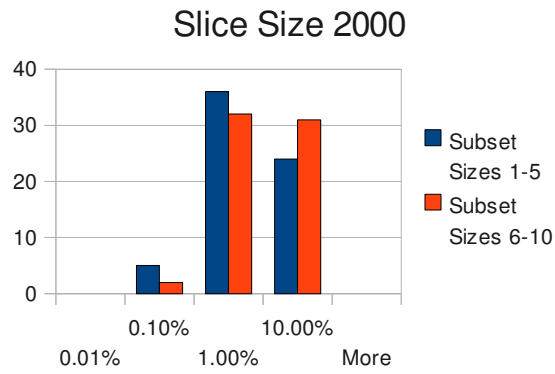

Figure 16: Subset Size and Power Estimation Accuracy, Slice Size 2000

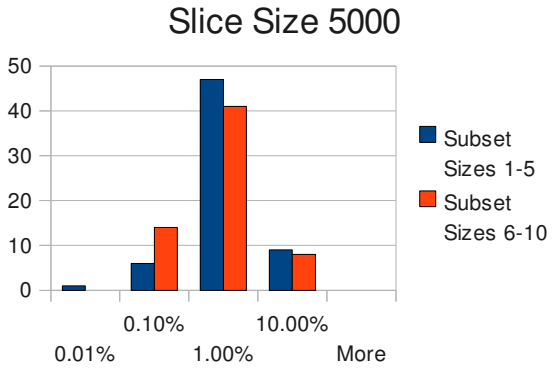

Figure 17: Subset Size and Power Estimation Accuracy, Slice Size 5000

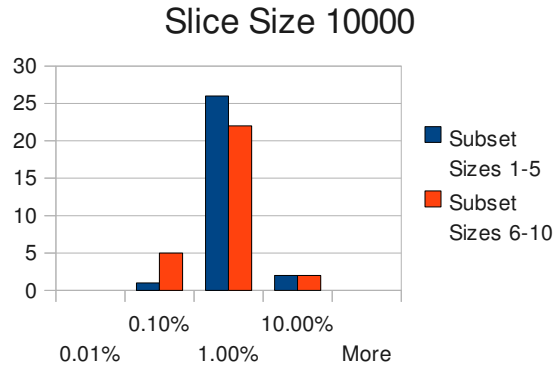

Figure 18: Subset Size and Power Estimation Accuracy, Slice Size 10000

Figures 11 through 14 show that cluster quality decreases as the feature subset size increases. However, figures 15 through 18 demonstrate that, in some cases, larger feature subset sizes result in greater power estimation accuracy. This demonstrates a lack of correlation between cluster quality and power estimation accuracy for small feature subset sizes.

5.1.3.5 Exhaustive Search Versus Genetic Algorithm Search

The genetic algorithm has been selected over the hill climbing method due to its greater accuracy, and over the exhaustive method due to its greater speed. In order to gauge the

performance of the genetic algorithm feature selection method, its results are compared to the results of an exhaustive search of the entire feature set.

The experiments are run using the following parameters:

1. ISODATA and Genetic Algorithm parameters:
    1. *ISODATA k-Max Values*: 10
    2. *ISODATA Termination Condition*: 0.3
    3. *ISODATA Max Iterations*: 50
    4. *ISODATA Merge Condition*: 1.0
    5. *ISODATA Split Condition*: 1.0
    6. *Genetic Algorithm Termination Condition*: 0.3
    7. *Genetic Algorithm Max Iterations: 50*
    8. *Percent of Features Selected*: 75%
2. Experiment parameters:
    1. *Execution Trace Slice Sizes*: 1000, 2000, 5000, 10000
    2. *Benchmarks*: Basic Math, FFT, Dijkstra, Q-Sort
    3. *Feature Selection*: Exhaustive Search and Genetic Algorithm
    4. *Known Power-Affecting Features*: With, Without

Figure 19 compares the cluster qualities returned from both exhaustive search and genetic algorithm feature selection for the Basic Math benchmark.

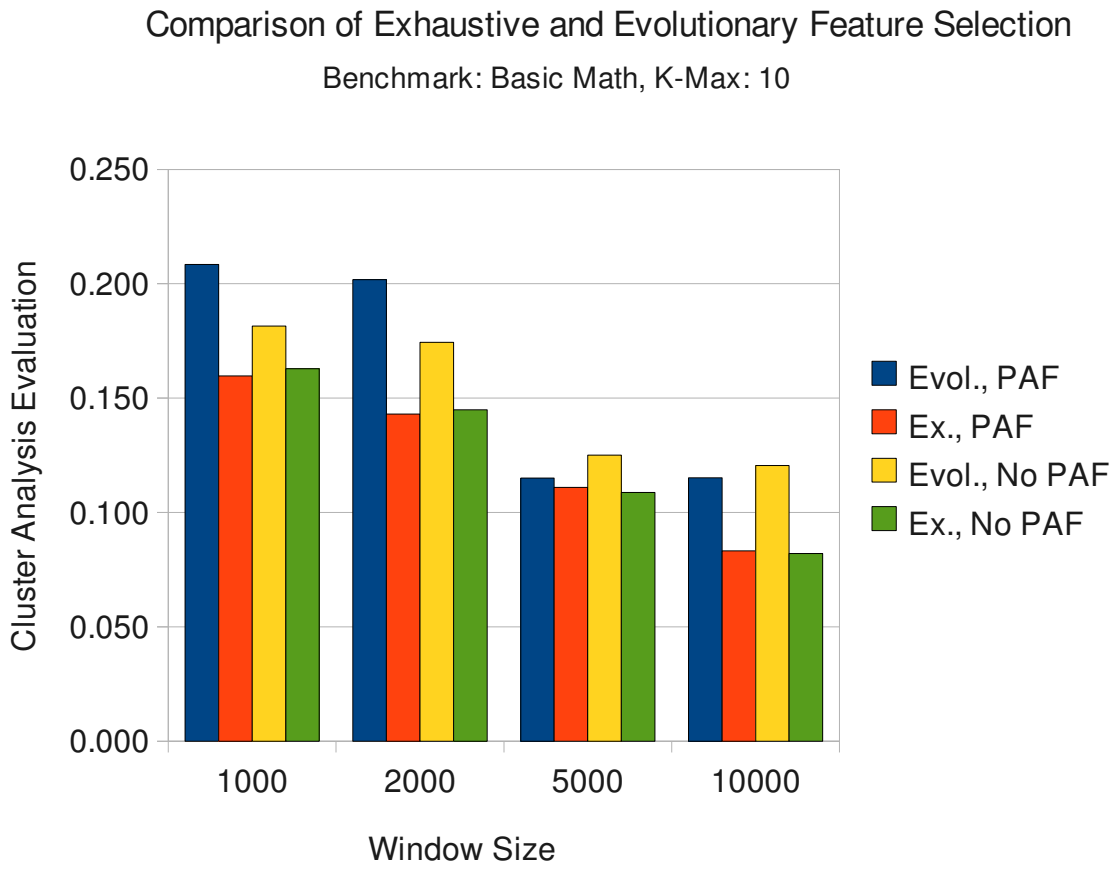

Figure 19: Exhaustive vs. GA Search, Basic Math Benchmark, k-Max: 10

In this figure, the evaluation differences between the exhaustive search and genetic algorithm search seem to be the most minor for the slice size of 5K.

Figure 20 also compares the cluster qualities returned from both exhaustive search and genetic algorithm feature selection, but for the Fast Fourier Transform benchmark.

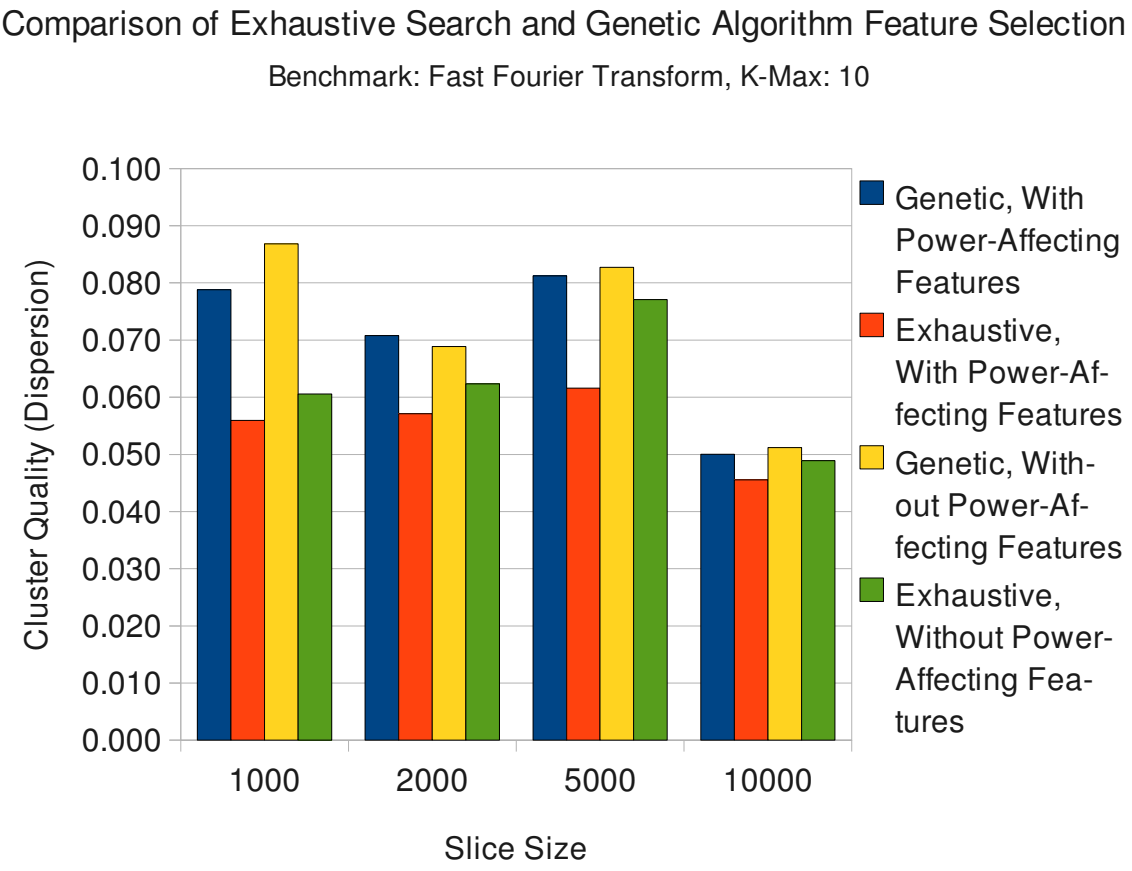

Figure 20: Exhaustive vs. GA Search, Fast Fourier Transform Benchmark, k-Max: 10

Unlike figure 19, the evaluation differences between the exhaustive search and genetic algorithm search, in this figure, seem to be the most minor for the slice size of 10K. Despite different slice sizes having the most minor evaluation differences, all of the other benchmarks demonstrated the exhaustive search returning higher cluster quality than the genetic algorithm. The only exception to this trend is the Q-Sort algorithm at slice size 2000 when using power-affecting features.

5.2 Affecting Factors of Power Estimation Accuracy

The following experiments describe the various factors that may have a significant effect

on power estimation accuracy.

5.2.1 ASTD Slice Size

Simulation of an instruction stream slice is preceded by a cache warming phase in which data is set in cache memory for processing before execution begins. The cache warming affects the overall energy estimation per simulation, which means that the overall cache warming effect will be much greater for smaller slices. These experiments demonstrate the cache warming effect being mitigated by the larger slice sizes.

Figure 21 demonstrates the effect of the ASTD slice size on power estimation accuracy for the Basic Math benchmark.

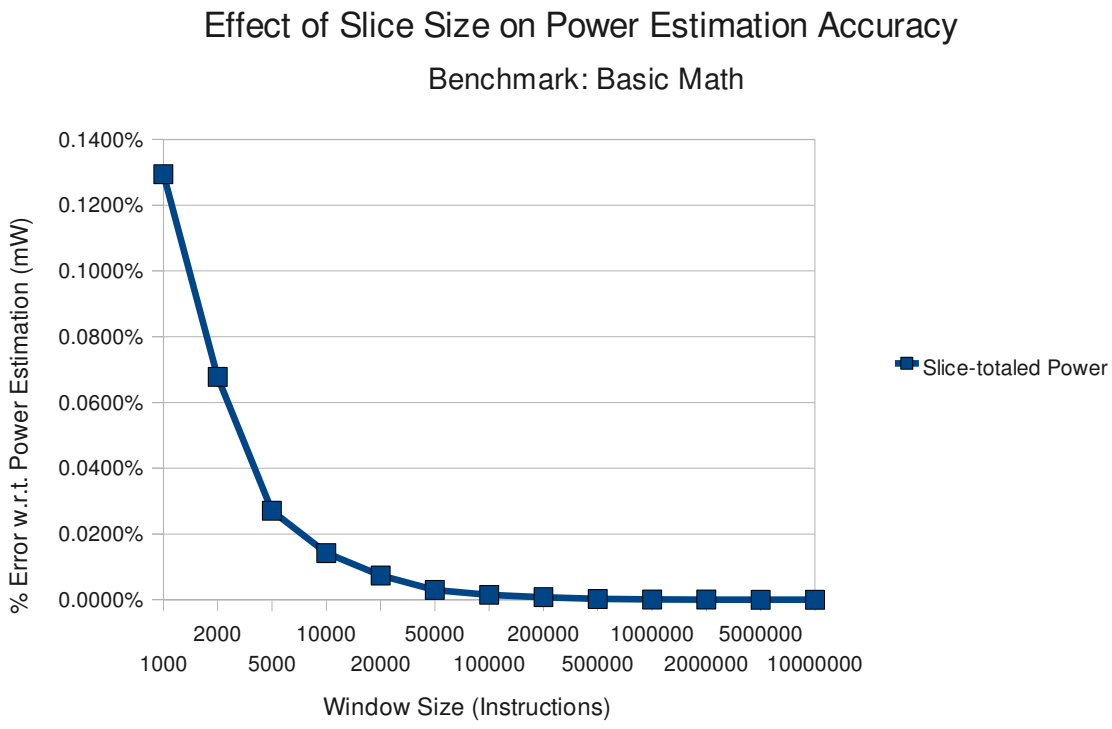

Figure 21: Effect of Slice Size: Basic Math Benchmark

In this figure, the percent error becomes less than 0.1% at a slice size of 2K, and less than 0.01% at a slice size of 20K. Also, the power estimation accuracy is very high.

Figure 22, as with figure 21, demonstrates the effect of the ASTD slice size on power estimation accuracy. This figure applies to the Q-Sort benchmark data, however.

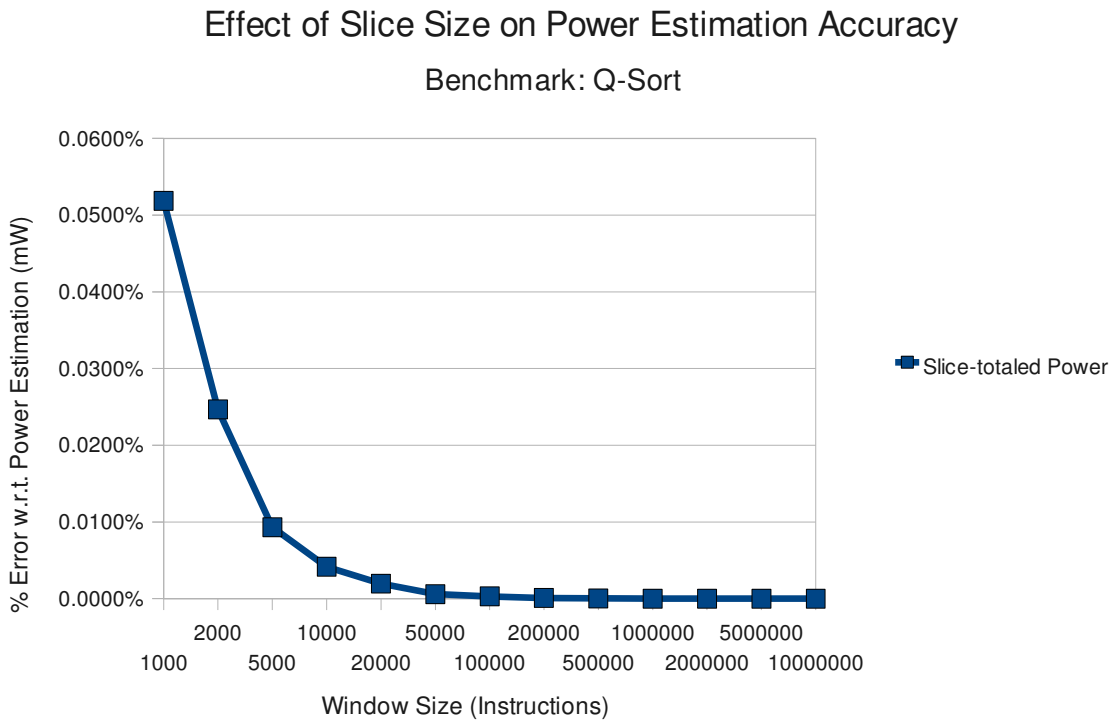

Figure 22: Effect of Slice Size: Q-Sort Benchmark

Overall, the cache warm-up effect is much smaller for the Q-Sort benchmark than it is for the Basic Math benchmark, given the low power estimation percent error. Experimentation on the other two benchmarks, Dijkstra and Fast Fourier Transform, also returned very a low power estimation percent error, at least an order of magnitude smaller than the average errors returned from experiments related to the three hypotheses.

5.2.2 ISODATA Slice Selection

This experiment determines the effectiveness of the ISODATA clustering when compared to randomly selected slices.

The experiments are run using the following parameters:

1. ISODATA and Genetic Algorithm parameters:
    1. *ISODATA k-Max Values*: 10, 20, 50, 100, 200, 500, 1000
    2. *ISODATA Termination Condition*: 0.3
    3. *ISODATA Max Iterations*: 50
    4. *ISODATA Merge Condition*: 1.0
    5. *ISODATA Split Condition*: 1.0
    6. *Genetic Algorithm Termination Condition*: 0.3
    7. *Genetic Algorithm Max Iterations: 50*
    8. *Percent of Features Selected*: 75%
2. Experiment parameters:
    1. *Execution Trace Slice Sizes*: 1000, 2000, 5000, 10000
    2. *Benchmarks*: Basic Math, FFT, Dijkstra, Q-Sort
    3. *Feature Selection*: No Feature Selection
    4. *Known Power-Affecting Features*: With, Without

Figure 23 compares random slice selection and ISODATA slice selection for the Basic Math benchmark.

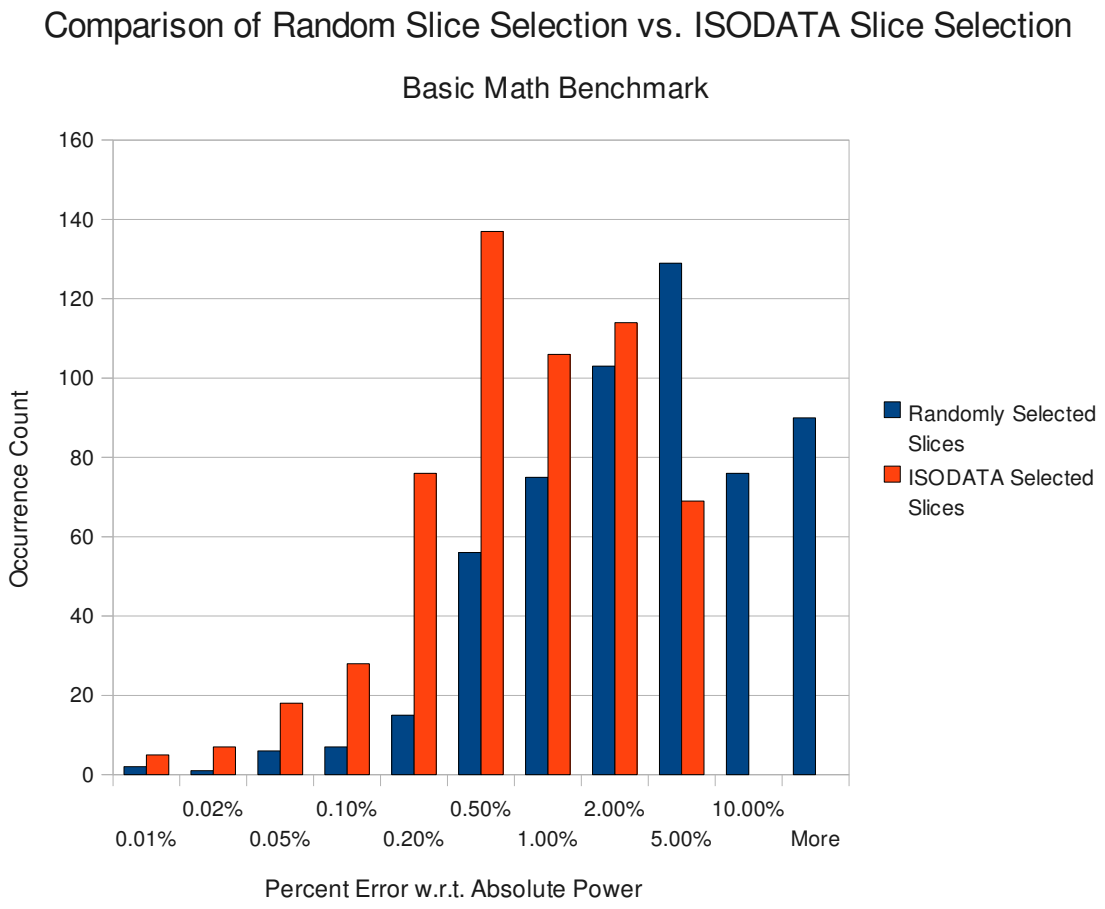

Figure 23: Effectiveness of ISODATA, Basic Math Benchmark

In this figure, the power estimation accuracy from the ISODATA selected slices is much higher than the power estimation accuracy from the randomly selected slices. Using ISODATA does not result in a power estimation error exceeding 5%, while using randomly selected slices results in power estimation errors exceeding 10%.

Figure 24 compares random slice selection and ISODATA slice selection, but for the Basic Math benchmark.

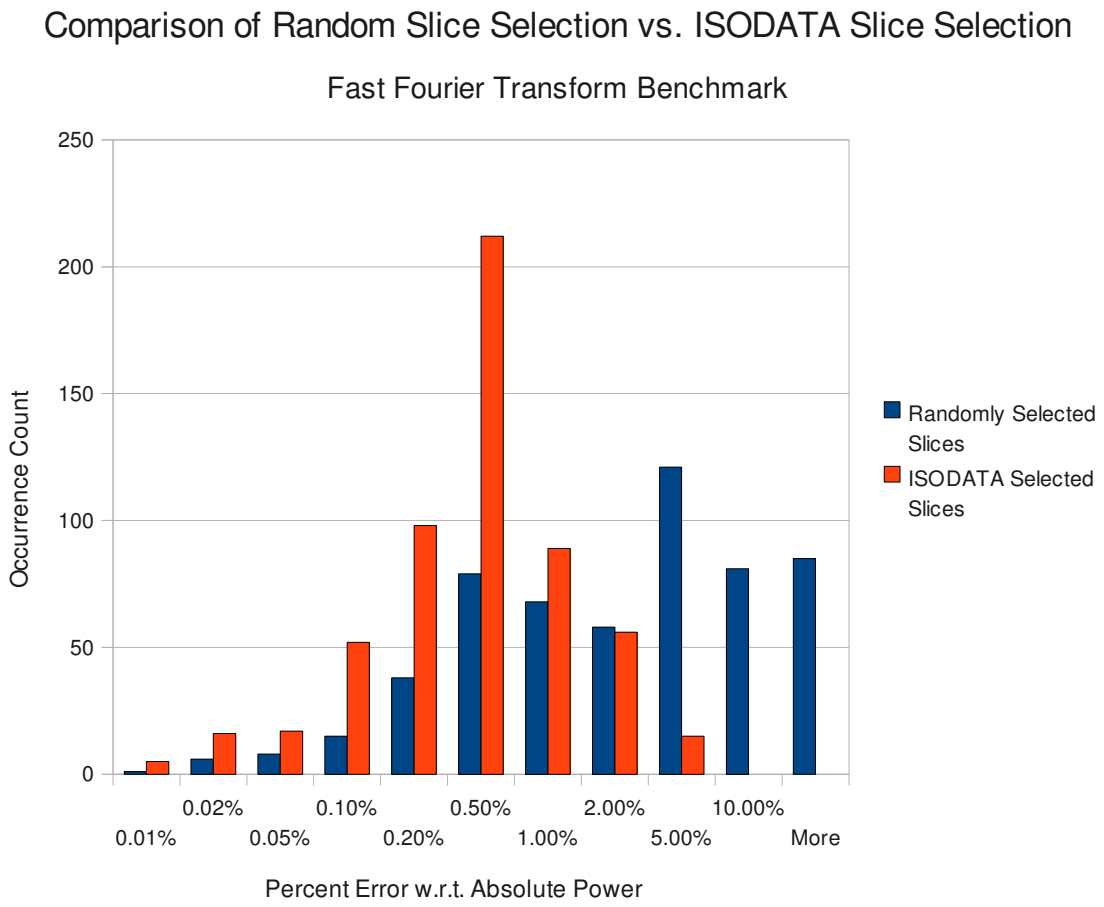

Figure 24: Effectiveness of ISODATA, Fast Fourier Transform Benchmark

Power estimation accuracy from the ISODATA selected slices is much higher than the power estimation accuracy from the randomly selected slices, as was observed in figure 23. Power estimation accuracy as a result of using of ISODATA does not exceed 5% error, while use of randomly selected slices results in power estimation accuracy exceeding 10% error. The results returned from experimentation with other two benchmarks, Dijkstra and Q-sort, demonstrate the same effect in using ISODATA.

5.2.3 Cluster Quality

The purpose of this set of experiments is to determine the effect of cluster quality on power estimation accuracy. In this set of experiments, cluster quality is the evaluation score returned by the ISODATA algorithm.

The experiments are run using the following parameters:

1. ISODATA and Genetic Algorithm parameters:
    1. *ISODATA k-Max Values*: 10, 20, 50, 100, 200, 500, 1000
    2. *ISODATA Termination Condition*: 0.3
    3. *ISODATA Max Iterations*: 50
    4. *ISODATA Merge Condition*: 1.0
    5. *ISODATA Split Condition*: 1.0
    6. *Genetic Algorithm Termination Condition*: 0.3
    7. *Genetic Algorithm Max Iterations: 50*
    8. *Percent of Features Selected*: 75%
2. Experiment parameters:
    1. *Execution Trace Slice Sizes*: 1000, 2000, 5000, 10000
    2. *Benchmarks*: Basic Math, FFT, Dijkstra, Q-Sort
    3. *Feature Selection*: No Feature Selection

4. *Known Power-Affecting Features*: With, Without

Figure 25 demonstrates the effect of cluster quality on power estimation accuracy for the Basic Math benchmark.

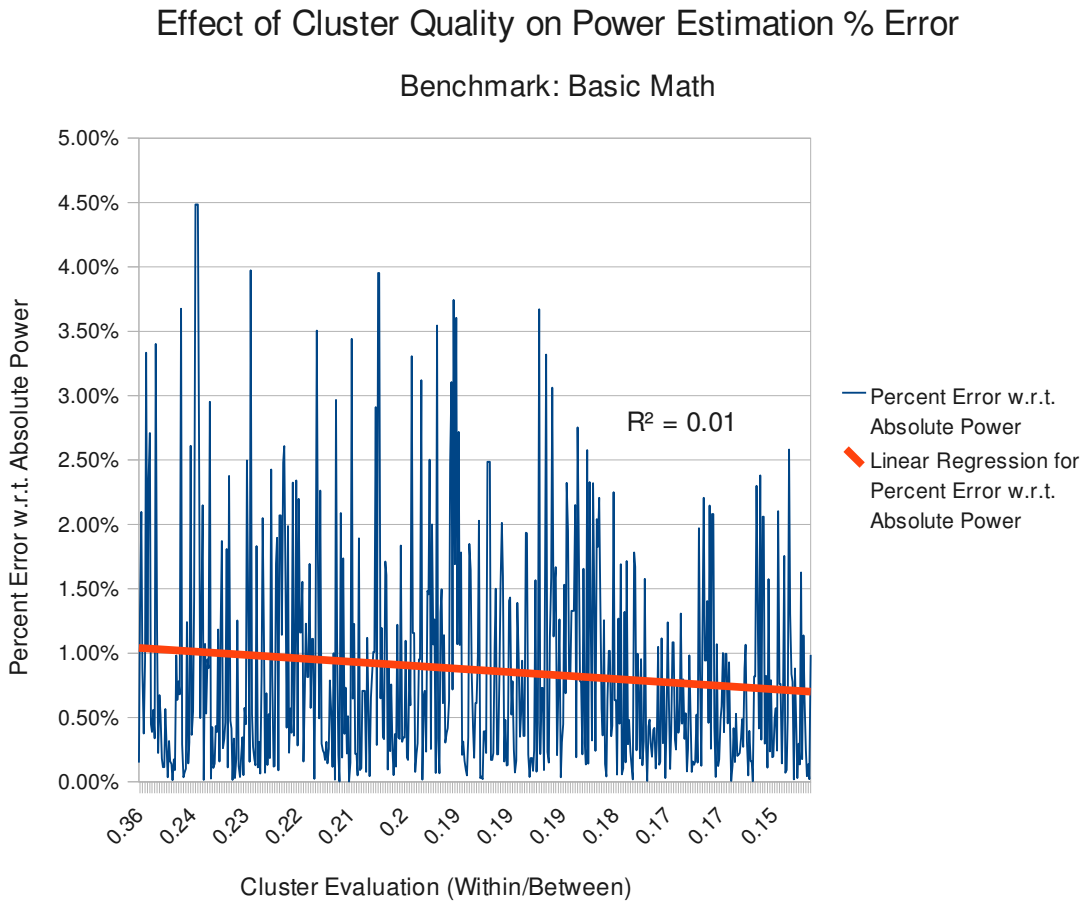

Figure 25: Effect of Cluster Quality, Basic Math Benchmark

In this figure, the linear regression coefficient is 0.01, indicating no correlation between power estimation accuracy and cluster quality. This lack of correlation may be due to the dispersion metric used to evaluate the quality of the cluster configuration. The Fast

Fourier Transform benchmark also achieved no demonstrable correlation between cluster quality and power estimation accuracy.

Figure 26 demonstrates the effect of cluster quality on power estimation accuracy for the Dijkstra benchmark, which returns better results than the previous benchmarks.

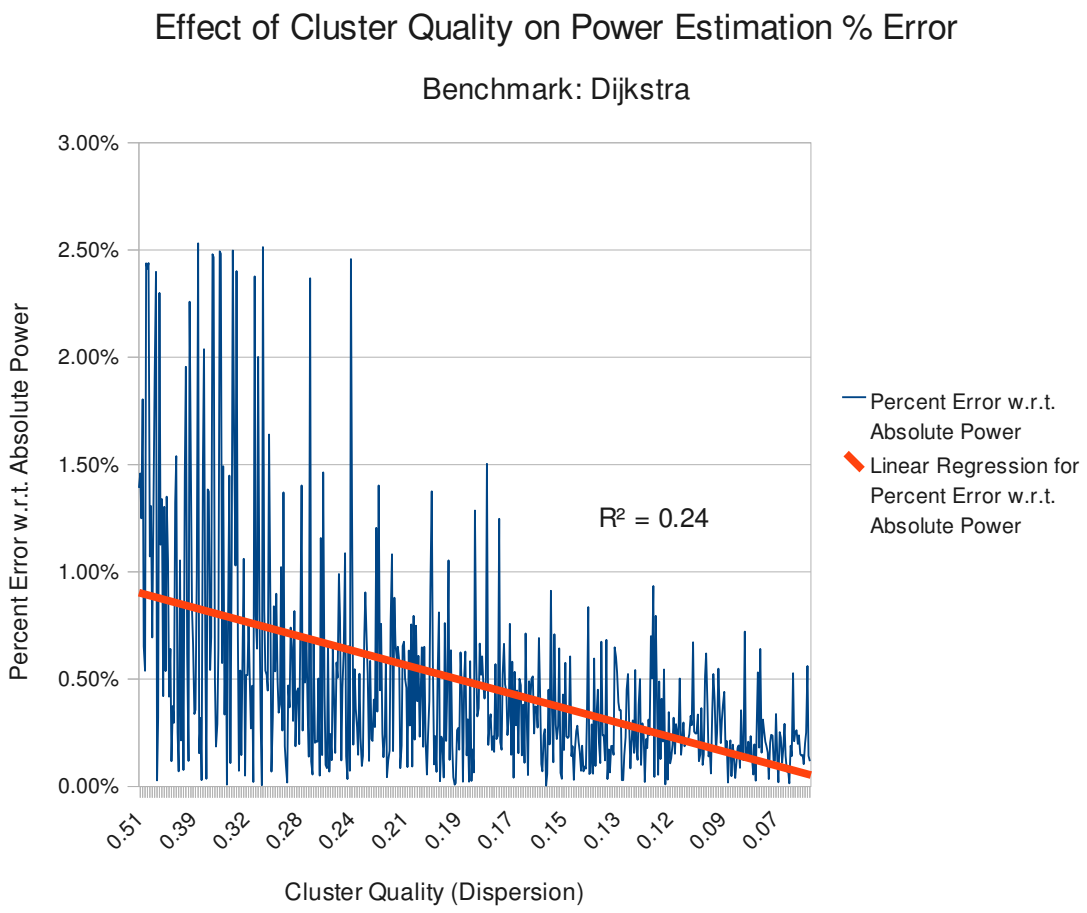

Figure 26: Effect of Cluster Quality, Dijkstra Benchmark

The regression coefficient is 0.24 for the data returned from experimenting with the Dijkstra benchmark, indicating significant correlation between power estimation accuracy

and cluster quality. This benchmark has the highest regression coefficient for all four benchmarks.

5.3 Verification of Hypotheses

The following experiments attempt to demonstrate verification of the three hypotheses.

5.3.1 ASTD Compression

This set of experiments compares the compression rate with the power estimation accuracy. Compression, in this case, is a product of the slice size and the number of clusters selected by ISODATA:

I : Total number of instructions
S : Slice size in instructions
K : Number of clusters selected by ISODATA

$$Compression = \frac{I}{K * S}$$

The experiments are run using the following parameters:

1. ISODATA and Genetic Algorithm parameters:
    1. *ISODATA k-Max Values*: 10, 20, 50, 100, 200, 500, 1000
    2. *ISODATA Termination Condition*: 0.3
    3. *ISODATA Max Iterations*: 50
    4. *ISODATA Merge Condition*: 1.0
    5. *ISODATA Split Condition*: 1.0
    6. *Genetic Algorithm Termination Condition*: 0.3

7. *Genetic Algorithm Max Iterations: 50*
8. *Percent of Features Selected*: 75%

2. Experiment parameters:
    1. *Execution Trace Slice Sizes*: 1000, 2000, 5000, 10000
    2. *Benchmarks*: Basic Math, FFT, Dijkstra, Q-Sort
    3. *Feature Selection*: No Feature Selection
    4. *Known Power-Affecting Features*: With, Without

In order to isolate the effect of cluster selection on power estimation accuracy, no feature selection methods are used. This results in 560 individual ISODATA runs per benchmark, for a total of 2240 individual runs. The results for each benchmark are described in the figures using a line graph, where the x-axis represents the compression ratio for each experiment in descending order, and the y-axis represents the corresponding power estimation accuracy.

Figure 27 demonstrates the effect of the compression ratio of the ASTD on the power estimation accuracy for the Basic Math benchmark.

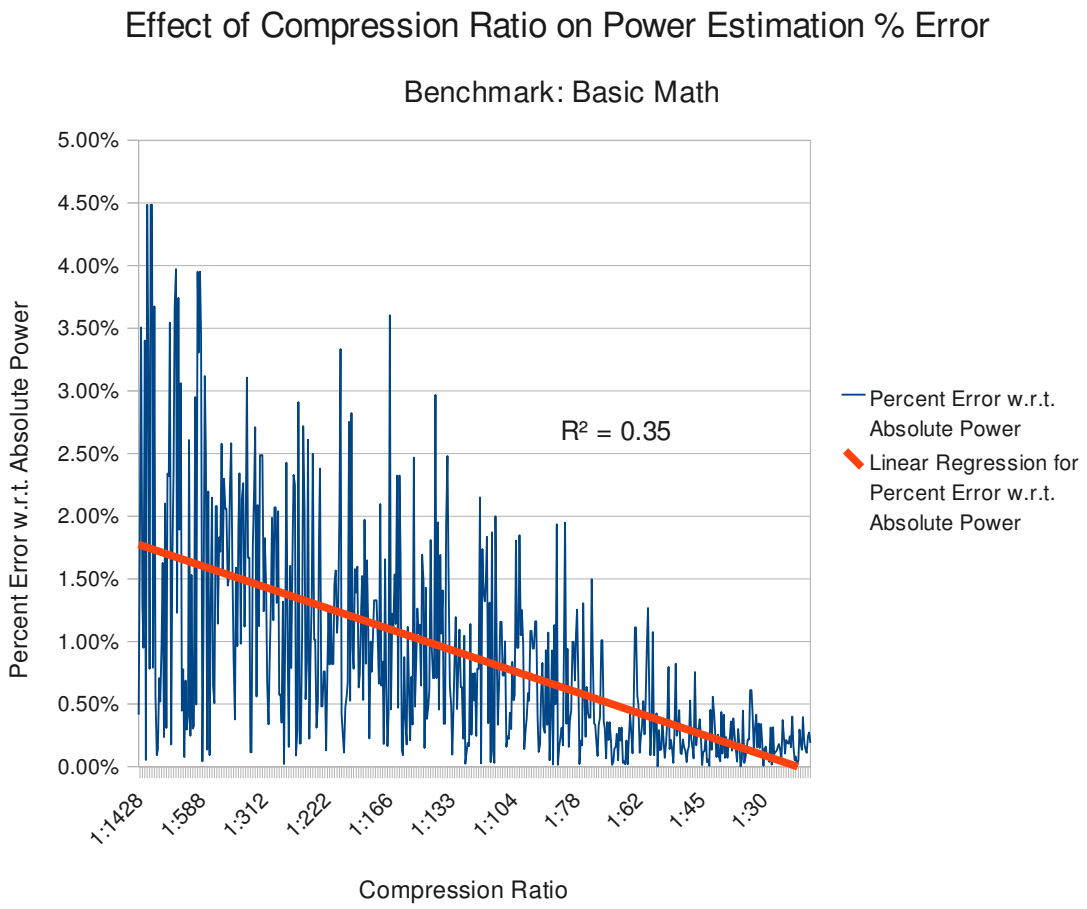

Figure 27: Compression Ratio, Basic Math Benchmark

In this figure, the overall power estimation accuracy increases as the compression ratio decreases, as indicated by a linear regression coefficient of 0.35.

Figures 28 through 31 represents a per-slice-size decomposition of figure 27 by slice size. This decomposition represents the effect compression on power estimation accuracy in greater detail.

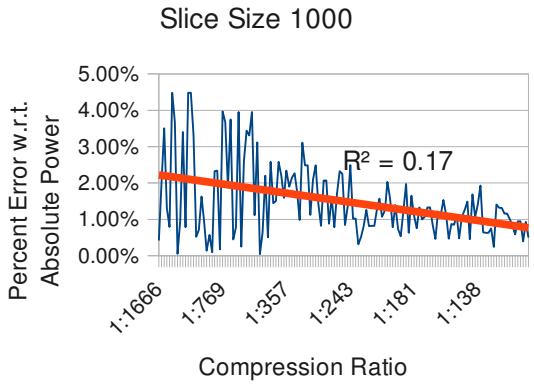

Figure 28: Compression Ratio, Basic Math Benchmark, Slice Size 1000

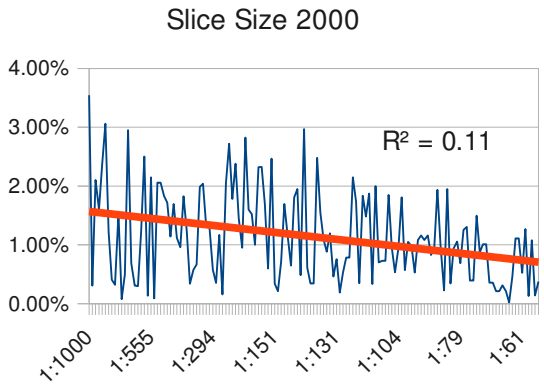

Figure 29: Compression Ratio, Basic Math Benchmark, Slice Size 2000

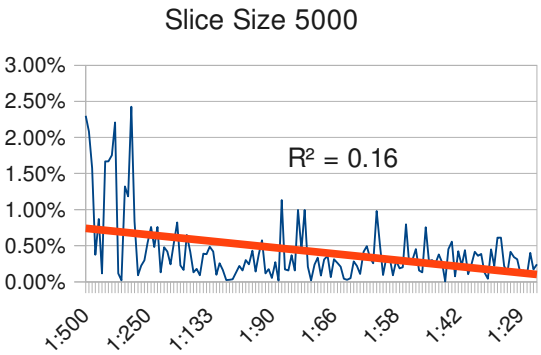

Figure 30: Compression Ratio, Basic Math Benchmark, Slice Size 5000

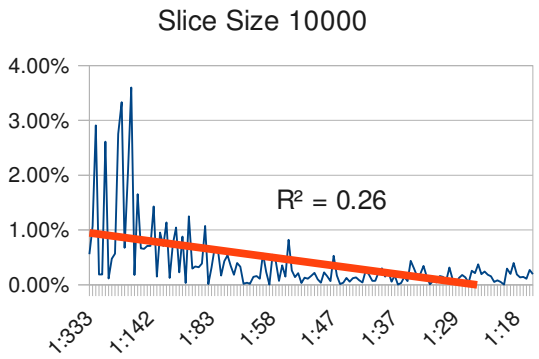

Figure 31: Compression Ratio, Basic Math Benchmark, Slice Size 10000

The power estimation accuracy increases as the slice size increases for these set of figures, as it did in figure 27. Slice sizes of 5000 and 1000 achieve much higher power estimation accuracy than slice sizes of 1000 and 2000. Additionally, while all of the regression coefficients are above 0.15, the slice size of 10000 returned a regression coefficient of 0.26.

For the Q-Sort benchmark data, figure 32 demonstrates the effect of the ASTD compression ratio on the power estimation accuracy.

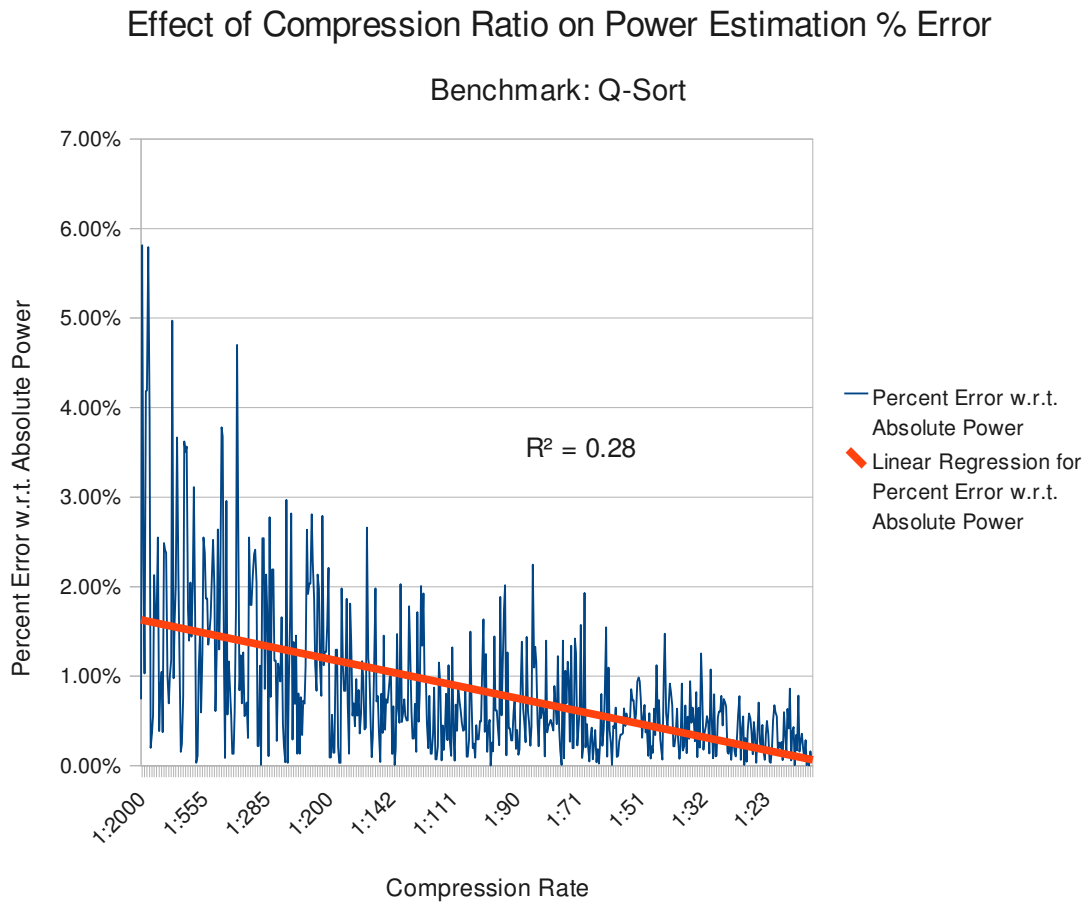

Figure 32: Compression Ratio, Q-Sort Benchmark

This figure demonstrates the overall power estimation accuracy increasing as the compression ratio decreases. The correlation is indicated by a linear regression coefficient of 0.28.

Figures 33 through 36 represent a decomposition of figure 32 by slice size.

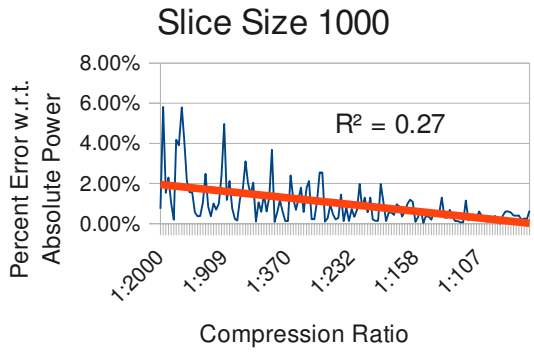

Figure 33: Compression Ratio, Q-Sort Benchmark, Slice Size 1000

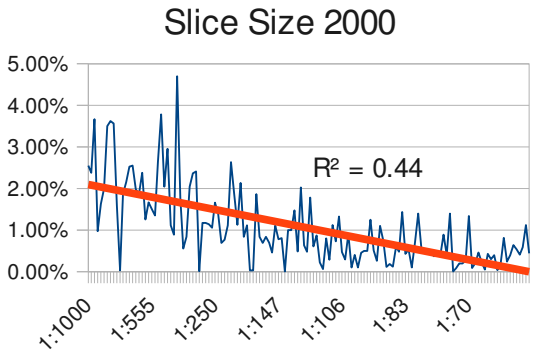

Figure 34: Compression Ratio, Q-Sort Benchmark, Slice Size 2000

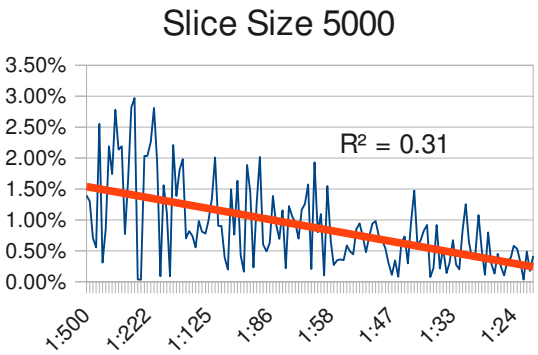

Figure 35: Compression Ratio, Q-Sort Benchmark, Slice Size 5000

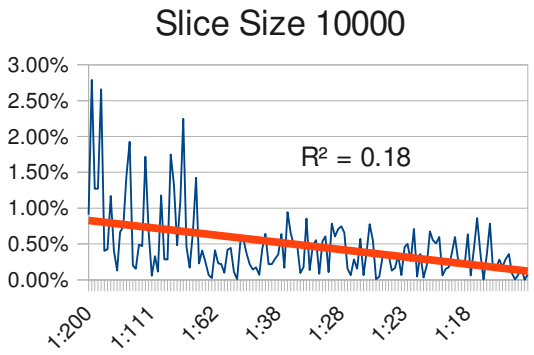

Figure 36: Compression Ratio, Q-Sort Benchmark, Slice Size 10000

For these figures, the power estimation accuracy increases as the slice size increases. For all slice sizes, the power estimation accuracy increases as the compression ratio decreases, all of which start out above 2.00% and decrease to below 2.00%. Power estimation accuracy for slice sizes 5000 and 10000 decrease below 1.00%.

5.3.2 Genetic Algorithm Feature Selection

The intent behind this set of experiments is to determine the effectiveness of using feature selection.

The experiments are run using the following parameters:

1. ISODATA and Genetic Algorithm parameters:
    1. *ISODATA k-Max Values*: 10, 20, 50, 100, 200, 500, 1000
    2. *ISODATA Termination Condition*: 0.3
    3. *ISODATA Max Iterations*: 50
    4. *ISODATA Merge Condition*: 1.0
    5. *ISODATA Split Condition*: 1.0
    6. *Genetic Algorithm Termination Condition*: 0.3
    7. *Genetic Algorithm Max Iterations: 50*
    8. *Percent of Features Selected*: 75%
2. Experiment parameters:
    1. *Execution Trace Slice Sizes*: 1000, 2000, 5000, 10000
    2. *Benchmarks*: Basic Math, FFT, Dijkstra, Q-Sort
    3. *Feature Selection*: Genetic Algorithm
    4. *Known Power-Affecting Features*: With, Without

Figure 37 demonstrates the effect of genetic algorithm feature selection on power estimation accuracy for the Basic Math benchmark data.

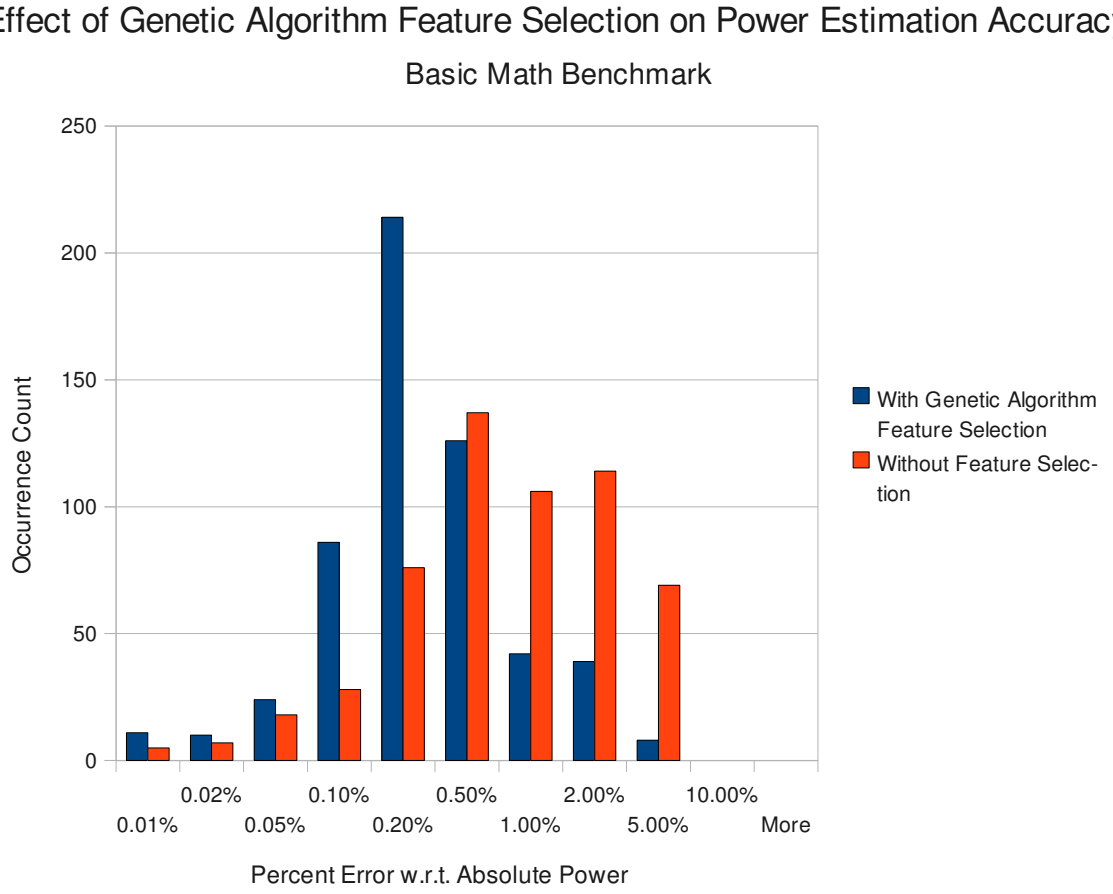

Figure 37: Effect of GA Feature Selection on Basic Math Benchmark

In this figure, the power estimation accuracy is significantly higher when the genetic algorithm feature selection is applied versus when it is not applied.

Figure 38, as with figure 37, demonstrates the effect of genetic algorithm feature selection on power estimation accuracy. However this figure demonstrates the effect for the Fast Fourier Transform benchmark data.

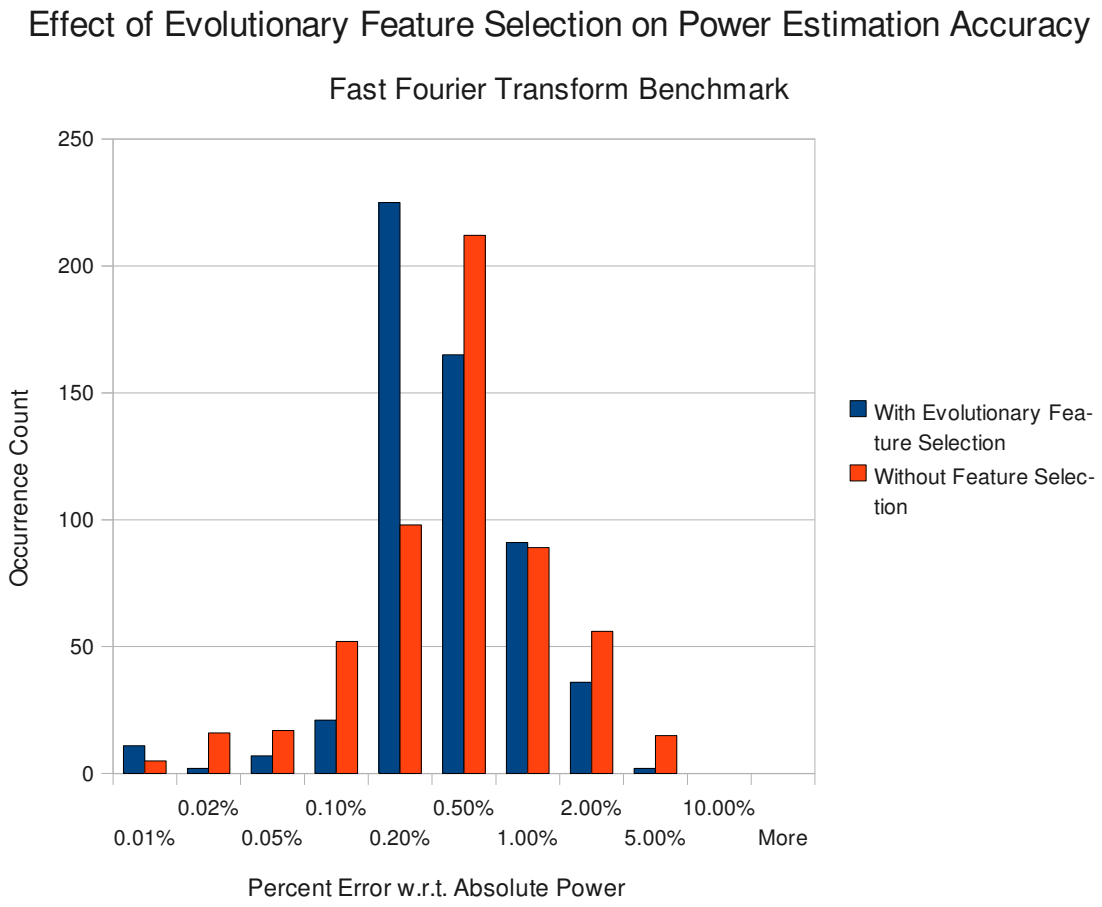

Figure 38: Effect of GA Feature Selection on Fast Fourier Transform Benchmark

In this figure, the advantage of the genetic algorithm feature selection is unclear. The trend from percent error range to percent error range does not give any indication of evolutionary feature selection providing an advantage in power estimation accuracy.

Figure 39 demonstrates the effect of genetic algorithm feature selection on power estimation accuracy for the Q-Sort benchmark.

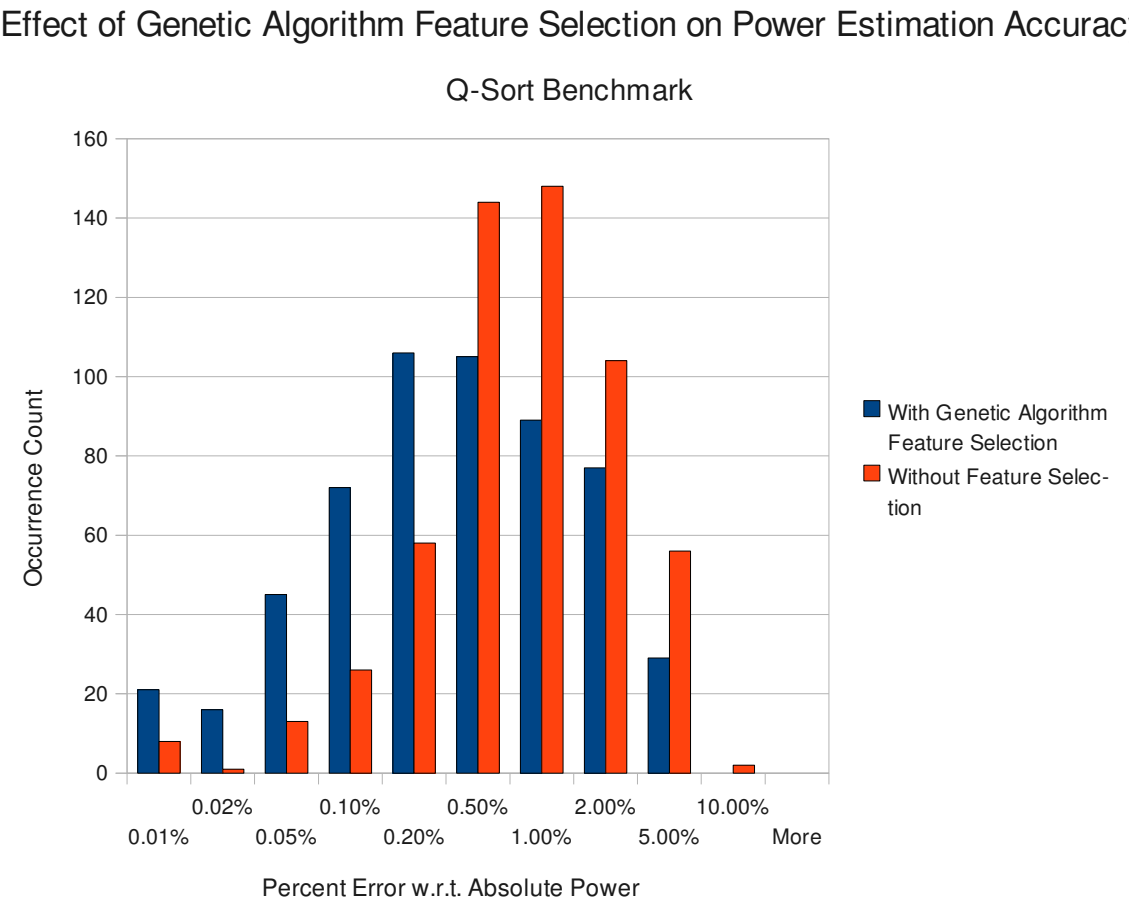

Figure 39: Effect of GA Feature Selection on Q-Sort Benchmark

In this figure, as with figure 37, the power estimation accuracy is significantly higher when the genetic algorithm feature selection is applied.

5.3.3 Known Power-Affecting Features

This set of experiments measures the effect of including the known power affecting features on power estimation accuracy after feature selection has been applied. Two sets of results are displayed for each figure: power estimation accuracy yielded from including the known power-affecting features in the complete feature set, and power estimation accuracy yielded from excluding the known power-affecting features from the complete

feature set.

The experiments are run using the following parameters:

1. ISODATA and Genetic Algorithm parameters:
    1. *ISODATA k-Max Values*: 10, 20, 50, 100, 200, 500, 1000
    2. *ISODATA Termination Condition*: 0.3
    3. *ISODATA Max Iterations*: 50
    4. *ISODATA Merge Condition*: 1.0
    5. *ISODATA Split Condition*: 1.0
    6. *Genetic Algorithm Termination Condition*: 0.3
    7. *Genetic Algorithm Max Iterations: 50*
    8. *Percent of Features Selected*: 75%
2. Experiment parameters:
    1. *Execution Trace Slice Sizes*: 1000, 2000, 5000, 10000
    2. *Benchmarks*: Basic Math, FFT, Dijkstra, Q-Sort
    3. *Feature Selection*: No Feature Selection
    4. *Known Power-Affecting Features*: With, Without

Figure 40 demonstrates the effect of power affecting features on power estimation accuracy, when included in the clustering and feature selection process, for slice sizes of

1000.

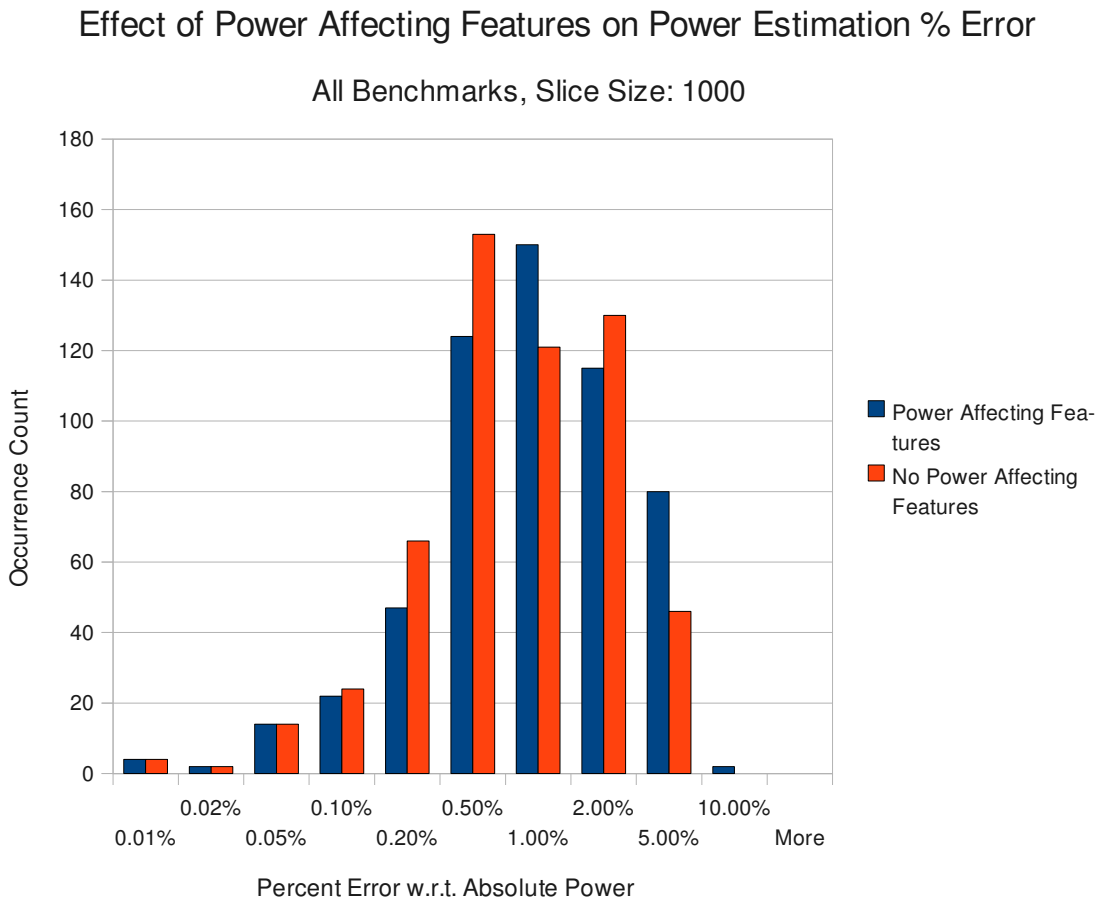

Figure 40: Effect of Power Affecting Features, Slice Size 1000

In this figure, determining whether or not power-affecting features result in higher power estimation accuracy is unclear for slice sizes of 1000.

Figure 41 demonstrates the effect of power affecting features on power estimation accuracy for slice sizes of 2000.

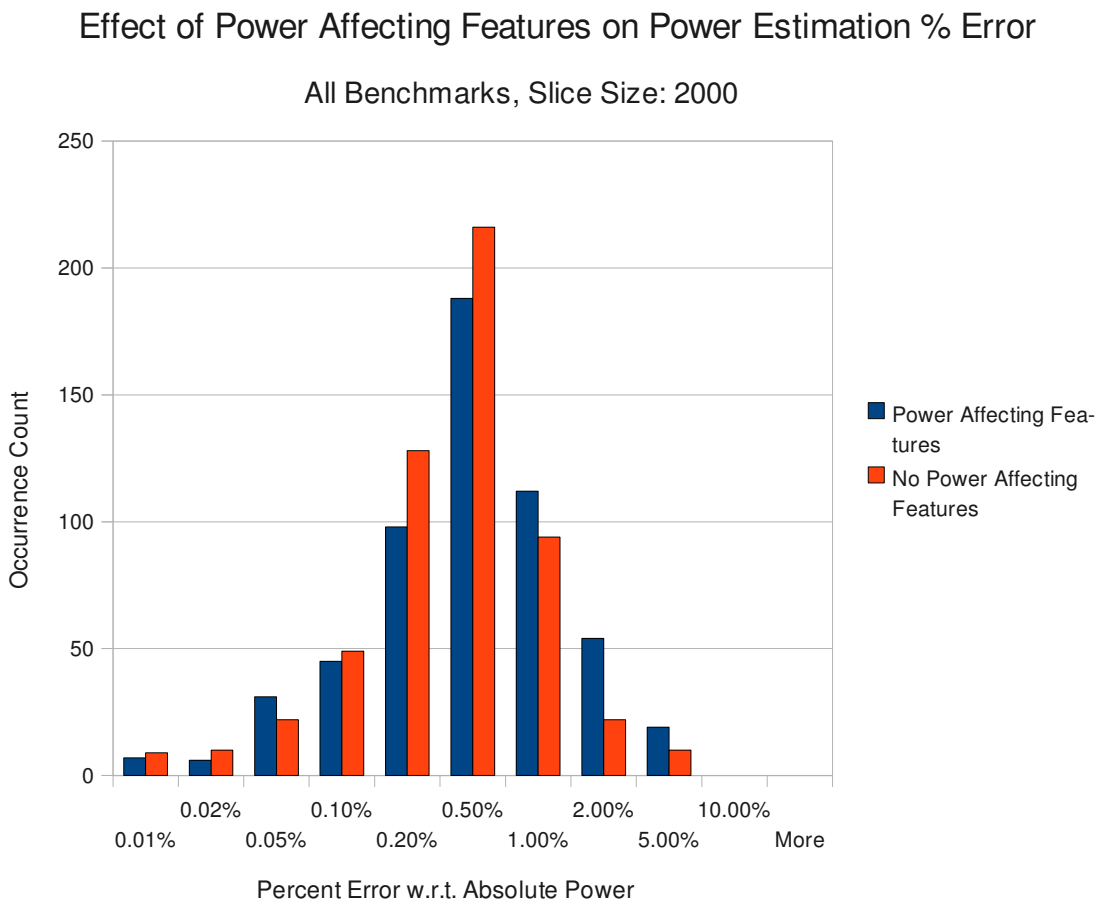

Figure 41: Effect of Power Affecting Features, Slice Size 2000

In this figure, as with figure 40, determining whether or not power-affecting features result in higher power estimation accuracy is unclear for slice sizes of 2000.

Figure 42 demonstrates the effect of power affecting features on power estimation accuracy for slice sizes of 5000.

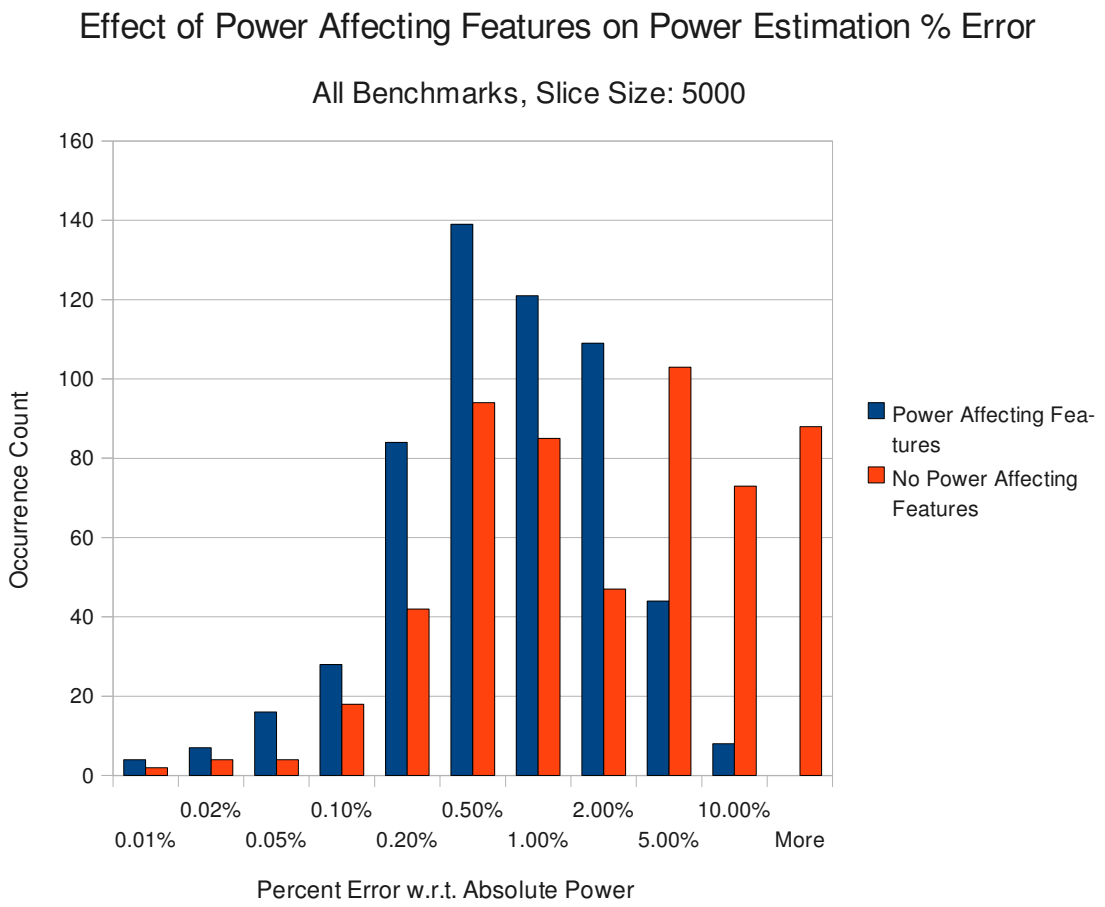

Figure 42: Effect of Power Affecting Features, Slice Size 5000

The inclusion of power-affecting features in the complete feature set consistently results in greater power estimation accuracy for slice sizes of 5000.

Figure 43 demonstrates the effect of power affecting features on power estimation accuracy.

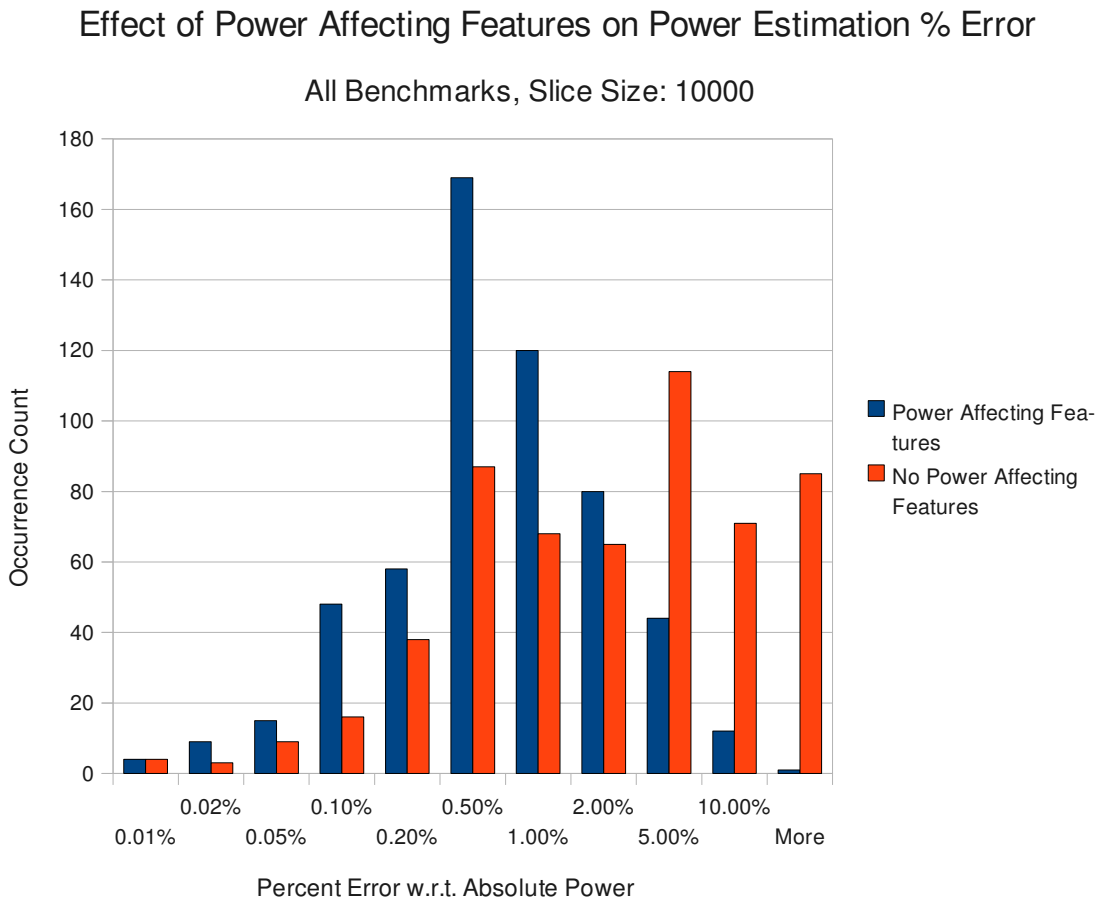

Figure 43: Effect of Power Affecting Features, Slice Size 10000

Similar to figure 42, figure 43 demonstrates that the inclusion of power-affecting features also results in greater power estimation accuracy for slice sizes of 10000.

6. ANALYSIS OF RESULTS

The following subsections analyze the results returned from the experimentation.

6.1 Algorithm Calibration and Evaluation

The following subsections are the results returned from the calibration and evaluation experiments.

6.1.1 Feature Selection Algorithm Evaluation

The exhaustive feature selection algorithm evaluates every possible feature subset, hence the large gap between the best and worst cluster qualities. Increasing feature subset sizes result in a factorial growth in the number of feature subsets evaluated, thus this gap rapidly grows in the first few feature subset sizes. However, the gap does not shrink symmetrically at the tail end of the feature subset sizes, as the clustering now must contend with many less-relevant features in the data set.

Beyond a certain feature subset size, adding a feature to the data does not effect the cluster quality in a significant way. In other words, there are a certain number of relevant features within the entire feature set, and all or most of those features have already been

discovered by feature selection. Thus, the addition of less relevant features at larger subset sizes should not effect the cluster quality significantly. Also, the increasing dimensionality of the data limits the optimization potential on the k-centroids, reducing any potential gains in cluster quality.

Rapidly decreasing cluster quality up to a certain subset size indicates the discovery of the minimal feature subset size, i.e. the feature subset size which contains only the most relevant features. The high cluster qualities achieved by feature selection at very low subset sizes do not correlate with the power estimation accuracy yielded at these subset sizes. Reducing the feature subset size too far eliminates relevant features, which results in less accurate power estimations.

6.1.2 Exhaustive Search Versus Genetic Algorithm Search

The cluster quality for exhaustive feature selection is consistently higher than the genetic algorithm feature selection in every case, except for the Q-Sort benchmark at slice size 2000. Exhaustive feature selection may not always return the best cluster quality, as the ISODATA algorithm is non-deterministic in its random initialization of the k means. However, the overwhelming majority of the cases demonstrate exhaustive feature selection resulting in the best cluster quality. These results are not surprising, given that the exhaustive feature selection searches all feature subsets, whereas the genetic algorithm only searches through some of them.

6.2 Affecting Factors of Power Estimation Accuracy

The following sections are the results returned from the experiments which demonstrate the factors that affect power estimation accuracy.

6.2.1 ASTD Slice Size

The experiment demonstrates cache warm-up having a greater effect on power estimation accuracy as the slice size decreases. The power estimation percent error is very small when compared to the power estimation percent error in other experiments. As such, slice size does not have a significant effect on power estimation accuracy for the experiments in our research.

6.2.2 ISODATA Slice Selection

The use of the ISODATA algorithm to determine representative slices significantly improves the accuracy of power estimation. The power estimation accuracy from the ISODATA algorithm is consistently higher than the power estimation accuracy from randomly selected representative slices.

6.2.3 Cluster Quality

Cluster quality has a small effect on power estimation accuracy for some of the benchmark applications. The effect is smaller than expected, possibly due to the dispersion metrics used in the ISODATA algorithm. Different dispersion metrics may result in more significant correlation between cluster quality and power estimation accuracy.

6.3 Verification of Hypotheses

These final subsections are the results returned from the calibration and evaluation experiments.

6.3.1 ASTD Compression

This experiment successfully demonstrates the trade-off between lossy compression and data integrity. In other words, a strong correlation is demonstrated between compression (reduced system level simulation throughput) and power estimation accuracy. The ISODATA algorithm thus yields a level of predictability in power estimation accuracy given a particular compression ratio.

6.3.2 Genetic Algorithm Feature Selection

This experiment successfully demonstrates that the genetic algorithm feature selection significantly improves the accuracy of power estimation for most of the benchmark tests. Most of the experiments demonstrated gains in power estimation accuracy from applying genetic algorithm feature selection.

6.3.3 Known Power-Affecting Features

Some of the slice sizes for the benchmarks demonstrated an improvement in power estimation accuracy by including the known set of power-affecting features. However, no observable trend is identified in order to determine why the improvement only occurs for some slices and not others. The reason for this may be specific to each benchmark's ASTD or to the result of slicing the ASTD into different sizes.

7. CONCLUSIONS

This thesis attempts to reduce system level simulation throughput while maintaining power estimation accuracy. Cluster analysis is used to identify similar ASTD slices generated from an architecture simulation of a benchmark test. Instruction stream slices which correspond to ASTD slice representatives of each cluster are run through system level simulation, returning the power estimation for each slice. These power estimations are multiplied by the number of slices in each cluster and then summed, according to equation 1, resulting in an approximation of the power estimation otherwise returned by simulating the entire benchmark test at the system level. Feature selection allows for further gains in power estimation accuracy by eliminating the less relevant features from the data set. Additional gains are made by prioritizing feature selection towards the features which are known to affect power. These three operations: cluster analysis, feature selection, and use of known power-affecting features form the three hypotheses.

The experiments are designed to verify the effectiveness of using cluster analysis, feature selection, and selection of known power-affecting features. The first category of experiments calibrate the algorithms and evaluate the feature selection algorithms. The

second category explores various factors which may affect power estimation accuracy. The third category is designed to directly verify the effectiveness of cluster analysis, feature selection, and use of known power affecting features. The results from these experiments align with expectations, and the overall results from the third category appear successful in verifying the three hypotheses.

The ISODATA algorithm is demonstrated as effective in selecting representative slices, resulting in accurate power estimation, for a number of reasons. The ISODATA algorithm results in greater power estimation accuracy than randomly selected representative ASTD slices. The ISODATA algorithm also results in increased power estimation accuracy as ASTD compression decreases. Power estimation accuracy achieved using the ISODATA algorithm conforms to the classic trade-off between lossy data compression and data integrity.

Feature selection is effective in further increasing the power estimation accuracy achieved from using the ISODATA algorithm. Results for most of the benchmarks clearly indicate the effectiveness of using feature selection. And finally, giving priority to the known power-affecting features when selecting the feature subset results in increased power for some some slice sizes, as these features are known to have the most significant bearing on power consumption. For those particular slice sizes, the results demonstrate that the advantage of using the known power-affecting features becomes significant for larger slice sizes.

As stated previously, the methodology described in this thesis is evaluated based on the three hypotheses:

1. The ISODATA algorithm is demonstrated as effective in achieving high power estimation accuracy.
2. Feature selection is demonstrated to be effective in achieving further gains in power estimation accuracy.
3. Prioritizing the use of power-affecting features results in greater power estimation accuracy.

Given the outcome of the experimentation, the first two hypotheses are verified to be successful in a majority of the experimental cases. The third hypothesis seems only effective in specific instances. However, as many questions about the successes demonstrated in experimentation inevitably arise, future work is required to further evaluate the boundaries and parameters of such success.

7.1 Future Work

The primary interest in future work is using a system level microprocessor simulation software to verify the results obtained within this thesis paper. The range of experimentation demonstrated in this paper provides a basis for further narrowing the focus of experimentation with system level simulation software, provided that the heuristic power estimation equation (Sunwoo et al., 2007) for estimating power using

architecture level simulation is reasonably accurate. Use of the system level simulation software would also provide insight on the practicality of our methodology with respect to time.

Other interesting applications for future work are in other heuristics and information theoretic methods as alternatives to the genetic algorithm feature selection and the ISODATA clustering algorithm. Other methods may result in improved feature selection, increased cluster quality, increased power estimation accuracy, and reductions in the time required for feature selection and cluster analysis. The methodology described in this paper is not dependent on any particular type of feature selection or cluster analysis algorithm. As such, many other algorithms could be substituted for those used in this thesis paper.

Research goals other than fast and accurate power estimation could also benefit from cluster analysis and feature selection. The methodology in this paper could be modified to apply to similar problems, such as the program similarity problem looked at by Joshi et al. (2006). It could also be modified to apply to a completely different field of research where data classification is necessary.

BIBLIOGRAPHY

Armstrong, J. R., Woodruff, G. (1977). Simulation Techniques For Microprocessors. In *Proceedings of the 14th Conference on Design automation,* 225-229.

Beatty, D. L., Bryant, R. E. (1994). Formally verifying a microprocessor using a simulation methodology. In *Proceedings of the 31st Annual Conference on Design Automation,* 596-602.

Brandolese, C., Fomacian, & W., Salice, F. (2000). An Instruction-Level Functionality-based Energy Estimation Model for 32-bit Microprocessors. In *Proceedings of 37th Design Automation Conference,* 346-350.

Chang, C. (1972) Dynamic Programming as Applied to Feature Selection In a Pattern Recognition System, In *Proceedings of the 3rd ACM Annual Conference, 1*66-171.

Cook, R. (2000). Detection of Influential Observations in Linear Regression. *Technometrics, 42*, 65-68.

Downey, A., & Feitelson, D. (1999). The Elusive Goal of Workload Characterization, *ACM SIGMETRICS Performance Evaluation Review, 26*, 14-29.

Freescale Semiconductor. (2010). ADL Project Release Directory. Retrieved from Freescale Semiconductor website: http://opensource.freescale.com/fsl-oss-projects/adl/.

Guyon, I., Elisseeff, A. (2003). An Introduction to Variable and Feature Selection, In *Journal of Machine Learning Research, 3*, 1157-1182.

Hennessy, J. L., Patterson, D. A. (2003). *Computer Architecture: A Quantitative Approach*, Third Edition, Morgan Kaufmann Publishers, Inc. ISBN 1558605967.

Hsieh, C., Chen, L., Pedram, M. (2001). Microprocessor Power Analysis by Labeled Simulation. In *Proceedings of the Conference on Design, Automation, and Test in Europe*, 182-189.

Jain, A., Murty, M., Flynn, P., (1999) Data Clustering: A Review. *ACM Computing Surveys, 31*, 264-323.

John, G., Kohavi, R., Pfleger, P. (1994). Irrelevant Features and the Subset Selection Problem. In *Proceedings of the 11th International Conference on Machine Learning*, 121-129.

John, L., Vasudevan, P., Sabarinathan, J. (1998). Workload Characterization: Motivation, Goals, and Methodology. *Workload Characterization: Methodology and Case Studies*, 3-14.

Joshi, A., Luo, Y., & John, L.K. (2007). Applying Statistical Sampling for Fast and Efficient Simulation of Commercial Workloads. In *IEEE Transactions on Computers, 56*, 1520-1533.

Joshi, A., Phansalkar, A., Eeckhout, L., & John, L.K (2006) Measuring Benchmark Similarity Using Inherent Program Characteristics. In *IEEE Transactions on Computers, 55*, 769-782.

Kahne, B. (2006). The ADL Trace Format: A Description of DAT Files, Freescale.

Kohavi, R., John, G. H. (1997). Wrappers for Feature Selection. In *Artificial Intelligence, 36*, 273-324.

Li, T., John, L. (2003). Run-time Modeling and Estimation of Operating System Power Consumption, In *Proceedings of the 31st ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems,* 160-171.

Luo, Y., Joshi, A., Phansalkar, A., John, L. K., & Ghosh, J. (2008). Analyzing and Improving Clustering Based Sampling for Microprocessor Simulation. In *Proceedings of the 17th International Symposium on Computer Architecture and High Performance Computing,* 193-200.

Phansalkar, A., & John, L. (2006). Performance Prediction Based on Inherent Program Similarity. In *Proceedings of the 15th International Conference on Parallel Architectures and Compilation Techniques,* 114-122.

Ravi, S., Raghunathan, A., & Chakradhar, S. (2003). Efficient RTL Power Estimation for Large Designs. In *Proceedings of the 16th International Conference on VLSI Design,* 431-439.

Rose, D.T. (2006). *Data Mining Methods and Models*. Wiley-IEEE Press.

Sherwood, T., Perelman, E., Hamerly, G., Calder, B. (2002). Automatically Characterizing Large Scale Program Behavior. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems,* 45-57.

Sherwood, W. (1977). Simulation hierarchy for microprocessor design. In *Proceedings of the Symposium on Design Automation and Microprocessors*, 44-49.

Sunwoo, D., Al-Sukhni, H., Holt, J. (2007). Early Models for System-Level Power Estimation. In *Proceedings of the 8th International Workshop on Microprocessor Test and Verification,* 8-14.

Tan, T., Raghunathan, A., Lakshminarayana, G., & Jha, N. (2001). High-level Software Energy Macro-modeling. In *Proceedings of Design Automation Conference*, 605-610.

Ting, S., Dy, J. (2004). A Deterministic Method for Initializing K-Means Clustering. In *Proceedings of the 16th International Conference on Tools with Artificial Intelligence,* 784-786.

Todi, T. (2001). SPEClite: Using Representative Samples to Reduce SPEC CPU2000 Workload. *IEEE International Workshop on Workload Characterization*, 15-23.

Wunderlich, R., Wenisch, T., Falsafi, B., & Hoe, J. (2003) SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling. In *Proceedings of 30th Annual International Symposium on Computer Architecture,* 84-95.

Xi, J., Huang, Z., & Zhong, P. (2005). Energy Macro-Modeling of Embedded Microprocessors Using SystemC. In *Proceedings of IEEE International Conference on Electro Information Technology*, 1-6.

Zhang, Y., & Zhang, G. (2009). Fast Gate-level Simulation and Power for High Performance Microprocessors. In *Proceedings of the 4th International Conference on Computer Science & Education,* 1155-1158.

Zhao, J., Wang, G., Wu, Z., Tang, H.,& Li, H. (2002) The Study on Technologies for Feature Selection. In *Proceedings of the 2nd International Conference on Machine Learning and Cybernetics,* 689-693.

VITA

Matthew Valdez Brock was born in Chicago, Illinois, on July 28th, 1982, the son of Beatrice Valdez Brock and Thomas William Brock. After completing his work at Tivy High School, Kerrville, Texas, in 2001, he entered Texas Lutheran University. He received the degree of Bachelor of Science from Texas Lutheran University in May 2005. During the following years he was employed as a graduate research assistant at Los Alamos National Laboratory in Los Alamos, New Mexico. In August 2005, he entered the Graduate College of Texas State University-San Marcos.

Permanent Address: 2002 Sharon Ln., Apt. B

Austin, Texas 78703

This thesis was typed by Matthew V. Brock.