POWER CONSUMPTION MODEL FOR DSP SYSTEMS

by

Gary Cameron

A thesis submitted in conformity with the requirements
for the degree of Masters of Engineering
Graduate Department of Engineering
Carleton University

Copyright © 2004 by Gary Cameron

Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-97462-6

Our file Notre référence

ISBN: 0-612-97462-6

The author has granted a non-exclusive license allowing the Library and Archives Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

PAGINATION ERROR.

TEXT COMPLETE.

ERREUR DE PAGINATION.

LE TEXTE EST COMPLET.

Abstract

Power Consumption Model for DSP Systems

Gary Cameron

Masters of Engineering

Graduate Department of Engineering

Carleton University

2004

The advent of compact yet sophisticated, portable electronic devices such as cellular phones, palmtop computers and electronic games have revolutionized modern life as the 21st century opens. Each generation of portable electronic devices pack ever increasing levels of functionality into a compact space. An assortment of ever more complex algorithms must be reconciled against the desire for more compact portable devices with longer battery times. There has been very little research to quantify or examine the possibility of reducing power consumption through software optimization.

This work examines power consumption for three very different DSPs, (Digital Signal Processors), which were made to loop on various test code segments, in order to derive a power consumption model. This demonstrates how a generic power consumption model can be applied towards accurately modelling the power dissipation for a wide range of CPUs running arbitrary code, and suggest strategies for improvement.

Acknowledgements

I would like to thank the following individuals for their invaluable help:

Dr. Trevor Pearce, my advisor Janeth and my parents, for their support and understanding.

Dedication

For Janeth, Kenneth, Tristan, Alex, and Janine.

Contents

1	Introduction	1
1.1	Statement of Thesis	4
1.2	Scope of work	5
1.3	Contributions	6
2	Background Information	7
2.1	Modelling Approaches	7
2.1.1	Low level hardware modelers	7
2.1.2	Empirical measurement of sample programs	8
2.1.3	System level modeling	9
2.2	Interpretation of SPSS output	10
2.2.1	Data Reduction SPSS Output Notes	11
2.2.2	Descriptive statistics table	12
2.2.3	Pearson Correlation	12
2.2.4	Significance 1-tailed	13
2.2.5	Number of data points	14
2.2.6	Variables entered or removed	14
2.2.7	Model Summary	15
2.2.8	ANOVA	18
2.2.9	Coefficients part A	19
2.2.10	Coefficients part B	21
2.2.11	Excluded variables	23
2.2.12	Coefficient Correlations	24
2.2.13	Collinearity Diagnostics	25

3	State of the Art in Power Optimization Studies	26
3.1	Hardware level modeling	26
3.2	Functional block modeling	27
3.3	System level modeling	29
4	Modeling	32
4.1	Modeling power consumption	32
4.2	Choosing a term of reference	36
4.3	Power consumption model	36
4.3.1	Creating a basic model	37
4.3.2	Basic model extensions	42
4.3.2.1	Pipeline and cache effects	42
4.3.2.2	Modeling branch prediction logic	53
4.3.2.3	Multiple masters	53
4.3.2.4	Variable clock rate	55
4.3.3	Compile time modeling	56
4.4	Regression Analysis	56
4.4.1	Regression diagnostics	57
5	Measurement	59
5.1	Measurement of energy consumption	60
5.1.1	Establishing the baseline zero power point	61
5.2	Development of test programs	62
5.3	Test setup	64
5.4	Post-processing	66
5.4.1	Assembly listing analysis	67
5.4.2	Conversion to tabular format	67
5.4.3	Converting voltage and current to energy	71
6	Case Studies	72
6.1	56303 Testing	73
6.1.1	56303 Model	73
6.1.2	Tabulation of 56303 observations	76
6.1.2.1	56303 assembly listing analysis	77
6.1.3	Analysis of 56303 results	88

6.1.3.1	Full 56303 regression	89
6.1.3.2	Internal busses only 56303 regression	95
6.1.3.3	Derived 56303 Power Consumption Model	96
6.2	C5402 Testing	96
6.2.1	C5402 Model	97
6.2.2	Tabulation of C5402 observations	100
6.2.2.1	Parsing C5402 assembly listings	101
6.2.3	Analysis of C5402 results	112
6.2.3.1	Full C5402 regression	112
6.2.3.2	Internal busses only C5402 regression	114
6.2.3.3	Derived C5402 Power Consumption Model	114
6.3	C6211 Testing	115
6.3.1	C6211 Model	115
6.3.2	Tabulation of C6211 observations	118
6.3.2.1	Parsing C6211 assembly listings	119
6.3.3	Analysis of C6211 results	132
6.3.3.1	Full C6211 regression	132
6.3.3.2	ALU only C6211 regression	133
6.3.3.3	Derived C6211 Power Consumption Model	134
7	Conclusions	135
7.1	Directions for future work	137
7.2	56301 specifics	138
7.3	C5402 specifics	139
7.4	C6211 specifics	139
	References	141
A	Shared Build Files	146
A.1	56303 shared build files	146
A.1.1	Link map file	146
A.1.2	Build file	149
A.2	C5402 shared build files	150
A.2.1	Linker map file	150
A.2.2	Project file	153

A.3	C6211 shared build files	155
A.3.1	Linker map file	155
A.3.2	Project file	156
A.3.3	C main file	157
B	List Parsing Programs	159
B.1	56300 Assembly Listfile Parsing Program	160
B.2	C5402 Assembly Listfile Parsing Program	175
B.3	C6211 Assembly Listfile Parsing Program	188
C	Octave Post Processing Files	202
C.1	56301 Results Processing	202
C.1.1	Raw Result to Tabular Data Conversion	202
C.1.2	Power Conversion Scaling	208
C.2	C5401 Results Processing	211
C.2.1	Raw Result to Tabular Data Conversion	211
C.2.2	Power Conversion Scaling	216
C.3	C6211 Results Processing	219
C.3.1	Raw Result to Tabular Data Conversion	219
C.3.2	Power Conversion Scaling	224
D	Processed Results	227
D.1	56303 Results	227
D.2	C5402 Results	245
D.3	C6211 Results	258
D.4	Data Reduction	264
D.4.1	56303 Data Reduction	264
D.4.1.1	Internal and External Memory Access	264
D.4.1.2	Internal Memory Access Only	281
D.4.2	C5402 Data Reduction	291
D.4.2.1	Internal and External Memory Access	291
D.4.2.2	Internal Memory Access Only	344
D.4.3	C6211 Data Reduction	369
D.4.3.1	C6211 Full	369
D.4.3.2	C6211 ALU Only	381

List of Tables

D.1	56303 Results Table Legend	227
D.2	56303 Results Table	228
D.3	C5402 Results Table Legend	245
D.4	C5402 Results Table	246
D.5	C6211 Results Table Legend	258
D.6	C6211 Results Table	259
D.7	56303 Data Reduction SPSS Output Notes	264
D.8	Descriptive Statistics	265
D.9	Pearson Correlation, part A	266
D.10	Pearson Correlation, part B	267
D.11	Significance: 1-tailed, part A	268
D.12	Significance: 1-tailed, part B	269
D.13	Number of data points, part A	270
D.14	Number of data points, part B	271
D.15	Variables Entered or Removed ^a	272
D.16	Model Summary	272
D.17	ANOVA ^f	273
D.18	Coefficients, part A ^a	274
D.19	Coefficients, part B ^a	275
D.20	Excluded Variables ^f	276
D.21	Coefficient Correlations ^a	279
D.22	Collinearity Diagnostics ^a	280
D.23	Data Reduction SPSS Output Notes	281
D.24	Descriptive Statistics	281
D.25	Pearson Correlation, part A	282
D.26	Pearson Correlation, part B	282

D.27 Significance: 1-tailed, part A	283
D.28 Significance: 1-tailed, part B	283
D.29 Number of data points, part A	284
D.30 Number of data points, part B	284
D.31 Variables Entered or Removed ^a	285
D.32 Model Summary	285
D.33 ANOVA ^f	285
D.34 Coefficients, part A ^a	287
D.35 Coefficients, part B ^a	287
D.36 Excluded Variables ^f	288
D.37 Coefficient Correlations ^a	289
D.38 Colinearity Diagnostics ^a	290
D.39 C54 Data Reduction SPSS Output Notes	291
D.40 Descriptive Statistics	292
D.41 Pearson Correlation, part A	293
D.42 Pearson Correlation, part B	294
D.43 Significance: 1-tailed, part A	295
D.44 Significance: 1-tailed, part B	296
D.45 Number of data points, part A	297
D.46 Number of data points, part B	298
D.47 Variables Entered or Removed ^a	299
D.48 Model Summary	300
D.49 ANOVA ^f	302
D.50 Coefficients ^a	305
D.51 Coefficients ^a	309
D.52 Excluded Variables ^o	313
D.53 Coefficient Correlations, part A ^a	318
D.54 Coefficient Correlations, part B ^a	324
D.55 Coefficient Correlations, part C ^a	330
D.56 Colinearity Diagnostics part A ^a	336
D.57 Colinearity Diagnostics part B ^a	340
D.58 C54 No External Memory Data Reduction SPSS Output Notes	344
D.59 Descriptive Statistics	344
D.60 Pearson Correlation, part A	345

D.61 Pearson Correlation, part B	346
D.62 Significance: 1-tailed, part A	347
D.63 Significance: 1-tailed, part B	348
D.64 Number of data points, part A	349
D.65 Number of data points, part B	349
D.66 Variables Entered or Removed ^a	350
D.67 Model Summary	351
D.68 ANOVA ^j	352
D.69 Coefficients ^a	354
D.70 Coefficients ^a	356
D.71 Excluded Variables ^j	358
D.72 Coefficient Correlations, part A ^a	361
D.73 Coefficient Correlations, part B ^a	364
D.74 Colinearity Diagnostics ^a	367
D.75 C6211 Data Reduction SPSS Output Notes	369
D.76 Descriptive Statistics	369
D.77 Pearson Correlation, part A	370
D.78 Pearson Correlation, part B	370
D.79 Significance: 1-tailed, part A	371
D.80 Significance: 1-tailed, part B	371
D.81 Number of data points, part A	372
D.82 Number of data points, part B	372
D.83 Variables Entered or Removed ^a	373
D.84 Model Summary	373
D.85 ANOVA ^f	374
D.86 Coefficients, part A ^a	375
D.87 Coefficients, part B ^a	376
D.88 Excluded Variables ^f	377
D.89 Coefficient Correlations ^a	379
D.90 Colinearity Diagnostics ^a	380
D.91 C62 ALU only Data Reduction SPSS Output Notes	381
D.92 Descriptive Statistics	381
D.93 Pearson Correlation	381
D.94 Significance: 1-tailed	382

D.95 Number of data points	382
D.96 Variables Entered or Removed ^a	382
D.97 Model Summary	382
D.98 ANOVA ^b	382
D.99 Coefficients ^a	383
D.100 Coefficients ^a	383
D.101 Coefficient Correlations ^a	383
D.102 Collinearity Diagnostics ^a	384

List of Figures

4.1	Bus capacitance power consumption model	34
4.2	Bus cycle model - return to neutral vs. direct switching	35
4.3	Simplified generic DSP	37
4.4	Write Through Cache Read Hit	45
4.5	Write Through Cache Read Miss	46
4.6	Write Through Cache Write	47
4.7	Copy Back Cache Write	48
4.8	Hit on L1 cache	50
4.9	Hit on L2 cache	51
4.10	Cache miss	52
4.11	Bus with multiple masters	54
5.1	Schematic of Test Setup	64
5.2	Frontal View of Test Setup	65
5.3	Test Setup with C6211 Board	66
6.1	56303 Block diagram[24]	74
6.2	Plot of TDRE vs. Energy	91
6.3	3D plot of TDRE and TDWE vs. Energy	92
6.4	Plot of TDWI vs. Energy	93
6.5	3D plot of TDWI and TDRI vs. Energy	94
6.6	C5402 Block diagram	98
6.7	C6211 Block diagram[33]	116

Chapter 1

Introduction

Despite the popularity of portable and hand-held electronic devices, to date there has been minimal study of what software optimizations could be done to extend battery life. There are a number of possible reasons for this - the need to run legacy software applications, (obviously true in the case of laptop PCs) and the difficulty in wringing the required functionality out of limited performance components. Unfortunately, it was not readily apparent what optimizations were even possible. Modern CPUs are extremely complex devices, and there may be millions of logic elements involved during the execution of a single machine instruction. Even if algorithm developers were conscious of the need to minimize power, there was no practical guidance for them on how to do it. Although processor hardware data sheets and programmer's guides may contain hundreds of pages, they typically list only typical and maximum power dissipation figures.

The ever increasing complexity of modern processors as driven by Moore's law, makes it difficult to precisely model power consumption on a per instruction basis. At first glance, it seems impossible to derive a power consumption model that is both reasonably accurate yet simple enough to be practical.

The purpose of this work is to develop and test generic bus activity based power consumption models, which could be of use to a compiler, yet general enough to be applicable across diverse architectures, with only the weighting factors changed. Ideally, this model can serve as a instruction weighting input to high level language compilers, which would enable it to select less power hungry instruction sequences. The derived models are tested on three diverse DSP architectures, to show that the techniques can be applied to radically different architectures.

There are a number of approaches that have been tried. For example, [35], [22], and [15] modeled only the memory subsystem while ignoring the variations between instructions. Some other papers such as [7] described a very detailed gate level simulation approach. [37] measured power consumption of individual instructions on a processor using test vector generators, however the physical measurement techniques would be difficult or impossible to use with modern processors, and there was no comparisons made across different processor families. This approach has the potential of generating the most accurate results, although it is expensive and often impractical. The results cannot be applied to an arbitrary program without re-running the simulation. It also fails to provide any useful information as to how the software or hardware can be improved.

In particular, [21], [19], and [20] measured power consumption by putting the processor in an infinite loop, and using an integrating ammeter, as described in [36]. However, this work was limited to optimizing code for a particular processor, and did not attempt to model the effect of varying address busses or data. There was no attempt to apply a generalized model across several diverse processor architectures. This is particularly important for a retargetable compiler such as gcc, where new processor families can be supported simply by plugging the correct coefficients into the power consumption model.

Once an accurate instruction level power consumption model has been established, it becomes possible to use this information to establish a power consumption based per instruction cost weighting. If the model is not overly complex, it may be built into an optimizing compiler, which can select instruction sequences weighted for minimal power consumption. This permits software optimization for minimal power consumption to be done, even while working in a high level language. It also becomes possible for code profilers to generate predicted power consumption statistics, once it is known how much CPU time is spent running a particular sequence of instructions.

In chapter 2, the various techniques for modeling power consumption modeling are described. They may be broadly lumped into the following categories:

1. Low-Level gate and transistor level hardware models
2. Empirical measurement of sample programs
3. System level measurements

Chapter 2 contains background and reference information. Section 2.1 examines the current state of research, categorized under the broad headings introduced in section 2. This paper focuses on the second approach. Section 2.2 describes how to interpret the SPSS¹ regression tables.

In chapter 4 a generic model is developed based on bus capacitance. Section 4.2 describes why power is given in terms of joules per CPU instruction. The generic model itself is presented in section 4.3. ANOVA (Anaysis of Variance) techniques used to validate the model are discussed in section 4.4.

Chapter 5, describes the methods used to perform experimental measurements. Section 5.3 contains a schematic and photo of the experimental setup. Section 5.4 briefly describes how measurements are extracted from assembly language listings.

Application of the generic model to the three targets selected for study in chapter 6. This chapter contains a high level summary of the regression results for each targeted DSP, as well as the final power consumption model. Statistically significant terms are mentioned, and any unexpected findings are discussed at length.

The findings are summarized in chapter 7, along with their implications. Possible avenues for further study in the field are touched upon here.

Memory map and build configuration files are given in appendix A. These are used to help generate the test programs from the source code.

The assembly listing parser programs are given in appendix B. The purpose for these are detailed in section 5.4.1.

Appendix C, contains a set of octave² programs, which were used to convert the raw data into a more useful tabular format, suitable for statistical analysis. These are discussed at greater length in section 5.4.2.

Appendix D contains all of the experimental data in a pre-processed, tabular form. The format and purpose of these tables are described by example in section 5.4.2.

¹©SPSS Inc.

²A GPL (Gnu Public License) matlab work-alike

1.1 Statement of Thesis

It is possible to develop a simple, generic bus-oriented model for analyzing DSP software power consumption, and apply it to an assortment of DSPs running in a target system, using only basic tools and commonly available information. The power consumption model can be generated and simplified by employing simple measurement and statistical data reduction techniques.

1.2 Scope of work

A simple power consumption model based on bus bit transitions was developed and tested for three different DSPs. The following effects were considered:

- Tri-state buses
- Instruction pipeline
- On chip versus off chip memory accesses
- ALU accesses
- Address bus transitions

The following effects were not considered, and might form the basis for future work:

- Interaction between terms, and higher order dependencies.
- Effect of varying power supply voltage
- Process variation between different CPUs of the same type.
- Varying master clock frequency
- Effect of cache hit or miss
- Effect of varying memory wait states
- Modeling the effect of DMA activity
- Penalties for executing code from external memory

1.3 Contributions

The following contributions were made through this work:

1. A practical bus-oriented power consumption model was developed.
2. Demonstration of how these techniques may be refined and applied to a variety of CPU architectures.
3. Accurate board level measurement techniques for collecting power consumption data, and isolating the software contribution were developed.
4. Practical application of the model and measurement techniques for three specific DSP systems.
5. The application of statistical techniques to validate the model and remove insignificant model terms was demonstrated.

Chapter 2

Background Information

2.1 Modelling Approaches

Approaches towards modeling software-dependent power consumption can be lumped into three broad categories, with various strengths and weaknesses.

2.1.1 Low level hardware modelers

The first approach is theoretically capable of generating extremely accurate power dissipation models, since the system is being modeled in extreme brute-force detail. Unfortunately, it suffers from a number of serious drawbacks, which may render it impractical for use in most cases:

- Typically only the CPU itself can be modeled in such detail. Unfortunately, the power effects of accessing off-chip memory and peripherals (which tend to be significant, as will be shown later) are totally ignored. Unfortunately, this makes the highly accurate CPU power consumption model virtually useless as a determinant of overall runtime system power consumption. To accurately describe the entire system, detailed hardware models would be needed for every piece of hardware that is accessed by the program during its normal operation.
- Detailed gate level models of the CPU are not normally provided by the manufacturer, as they fear exposing sensitive trade secrets to competitors. If this information is provided at all, it would only be to large customers with high

potential sales volume, who have signed non-disclosure agreements. This sort of information is openly available for a select few processor cores, but this is the exception, rather than the rule. In order to model the entire system, these problems are compounded, as the CPU, memory and glue logic (if any) are usually manufactured by different companies, who will be reluctant to risk compromising their trade secrets.

- The computational power required to run a gate level simulation of any sophisticated CPU running a meaningful program is extremely high. This is all the more true when simultaneously trying to simulate the entire CPU, cache, and memory subsystem. For all but the most trivial simulations, this approach quickly becomes prohibitive in terms of cost and/or simulation time.
- The simulation is so complex, that it is difficult or impossible to derive a predictive model from it. Without actually running the test code in the simulator, it cannot be predicted in advance what the outcome will be.

2.1.2 Empirical measurement of sample programs

The second approach breaks the CPU and peripherals into a collection of black boxes. No detailed knowledge is assumed or required about the innards of each box. The behavior of each component is determined by analysis of the overall power consumption, given knowledge of the inputs (program and input data) and outputs (results) of each box. Each component is assumed to consume a fixed amount of power, plus a variable factor governed by the supplied control and data signals. Advantages of this technique are:

- The system does not need to be modeled down to the individual gate level. This makes it possible to analyze more realistic programs without lengthy simulations. Naturally, given detailed information available regarding the functional blocks in the system, it can be better modeled. While this may be helpful in establishing a very accurate model, it is not required.
- The system power consumption model can provide guidance to power optimizing compilers. While it is utterly impractical for a compiler to test the effects of a candidate instruction sequence on a gate level model of the system, it

would be possible to check it against a simplified model covering the major functional blocks.

The major difficulties with this approach are:

- How to accurately measure the cycle by cycle power consumption of each individual functional block? This can seem to be an impossible chore, as many component blocks (such as the ALU, cache, or internal register bank) reside on chip, with no direct access to them. Even when the blocks reside in separable packages, it is extremely difficult to accurately measure instantaneous power consumption during CPU instructions that last only a few nanoseconds. Complicating things further - most modern CPUs have multiple power and ground pins. This is done to ensure even distribution of power across the die. The traditional technique of current measurement - lifting an IC pin, inserting a series resistor and measuring the current across it is no longer practical.
- Many modern CPUs employ pipelined instruction fetch, decode, and execution. Even when accurate measurements are possible, the effect of individual instructions can be spread across many CPU cycles, and blended with the effects of previous and subsequent instructions.
- How should the power consumption of each functional block be modeled? And how can this model then be checked for accuracy?

2.1.3 System level modeling

The third approach models power consumption based on CPU, cache, and memory interactions. Variations between instructions are treated as insignificant. This technique is helpful in evaluating the tradeoffs between various cache sizes and architecture, but does not address any potential benefits that may be derived by modifying individual instructions. Each instruction is assumed to consume approximately the same amount of power. Of course this is not true, but the model assumes that the cycle by cycle variation of power consumed by running a block of code out of a given block of memory will tend to a fixed average value, and that only the cache and memory architecture matters. The advantages of this approach are:

- The power consumption model is very simple. The compiler only needs to optimize cache hit ratios, and allocate CPU intensive functions and critical interrupts to faster (on chip) memory. Consequently, only a small number of test programs are required to build up the model.
- There is no data dependencies in the model. While the program instruction sequence may be perfectly known, the data to be operated on is usually indeterminate.
- The effect of trading off cache for larger on chip memory, (an option with the Motorola 56300 DSP) varying clock rates and external memory wait states can be easily seen.

Unfortunately, the technique also has some limitations, that preclude most compile time optimization:

- If some instructions are significantly more power hungry than others, opportunities for optimizing them out will be missed.
- Interactions between instructions cannot be modeled using this technique. For instance, repeating an instruction twice will not cause any bit transitions, while pairing two unrelated instructions will cause on average, half the program data space bits to transition.

2.2 Interpretation of SPSS output

The SPSS results are presented as a set of 12 automatically generated tables for each regression. They may be found in appendix D.4 These are:

1. SPSS output notes
2. Descriptive Statistics
3. Pearson Correlation
4. Significance, 1-tailed
5. Number of data points

6. Variables entered or removed
7. Model summary
8. ANOVA
9. Coefficients
10. Excluded Variables
11. Coefficient Correlations
12. Colinearity Diagnostics

In some cases, it was necessary to split wide tables into two or more parts, in order to fit them on the page. The purpose behind each table entry are discussed at length below. The $\hat{B}$ column of the last row of the coefficients table are the most important, as this contains the weighting factors for the final, and usually most accurate stepwise regression. But the information contained in many of the other tables are helpful in identifying troublesome interdependencies as described in section 4.4.1. They contain background information that may be helpful in determining how the results were obtained, and why some model terms were rejected. They also give helpful indications as to how the various model terms interact with each other. They can also provide clues as to what happened when the results are unexpected.

2.2.1 Data Reduction SPSS Output Notes

This is the first table produced from the SPSS regression. An example of this may be found in table D.7. This table does not contain much of interest, but has been provided for completeness. It contains general information such as when the results were tabulated, whether any special pre-processing filter or weighting function was used. It also lists the total number of data points used, and describes how the regression is done. In all cases, stepwise regression was used, with candidate terms added one at a time with each regression step if their significance The dependent variable, (ENERGY) as well as all independent variables (model terms) are listed. The exact invocation lines of all SPSS commands used to generate the results are also given. The command syntax is fully described in the SPSS manual[32].

2.2.2 Descriptive statistics table

An example of this may be found in table D.8. It lists the mean and standard deviation for all variables, as well as the number of data points for each model term variable.

These values are usually not very interesting, but have been provided for completeness. It is useful to compare the mean with the standard deviation of each term. Ideally, the standard deviation of each dependent variable should be fairly large with respect to the mean. This indicates that there exists a wide spread of data points for this term, making it easier to determine if a dependency exists. A large standard deviation with the dependent variable indicates that at least some of the software modifications affected the overall power consumption. If the software changes had no effect, the standard deviation of the energy would be zero.

In all of the experiments, every data point contains values for all variables so they are always the same.

2.2.3 Pearson Correlation

The Pearson correlation table describes how well each variable is correlated with each other[17]. It has no bearing on the actual regression output, but is useful in identifying problem interdependencies as discussed in section 4.4.1. Table D.9 shows a typical example. It may be defined as:

$$r = \frac{\sum_{i=1}^n x_i y_i - \frac{\sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n}}{\sqrt{(\sum_{i=1}^n x_i^2 - \frac{(\sum_{i=1}^n x_i)^2}{n})(\sum_{i=1}^n y_i^2 - \frac{(\sum_{i=1}^n y_i)^2}{n})}} \quad (2.1)$$

where

n is the total number of sample points taken

x_i is the i th value of the specified dependent variable.

y_i is the i th measurement of the independent variable. (Energy)

This value varies between $+/-1$, with 1 indicating a perfect positive linear correlation, 0 indicating no correlation at all, and -1 indicating a perfect negative correlation. Ideally, all independent variables should be uncorrelated with each other, so

their effect on energy can be evaluated independently, and show a strong positive correlation with energy. Note that if the energy does not show a strong correlation with a model term, it will very likely be omitted from the final model. This would suggest that the model term does not contribute to power consumption, more data points are needed, or the measurements were not accurate enough to reveal the dependency.

2.2.4 Significance 1-tailed

Since the object is to find only positive correlations between the dependent and independent variables, a single tailed significance test is used. Table D.11 is a typical example. The data is presented in matrix form, so that it may be seen whether the dependent variables are statistically significant predictors of each other. The entire table has been provided for completeness, but usually only the first column is of interest. The first column shows the probability each model term does not affect energy, when taken in isolation. The table lists α values for dependencies between each variable, where $(1 - \alpha)$ is the probability the two terms are correlated. To improve readability, only 3 significant digits are printed, so very small values are shown as 0. An entry of 0 or a very small number indicates that the variables in the associated row and column affect each other to a very high degree of probability.

Ideally the values should be zero or very small for the first row and column, indicating that all model terms are statistically significant predictors of energy. Large values are preferable for the other entries, which indicates the dependent terms do not affect each other. In practice there will be many low or 0 entries, since some interdependencies are unavoidable, given a large number of sample points.

The t statistic is first calculated from the correlation[1]:

$$t = r \frac{\sqrt{n-2}}{\sqrt{1-r^2}} \quad (2.2)$$

This result is used to compute the significance table, by integrating over the Student's t-distribution probability distribution function (PDF), with $n - 2$ degrees of freedom, where n is the total number of sample points. The t distribution PDF with

k degrees of freedom can be described by the following equation[43]:

$$t_k(x) = \frac{\Gamma(\frac{k+1}{2})}{\sqrt{k\pi}\Gamma(\frac{k}{2})(1 + \frac{x^2}{k})^{(k+1)/2}} \quad (2.3)$$

With the Γ function defined as[42]:

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \quad (2.4)$$

The significance table entries are calculated as follows:

$$\alpha_{1-tailed} = \int_t^\infty t_{n-2}(x) dx \quad (2.5)$$

2.2.5 Number of data points

The number of data points for each variable is given in this table. The values are presented in matrix format, listing the number of data points that can be used for calculating interactions between each term. It is not particularly interesting or useful to look at, since the entries are always the same for these experiments. Table D.13 is a typical example. This value is always used for n, when computing the degrees of freedom for interdependency statistics. Note that it is possible to analyze the data by treating variables which were not varied in test cases as missing. It would be reasonable to treat *DWE*, *DWI*, ΔDWE and ΔDWI as missing in tests designed to exercise reads, although this was not done. The reverse case is also true. Likewise, *DWE*, ΔDWE , *DRE*, and ΔDRE can optionally be considered as missing cases for internal memory tests. This would raise the α values for the affected variables, (since there are fewer data points to work with) and might possibly uncover some subtle dependencies that were masked by more significant factors.

2.2.6 Variables entered or removed

At each iteration, one term is added to, or removed from the model. It is important for understanding how the results were generated, as it shows how the energy consumption model evolved with each iteration. Terms are dropped in the next step if its 1-tailed F statistic value falls below 90%. Terms are added for F values indicating a significance of more than 95%. Table D.15 shows a typical example of

this. The F, or Fisher statistic value is defined in terms of the model and residual mean square as[16][41]:

$$F = \frac{MS_m}{MS_r} \quad (2.6)$$

where

$$MS_m = \frac{SS_m}{df_m} \quad (2.7)$$

$$MS_r = \frac{SS_r}{df_r} \quad (2.8)$$

$$df_m = p \quad (2.9)$$

$$df_r = n - p - 1 \quad (2.10)$$

and

n is the number of sample points taken

p is the number of dependent (predictor) variables.

The sum of squares terms are defined as:

$$SS_m = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 \quad (2.11)$$

$$SS_r = \sum_{i=1}^n (y_i - \hat{y})^2 \quad (2.12)$$

where

$\bar{y}$ is the arithmetic mean of all y (energy) values

$\hat{y}$ is the value of y predicted by the model.

Further references to the F distribution may be found in [41], and [16].

2.2.7 Model Summary

This table contains many useful diagnostics to show how the model improved with each iteration. Table D.16 is a typical example. The first column lists the iteration number for each model. The second column correlation coefficient R , which is

defined as:

$$R = \frac{COV(x, y)}{\sigma_x \sigma_y} \quad (2.13)$$

where,

$COV(x, y)$ is the covariance between the independent variable (x) and the dependent variable (y).

σ_x is the standard deviation of the independent variable

σ_y is the standard deviation of the dependent variable

R will vary between +/-1, with 1 indicating a perfect fit. The R value should increase with each iteration, and should be close to 1 when the model is complete. The third column lists the residual sum of squares, R^2 . The residual sum of squares, (sometimes known as the error sum of squares) R^2 is defined as[18]:

$$R^2 = \frac{SS_t - SS_r}{SS_t} \quad (2.14)$$

where SS_r is defined in equation 2.12, and SS_t is:

$$SS_t = \sum_{i=1}^n (y_i - \bar{y})^2 \quad (2.15)$$

The next column is the adjusted R^2 , which allows for the degrees of freedom associated with the sum of squares. The adjusted R^2 value is defined as[18]:

$$R_{adj}^2 = 1 - (1 - R^2) \left(\frac{n-1}{n-p} \right) \quad (2.16)$$

where,

R^2 is the non-adjusted residual sum of squares, as defined in equation 2.14.

n is the total number of sample points

p is number of dependent (predictor) variables used to develop the model.

The value of R^2 will always increase with sample size. The adjusted R^2 value is generally considered a more accurate representation of fit, since it compensates

for the degrees of freedom of the sum of squares. If the adjusted R^2 differs significantly from R^2 , this indicates that the dependent variable is not well defined by the independent variables[28]. In this case, a large deviation would suggest that the proposed model is poor or incomplete, and that there are some large, unknown factors that significantly contribute to the energy consumption, yet are missing from the model.

The remaining columns show how the model changes from one iteration to the next. There should be progressively less change each time a variable is added or removed, since model terms are added in order of decreasing significance.

The standard error of the estimate (SEE) gives the expected error in the predicted value of the dependent variable, in this case energy. The SEE estimate is given by[6]:

$$SEE = \sigma_{energy} \sqrt{1 - R^2} \quad (2.17)$$

where

σ_{energy} is the standard deviation of the energy readings

R^2 is given in equation 2.16.

The SEE value should value should drop as more terms are added, and the model is refined. If all measurement were perfect with unlimited accuracy, and the model described the physical system exactly, then this term would be zero. In reality, measurements have limited accuracy, and there is random noise in the measurement that cannot be totally filtered out. There may well be other dependencies, or higher order effects not accounted for in the model. For instance, what if the relationship between TDRE and energy had a significant $O(x^2)$ or an $O(x^3)$ term? This would not be accounted for in the model, and would show up here as a prediction error.

The R^2 change column shows how the R^2 value is affected by each change to the model. This value should decrease down the table, since decreasingly significant terms are added with each pass.

df1 and df2 are the degrees of freedom associated with the independent and dependent variable. For n samples with p predictors, $df1 = p$, and $df2 = n - p - 1$. In this table, the F statistic is computed for each added or deleted model term individually, one at a time. Hence, the p value is always 1.

The PDF for the F statistic is[41]:

$$f_{n,m}(x) = \frac{\Gamma(\frac{n+m}{2})n^{n/2}m^{m/2}}{\Gamma(\frac{n}{2})\Gamma(\frac{m}{2})} \frac{x^{n/2-1}}{(m+nx)^{(n+m)/2}} \quad (2.18)$$

where the Γ function was defined in equation 2.4.

The last column in the model summary contains the F statistic significance factor. This indicates the probability of error at each iteration, as terms are added or deleted. When terms are added it indicates the probability that the newly added term is not a predictor of energy. For iterations that remove entries the probability that the removed variable was significant in predicting energy is given instead. The significance factor can be interpreted as the probability of error.

When terms are deleted from the model, the probability of error is high, because the deleted term may still be a significant predictor of energy. It just isn't significant enough to be retained in the model at the chosen confidence level.

The significance factor is found by integrating the F distribution PDF from the F statistic value computed in the F change column to ∞ :

$$Sig = \int_F^{\infty} f_{p,(n-p)}(x)dx \quad (2.19)$$

2.2.8 ANOVA

The Analysis of Variance (ANOVA) table provides additional details about each model iteration. Table D.17 is a typical example. The first column contains the iteration number, as before. The second column contains the model, residual, and total sum of squares, as defined in equations 2.11, 2.12 and 2.15. The sum of squares are not particularly useful or informative. They are normally computed as an intermediate step to computing mean square, which in turn is used to derive the F statistic data.

The next column contains the degrees of freedom for each statistic. The total degrees of freedom will always equal the number of samples. The model and residual degrees of freedom will sum to the total. Note that in this table, the model degrees of freedom is now p , and is equal to the total number of dependent variables. This table shows the statistics for the entire multiple dependency model as it is built, instead of examining each term separately.

The mean squares are computed by dividing the sum of squares by the degrees of freedom.

The second last column contains the F statistic, as defined by equation 2.6 for each model. It should be noted that even though the mean square of the residuals are dropping, the F statistic may drop as well, because the degrees of freedom rise when more model terms are added.

The last column reveals the statistical probability of error for the model as it is built, in accordance with equation 2.19. Many table entries may show zero, but it will in practice be a very small positive number for large values of F. A small number indicates that there is a very high confidence that the model is statistically significant in predicting the resultant energy consumption.

2.2.9 Coefficients part A

This is the most important table that is produced by the regression, as it contains the coefficients, or weighting factors for each model term. Statistics regarding each coefficient are generated at each iteration. The coefficients table must be split apart, because it is too wide to fit on one page. Table D.18 is a typical example. It gives a very detailed breakdown as to the regression weighting and statistical significance of each dependent variable. This is done for all model iterations during the stepwise regression. This is by far the most important and useful table, as it contains the coefficients that were experimentally derived for each dependent variable in equation 4.1. The coefficients for all excluded variables are zero, since they do not appear in the model.

The iteration number is given in the first column, of both coefficient table halves. As with the previous tables, it shows how the model was refined from one iteration to the next.

The second column lists all dependent variables that were used to build each model iteration. The constant term is listed first. This should correspond to the per-loop energy consumption when all other variables are zero. For a linear regression with only one dependent variable, this would be the y intercept.

The values given in the third column are the most important of all. These are the derived terms for equation 4.1. The coefficients will be smaller for less important terms, showing a very weak Pearson correlation. The $\hat{B}$ values will have the same

sign as the Pearson coefficient. As mentioned in section 2.2.3, a negative dependency would not be expected according to the model, and would likely be explained by an unwanted strong correlation between two dependent variables.

The values of B_o to B_p are computed by forming an augmented matrix X , with all 1's in the first row, and the measurements for all p dependent variables in subsequent rows. A vector of $(p + 1)$ elements is required, because there are p dependent variables, plus a constant term. The observed energy readings are collected in a 1 by n matrix Y . The coefficient matrix $\hat{B}$ then becomes[31]:

$$\hat{B} = (XX)^{-1}XY \quad (2.20)$$

The standard error column lists the standard error for each term. Using the augmented matrix X as before, it is possible to compute a matrix $\hat{B}$ with the variances of each predictor along the diagonal, and covariance terms on the off-diagonal elements[23]:

$$Cov(\hat{B}) = MS_r^2(X^T X)^{-1} \quad (2.21)$$

The standard error vector can be extracted from this matrix by performing the following operation:

$$SE(\hat{B}) = \sqrt{Cov(\hat{B})IJ} \quad (2.22)$$

where

$Cov(\hat{B})$ is the 1 by $(p + 1)$ covariance vector from equation 2.21.

I is the $(p + 1)$ by $(p + 1)$ identity matrix.

J is a 1 by $(p + 1)$ vector containing all 1 entries.

$SE(\hat{B})$ is the 1 by $(p + 1)$ vector containing the standard error for each coefficient.

p is number of dependent (predictor) variables used to develop the model.

The Standardized coefficients β , are often used to compare the relative sensitivity of the output with respect to each variable. The Y intercept is removed, and the independent variable is scaled to have a standardized normal distribution with mean 0, and variance of 1. It is then possible to see the relative contribution of each term, to the overall power consumption. Standardized coefficients makes it possible

to compare the relative importance of the dependency terms between different processors. They may be computed from the unnormalized coefficients, and standard deviations of the independent and dependent variables, as follows[25][44]:

$$\beta_i = \hat{B}_i \frac{\sigma_y}{\sigma_x} \quad (2.23)$$

The t statistic indicates the corresponding point on the $t_n(0, 1)$ distribution curve for each coefficient. Higher t scores means a greater probability that the corresponding parameter is significant in determining energy consumption. The t score is the criteria used to drop or include dependent variables in the model. The formula for the t-statistic is simply:

$$t_i = \frac{\hat{B}_i}{SE(\hat{B})} \quad (2.24)$$

The values in the significance column may be computed by integrating the Student's t function PDF with n degrees of freedom, over the region between the t score given in the previous column, and ∞ :

$$Sig = \int_t^{\infty} t_n(x) dx \quad (2.25)$$

where the Student's t distribution PDF was given in equation 2.3. As with significance column in previous tables, this indicates the probability that the given model term is not significant, and does not affect the energy.

2.2.10 Coefficients part B

Table D.19 is a typical example. The first two columns of the Coefficients table part B, also contain the iteration number and coefficient name. This makes it easier to cross reference data from part A of the table. The 95% confidence interval for $\hat{B}$ give an upper and lower bound for each coefficient term. This illustrates the absolute, un-scaled uncertainty levels associated with each term. It is calculated by:

$$\hat{B}_{upper} = \hat{B} - (SE(\hat{B})t_n^{-1}(0.95)) \quad (2.26)$$

$$\hat{B}_{lower} = \hat{B} + (SE(\hat{B})t_n^{-1}(0.95)) \quad (2.27)$$

$t_n^{-1}(0.95)$ refers to the inverse of the Student's t distribution, whose PDF was given in equation 2.3. For large values of n , $t_n^{-1}(0.95) \simeq 1.650$

The zero order correlations are useful to show how strongly the output is correlated with each dependent variable. These are correlations that are calculated for bivariate relationships[5]. The values are identical to those found for the corresponding dependent variables in the first column of the Pearson correlation matrix, as defined by equation 2.1.

The partial correlations indicate the strength of the dependency between the energy and the given model variable, when the common effect of other variables are removed[11]. This may be helpful if strong dependencies were found between dependent variables, to see if they still affect the energy consumed when this is accounted for. A formula for calculating partial correlation coefficient between X and Z with Y held constant is[9]:

$$r_{XZ|Y} = \frac{r_{XZ} - r_{XY}r_{YZ}}{\sqrt{(1 - r_{XY}^2)(1 - r_{YZ}^2)}} \quad (2.28)$$

Note that if there are multiple Y 's, then the formula may be applied iteratively. For example computation of $r_{XZ|Y_0Y_1}$, first compute $r_{XZ|Y_0}$, then let $r_{XZ|Y_0}$ replace r_{XZ} and replace Y_0 with Y_1 . The partial correlation will be the same as the β term, if the variable is completely independent of (orthogonal to) all others. If there is a statistical correlation with other dependent variables, then the partial correlation may differ, or even change sign. This is most likely to happen with small values of β , and a large t value, indicating a very poor correlation with low statistical significance. If this is not the case, then it could mean one of several things:

1. More experimental points are needed, to change one of the dependent variable independently of the others, where the strong correlation was found
2. Two or more dependent terms inherently have a strong correlation, resulting in causal interference[11].
3. There were undiscovered mistakes (systematic errors) made in doing some of the measurements
4. The experiment has uncovered an effect that is not explained by the model

The difference between the squared semi-partial (part) and partial correlation is that in part correlation, the contribution of variance by each controlled variable is removed from the independent variables. In partial correlation, the variance contributed by the controlled variable(s) are removed from both the dependent and independent terms[11]. When the corresponding term is removed from the model equation, R^2 will decrease by exactly this amount.

The two colinearity statistics columns are helpful to check if dependent variables are highly correlated with each other. A common rule of thumb states that the variance inflation factor (VIF) should be less than 10, and/or the tolerance should be more than 0.1[10]. The tolerance is simply the reciprocal of the VIF. To compute the VIF for each independent variable, a regression is done with the selected term replacing energy as the dependent variable, and the residual sum of squares computed. The VIF can then be calculated via the following formula[4]:

$$VIF_i = \frac{1}{1 - R_i^2} \quad (2.29)$$

where R_i^2 is the residual sum of squares for using the i th parameter as the dependent term. The VIF and tolerance are mainly useful as diagnostic tools, to verify that the stepwise regression results makes sense, or to help discover why it did not.

2.2.11 Excluded variables

Statistics for all variables that were excluded from each model iteration are found here. Table D.20 is a typical example of this. This is mainly useful for post regression analysis, to determine why a particular candidate model term was not included. It is most instructive to examine the last row of this table, to see what terms were excluded when the regression was complete, and try to discover why they were discarded.

The first column of the excluded variables table lists the stepwise regression model iteration number, as with the earlier tables. The second column contains all dependent variables that were omitted from the model, on each pass.

The standardized coefficients (β) values for each rejected term are given in column 3, as defined by equation 2.23. The values supplied are what the standardized coefficient for the term would be, if it had been included in the result at this stage.

The entries for the last row of this table will normally be small, (less than 0.05) unless they were highly correlated with another model term, or the variance associated with it was high. A very small β value indicates that varying the parameter has minimal effect on the overall energy consumption.

The t statistic and significance columns are generated in the same manner as with the included coefficients, using equations 2.2 and 2.25. Since the criteria for inclusion is a 95% confidence level, the value in the significance column must be less than 0.05 for the term to be included.

Partial correlations for discarded terms are also computed in the same way as they are for dependents that were used in the model. Normally these will also show a very small correlation with energy, unless it was strongly correlated with another variable, that was included in the model. As described earlier, if the partial correlation is substantially different than the β value given, it indicates the variable is strongly correlated with at least one other term. One needs to look at the coefficient correlations table, described in section 2.2.12 to see what other terms it is correlated with.

The VIF and tolerance columns are computed as described previously. A brief scan of these results will indicate if there was any strong interdependencies found with terms that were omitted from the regression. As with the included terms, the VIF should be less than 10 for a stable regression.

The minimum tolerance value is the lowest tolerance value of all dependent variables in the regression, assuming the excluded variable was added back into the model.

2.2.12 Coefficient Correlations

This table shows the covariance and correlation between all included terms, at each iteration. Table D.21 is a typical example. The correlation is more useful, as it normalizes the covariance to ± 1 . The correlation may be positive or negative, but ideally should all be much less than 1 for all non-diagonal values. The correlation on the diagonals will all be 1, since all variables are perfectly correlated with themselves. The formulae for covariance and correlation are[2][3]:

$$Cov(X, Y) = \frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y}) \quad (2.30)$$

$$Cor(X, Y) = \frac{Cov(X, Y)}{\sigma_X \sigma_Y} \quad (2.31)$$

2.2.13 Colinearity Diagnostics

This table helps to expose and explore problems with collinear dependent variables. Table D.22 is a typical example. The dependent terms, (including the constant) are converted into orthogonal determinants of Y . Column 3 lists the eigenvalues of this matrix. Any eigenvalues which are close to 0 indicates that there is a dimension that does not significantly contribute to the variance of the energy. Multiple eigenvalues close to 0 indicates that the cross product matrix is poorly conditioned, which indicates multiple collinear terms.

The condition index summarizes the findings, and a common rule of thumb is that a condition index over 15 indicates a possible multicollinearity problem. If a factor has a high condition index, one looks in the variance proportions column to see if it accounts for a sizable proportion of variance in two or more variables. If it does, multicollinearity is a problem[12]. The condition index is defined as:

$$k_i = \sqrt{\frac{\lambda_{max}}{\lambda_{min}}} \quad (2.32)$$

Issues of colinearity can be dealt with by dropping one of the offending terms from consideration, or doing more experiments to generate additional data points which separate the terms.

For some regressions, the variance properties of the colinearity diagnostics table had to be split out, because there were too many terms. For each model iteration, the total model variance is scaled to 1, so that the relative contribution of each dimension can be seen. The contribution of each independent variable (plus the constant term) to every dimension can be seen as well, by looking down the associated column. The variances in each column will add to 1, for each model iteration.

Chapter 3

State of the Art in Power Optimization Studies

There has been a substantial amount of prior work in this field. For the most part, they can be lumped into the three main categories described in the previous chapter. The one exception found to this was a very broad survey paper[40]. This briefly describes many general techniques to minimize power consumption, from transistor level to operating system scheduling. Use of gray and bus invert codes on busses to minimize hamming distance is mentioned, as well as using sign/magnitude representation as opposed to 2's complement, for data which is temporally correlated. Compiler optimizations are also introduced in a very general way.

3.1 Hardware level modeling

Chaitali Chakrabarti and Dinesh Gaitonde modeled an entire CPU at gate-level for the purposes of a power consumption study, using gate level modeling tools to estimate power consumption of a 68HC11 microcontroller [7]. Their power consumption model was checked by comparing the predicted power consumption estimates against their measured value, while running sample programs. They concluded that data correlation did not affect the energy consumption of most instruction, and that the same instruction incurs a different cost, depending on the gate level implementation. This research was intentionally based on a simplified processor where the effects of pipeline stalls, cache misses, etc. do not come in to play.

3.2 Functional block modeling

Researchers in this area employed a variety of diverse techniques which attempted to model functional blocks and examine the effect of individual instructions on overall power consumption.

The use of instruction scheduling algorithms to minimize the hamming distance between instruction groupings was investigated by [8], as a special case of the constrained traveling salesman problem. A number of instruction ordering algorithms were compared for some small, industry standard benchmark SuperSPARC programs.

Cheng-Ta Hsieh, Massoud Pedram, Gaurav Mehta and Fred Rastgar[14], showed how profiling tools can be applied to Spec95 programs running on a super-scalar pipelined processor. The power consumption model is derived by measuring power dissipation in known benchmarks, and extracting bit level program statistics to make predictions for other programs. Extension to more complex architectures, such as VLIW was left for future study.

A power consumption model was developed by [21], based on hamming distance between bit transitions in the instruction field by running a series of test programs, and doing a regression analysis. They were able to resolve energy consumption for instructions down to the pipeline stage level in an ARM7 processor. They claimed they were able to model energy consumption for an arbitrary instruction grouping to an accuracy of 2.5%. This work used experimental measurement and regression analysis to accurately model power consumption of all data processing instructions within an ARM7 processor. The effect of instruction fetch address, bit encoding, data dependencies and pipe stage dependent effects were considered, although the study did not investigate the effect of pipeline stalls and I/O.

Power consumption and instruction scheduling techniques on a Fujitsu DSP were examined by [19]. The base power consumption of each instruction was measured by putting the processor in an infinite loop. Pairs of instructions were looped together to estimate average overhead costs for transitioning between different types of instructions. Thus, a crude instruction level power consumption model was derived, based on a fixed instruction cost, plus the average cost of transitioning to a new instruction. No data dependencies were included with the study. The effect of swapping operands where one operand is constant, (common in filter algorithms)

was explored, although the results were very specific to the particular DSP under study, and would not be applicable to a retargetable compiler.

The same author, [20] later expanded on the previous work. In this study, it was observed that non adjacent instructions can affect the ALU data-path in such a manner as to raise power consumption, something that the previous work did not consider. Register transfer list (RTL) transitions were used to model power consumption. With the particular DSP under study, it was determined that the hardware multiplier used was a major source of power dissipation. The effect of data dependencies was noted, but not modeled. Instead, average values were used. Likewise, an average power cost was assigned to each instruction class for simplicity. Significant power optimization was shown to be possible on the particular DSP under study by use of architectural features such as dual memory access, instruction packing, and operand swapping.

The effect of software on overall power consumption was discussed by [26]. This was more of a white paper surveying earlier research, as opposed to an independent study. A number of points were touched upon, including memory access penalties, system bus capacitance, ALU costs, and miscellaneous overhead such as clock distribution logic. He discusses a number of analysis techniques ranging from gate simulations to instruction level power analysis. This paper is an excellent starting point for any reader trying to familiarize themselves with software power optimization studies.

Jeffrey Russel [27], showed a link between minimal execution time, and minimal power consumption for two 32 bit RISC processors. Power consumption was measured by use of a precision resistor placed in series with V_{cc} for the processor, connected to an oscilloscope.

An energy consumption profiling tool was presented by [30], that was used to implement an MPEG 3 audio decoder. Techniques such as profiling execution time to determine where algorithms need to be hand optimized, and avoiding use of floating point operations, (especially using software emulation) were discussed.

Vivek Tiwari looked at power consumption of individual 80486 instructions were measured in [36], along with inter instruction effects and data dependency estimates. The power consumption effects of the DRAM memory during cache misses was also examined. His work was then applied to a second processor, based on the SPARC architecture. This was subsequently expanded on in [37]. This time

he looked at the 486DX2 architecture in still greater detail. CPU power consumption was estimated by isolating power to the CPU from the rest of the system. The CPU was put into an infinite loop with an integrating ammeter attached. Unfortunately, this technique is not practical or even possible with many modern CPUs. The SPARC processor was tested using a processor chip connected to a hardware test vector generator. Base power consumption as well as instruction interaction effects were studied in detail. It was observed in all three processors that external memory access is costly in terms of power. The power reduction effects of caches were also examined, and found to be significant. Reordering instructions to reduce power consumption was studied, as well as the effect of exploiting architecture specific parallelism.

Profiling techniques were described in [38], to identify instruction sequences that require significant amounts of current. This was done in order to limit the amount of power dissipation that can occur in any given cycle. Their results suggested performance must often be scaled back in order to minimize power consumption spikes. The chief benefit of this would be to prevent momentary current spikes from causing power distribution problems, which result in erratic operation. Of course, similar benefits may be had by selectively reducing the CPU clock frequency, introducing memory wait states, or adding `nop` cycles. The primary benefits of the software detuning techniques described here, is to reduce current spikes caused by software looping on energy hungry instruction sequences.

Graphing techniques for improving loop scheduling algorithms were examined by [45], as a means to optimize power consumption. Weighted data flow graphs were used to model iterative loop algorithms, and adjusted in an attempt to reduce the number of state transitions necessary. A number of DSP filter algorithms were used as case studies.

3.3 System level modeling

The following researchers described system level modeling techniques, where the effects of individual instructions were not considered. The approaches used are not directly relevant to the approach taken in this work, but are still of related interest. Some of the techniques described (especially cache hit ratio and dynamic memory modeling) could be added to the functional block model derived in this paper.

Inki Hong, Gang Qu, Miodrag Potkonjak and Mani B. Srivastava developed techniques for dynamically varying supply voltage on the fly within a real time system were evaluated in [13]. The Vcc slew rate was limited to 1 volt per 10,000 instructions, so this technique cannot be applied on an instruction by instruction basis. Instead, supply voltage was scheduled by task with known deadlines.

Milind Kamble and Kanad Ghose examined the accuracy of multilevel cache power consumption estimates using a specialized simulator[15]. The focus of their work was in trying to accurately simulate the cache power consumption using average hit rate statistics.

Yanbing Li and Jörg Henkel looked at power consumption of embedded systems from a combined hardware and software perspective[22]. A sophisticated transistor level split instruction/data cache model was added to a DRAM memory model. Software instruction level power estimates were made by assuming a fixed base power consumption value for each individual instruction. The effects of varying data and instruction cache size on power consumption were evaluated.

A methodology for cycle-accurate simulation of energy consumption was developed by Tajana Šimunić, Luca Benini, and Giovanni De Micheli[29], for an ARM processor. Power consumption models of a two layer cache were developed. This model did not detail power consumption variations between instructions. The processor was considered to be either active or idle waiting for the cache to be filled. Despite this, they claim that their tool was able to predict power consumption to within a 5% accuracy.

The first serious attempt to model instruction level power consumption within an embedded system was done by Vivek Tiwari, Sharad Malik, and Andrew Wolfe[35]. The measurements were done on the CPU in tandem with the DRAM subsystem - no attempt was made to separate the power consumption of the CPU from the memory subsystem, although the effect of pipeline stalls and instruction cache misses were estimated. No investigation was made to model or determine the effect of data and address line transitions. Their work revolved around the 486DX2 processor.

Power dissipation from a hardware/software codesign perspective were studied by N. Vijaykrishnan, M. Kandemir, M. J. Irwin, H.S. Kim, and W. Ye[39]. Each major functional unit within the system is modelled with respect to power consumption. Tradoffs of compiler optimizations such as loop unrolling algorithms and register reuse strategies were discussed, but not in great detail. The primary focus of the

work involved the memory and cache subsystem.

Chapter 4

Modeling

In order to be useful, the generic power consumption model is developed with the following objectives in mind:

- Accurately model power consumption
- Applicable to any system running arbitrary software
- Simple to compute

Unfortunately these objectives tend to be mutually exclusive. In order to create a generic model, it is necessary to ignore the architectural details of particular targets wherever possible, and focus on attributes they have in common.

4.1 Modeling power consumption

Developing a power consumption model of an arbitrary system without resorting to gate level modeling requires a few assumptions:

The first assumption is that circuits driving inactive functional blocks and busses consume a small, fixed amount of current, once they have been put into a given state. Of course there are millions of individual components on a DSP, and the combined effect of their individual leakage currents may not be negligible. It is assumed that it can be modeled as a small, constant value so long as there are no state changes occurring. In practice, there may be slight parasitic effects from nearby active circuits, as well as differences between components due to normal manufacturing tolerances. This is treated as an unknown constant term that must be experimentally derived

for each processor. This constant can be based on the CPU halted, but clock running state with zero bus activity. Some DSPs intended for ultra low power applications have a stop state, where the system clock is completely inactive. Power consumption in this state can be obtained directly from the component data sheet, and does not require any detailed modeling.

For the purposes of this model, it is assumed that the majority of the power dissipation within the system is consumed by changing the state of internal and external address and data busses. In order to maximize performance in the smallest possible die area, related blocks of logic are placed as closely together as possible. But busses must be used to carry signals between on chip blocks, or the outside world. Since they must carry information a correspondingly greater distance, they have a higher effective capacitance. Switching them at the same speed requires higher currents.

Figure 4.1: Bus capacitance power consumption model

State A, shows the bus in a neutral (high impedance) state. In figure 4.1 B, a logic 1 state is asserted (assuming positive logic) by turning on the upper transistor, and allowing current to flow onto the bus. State C shows the state with the bus fully charged; no current needs to flow in order to maintain the logic 1 state. State D shows what happens when a logic 0 is subsequently written out to the bus. The lower transistor is turned on, and the bus capacitance must be discharged by draining current away. Note that current only flows in states B and D, when the voltage on the bus transitions. In practice there will be a small amount of leakage, as well as noise, crosstalk and other parasitic effects that will cause current to flow.

Figure 4.2: Bus cycle model - return to neutral vs. direct switching

A

B

Power consumption data dependency is affected by whether or not the bus reverts to the neutral state between cycles. In figure 4.2 A, the bus reverts to the neutral state before new data is transmitted across it. In this case, the power consumed per cycle does not depend on the data transmitted in the previous cycle. Energy consumption is directly proportional to the number of active bus lines. If the bus does not return to the high impedance state as in 4.2 B, then no energy is required to maintain the same logic level. Energy will be required if the logic level is changed from a high to low, or vice versa. In this case, there is a strong dependency on the number of bit level transitions between adjacent bus cycles. If the bus partially returns to the neutral state between cycles, then there will be an dependency on both the total number of bits and the bit transitions.

It should also be noted that different types of busses will behave differently, and must be treated as separate in the power consumption model. Busses that run off chip would normally be expected to have higher capacitance, and thus requiring more current to drive. It may be difficult or impossible to determine exactly how a given bus will behave, so the empirical model assumes both a dependency term for the number of active bits on a bus, as well as the number of bit transitions.

Since the performance penalties for external memory access are higher, most real world applications avoid running code out of external memory. If the application does not fit, then effort is normally made to ensure the most CPU intensive and time critical routines reside on chip. In order to limit the complexity of the three

DSP models, tests involving code fetched from off chip memory or ROM were not done, although the generic bus model supports it. These cases are of lesser interest, and have been left as a possible avenue for future work in this area.

4.2 Choosing a term of reference

Before a model can be developed, there is one more preliminary step - selecting units of reference. Normally, power consumption is specified in terms of joules per second, or watts. But when attempting to measure energy consumed by a particular sequence of machine instructions, it becomes more natural to measure energy consumed per CPU clock cycle. Compilers can easily count clock cycles when deciding between two candidate optimization paths, but without knowledge of the actual clock frequency, cannot correlate it to real time.

There is another weighting factor to be considered. How should instructions be weighted that require different numbers of clock cycles to compute? Instructions with more complex addressing modes typically fall into this category. Even simple instructions that miss the cache, or directly access external memory normally incur one or more wait states. This factor must be taken into account.

In a typical embedded system where power is a concern, the DSP is typically called upon to perform a specific task within a given period of time. If the task can be done in a smaller number of machine cycles, the DSP can either be put into a halted state which consumes negligible power, or run at a lower clock frequency to achieve the same benefit. Therefore, it makes more sense to weight power consumption in terms of clock cycles, rather than machine instructions. The measured power consumed while running each CPU instruction must be multiplied by the number of clock cycles required. This ensures that the optimization will apply the correct penalty weighting against slow instructions, since the DSP will spend more time running them (and more time actively consuming power) in order to complete the desired task.

4.3 Power consumption model

Once a bus state transition-based energy consumption model has been established, which busses are to be considered? A close examination of the DSPs architectural

block diagram is helpful here. It does not do any harm to consider more busses than may be necessary when proposing a candidate model. This does tend to make the up front experimental analysis a bit more difficult, but any factors which do not provide significant contribution to the power consumption will be subsequently removed through statistical reduction.

4.3.1 Creating a basic model

Consider a hypothetical generic DSP with some internal memory, and a simple external address and data bus, as shown in figure 4.3.

Figure 4.3: Simplified generic DSP

Even with a unified Von Neumann memory model, at some point the program and data must be routed to separate busses, so they are modeled separately. Each bus makes a different contribution to the overall power consumption, so each has a

different weighting factor. Busses which are empirically found to have a negligible effect on the overall power measurement are removed from the model.

This assumes a-priori knowledge as to what the previous state of the bus really is. What if it was previously in the high impedance (tri state) mode? In this case, there will be current flow regardless whether a 1 or a 0 bit is written. Since it is a trivial matter to count the total number of bits in a bus, another weighting factor may be added that counts the total number of bits being driven on a particular bus. If it turns out this is not a dependency, then it should vanish when the model is analyzed, as described in subsection 4.4.

Read and write cycles to memory are modeled separately, since they may require different amounts of energy. For instance, if the output buffers on the CPU were weaker than the output buffers on the memory, then a read cycle may consume more power than a write. This would certainly be true if the CPU employed a write through cache, because every write cycle would go to main memory, while most reads would hit the data cache.

When a program instruction is fetched, an address must be broadcast over the program space address bus. The data (representing a program instruction) is then fetched from that location. If the instruction requires data to be fetch from or written to memory, then the number of bit transitions over the bus used to broadcast the address, as well as the number of bit transitions within the data bus itself should be counted. It typically requires more energy to write data to a bus than it does to read from it, so writes and reads are assigned separate weightings. The ALU data bus must be considered as well. The energy model now becomes:

$$\begin{aligned}
 E_{instruction} = N * (& K + K_{\Delta PAI} \Delta PAI + K_{TPD} TPD + K_{\Delta PDI} \Delta PDI + \\
 & K_{\Delta PAE} \Delta PAE + K_{\Delta PDE} \Delta PDE + K_{\Delta DAI} \Delta DAI + \\
 & K_{TDWI} TDWI + K_{\Delta DWI} \Delta DWI + K_{TDRI} TDRI + \\
 & K_{\Delta DRI} \Delta DRI + K_{\Delta DAE} \Delta DAE + K_{TDWE} TDWE + \\
 & K_{\Delta DWE} \Delta DWE + K_{TDRE} TDRE + K_{\Delta DRE} \Delta DRE + \\
 & K_{TAS} TAS + K_{\Delta AS} \Delta AS + K_{TAD} TAD)
 \end{aligned} \tag{4.1}$$

where:

N is the number of clock cycles required for the instruction to complete.

K is the constant (baseline) power consumption of the CPU.

$K_{\Delta PAI}$ is the weighting factor for energy contribution due to ΔPAI .

ΔPAI is the internal memory program space address bits that changed between the last two instructions.

K_{TPD} is the weighting factor for energy contribution due to TPD .

TPD is the internal memory program space data bits that are used to represent the instruction.

$K_{\Delta PDI}$ is the weighting factor for energy contribution due to ΔPDI .

ΔPDI is the internal memory program space data bits that have changed between the last two instructions.

$K_{\Delta PAE}$ is the weighting factor for energy contribution due to ΔPAE .

ΔPAE is the external memory program space address bits that changed between the last two instructions.

$K_{\Delta PDE}$ is the weighting factor for energy contribution due to ΔPDE .

ΔPDE is the external memory program space data bits that have changed between the last two instructions.

$K_{\Delta DAI}$ is the weighting factor for energy contribution due to ΔDAI .

ΔDAI is the internal memory data space address bits that changed between the last two instructions.

K_{TDWI} is the weighting factor for energy contribution due to $TDWI$.

$TDWI$ is the internal memory data space data bits that have been written by the current instruction.

$K_{\Delta DWI}$ is the weighting factor for energy contribution due to ΔDWI .

ΔDWI is the internal memory data space data bits that have changed due to a write between the last two instructions.

K_{TDRI} is the weighting factor for energy contribution due to $TDRI$.

$TDRI$ Is the internal memory data space data bits that have been read by the current instruction.

$K_{\Delta DRI}$ is the weighting factor for energy contribution due to ΔDRI .

ΔDRI is the internal memory data space data bits that have changed due to a read between the last two instructions.

$K_{\Delta DAE}$ is the weighting factor for energy contribution due to ΔDAE .

ΔDAE is the external memory data space address bits that changed between the last two instructions.

K_{TDWE} is the weighting factor for energy contribution due to $TDWE$.

$TDWE$ is the external memory data space data bits that have been written by the current instruction.

$K_{\Delta DWE}$ is the weighting factor for energy contribution due to ΔDWE .

ΔDWE is the external memory data space data bits that have changed due to a write between the last two instructions.

K_{TDRE} is the weighting factor for energy contribution due to $TDRE$.

$TDRE$ is the external memory data space data bits that have been read by the current instruction.

$K_{\Delta DRE}$ is the weighting factor for energy contribution due to ΔDRE .

ΔDRE is the external memory data space data bits that have changed due to a read between the last two instructions.

K_{TAS} is the weighting factor for energy contribution due to TAS .

TAS is the total ALU source bits that have been written by the current instruction.

$K_{\Delta AS}$ is the weighting factor for energy contribution due to ΔAS .

ΔAS is the ALU source bits that changed between the last two instructions.

K_{TAD} is the weighting factor for energy contribution due to TAD .

TAD is the ALU destination bits that have been driven by the current instruction.

$E_{instruction}$ is the energy in nanojoules, required to execute each instruction.

The program will be running in either internal or external memory, so either ΔPAI and ΔPDI , or ΔPAE and ΔPDE in formula 4.1 will be zero. For some CPUs, it may be possible to simultaneously fetch or write data from both internal and external memory busses at the same time. Read and write cycles to the same memory space within one instruction do not normally occur, unless atomic read/modify/write operations are supported.

As mentioned in section 4.1, test cases where code is fetched from external memory are omitted. After deleting these terms, the model now becomes:

$$\begin{aligned}
 E_{instruction} = N * (&K + K_{\Delta PA}\Delta PA + K_{TPD}TPD + K_{\Delta PD}\Delta PD + \\
 &K_{\Delta DAI}\Delta DAI + K_{\Delta DAE}\Delta DAE + K_{TDRI}TDRI + \\
 &K_{\Delta DRI}\Delta DRI + K_{TDWI}TDWI + K_{\Delta DWI}\Delta DWI + \\
 &K_{TDRE}TDRE + K_{\Delta DRE}\Delta DRE + K_{TDWE}TDWE + \\
 &K_{\Delta DWE}\Delta DWE + K_{\Delta DWE}\Delta DWE + K_{TDRE}TDRE + \\
 &K_{\Delta DRE}\Delta DRE + K_{TAS}TAS + K_{\Delta AS}\Delta AS + \\
 &K_{TAD}TAD + K_{\Delta AD}\Delta AD)
 \end{aligned} \tag{4.2}$$

where:

$K_{\Delta DA}$ replaces $K_{\Delta DAI}$ in equation 4.1 and is the weighting factor for energy contribution due to ΔDA .

ΔDA replaces ΔDAI in equation 4.1 and is the data space address bits that changed between the last two instructions.

$K_{\Delta PA}$ replaces $K_{\Delta PAI}$ in equation 4.1 and is the weighting factor for energy contribution due to ΔPA .

ΔPA is the program space address bits that changed between the last two instructions.

$K_{\Delta PD}$ replaces $K_{\Delta PDI}$ in equation 4.1 and is the weighting factor for energy contribution due to ΔPD .

ΔPD is the number of bits in the program store data space that changed between the last two instructions.

4.3.2 Basic model extensions

It is possible to extend the basic model, in order to model other CPU activity which affects power consumption. Most modern processors contain one or more level of cache, and have instruction pipelines, enabling them to operate at higher speeds. Many systems also contain special peripherals such as DMA controllers, or coprocessors. This must be accounted for, in order to create an accurate model.

4.3.2.1 Pipeline and cache effects

Most modern CPUs (including DSPs) employ an instruction pipeline to speed up executions. In normal straight line code execution, it is fairly easy to see that this does not affect the model in any way. An address is presented on the address bus, and an instruction is fetched. In subsequent cycles the instruction is executed, which may cause transitions on the data and ALU buses. The fact that transitions did not occur on the program address space, program data space, data or ALU buses at the same point of time is of no consequence, as far as power consumption is concerned. But what happens at a branch or loop point? Typically, the CPU will fetch one or more instructions past the branch instruction, before the branch decodes and executes. In this case, program space address and data bus transitions occurring as a result of instructions that are fetched after the branch must be considered. Sometimes the fetched instructions are discarded. If this is the case, then the program space bus transitions caused by the discarded instructions are the only factor to consider. If the post branch instructions are in fact executed, the full effects of the instruction on all other busses must be included in the model.

Caches will certainly complicate the model. The power consumption model changes quite drastically, depending whether or not a hit or miss occurs on cache.

Accurate power consumption models must determine whether or not a hit or miss occurs on the instruction and/or data caches. This can be modeled by treating the busses from the processor to the cache separate from the bus between the cache and main memory. (either internal or external memory)

After adding separate single level instruction and write through data caches, equation 4.2 now becomes:

$$\begin{aligned}
E_{instruction} = N * (&K + K_{\Delta PA_h} \Delta PA_h + K_{\Delta PA_m} \Delta PA_m + \\
&K_{TPD_h} TPD_h + K_{TPD_m} TPD_m + \\
&K_{\Delta PD_h} \Delta PD_h + K_{\Delta PD_m} \Delta PD_m + \\
&K_{\Delta DAI_h} \Delta DAI_h + K_{\Delta DAI_m} \Delta DAI_m + \\
&K_{\Delta DAE_h} \Delta DAE_h + K_{\Delta DAE_m} \Delta DAE_m + \\
&K_{TDRI_h} TDR_h + K_{TDRI_m} TDRI_m + \\
&K_{\Delta DR_h} \Delta DR_h + K_{\Delta DRI_m} \Delta DRI_m + \\
&K_{TDWI_h} TDWI_h + K_{TDWI_m} TDWI_m + \\
&K_{\Delta DWI_h} \Delta DWI_h + K_{\Delta DWI_m} \Delta DWI_m + \\
&+ K_{TDRE_m} TDRE_m + \\
&+ K_{\Delta DRE_m} \Delta DRE_m + \\
&K_{TDWE_h} TDWE_h + K_{TDWE_m} TDWE_m + \\
&K_{\Delta DWE_h} \Delta DWE_h + K_{\Delta DWE_m} \Delta DWE_m + \\
&K_{TDR_h} TDR_h + K_{TDRE_m} TDRE_m + \\
&K_{\Delta DR_h} \Delta DR_h + K_{\Delta DRE_m} \Delta DRE_m + \\
&K_{TAS} TAS + K_{\Delta AS} \Delta AS + \\
&K_{TAD} TAD + K_{TAD} TAD)
\end{aligned} \tag{4.3}$$

Here, the h and m subscript terms are used to denote power consumption during cache hit and miss cycles. Note that if a read cycle hits on cache, then it does not matter if the code or data fetch is from internal or external RAM space. Since a write through cache architecture posts data to both cache and physical memory, internal and external memory write cycles that hit on cache are treated separately.

Copy back data caches are much more difficult to model, but are comparatively rare with DSPs. Data is only written out to physical memory when the cache is cleared to accept new data, or it is explicitly forced. This is best seen with a comparative illustration.

Figure 4.4: Write Through Cache Read Hit

Figure 4.5: Write Through Cache Read Miss

These diagrams show a very simplified view of bus activity during cache read. If the data to be read from memory is found in cache, then there is no activity on the address or data busses running between main memory and the cache. If the data is not found, then both the CPU to cache and cache to main memory address and data busses are active. Read cycle behavior is identical for both copy back and write through caches.

Figure 4.6: Write Through Cache Write

Figure 4.7: Copy Back Cache Write

With a write through cache, data is always posted to main memory as well as the cache. Both the CPU to cache and cache to main memory address and data busses are active for every write cycle. But with a copy back architecture, data is only written out to main memory when it becomes necessary to flush data from the cache. This activity is shown as a dotted line between cache and main memory in figure 4.7. Note that the data and associated tag address being flushed out differs from the address and data being written in on the CPU to cache bus. The model must have knowledge of the associativity and replacement algorithm in the cache layer in order to predict what address and data pairs will be flushed, and how often this would be expected to occur.

Equation 4.3 is still used, except that a careful track must be kept of cache activity. The model must accurately predict if and when data is flushed out of cache. The DAI_m , DAE_m , $TDWI_m$, ΔDWI_m , $TDWE_m$, and ΔDWE_m terms are non-zero only when the cache is flushed. None of the DSPs studied included a data cache, so the modeling becomes much simpler.

Modeling CPUs with a complex, multilevel cache is possible, but can be cumbersome. Consider the three cases below:

Figure 4.8: Hit on L1 cache

Figure 4.9: Hit on L2 cache

Figure 4.10: Cache miss

In figure 4.8, the data was found in the L1 cache. Only the busses running between the CPU and L1 cache are active. In figure 4.9, the data was found in L2, but not L1. The busses between the CPU and L1 as well as the L1 to L2 busses are active. Figure 4.10 shows the case where both caches miss, and the data must be retrieved from main memory. In this case, there are three bus groups active: CPU to L1, L1 to L2, and L2 to main memory.

Since multilevel caches are normally implemented with a copy back architecture, determining exactly when write flush activity occurs can be quite difficult. The model in equation 4.3 can easily be expanded by using $h_1 \dots h_n$ terms to represent hit on the n th level of cache.

Automated machine code analysis tools are essential for modeling bus activity with complex architectures. If the model becomes unwieldily, it may help to remember that longer busses need a higher current drive than short busses. Since the L1 cache is the smallest, the effect of bus activity between the CPU and the first cache layer can probably be neglected if simplification is required.

4.3.2.2 Modeling branch prediction logic

Branch prediction logic was not used in any of the DSPs under study. But since it affects pre-fetch behavior which in turn affects bus activity, it is important to know in advance how the branch prediction logic will behave. If this is not practical, then the modeler can approximate by assuming backwards branches will be taken, while forward branches will not. For short looping code, the branch likely state will be set correctly after the first iteration.

4.3.2.3 Multiple masters

Although they are outside the scope of this research, some of the studied DSPs included on chip DMA controllers. The issues of modeling busses driven by multiple masters also includes multiple CPUs. At first glance, it may seem impossible to accurately model what happens with two or more masters driving a single bus through arbitration logic. There is no way of knowing exactly when the other master will win the arbitration, and take control of main memory. When this happens, the CPU will stall as soon as it requires access to the contested resource. When this happens, its power consumption will drop to the baseline level.

As it turns out, it isn't necessary to know exactly when this happens, as long as the assumption is made that there will be sufficient memory bandwidth for all masters to complete their task. For instance, assume that the CPU spends 80% of its time processing data, 10% of its time waiting for DMA to complete, and another 10% of its time in an idle state. CPU power consumption is calculated for the 80% of the time it is active. The DMA controller activity is modeled separately, and the power consumption for having it active 10% of the time is computed. It does not matter at what point of time the CPU or DMA controller has mastery of the bus, except at the transition point, when bus mastery changes. If the data and address transitions caused by the second master is not known in advance, (the usual case) it is safest to assume half of the data and address bits will change state during the transition. If the bus masters rarely change, (thousands of clock cycles or longer) then the transition effects caused by changing masters are negligible, and may be safely ignored.

Figure 4.11: Bus with multiple masters

The multiple master bus (shown in figure 4.11 A using round robin arbitration) appears similar to the single bus master diagram in figure 4.2. In the case of figure 4.11 B, the bus reverts to the high impedance state between change of master. In this case, the power consumption for both masters can be computed independently. In figure 4.11 C however, power consumption may depend on the difference between the data last broadcast by the old master, and the new one. If the transitions are comparatively rare, (many hundreds or thousands of bus cycles apart) then they can be safely neglected. If bus mastery changes frequently, (as with round-robin arbitration) then it will be necessary to estimate the average main memory address and data bus bit transition rate, since they are likely to be uncorrelated.

In most cases, there is no software optimization that can be done to improve the situation. If some knowledge is available about the nature of the data from the other master, (e.g. DMA from an 8 bit source to a 32 bit wide bus, where the upper bits are zero) this information can be used to improve the estimate.

4.3.2.4 Variable clock rate

Some processors have the ability to vary their system clock rate on the fly. Power consumption increases linearly with clock rate, so equation 4.1 must be modified by adding a clock constant multiplier to the non-constant factors:

$$\begin{aligned}
 E_{instruction} = N * K_f f & (K + K_{\Delta PAI} \Delta PAI + K_{TPD} TPD + K_{\Delta PDI} \Delta PDI + \\
 & K_{\Delta PAE} \Delta PAE + K_{\Delta PDE} \Delta PDE + K_{\Delta DAI} \Delta DAI + \\
 & K_{TDWI} TDWI + K_{\Delta DWI} \Delta DWI + K_{TDRI} TDRI + \\
 & K_{\Delta DRI} \Delta DRI + K_{\Delta DAE} \Delta DAE + K_{TDWE} TDWE + \\
 & K_{\Delta DWE} \Delta DWE + K_{TDRE} TDRE + K_{\Delta DRE} \Delta DRE + \\
 & K_{TAS} TAS + K_{\Delta AS} \Delta AS + K_{TAD} TAD)
 \end{aligned} \tag{4.4}$$

where:

K_f is the weighting factor for the system clock frequency

f is the system clock frequency in Herz.

All other terms are exactly as described in equation 4.1.

4.3.3 Compile time modeling

If the model defined by equation 4.1 is used as a basis for compile time optimization, the values of ΔDWI , ΔDRI , ΔDWE , and ΔDRE , ΔAS , and ΔAD are usually unknown, unless the CPU is reading constant data that is pre-determined at compile time. For table lookup and pointer operations, the values of ΔDAI and ΔDAE may be unknown as well. The compiler can assume that half of the bits are likely to change state, or it may take advantage of any partial knowledge it may have. In byte wide operations for instance, a maximum of 8 and an average of 4 bits will change state. If the CPU is processing a 16 element array of aligned values with an index that is unknown at compile time, then only 4 bits of the data address space are indeterminate.

4.4 Regression Analysis

After the experimental results have been reduced to tabular form, they can be fed into SPSS¹ for a simple regression analysis.

In all cases, stepwise least squares regression is used with no pre-processing filter or weighting. During each iteration, model terms are added or removed one at a time. Terms were only added if their significance exceeded a 95% confidence level ($\alpha = 0.05$). If adding more terms caused the confidence level of any term to fall below 90%, it is removed from the model in the following step. The process is halted when there are no remaining terms with greater than 95% confidence to be added, or terms with less than 90% confidence to be deleted. In some cases the analysis is repeated with all external access data points deleted, to check if they were swamping any smaller scale effects from internal bus transitions.

The regression analysis extracts linear relationships between the dependent variables, (model terms such as TPD in equation 4.1) and the normalized energy consumption. This proves whether or not each variable term in the proposed model is significant for each DSP under study, and that the model is relevant. At this point, each of the constant terms in equation 4.2 will be determined, with statistically insignificant terms assigned a weighting of zero, effectively removing it.

Candidate models with a large number of terms, can be analyzed in a piecewise

¹©SPSS Inc.

fashion. The analysis and statistical reduction can first be done with half of the dependent terms. The analysis can then be repeated with the second half of the terms, plus any of the first half that were shown to be statistically significant. Alternatively, the analysis can be split in terms of internal versus external bus analysis. Since external memory busses are apt to require more current to drive than on chip busses, it makes sense to analyze them separately. The major effects of external bus activity may mask out the more subtle power dependencies between on chip busses.

Once this step is complete, a simplified model remains which has been verified as correct, and may be used for accurately predicting instantaneous power consumption for the target while running an arbitrary segment of code.

4.4.1 Regression diagnostics

It can be dangerous and misleading to blindly accept the results of a least squares regression, without performing some additional sanity checks. The most common problems are outliers, and collinear dependent variables.

Some formal techniques exist for the analysis of outliers, but fortunately they were not required here. The experiments were carefully prepared in advance, so they could be run at one sitting, replicate measurements were taken of every test run, and the data is carefully examined while it is being collected. These techniques helped to prevent outliers from happening in the first place.

Some dependencies between variables certainly do exist, as mentioned earlier in section 5.2. Ideally, if two dependent variables showed a very strong correlation, it would be desirable to repeat the experiment after devising additional test cases that altered one of the affected variables, while avoiding similar change in the other variable. It is impossible to devise test programs that completely remove variable interdependencies, but it is sometimes possible to minimize them. To see how these interactions play out, consider the following case: Altering ΔDRI requires modification of the source code. This in turn changes ΔPD . Although care must be taken to try and develop some test cases that altered one dependent variable to the exclusion of all others, residual dependencies will remain.

A strong correlation between two dependent variables makes it difficult to determine how each variable separately affects overall energy consumption. This can lead

to indeterminate model coefficients, or a stepwise regression that causes coefficients to change radically as terms are added or deleted. To see this, consider the extreme case where variables are perfectly correlated:

$$Y = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 \quad (4.5)$$

where,

$$X_2 = kX_1 \quad (4.6)$$

It is impossible to derive fixed values for coefficient estimators $\hat{\beta}_1$ and $\hat{\beta}_2$, because an infinite family of solutions exist. If the correlation between the two is high but not quite perfect, a least squares regression is possible. In this case however, $\hat{\beta}_1$ may vary considerably, depending on the value selected for $\hat{\beta}_2$ and vice-versa.

Fortunately, interaction between dependent variables is not ruinous unless it becomes severe, and the criteria for determining this are described in the following section. In the very rare case where correlation between two terms becomes troublesome, it makes sense for one of them to be dropped from the regression.

Chapter 5

Measurement

It is not possible to directly measure power consumption. Power consumption is determined by the application of Ohm's law using simultaneous measurements of voltage and current.

Voltage measurements are comparatively simple, since the supply voltage does not vary from cycle to cycle. While it is possible to design processors that conserve power by varying supply voltage, it is not a common design practice. Supply voltage is constant and fixed for all of the platforms used in this study. The normal practice is for supply voltage to be carefully regulated with large bypass capacitors near the power pins to ensure that it does not vary over time.

Current measurements are much more problematic, since they can and do vary substantially from cycle to cycle. Conventional current measurement techniques require adding a small resistance in series with the power supply to the DSP itself. Unfortunately, this is not often practical as many DSPs now use multiple ground and power pins. The popularity of high density BGA (Ball Grid Array) packages means that the power pins themselves are inaccessibly buried underneath the component.

Instantaneous cycle by cycle current must be read graphically from an oscilloscope, and it is very difficult to obtain a high degree of accuracy using this approach. This technique is susceptible to many subtle sources of error, such as lead inductance, ground bounce, and noise. Instantaneous measurements of current over a clock cycle time lasting a few nanoseconds is exceedingly difficult. It may not even be possible to externally determine when a clock cycle begins, if the system clock is generated internally via a phase lock loop from a low frequency reference.

There are yet more complications - as instructions bubble through the fetch,

decode and execution pipelines, the effects of several instructions will intermingle. It is not possible to determine if a power spike was the result of the instruction currently being executed, the subsequent one being decoded, or the one after that which is now being fetched.

5.1 Measurement of energy consumption

It is possible to avoid the need for instantaneous power measurement, by repeatedly looping on the instruction sets to be bench-marked. This allows the use of average, as opposed to instantaneous current measurement, to $5\frac{1}{2}$ digits of precision using the test setup pictured in section 5.3. If an integrating ammeter is available, it may improve the measurement accuracy still further. The power consumption is obtained by simultaneously measuring the average voltage and current supplied to a low cost evaluation board running a series of test programs. Three replicates of each measurement are taken to further improve accuracy. These measurements are then converted to energy, specified as joules per loop, which may then be converted to joules per clock cycle.

Measuring average power consumed by an entire system to a high degree of accuracy is comparatively easy. In order to reduce the impractical problem of measuring per cycle energy to one of measuring average power, the DSP is forced to loop a sequence of instructions. As long as the exact clock frequency the DSP is operating at is known, then the measured energy per cycle is:

$$T_{cycle} = \frac{I_{avg} V_{avg}}{F} \frac{j}{cycle} \quad (5.1)$$

I_{avg} is the measured current flowing into the test board

V_{avg} is the measured voltage supplied to the test board

F is the DSP clock frequency

T_{cycle} is the energy per clock cycle used by the entire board

Unfortunately, this measurement includes some unwanted factors. The evaluation boards include components such as serial interfaces, flash memory, analog amplifiers, and LEDs. If the target board includes local power regulation, this will

also affect the results. It is not desirable to measure the power consumed by the CPU in isolation, as it is important to consider the power consumed by external memory activity, which will depend on the characteristics of the memory data is being fetched from.

5.1.1 Establishing the baseline zero power point

The baseline zero of power consumption $I_{baseline}$ is found by measuring the board power consumption with the CPU in the halted but clock running state. The lower power stop (clock halted) state is not appropriate, because the clock is also used to drive some peripheral components. If the clock is not present, then the baseline power consumed by these devices would drop even further, as documented in the device data-sheet. This results in an artificially low value for $I_{baseline}$, which skews the model. The revised formula for converting power becomes:

$$E_{cycle} = \frac{(I_{avg} - I_{baseline})V_{avg}}{F} \quad (5.2)$$

$$E_{instruction} = NE_{cycle} \quad (5.3)$$

$$E_{instruction} = \frac{N((I_{avg} - I_{baseline})V_{avg})}{F} \quad (5.4)$$

E_{cycle} is the energy consumed by the DSP per clock cycle.

N is the number of clock cycles required for the instruction to complete

$E_{instruction}$ energy consumed running a particular instruction

I_{avg} average system power consumption, measured while looping on the test program.

If I_{avg} , $I_{baseline}$, and V_{avg} are measured experimentally, and F is known, then it is possible to determine the per cycle joules of energy that are consumed while running the program.

This technique works even if the assumption of a constant supply voltage does not hold, as long as it is possible to measure the average value. It is possible to test hypothetical processors that continuously vary their supply voltage, as long as it is possible to measure the average value while looping on an arbitrary group of instructions.

5.2 Development of test programs

Next, a set of test programs must be developed for the target system. Typically, a hundred or more small test programs are needed to generate sufficient data points for each DSP. The programs were written so that the DSP loops on the instructions of interest, and that the code was executing out of on chip memory. The instructions past the end of the loop must also be padded with consistent data, in order to ensure the pre-fetch buffer is always filled with the same values, even if the code is never executed. The code was also written to ensure that the processing loop begin and ends at a fixed, known address with optimum word, cache, and memory block alignment.

The exact number of bit transitions that occur during the loop on every bus must be precisely known, or can be calculated from the assembly language listing. To help with automated post-processing, special labels were added to the program code, to identify the test loop. The exact number of CPU cycles that are required to execute the loop can be easily determined from the data-sheet. The average bit transitions per clock cycle on a given bus becomes:

$$\Delta B_{cycle} = \Delta B_{loop} / C_{loop} \quad (5.5)$$

where:

ΔB_{cycle} is average number of bit transitions on the given bus during a clock cycle, while the loop is running

ΔB_{loop} is the number of bit transitions on the given bus throughout the loop

C_{loop} is the number of cycles per loop.

Ideally, these programs should try to vary each of the model terms independently. Unfortunately this is not always possible in practice. In order to test the effect of strength reduction in the ALU stage, different CPU machine instructions must be used. This in turn affects ΔDRI . In order to see the effect of ΔDAI by shuffling the order in which data is fetched, the address offsets will change, which affects ΔDRI . In practice, there will be numerous small to moderate inter-dependencies between model terms. Limited correlations between the dependent variables do not

affect the regression, as long as the interdependencies are not extreme. Techniques for detecting this will be detailed in section 4.4.1.

Three replicate power consumption measurements were taken for every test program. This was done for a number of reasons:

- It helps to quickly identify and prevent statistical outliers from cropping up, before the analysis is done
- It is possible to get an approximate idea how much noise is present in the measurements, and how accurate the results will be, prior to analysis.
- It gives more information on the variance of the dependent variable. (Energy)
- It increases the total number of data points taken for each term, which helps to improve the accuracy of the regression, and identify more subtle dependencies.

5.3 Test setup

A schematic of the test setup is shown in figure 5.1 below. Note the $4700\mu\text{F}$ capac-

Figure 5.1: Schematic of Test Setup

itor connected across power to the board under test, which served to reduce ripple noise. A ceramic capacitor can be added in parallel to reduce high frequency noise if necessary. Ideally, the capacitor should be situated as close as possible to the ammeter. In this case, the HP-34401A has a high frequency rejection filter, which was enabled for these tests. All of the target boards had their own on board regulators, so there was very little high frequency switching noise coupled back through the power connections. This also meant the exact value of the power supply voltage did not matter for the experiments, as long as it was within the range the on board regulator could handle, and remained stable between tests.

Figure 5.2 shows a frontal view of the test setup. The HP-E3610A power supply is sitting on top - the HP-34401A Multimeter is on the bottom. The $4700\mu\text{F}$ capacitor is clearly visible.

Figure 5.2: Frontal View of Test Setup

Figure 5.3 shows another view of the setup, with the C6211 board under test.

Figure 5.3: Test Setup with C6211 Board

5.4 Post-processing

Once the energy measurements have been taken, the compiler generated assembly language listing for each test program must be analyzed to extract each of the terms in equation 4.2. Attempting to do this by hand is tedious and error prone, even for a small number of test programs. Bit transition statistics on up to 15 busses must be extracted from each program listing, and there are hundreds of test programs required. Clearly, some form of automation is necessary.

5.4.1 Assembly listing analysis

A bash shell script and C++ program was developed to read and analyze each assembly language listing file in turn. For each listing file, the parsing program identifies the beginning and end of the main loop. The number of loop lines and loop instruction cycles are reported, along with the activity on each bus. The analysis must also compute how many CPU cycles are required for each instruction. For each bus, both the total number of bits as well as the bit transitions through the main processing loop are counted. Place-holders are left for manually entering the measured current, for each experiment. Since the assembly language listing syntax differs for each processor family, separate compiler output list parsing programs had to be developed for each DSP under study. The parser source code is shown in appendix B.

5.4.2 Conversion to tabular format

At this point, the parsing program output for all three DSPs under study are presented in the same list format. This must be converted to a tabular format, as required by the least squares regression software.

Three (matlab compatible) octave programs were used to read the parsed report files, and convert them to the required tabular format. Three programs were needed, because the file naming convention and parser output differed slightly between the three DSPs.

The source listings for these are presented in appendix C. This program takes clues from the filename itself to determine whether or not the data space resided in external memory. Octave was chosen for this step because of its superior matrix processing abilities; It is both intuitive and convenient to treat the desired output format as a matrix. Each experiment adds three rows to the matrix - one for each replicate current measurement.

Since the format of the results table is similar for all three DSPs, only the first few lines of the C54 raw results table are reproduced on the following page. The first column contains the count of loop lines. The second column translates this to CPU cycles, for the particular processor.

L.	Cyc.	D_PA	T_PD	D_PD	D_DA	T_DRI	D_DRI	T_DWI	D_DWI	T_DRE	D_DRE	T_DWE	D_DWE	T_AS	D_AS	T_AD	Current
34	48	112	544	114	0	0	0	0	0	256	0	0	0	0	0	0	250.230
34	48	112	544	114	0	0	0	0	0	256	0	0	0	0	0	0	251.150
34	48	112	544	114	0	0	0	0	0	256	0	0	0	0	0	0	251.410
34	48	112	544	112	16	0	0	0	0	256	0	0	0	0	0	0	256.350
34	48	112	544	112	16	0	0	0	0	256	0	0	0	0	0	0	256.210
34	48	112	544	112	16	0	0	0	0	256	0	0	0	0	0	0	256.340
34	48	112	544	128	24	0	0	0	0	256	0	0	0	0	0	0	257.070
34	48	112	544	128	24	0	0	0	0	256	0	0	0	0	0	0	257.040
34	48	112	544	128	24	0	0	0	0	256	0	0	0	0	0	0	256.890
34	48	112	544	126	28	0	0	0	0	256	0	0	0	0	0	0	257.210
34	48	112	544	126	28	0	0	0	0	256	0	0	0	0	0	0	257.140
34	48	112	544	126	28	0	0	0	0	256	0	0	0	0	0	0	257.320
34	48	112	544	142	30	0	0	0	0	256	0	0	0	0	0	0	256.960
34	48	112	544	142	30	0	0	0	0	256	0	0	0	0	0	0	257.200
34	48	112	544	142	30	0	0	0	0	256	0	0	0	0	0	0	257.080
34	32	112	544	114	0	0	0	0	0	0	0	256	0	0	0	0	257.350
34	32	112	544	114	0	0	0	0	0	0	0	256	0	0	0	0	257.380
34	32	112	544	114	0	0	0	0	0	0	0	256	0	0	0	0	257.390
34	32	112	544	112	16	0	0	0	0	0	0	256	0	0	0	0	260.440
34	32	112	544	112	16	0	0	0	0	0	0	256	0	0	0	0	260.670
34	32	112	544	112	16	0	0	0	0	0	0	256	0	0	0	0	260.600
34	32	112	544	128	24	0	0	0	0	0	0	256	0	0	0	0	260.930
34	32	112	544	128	24	0	0	0	0	0	0	256	0	0	0	0	261.120
34	32	112	544	128	24	0	0	0	0	0	0	256	0	0	0	0	261.540
34	32	112	544	126	28	0	0	0	0	0	0	256	0	0	0	0	261.730
34	32	112	544	126	28	0	0	0	0	0	0	256	0	0	0	0	261.500
34	32	112	544	126	28	0	0	0	0	0	0	256	0	0	0	0	261.510
34	32	112	544	142	30	0	0	0	0	0	0	256	0	0	0	0	261.630
34	32	112	544	142	30	0	0	0	0	0	0	256	0	0	0	0	261.370
34	32	112	544	142	30	0	0	0	0	0	0	256	0	0	0	0	261.630
34	48	112	544	112	16	0	0	0	0	256	256	0	0	0	0	0	258.510
34	48	112	544	112	16	0	0	0	0	256	256	0	0	0	0	0	258.760
34	48	112	544	112	16	0	0	0	0	256	256	0	0	0	0	0	258.380
34	48	112	544	128	24	0	0	0	0	256	256	0	0	0	0	0	259.160
34	48	112	544	128	24	0	0	0	0	256	256	0	0	0	0	0	259.170
34	48	112	544	128	24	0	0	0	0	256	256	0	0	0	0	0	259.350
34	48	112	544	126	28	0	0	0	0	256	256	0	0	0	0	0	259.460
34	48	112	544	126	28	0	0	0	0	256	256	0	0	0	0	0	259.310
34	48	112	544	126	28	0	0	0	0	256	256	0	0	0	0	0	259.400
34	48	112	544	142	30	0	0	0	0	256	256	0	0	0	0	0	259.500
34	48	112	544	142	30	0	0	0	0	256	256	0	0	0	0	0	259.530
34	48	112	544	142	30	0	0	0	0	256	256	0	0	0	0	0	259.190
34	32	112	544	122	0	0	0	0	0	0	256	256	0	0	0	0	261.670
34	32	112	544	122	0	0	0	0	0	0	256	256	0	0	0	0	261.600
34	32	112	544	122	0	0	0	0	0	0	256	256	0	0	0	0	261.630
34	32	112	544	120	16	0	0	0	0	0	0	256	256	0	0	0	265.240
34	32	112	544	120	16	0	0	0	0	0	0	256	256	0	0	0	264.980
34	32	112	544	120	16	0	0	0	0	0	0	256	256	0	0	0	265.150

Next, a second, smaller octave program was run on the raw results file. This program accepts four arguments:

1. The raw data matrix filename
2. Power supply voltage, as measured by the HP-E3610A
3. The master clock frequency
4. The scaled output filename

The raw data file is read into a matrix. Next, the first column (loop lines) was deleted, as it is not used in the regression, although the average loop length is printed to stdout. Finally, the data in the current measurement column is converted to nanojoules per loop from the supplied clock frequency and voltage, loop cycle and instruction count columns. The resultant matrix (with headers) is written out to the supplied output filename in a tabular text format suitable for use by the SPSS regression analysis program. A sample of the output (from table D.4) is reproduced on the following page:

D_PA	T_PD	D_PD	D_DA	T_DRI	D_DRI	T_DWI	D_DWI	T_DRE	D_DRE	T_DWE	D_DWE	T_AS	D_AS	T_AD	Energy
112	544	114	0	0	0	0	0	256	0	0	0	0	0	0	1.2962
112	544	114	0	0	0	0	0	256	0	0	0	0	0	0	1.3774
112	544	114	0	0	0	0	0	256	0	0	0	0	0	0	1.4003
112	544	112	16	0	0	0	0	256	0	0	0	0	0	0	1.8362
112	544	112	16	0	0	0	0	256	0	0	0	0	0	0	1.8238
112	544	112	16	0	0	0	0	256	0	0	0	0	0	0	1.8353
112	544	128	24	0	0	0	0	256	0	0	0	0	0	0	1.8997
112	544	128	24	0	0	0	0	256	0	0	0	0	0	0	1.8971
112	544	128	24	0	0	0	0	256	0	0	0	0	0	0	1.8838
112	544	126	28	0	0	0	0	256	0	0	0	0	0	0	1.9121
112	544	126	28	0	0	0	0	256	0	0	0	0	0	0	1.9059
112	544	126	28	0	0	0	0	256	0	0	0	0	0	0	1.9218
112	544	142	30	0	0	0	0	256	0	0	0	0	0	0	1.8900
112	544	142	30	0	0	0	0	256	0	0	0	0	0	0	1.9112
112	544	142	30	0	0	0	0	256	0	0	0	0	0	0	1.9006
112	544	114	0	0	0	0	0	0	0	256	0	0	0	0	1.3631
112	544	114	0	0	0	0	0	0	0	256	0	0	0	0	1.3650
112	544	114	0	0	0	0	0	0	0	256	0	0	0	0	1.3656
112	544	112	16	0	0	0	0	0	0	256	0	0	0	0	1.5563
112	544	112	16	0	0	0	0	0	0	256	0	0	0	0	1.5706
112	544	112	16	0	0	0	0	0	0	256	0	0	0	0	1.5663
112	544	128	24	0	0	0	0	0	0	256	0	0	0	0	1.5869
112	544	128	24	0	0	0	0	0	0	256	0	0	0	0	1.5988
112	544	128	24	0	0	0	0	0	0	256	0	0	0	0	1.6250
112	544	126	28	0	0	0	0	0	0	256	0	0	0	0	1.6369
112	544	126	28	0	0	0	0	0	0	256	0	0	0	0	1.6225
112	544	126	28	0	0	0	0	0	0	256	0	0	0	0	1.6231
112	544	142	30	0	0	0	0	0	0	256	0	0	0	0	1.6306
112	544	142	30	0	0	0	0	0	0	256	0	0	0	0	1.6144
112	544	142	30	0	0	0	0	0	0	256	0	0	0	0	1.6306
112	544	112	16	0	0	0	0	256	256	0	0	0	0	0	2.0268
112	544	112	16	0	0	0	0	256	256	0	0	0	0	0	2.0488
112	544	112	16	0	0	0	0	256	256	0	0	0	0	0	2.0153
112	544	128	24	0	0	0	0	256	256	0	0	0	0	0	2.0841
112	544	128	24	0	0	0	0	256	256	0	0	0	0	0	2.0850
112	544	128	24	0	0	0	0	256	256	0	0	0	0	0	2.1009
112	544	126	28	0	0	0	0	256	256	0	0	0	0	0	2.1106
112	544	126	28	0	0	0	0	256	256	0	0	0	0	0	2.0974
112	544	126	28	0	0	0	0	256	256	0	0	0	0	0	2.1053
112	544	142	30	0	0	0	0	256	256	0	0	0	0	0	2.1141
112	544	142	30	0	0	0	0	256	256	0	0	0	0	0	2.1168
112	544	142	30	0	0	0	0	256	256	0	0	0	0	0	2.0868
112	544	122	0	0	0	0	0	0	256	256	0	0	0	0	1.6331
112	544	122	0	0	0	0	0	0	256	256	0	0	0	0	1.6288
112	544	122	0	0	0	0	0	0	256	256	0	0	0	0	1.6306
112	544	120	16	0	0	0	0	0	0	256	256	0	0	0	1.8563
112	544	120	16	0	0	0	0	0	0	256	256	0	0	0	1.8400
112	544	120	16	0	0	0	0	0	0	256	256	0	0	0	1.8506
112	544	136	24	0	0	0	0	0	0	256	256	0	0	0	1.8812

5.4.3 Converting voltage and current to energy

A third octave program reads the raw tabular data and together with the supply voltage and clock frequency, generates another tabular report, with current converted to nanojoules per loop. The baseline CPU idle current consumption is subtracted off, since the DSP is consuming negligible power at this point. This omits the power consumed by unused evaluation board peripherals, regulator losses, etcetera. The loop and cycles columns are removed from this table, as they are only used to normalize the energy consumption, and are not needed for the final analysis.

At this point, all units are expressed in terms of per loop values. On the surface, it may seem odd to do the regression this way, but it is reasonable as long as all units are consistent. In fact, there are a number of good reasons for it:

1. It is simple to scale E_{loop} to $E_{instruction}$ simply by dividing the constant term by the average number of loops lines in in the regression, and assuming per-instruction, instead of per-loop values for all variable terms
2. With per-loop regression, all dependent variables are integer values. Scaling results to per-instruction values would cause most result rows to have non-integral values. These would be subject to rounding errors.
3. The results table is already quite wide - using floating point values in most columns would mean it could not be made to fit across a single page.

Chapter 6

Case Studies

For each DSP, the proposed solution involves a four step process:

1. Develop the model
2. Create test programs
3. Measure the results
4. Employ appropriate statistical techniques to determine the model coefficients and eliminate insignificant terms.

Firstly, the block diagram is studied to determine all of the major data-flow paths. From this, the power consumption model is derived. Each bus is assumed to consume an amount of power proportional to the number of bit state transitions, every time data flows across it. The result is a linear equation with N unknowns, where N is the number of active busses in the system.

In order to solve for each unknown, a series of test programs must be developed to exercise each bus. Ideally it would be best to stimulate each bus independently, but this is impossible. Since the program must loop in order to do the measurement, all bus activity throughout the loop must be accounted for. Care must be taken to account for any bus activity that occurs as a result of continuing to fill the instruction pipeline at the end of the branch, as well as any cache effects. This is a tedious and error prone process which is best handled with automated tools (see appendix B) for analyzing the raw machine code. The programs should be compiled to locate the loop at the same point in memory to eliminate the effects of different cache alignment.

Next comes the actual measurements. These must be done using very accurate instruments under carefully controlled conditions. Changes in ambient temperature, power supply voltage or crystal frequency drift can introduce errors. Allowing the system to fully stabilize for many hours before measurements are taken, and running all tests in a single sitting helps to minimize these effects. Taking replicate measurements provides important information for data reduction.

Once the measurements have been taken, the data is distilled to tabular form, via the parsing program, and the stepwise regression is done. Insignificant model elements are discarded, and weighting terms can be derived for each remaining model term. If the power consumed by an activity on a particular bus is shown to be insignificant, it must be discarded from the final results.

6.1 56303 Testing

The first DSP board used in this study is based on the Motorola 56303. The generic power consumption model is mapped to this particular device, and a series of assembly language test programs are developed to exercise each model parameter as independently as possible. The target board is then hooked up to the ammeter and power supply as described in section 5.3. The results of the experiment are processed using the methods of section 4.4, and the results of the analysis are shown in section D.1. The final best case model resulting from the regression is equation 6.2.

6.1.1 56303 Model

The architectural block diagram is shown in figure 6.1. With this DSP, there are in fact two internal data busses, (X and Y) and one program bus. For the purposes of this experiment, there was no attempt to try to measure the power consumption effects of having the internal DMA bus active. It is possible to configure some of the internal memory to function as cache, although the system under test was not configured this way.

Equation 4.2 is applied to the 56303 DSP by examining the block diagram of figure 6.1, and identifying the relevant buses. Since the data bus is divided into separate X and Y spaces, the data space is considered to be the aggregate of X and Y. If X and Y data buses are both actively reading from on chip memory in a one

Figure 6.1: 56303 Block diagram[24]

instruction loop, then $TDRI$ will be $24 + 24 = 48$, since each bus is 24 bits wide. If all of the bits are changing state, ΔDRI will also be 48. If the loop is n instructions long, and the same bus activity is occurring each instruction, then these values are multiplied by n .

Data address bus transitions are considered to be the aggregate sum of X and Y address transitions. If six bits change on XAB and four bit changes on YAB, then ΔDA is considered to be $6 + 4 = 10$.

The ALU itself is buried in the data ALU block of figure 6.1. While the ALU to accumulator connection is not explicitly shown, it is apparent from the instruction mnemonics themselves when data is moving to and from the ALU. The destination of any ALU operation is always an accumulator, although AD (ALU destination size)

ranges from 24 to 56 bits. If the source of an ALU operation is X or Y memory, then AS is 24 bits. If ALU operations are done on both X and Y operands, then AS is $24 + 24 = 48$ bits. MAC (Multiply Accumulate) ALU operations involve both X and Y operands, plus a full 56 bit accumulator, so AS would be 104 bits in this case.

From figure 6.1, it can be seen that the program address and data buses are PAB and PDB.

The following terms from equation 4.2 have been mapped onto the 56303 block diagram in figure 6.1, and included in the 56303 energy consumption model:

ΔPA is the program space address line transitions per loop (internal memory only) on the PAB bus in figure 6.1.

TPD is the total program code lines in the test loop, multiplied by the width of the PDB bus in figure 6.1.

ΔPD represents program space data line transitions per loop (internal memory only) on the PDB bus in figure 6.1.

ΔDA is the data space address line transitions per loop (combined internal and external) on the DAB bus in figure 6.1.

$TDRI$ is the total number of data space internal memory bits read during the test loop. This is the aggregate sum in figure 6.1 of XDB and YDB bits read from X or Y internal memory during the test loop.

ΔDRI is the data space internal memory read bits that changed state during the test loop. This is mapped to the sum of transitions on XDB and YDB in figure 6.1.

$TDWI$ is the total data space internal memory write bits over the test loop. This is mapped to the sum of XDB and YDB of figure 6.1. Each bit of data written to the on chip X or Y data RAM during the loop is counted here.

ΔDWI indicates the data space internal memory write bit transitions. This is the count of state changes per loop on the XDB and YDB, for data written to the X or Y RAM from figure 6.1.

$TDRE$ is the total number of data space external memory read bits. This is the aggregate sum of bits read from off chip memory through the external data bus switch in figure 6.1 during the test loop.

ΔDRE represents data space external memory read bit transitions. This is the number of times data on XDB and YDB read from off chip via the external data bus switch in figure 6.1, changed state during the test loop.

$TDWE$ represents total data space external memory write bits. This is the aggregate sum of bits written to off chip memory through the external data bus switch in figure 6.1 during the test loop.

ΔDWE is the data space external memory write bit transitions. This is the number of times the XDB and YDB data written to off chip through the external data bus switch in figure 6.1, changed state during the test loop.

TAS is the aggregate total ALU source bits, written to the ALU block in figure 6.1 from XDB, YDB, and accumulators during the test loop.

ΔAS is ALU source bit transitions for data written to the ALU block in figure 6.1 during the test loop.

TAD is the total ALU destination bits. This is the total number of bits sent back from the ALU to either accumulator, during each test loop.

$Energy$ is given in nanojoules consumed per test loop.

The SPSS software used for analysis limits the number of independent variables. For this reason, ΔDA is a combination of the ΔDAI and ΔDAE terms, to allow room for simultaneous examination of the ALU block dependency. Arguably, it could be analyzed as a separate regression, although this would entail combining and weighting the results of multiple experiments.

6.1.2 Tabulation of 56303 observations

Once the 56303 test programs have been run and all current measurements have been logged, the data must be tabulated for regression analysis. One example from the complete 56303 suite is shown below:

6.1.2.1 56303 assembly listing analysis

```

        page    132,60
;*****
;
; File: EBLAR3.s
;       This tests the DSP power consumption while reading an alternating
; bit pattern from 8 addresses in external L memory using a binary
; pattern in a 16 instruction loop.
;
;=====
; Revisions
; Jan 22 2001   Gary Cameron
;       Creation
;
;*****
        nolist
        include 'ioequ.asm'
        list

;*****
;  XREFs
;*****

;*****
;  Data
;*****

PATTERN equ    $555555          ; The data to be written out

LOOP_SZ equ    $10              ; The loop size

        org     L:$1000
TARGET ds      LOOP_SZ          ; The block of test memory

        org     p:$100

```



```

;*****
;
; Function: EBLAR3
;
; Arguments: none
;
; Returns: nothing
;
;=====
; Authours
; Jan 22 2001   Gary Cameron
;      Creation
;
;*****
Start
EBLAR3
                                ; Initialize SC0, SC1 as GPIO inputs
movep    #$831,x:<<M_AARO      ; Set up SRAM configuration
movep    #$040003,x:<<M_PCTL    ; PLL = 4 x 16.9344Mhz = 67.7Mhz
movep    #$012421,x:<<M_BCR     ; one wait state for external spaces
movep    #$000E07,X:<<M_IPRC    ; IRQA/IRQD/SSI level 3 interrupts.
                                ; edge sensitive

move     #0,omr
movec    #0,sp

;; Set up for the loop here
move     #PATTERN,a
move     #PATTERN,a0
tfr      a,b
neg      a
dec      a
move     #TARGET,r0
move     #TARGET+1,r1
move     #TARGET+2,r2
move     #TARGET+3,r3
move     #TARGET+4,r4

```

```

move    #TARGET+5,r5
move    #TARGET+6,r6
move    #TARGET+7,r7
move    a,l:(r0)      ; Prefill the target memory
move    b,l:(r1)
move    a,l:(r2)
move    b,l:(r3)
move    a,l:(r4)
move    b,l:(r5)
move    a,l:(r6)
move    b,l:(r7)
move    a,l:(r0)
move    b,l:(r1)
move    a,l:(r2)
move    b,l:(r3)
move    a,l:(r4)
move    b,l:(r5)
move    a,l:(r6)
move    b,l:(r7)
bra     <Lp

```

```

;;; Place it here so that the loop is nicely aligned
org p:$1FE

```

```

Lp:

```

```

dor     forever,LpEnd
move    l:(r0),a
move    l:(r1),a
move    l:(r2),a
move    l:(r3),a
move    l:(r4),a
move    l:(r5),a
move    l:(r6),a
move    l:(r7),a
move    l:(r0),a
move    l:(r1),a
move    l:(r2),a

```

```

        move    l:(r3),a
        move    l:(r4),a
        move    l:(r5),a
        move    l:(r6),a
        move    l:(r7),a
LpEnd:
        nop                                ; Pipecleaner
        nop                                ; Pipecleaner
        rts                                ; Will never get here$

```

This particular test accesses the 48 bit L: memory space in on chip memory. (48 bit L: memory accesses to external memory are possible, but are split into two 24 bit operations by the memory controller.)

The first few instructions in the program set up the external bus clock, some I/O and interrupt registers, and configures the internal RAM to be used as addressable memory, instead of cache. It also clears the on chip stack. This initialization code is common to all tests:

```

                                ; Initialize SC0, SC1 as GPIO inputs
movep    #$831,x:<<M_AARO      ; Set up SRAM configuration
movep    #$040003,x:<<M_PCTL    ; PLL = 4 x 16.9344Mhz = 67.7Mhz
movep    #$012421,x:<<M_BCR     ; one wait state for external spaces
movep    #$000E07,X:<<M_IPRC    ; IRQA/IRQD/SSI level 3 interrupts.
                                ; edge sensitive

move     #0,omr
movec    #0,sp

```

The next few instructions, load the A accumulator with the constant PATTERN, and generates the 1's complement of the value in the B accumulator. When the two accumulators are alternately written out to sequential locations in memory, the result is an alternating bit pattern:

```

;; Set up for the loop here
move     #PATTERN,a
move     #PATTERN,a0
tfr      a,b
neg      a
dec      a

```

Next, the index registers are pre-loaded with the target addresses.

```

move    #TARGET,r0
move    #TARGET+1,r1
move    #TARGET+2,r2
move    #TARGET+3,r3
move    #TARGET+4,r4
move    #TARGET+5,r5
move    #TARGET+6,r6
move    #TARGET+7,r7

```

Next, the alternating bit pattern is written out to internal memory:

```

move    a,l:(r0)          ; Prefill the target memory
move    b,l:(r1)
move    a,l:(r2)
move    b,l:(r3)
move    a,l:(r4)
move    b,l:(r5)
move    a,l:(r6)
move    b,l:(r7)
move    a,l:(r0)
move    b,l:(r1)
move    a,l:(r2)
move    b,l:(r3)
move    a,l:(r4)
move    b,l:(r5)
move    a,l:(r6)
move    b,l:(r7)

```

This memory pre-fill sequence is done for all of the 56303 L: space tests, even if the filled locations are not all used on a particular test. In this case, only the first 48 bit L: storage location is referenced in the loop. Now that the pre-loop setup is complete, the CPU branches to the target loop, which has been aligned on an even 512 byte boundary:

```

bra     <Lp

```

```

;;; Place it here so that the loop is nicely aligned
    org p:$1FE
Lp:

```

Note that the branch target address is hex 0x1FE instead of 0x200, because the `dor` instruction that initiates looping is not executed as part of the loop itself, and the intent is to keep the code that actually does the looping aligned. It is possible to run the test with the loop origin set at an arbitrary location, but aligning the loops minimizes ΔPA . Minimizing ΔPA reduces the colinearity between it and other dependent variables, and makes it easier to spot outliers.

Finally, the loop code itself:

```

    move    l:(r0),a
    move    l:(r1),a
    move    l:(r2),a
    move    l:(r3),a
    move    l:(r4),a
    move    l:(r5),a
    move    l:(r6),a
    move    l:(r7),a
    move    l:(r0),a
    move    l:(r1),a
    move    l:(r2),a
    move    l:(r3),a
    move    l:(r4),a
    move    l:(r5),a
    move    l:(r6),a
    move    l:(r7),a
LpEnd:
    nop                ; Pipecleaner
    nop                ; Pipecleaner
    rts                ; Will never get here

```

The intent here was not to develop the most efficient algorithm, as the fetch could be done while looping on only the first two instructions. In fact, some other test programs were written with varying loop length, which adds more data points to the result. Repeating the same test with a different addressing mode gives two sets

of energy measurements, with different values of ΔPD . If the loop length changes then the ΔPA , and $TotalPD$ is altered. This provides additional data points where the effect of varying these terms can be evaluated. Since an alternating bit pattern was written to memory before the loop was started, every read will fetch data which alternates between 0x555555555555 and 0xAAAAAAAAAAAA. This means all 48 bits on the L: data bus are changing with every instruction. From this, it can be seen that the intent of this particular test is to vary ΔDRI as much as possible. As compared with a single instruction loop, ΔPA will change a number of times. The `LpEnd:` label marks the location of the last instruction plus one to be fetched in the loop[24].

The two `nop` instructions after the `LpEnd:` label are there because the CPU continues to fetch the next two instructions past the end of the end of loop[24]. These two instructions are subsequently discarded as the end of branch takes effect, and the program counter is rolled back to the beginning of the loop. But since they are still fetched from memory they do consume bus cycles, hence must be included in the power analysis. If these instructions were not present, whatever random data happened to be in memory would be fetched from these addresses. This would introduce irrecoverable errors in the data, and throw off the regression. For this reason, it is critical to the analysis that the two instructions following the end of loop contain known data, even though they do not affect operation of the loop.

Care must also be taken when using different addressing modes, to check if the instruction cycle time changes. If it does, then the analysis must allow for this, as per equation 5.3.

The final `rts` instruction is not really necessary, since the loop is infinite.

Once the test program is assembled, a program listing is generated with the `.lst` extension. The listing files are not shown in the appendices, because they are quite long and are automatically generated from the assembly listings. The `.lst` file for this particular program is given on the following page:

```

2          ;*****
3          ;
4          ; File: EBLAR3.s
5          ;       This tests the DSP power consumption while read
6          ; bit pattern from 8 addresses in external L memory usi
7          ; pattern in a 16 instruction loop.
8          ;
9          ;=====
10         ; Revisions
11         ; Jan 22 2001   Gary Cameron
12         ;       Creation
13         ;
14         ;*****
619
620         ;*****
621         ;   XREFs
622         ;*****
623
624         ;*****
625         ;   Data
626         ;*****
627
628         555555          PATTERN    equ    $555555          ;
629
630         000010          LOOP_SZ    equ    $10              ;
631
632         L:001000          org      L:$1000
633         L:001000          TARGET    ds      LOOP_SZ        ;
634
635         P:000100          org      p:$100
636         ;*****
637         ;
638         ; Function: EBLAR3
639         ;
640         ; Arguments: none
641         ;
642         ; Returns: nothing
643         ;
644         ;=====
645         ; Authours
646         ; Jan 22 2001   Gary Cameron
647         ;       Creation
648         ;
649         ;*****
650         Start
651         EBLAR3
652
653         P:000100 08F4B9 [2 - 2]          movep          #$831,x:<<M_AARO          ;
654                000831
654         P:000102 08F4BD [2 - 4]          movep          #$040003,x:<<M_PCTL          ;
654                040003

```

```

655      P:000104 08F4BB [2 -      6]      movep      #$012421,x:<<M_BCR      ;
          012421
656      P:000106 08F4BF [2 -      8]      movep      #$000E07,X:<<M_IPRC      ;
          000E07
657                                          ;
658      P:000108 0500BA [1 -      9]      move      #0,omr
659      P:000109 0500BB [1 -     10]      movec      #0,sp
660
661
662      P:00010A 56F400 [2 -     12]      move      #PATTERN,a
          555555
663      P:00010C 50F400 [2 -     14]      move      #PATTERN,a0
          555555
664      P:00010E 200009 [1 -     15]      tfr      a,b
665      P:00010F 200036 [1 -     16]      neg      a
666      P:000110 00000A [1 -     17]      dec      a
667      P:000111 60F400 [2 -     19]      move      #TARGET,r0
          001000
668      P:000113 61F400 [2 -     21]      move      #TARGET+1,r1
          001001
669      P:000115 62F400 [2 -     23]      move      #TARGET+2,r2
          001002
670      P:000117 63F400 [2 -     25]      move      #TARGET+3,r3
          001003
671      P:000119 64F400 [2 -     27]      move      #TARGET+4,r4
          001004
672      P:00011B 65F400 [2 -     29]      move      #TARGET+5,r5
          001005
673      P:00011D 66F400 [2 -     31]      move      #TARGET+6,r6
          001006
674      P:00011F 67F400 [2 -     33]      move      #TARGET+7,r7
          001007
675      P:000121 486000 [1 -     34]      move      a,l:(r0)
676      P:000122 496100 [1 -     35]      move      b,l:(r1)
677      P:000123 486200 [1 -     36]      move      a,l:(r2)
678      P:000124 496300 [1 -     37]      move      b,l:(r3)
679      P:000125 486400 [1 -     38]      move      a,l:(r4)
680      P:000126 496500 [1 -     39]      move      b,l:(r5)
681      P:000127 486600 [1 -     40]      move      a,l:(r6)
682      P:000128 496700 [1 -     41]      move      b,l:(r7)
683      P:000129 486000 [1 -     42]      move      a,l:(r0)
684      P:00012A 496100 [1 -     43]      move      b,l:(r1)
685      P:00012B 486200 [1 -     44]      move      a,l:(r2)
686      P:00012C 496300 [1 -     45]      move      b,l:(r3)
687      P:00012D 486400 [1 -     46]      move      a,l:(r4)
688      P:00012E 496500 [1 -     47]      move      b,l:(r5)
689      P:00012F 486600 [1 -     48]      move      a,l:(r6)
690      P:000130 496700 [1 -     49]      move      b,l:(r7)
691      P:000131 050D8D [4 -     53]      bra      <Lp
692

```



```

694      P:0001FE                                org      p:$1FE
695                                          Lp:
696      P:0001FE 000202 [5 -      58]      dor      forever,LpEnd
        000011
697      P:000200 48E000 [1 -      59]      move      1:(r0),a
698      P:000201 48E100 [1 -      60]      move      1:(r1),a
699      P:000202 48E200 [1 -      61]      move      1:(r2),a
700      P:000203 48E300 [1 -      62]      move      1:(r3),a
701      P:000204 48E400 [1 -      63]      move      1:(r4),a
702      P:000205 48E500 [1 -      64]      move      1:(r5),a
703      P:000206 48E600 [1 -      65]      move      1:(r6),a
704      P:000207 48E700 [1 -      66]      move      1:(r7),a
705      P:000208 48E000 [1 -      67]      move      1:(r0),a
706      P:000209 48E100 [1 -      68]      move      1:(r1),a
707      P:00020A 48E200 [1 -      69]      move      1:(r2),a
708      P:00020B 48E300 [1 -      70]      move      1:(r3),a
709      P:00020C 48E400 [1 -      71]      move      1:(r4),a
710      P:00020D 48E500 [1 -      72]      move      1:(r5),a
711      P:00020E 48E600 [1 -      73]      move      1:(r6),a
712      P:00020F 48E700 [1 -      74]      move      1:(r7),a
713                                          LpEnd:
714      P:000210 000000 [1 -      75]      nop
715      P:000211 000000 [1 -      76]      nop
716      P:000212 00000C [3 -      79]      rts
717
718
719

0      Errors
0      Warnings

```

The listing file contains all of the information needed to extract the information required for regression. The first column contains an arbitrary line number. The second column contains the memory space, (P, X, Y, or L) address, followed by the data stored at the given address.

The small bash shell script was created to iterate through all of the program listings in a given directory tree, and pipe each `.lst` file to stdin of the parsing program. The output of the parsing program (stdout) is piped to a text file, named `56kresults.txt`

```
find . -type f -name \*.lst -exec parse56k.exe '{}' ';' > 56kresults.txt
```

The 56300 assembly file parsing program source code is show in appendix B.1, and will not be reproduced here, although a brief discussion of its operation is in order. The parsing program looks for the beginning and end of the loop, as denoted

by the **dor** instruction and the **LpEnd:** label. In some test cases, the loop may be terminated by a **bra** instruction. The parsing program first extracts the filename less extension from stdin, and prints it.

The program then opens the listing file and begins processing lines. The parser has three states of operation:

FIND_LP: This is the initial program state. Listing lines are read, until the beginning of the loop is found. If a modulo register load is found, then this information is parsed and saved, as this affects the ΔRD and ΔWD values, and modulo mode addressing is used for some for the test programs. Once the beginning of the loop is detected, the parser enters the **IN_LP** state.

IN_LP: For every loop instruction, the values of the data and program space fields are saved, for subsequent processing. A running count of loop lines is kept. In some cases, a loop instruction may require multiple addresses. When this happens, the address field is omitted from the listing at the second address, so the parser must compute it. None of the loop instructions for this test require multiple addresses, although the **dor** instruction itself does. The parser must check if an L memory space reference is being done, as the values for *TotalRD*, *TotalWD*, ΔRD and ΔWD will double, since the L bus is 48 bits wide. Automated checks for ALU source and destination references are not made - as there are only a limited number of ALU tests done, these are computed manually. The **LpEnd:** label signifies the end of the loop, and the parser enters the final stage of processing.

PARSED_LP: In this state, the loop statistics are computed, and echoed to stdout. Note that the program cycles, current readings, ΔRWD , (Read or Write Δ) *ALUSrc*, *ALUDst.*, ΔALU fields are not computed automatically; they had to be manually entered later.

When the end of listing file is reached, the parsing program state is checked as a basic sanity to ensure that it reached the end of loop state. Failure to reach this state means either the loop beginning or end was not identified. If a parsing failure was found, a warning is printed to stderr.

The output of the parsing program after missing fields are manually entered appear as follows:

```

File: EBLAR3
Loop Lines: 16
Cycles: 64
Current 1: 448.45
Current 2: 448.39
Current 3: 448.45
Delta PA Bits: 30
Total PD Bits: 384
Delta PD Bits: 28
Delta RA Bits: 28
Delta WA Bits: 0
Total RD Bits: 768
Total WD Bits: 0
Delta RWD Bits: 768
ALU Src Bits: 0
ALU Dst Bits: 0
Delta ALU Bits: 0
ALU Paths: 0

```

At this stage, the results from parsing all of the test runs will be combined into one very large file, suitable for post-processing as described in subsection 5.4.2.

6.1.3 Analysis of 56303 results

The only statistically significant variables in the model were $TDRE$, $TDWE$, ΔDWE , ΔDRE , and ΔDA . Write cycles consume more energy than reading, which isn't surprising. Since ΔDA is significant for external memory accesses, power consumption can be optimized by keeping data localized. It would also be worth arranging external memory data so that it can be fetched in grey code fashion.

6.1.3.1 Full 56303 regression

The most surprising result is that from table D.18 is that $K_{\Delta DA}$ is negative. This is completely contrary to the predictions of the model. This is most likely caused by causal interference between the model terms[11]. It should also be noted that a dependency at the 95% confidence level still leaves a 5% probability of error. The t statistic value from table D.18 reveals that the dependency is very weak. A look at the significance column, shows that it barely made it into the model, with a significance level of 0.048

As described in section 2.2.3, the Pearson correlation table D.9 details the degree of correlation between variable terms. Despite efforts to separate dependent variables, some dependent variables were highly correlated. Usually there is a logical reason for this, and it only becomes problematic to the regression when it causes the VIF (from the coefficients table, to be described later) to become extremely high. Note that a small but definite negative correlation exists between $TDRE$, $TDWE$ and ΔDWE . This points to a small flaw in the development of the test programs, as these are both dependent variables which should be totally unrelated. The programs just happened to be written such that the writes to external memory occurred at the time when a lower than average number of program space address bits were changing. Since these are all independent variables under experimental control, their correlation between them should (ideally) be zero. When the data was re-analyzed with all external memory data points deleted, the dependency vanishes. This demonstrates that although the analysis techniques are powerful, great care must be taken in devising the test programs and analyzing the results.

It can be seen from table D.9, that ΔPA is now perfectly correlated with TPD! This is obviously a problem, and a model could never be built with both of these terms present. It makes sense that the two terms would have a high correlation, since every time a program instruction is fetched, the program counter address changes. Unlike the case with the data bus, this is inherent to CPU activity. One cannot change how instructions are fetched within the loop. By doing more experiments it would be difficult, but not impossible to separate the two terms. TPD always increases linearly with loop length. ΔPA also increases with loop length, but not in a linear fashion. The least significant address bit will change with every instruction, but the n th order program counter bit will change every 2^n th instruction. It would

also be possible to move the beginning of the loop for a few of the tests, which would ensure that a different number of program space address bits changed. Ideally, the offset version would have a large number of address bit transitions right in the middle of the loop.

Table D.9 also reveals that the correlation between TAS and TAD is so high (0.96), it would be impossible to build a model containing both terms. A moderately high correlation exists between ΔPA and ΔDA as well. Undesirable correlations were also present between TPD and ΔDA .

Since there are no real internal bus activity dependencies that can be detected with this DSP, the potential for software optimization is restricted to off chip activity. It might seem that the design of the CPU is such that the energy consumed by on chip bus activity is insignificant compared with the energy required to drive the rest of the CPU. But an examination of the Pearson correlations shown in table D.9 reveals a negative correlation between TAS , DAS , TAD and other dependent variables, that shouldn't be there.

Without having detailed knowledge of how the chip was designed, it is difficult to speculate exactly why no statistically significant on chip and ALU bus dependencies were exposed. It may be that with the system clock running, a large portion of the core is active anyway, which might swamp the effect of the busses. Note that there is a large negative correlation between TAD and TPD that shouldn't be there. Test programs that exercised ALU operations were shorter than average, and the experiment would have benefitted from a few longer ALU test loop programs.

The first row of the significance table D.11 is quite interesting. Viewed in isolation, all terms appear to have a significant correlation with energy. But as the model is built up in the stepwise regression, most of them lose their importance.

The latter half of the coefficients table D.18 is worthy of minor discussion. Note that the uncertainty in the final model regarding $\hat{\beta}$ for ΔDA is quite high, and the partial correlation is weak. This lends additional weight to suspicions regarding the significance of this term. The tolerance and VIF statistics look good for each step, so the regression was stable.

The excluded variables table D.20 shows that most of the dependents have lost significance by the 5th model iteration. TAS and/or TAD (very highly correlated) *almost* make the list, The threshold was a significance of less than or equal to 0.05%, and TAD ended up with 0.074%.

The correlation coefficient table D.21, indicates a moderately strong negative correlation between the coefficients of ΔDRE and TDRE, as well as ΔDWE and TDWE. This is not surprising, as the terms themselves have a moderately strong positive Pearson correlation.

The colinearity diagnostics table D.22 reveal that the constant term and ΔDA are both strongly aligned with the same dimensional axis. This also suggests that ΔDA can be deleted, and that the 4th model should be used instead. The ANOVA (Analysis of Variance) table shows that the residual mean square is very close for both models.

A graphs of $TDRE$ versus energy is shown in figure 6.2 below: The wide variation

Figure 6.2: Plot of TDRE vs. Energy

in energy when $TDRE$ was zero shows that other factors have a substantial effect on energy. As $TDRE$ is increased, the loop energy can clearly be seen to increase as well. $TDWE$ is the second most significant factor affecting energy. In figure 6.3 both TDRE and TDWE are plotted together against energy: The error markers on the top of each pyramid indicate the 95% confidence region. The error bars will be

Figure 6.3: 3D plot of TDRE and TDWE vs. Energy

larger in places where there are fewer data points. Clearly the average energy is decreasing when TDRE and TDWE are both zero.

The 2D plot of $TDWI$ versus energy is shown in figure 6.4 for comparison.

Figure 6.4: Plot of TDWI vs. Energy

There is a huge spike at zero, since there are a lot of variables affecting energy when TDWI is zero. Despite this a weak positive correlation with energy can be inferred. But table D.9 indicates a substantial negative correlation. Clearly the wide variation in energy when TDWI is zero is skewing the results. In this case, it would have made sense to treat tests involving external memory as data missing, instead of valid $TDWI = 0$ entries. This is probably true for all internal memory tests. For comparison, the 3D plot of TDRI and TDWI versus energy is shown in figure 6.5 below:

Figure 6.5: 3D plot of TDWI and TDRI vs. Energy

If the large average energy value at the origin is ignored, weak positive trends can be seen in both axis, which was not indicated by table D.9.

6.1.3.2 Internal busses only 56303 regression

The 56303 data was re-analyzed in table D.4.1.2, this time omitting any tests that accessed external memory. Because the act of driving external busses consumes significant amounts of power, it is possible that the lesser contribution of internal bus activity might become lost in the statistical noise. As it turned out, dependencies with ΔPA for internal memory, as well as very slight dependencies on TDRI and TDWI.

From the Pearson correlation table D.25, it can be seen that TPD and ΔPA are still perfectly correlated. The high correlation between ΔDA and ΔPA , as well as TPD that was seen in table D.9 continues to manifest itself. Since this is so, it might have made sense to replace ΔPA with TPD in the regression, since it is easier to calculate.

It is interesting to compare tables D.11 and D.27. The significance of ΔPA and TPD is greatly decreased. ΔPD , ΔDA , ΔDRI , ΔDWI , TAS, ΔAS now become insignificant predictors of energy. TDRI, TDWI, and TAD are also insignificant, unless a whopping 25% margin of error is deemed acceptable! By deleting the external bus measurements, the number of data points has been reduced, hence so has the degrees of freedom. More importantly, the variances of the dependent variables are higher, since the external bus tests were treated as valid zero entries for the internal bus measurements. This can be clearly seen by comparing tables D.8 and D.24.

Table D.32 shows that the adjusted R^2 statistic does not approach unity, as it did with the full regression. This shows that in this case, the dependent variables are very weak predictors of energy.

Turning to the coefficients themselves, it may seem strange at first, that the baseline constant term is different than it was for the external memory tests. But it must be remembered, that this represents the baseline energy consumption over the test loop, not the energy per program cycle. The value is lower, because the average loop length is shorter for the internal memory tests. A comparison of tables D.8 and D.24 reveals that this is indeed so.

Table D.34 reveals that the coefficients for TDRI and TDWI are very slightly negative. This certainly seems at odds with the proposed power consumption model, and is difficult to explain without knowing details of the on chip circuitry. It may be possible, that while on chip memory is being written or read, other power hungry portions of the chip are switched off. One can only speculate without doing further experiments, and shows the importance of keeping an open mind with respect to the results.

6.1.3.3 Derived 56303 Power Consumption Model

Since table D.18 shows the last regression iteration contains a negative value of $\hat{B}$ for ΔDA , the values from the previous (4th) iteration shall be used instead:

$$\begin{aligned} E_{loop} = & 14.617 + (9.206 * 10^{-2})TDRE + (7.653 * 10^{-2})TDWE + \\ & (2.154 * 10^{-2})\Delta DWE + (1.650 * 10^{-2})\Delta DRE \end{aligned} \quad (6.1)$$

where E_{loop} is specified in units of nanojoules per loop.

This may be converted from E_{loop} to $E_{instruction}$ as described in section 5.4.3, by dividing the constant term by the average number of instructions per loop, which was manually computed to be 12.621 for the 56303 test programs. This yields:

$$\begin{aligned} E_{instruction} = & 1.158 + (9.206 * 10^{-2})TDRE + (7.653 * 10^{-2})TDWE + \\ & (2.154 * 10^{-2})\Delta DWE + (1.650 * 10^{-2})\Delta DRE \end{aligned} \quad (6.2)$$

where $E_{instruction}$ is specified in units of nanojoules per instruction.

6.2 C5402 Testing

The second target DSP chosen for study is the Texas Instruments C5402. As with the 56303, the generic power consumption model is mapped to this device, and a series of assembly language test programs are developed to exercise each model parameter as independently as possible. The target board is connected to the ammeter and power supply as described in section 5.3. Experimental results are processed using the methods of section 4.4, and the results of the analysis are shown in section D.2. The final best case model resulting from the regression is equation 6.4.

6.2.1 C5402 Model

The C5402 is an older, 16 bit DSP that is still in common use. The C5402 contains 16 bit wide program and data buses, a 17 by 17 bit multiplier, and a 40 bit ALU.

The architectural block diagram is shown in figure 6.6 on the following page. (on chip memory is omitted for clarity) A number of specialized internal busses can be seen. There was no attempt made to model these separately. The analysis was based on the simpler, generic consumption model in equation 4.2.

From figure 6.6, it is apparent that the data bus is DB, while the data address bus is DAB. DB maps to DRI , DWI , DRE , DWE in equation 4.2, depending on whether a read or write operation is occurring, and whether or not the read or write cycle is targeted at an off-chip address. Transitions on DAB map to ΔDA in equation 4.2. For this processor, all data busses are 16 bits wide.

The program data and address buses of figure 6.6 are easy to identify. PAB is the program address bus, while PB is the program data bus. PB maps directly to TPD in equation 4.2, while ΔPA equals transitions on figure 6.6 PAB.

Figure 6.6 is very confusing when it comes to ALU operations. The barrel shifter, adder, and multiplier are shown as separate from the ALU. Instead of trying to model all of these devices separately, any arithmetic or logical operation is considered to be ALU activity. AS is considered to be the number of operand bits used in calculations during the test loop, while AD is considered to be the number of bits in the result. If a test loop used six ALU operations of 16 bits with 40 bit results, then AS would be $6 \times 16 = 96$, while AD would be $6 \times 40 = 240$. ΔAS would equal the number of bits that changed in the ALU source data during the test loop.

Figure 6.6: C5402 Block diagram

The following terms from the generic model equation are included in the C5402 energy consumption analysis:

ΔPA is the number of program space address line transitions (internal memory only) that occur during the test loop on the PAB bus.

TPD is the total number of program instruction lines in the test loop, multiplied by the instruction width (16 bits).

ΔPD is the total number of bit transitions on the PD bus from figure 6.6 during each test loop.

ΔDA is the transitions per test loop on DAB from figure 6.6. (composite of internal and external access)

$TDRI$ is the number of times data was read from internal memory during the test loop, multiplied by the width of the data bus. (16 bits)

ΔDRI represents the number of state changes for each bit of data read from internal memory during the test loop.

$TDWI$ is the number of times data was written to internal memory during the test loop, multiplied by the data bus width. (16 bits)

ΔDWI represents the number of state changes on all bits written to internal memory during the test loop.

$TDRE$ is the number of times data was read from off-chip memory during the test loop, multiplied by the width of the data bus. (16 bits)

ΔDRE is the number of state changes for each bit of data read from external memory during the test loop.

$TDWE$ is the number of times data was written out to off-chip memory during the test loop, multiplied by the width of the data bus. (16 bits)

ΔDWE represents the number of state changes on all bits written to internal memory during the test loop.

TAS is the total number of arithmetic operand bits to be processed during each program test loop pass.

ΔAS represents the number of state changes among the arithmetic operand bits processed during each test loop.

TAD is the number of arithmetic result bits processed during each program test loop pass. (The C5402 ALU and adder are both 40 bits wide)

$Energy$ is given in normalized nanojoules per loop

As before, the ΔDA term is a combination of the ΔDAI and ΔDAE terms, to allow room for simultaneous examination of the ALU block dependency.

6.2.2 Tabulation of C5402 observations

Once the C5402 test programs have been run and all current measurements have been logged, the data must be tabulated for regression analysis. One typical test program from the complete C54 suite is shown below:

6.2.2.1 Parsing C5402 assembly listings

```

;*****
;*
;* File: Ebar4.asm
;*
;*      Test power consumption while the CPU is reading an alternating
;* pattern in a 16 instruction repeat block
;*
;* Regs Modified      : A1,A2
;* Regs Used          : A1,A2
;* Local Frame Size   : 0 Args + 0 Auto + 0 Save = 0 byte
;*
;=====
;* Revision History
;* Mar 14 2001  Gary Cameron
;*              - Creation
;*
;*****
;***** ASSEMBLY CODE: *****

        .mmregs

        .sect  ".const"

        .sect  "asm_extern"

; Reserve 16 words of uninitialized data, aligned on a longword bound
        .align 100h

Data:

        .sect  "asm_text"

ST0_SET .set    01800h
ST1_SET .set    0a900h
PatAA   .set    0aaaah

```


[illegible]

```

        stl b,          *ar5+%
        b _outerLoop

        .sect    "asm_text2"

_outerLoop
        st          #BRC_SET, BRC

        rptbd    EndLp
        nop
        nop
        ;; == The loop body begins here
        ld *ar5(0),a
        ld *ar5(1),a
        ld *ar5(2),a
        ld *ar5(3),a
        ld *ar5(4),a
        ld *ar5(5),a
        ld *ar5(6),a
        ld *ar5(7),a
        ld *ar5(8),a
        ld *ar5(9),a
        ld *ar5(10),a
        ld *ar5(11),a
        ld *ar5(12),a
        ld *ar5(13),a
        ld *ar5(14),a
        EndLp
        ld *ar5(15),a
        ;; == The loop body ends here

        nop                      ; Pipecleaners
        nop
        b _outerLoop            ;Reinintialize the block repeat count
        ;; == Should never get here
        ret

```


At this point, the CPU is now initialized, and the test pattern has been written out. The next instruction is an unconditional branch to the section of code containing the test loop:

```
b _outerLoop
```

At the beginning of the outer loop, the block repeat count is set. Unfortunately, it is not possible to set up an infinite loop with this processor, using the hardware looping instructions. Use of a branch to terminate the loop is undesirable, because it introduces pipeline delays. Instead, a nested double loop is used. The value in the BRC register is one less than the number of times the loop will execute. The inner loop runs for $0xFFFF + 1$ or 65,536 iterations before falling through to the outer loop branch. At this point, the BRC register is reloaded, and the inner loop runs for another 65,536 iterations[34].

Since the outer loop executes only once every 65,536 times through the inner loop, it has a negligible effect on the power consumption measurements, and is ignored. The inner loop setup code is given below:

```
_outerLoop
    st                #BRC_SET, BRC

    rptbd    EndLp
    nop
    nop
```

The first two instructions after the `rptbd` instruction are not part of the inner loop. They are only executed when the inner loop is entered.

The code within the inner loop code that is used for experimental calculation is:

```

;; == The loop body begins here
ld *ar5(0),a
ld *ar5(1),a
ld *ar5(2),a
ld *ar5(3),a
ld *ar5(4),a
ld *ar5(5),a
ld *ar5(6),a
ld *ar5(7),a
ld *ar5(8),a
ld *ar5(9),a
ld *ar5(10),a
ld *ar5(11),a
ld *ar5(12),a
ld *ar5(13),a
ld *ar5(14),a
EndLp
ld *ar5(15),a
;; == The loop body ends here

nop                                ; Pipecleaners
nop

```

With every instruction, the address pointer register `ar5` is incremented in a modulo 16 fashion after the data at the indicated address is loaded into accumulator `a`. The various C5x addressing modes are discussed at length in the mnemonic instruction set manual[34]. Since an alternating bit pattern was set up in memory as part of the initialization sequence, ΔDR will be maximized. The CPU is also fetching the same instruction repeatedly, so ΔPD will be very small - the only transition will be with the pipe-cleaner instructions. ΔPD would be exactly 0 if the pipe-cleaners were replaced with the same instructions as in the loop, (which would also reduce its colinearity with other variables) but using `nop` makes it easier to parse automatically. The pipe-cleaner `nop` instructions serve the same purpose as they did with the 56303 DSP. It is necessary to define known program memory

values past the loop end, since they are fetched. They are ultimately discarded from the instruction queue prior to execution unless the inner loop has terminated.

When the inner loop terminates, the pipe-cleaner instructions are finally executed, along with the outer loop branch:

```
    b _outerLoop          ;Reinitalize the block repeat count
    ;; == Should never get here
ret
```

This restarts the inner loop. As mentioned in the source code comments, the final `ret` instruction should never be executed. As with the 56k DSP, the test program is assembled, and a program listing is generated with the `.lst` extension. The `.lst` file for this particular program is shown on the following page:

```

1          ;*****
2          ;*
3          ;* File: Ebar4.asm
4          ;*
5          ;*   Test power consumption while the CPU is reading an alternating bit
6          ;* pattern in a 16 instruction repeat block
7          ;*
8          ;* Regs Modified   : A1,A2
9          ;* Regs Used      : A1,A2
10         ;* Local Frame Size : 0 Args + 0 Auto + 0 Save = 0 byte
11         ;*
12         ;=====
13         ;* Revision History
14         ;* Mar 14 2001 Gary Cameron
15         ;*           - Creation
16         ;*
17         ;*****
18         ;***** ASSEMBLY CODE: *****
19
20         .mmregs
21
22 000000          .sect  ".const"
23
24 000000          .sect  "asm_extern"
25
26         ; Reserve 16 words of uninitialized data, aligned on a longword boundary.
27         .align 100h
28 000000  Data:
29
30 000000          .sect  "asm_text"
31
32      1800 ST0_SET .set  01800h
33      A900 ST1_SET .set  0a900h
34      AAAA PatAA  .set  0aaaah
35      5555 Pat55  .set  05555h
36      0000 Pat0   .set  00000h
37      FFFF BRC_SET .set  0FFFFh
38
39         ; Enable external SRAM
40      0004 CNTL2 .set  00004h
41 000000 0000 EN_XMEM .uint 00000h
42         .def _main
43
44         ;*****
45         ;* Function: _main
46         ;*****
47 000001  _main:
48 000001 7500+      portw EN_XMEM, CNTL2      ;Enable external memory
49         000002 0004
48 000003 7619      st #16,BK      ;Define size of circular buffer
49         000004 0010

```

```

50 000005 7600      st #0,IMR
    000006 0000
51 000007 7706      stm #ST0_SET,ST0
    000008 1800
52 000009 7707      stm #ST1_SET,ST1
    00000a A900
53 00000b F020      ld #PatAA,      a
    00000c AAAA
54 00000d F120      ld #Pat55,      b
    00000e 5555
55 00000f 7615      st #Data,      ar5
    000010 0000+
56 000011 80D5      stl a,          *ar5+%
57 000012 81D5      stl b,          *ar5+%
58 000013 80D5      stl a,          *ar5+%
59 000014 81D5      stl b,          *ar5+%
60 000015 80D5      stl a,          *ar5+%
61 000016 81D5      stl b,          *ar5+%
62 000017 80D5      stl a,          *ar5+%
63 000018 81D5      stl b,          *ar5+%
64 000019 80D5      stl a,          *ar5+%
65 00001a 81D5      stl b,          *ar5+%
66 00001b 80D5      stl a,          *ar5+%
67 00001c 81D5      stl b,          *ar5+%
68 00001d 80D5      stl a,          *ar5+%
69 00001e 81D5      stl b,          *ar5+%
70 00001f 80D5      stl a,          *ar5+%
71 000020 81D5      stl b,          *ar5+%
72 000021 F073      b _outerLoop
    000022 0000+
73
74 000000      .sect    "asm_text2"
75
76 000000      _outerLoop
77 000000 761A      st          #BRC_SET, BRC
    000001 FFFF
78
79 000002 F272      rptb    EndLp
    000003 0024+
80 000004 F495      nop
81 000005 F495      nop
82                  ;; == The loop body begins here
83 000006 10E5      ld *ar5(0),a
    000007 0000
84 000008 10E5      ld *ar5(1),a
    000009 0001
85 00000a 10E5      ld *ar5(2),a
    00000b 0002
86 00000c 10E5      ld *ar5(3),a
    00000d 0003
87 00000e 10E5      ld *ar5(4),a

```



```

00000f 0004
88 000010 10E5      ld *ar5(5),a
000011 0005
89 000012 10E5      ld *ar5(6),a
000013 0006
90 000014 10E5      ld *ar5(7),a
000015 0007
91 000016 10E5      ld *ar5(8),a
000017 0008
92 000018 10E5      ld *ar5(9),a
000019 0009
93 00001a 10E5      ld *ar5(10),a
00001b 000A
94 00001c 10E5      ld *ar5(11),a
00001d 000B
95 00001e 10E5      ld *ar5(12),a
00001f 000C
96 000020 10E5      ld *ar5(13),a
000021 000D
97 000022 10E5      ld *ar5(14),a
000023 000E
98 000024      EndLp
99 000024 10E5      ld *ar5(15),a
000025 000F
100                ;; == The loop body ends here
101
102 000026 F495      nop                ; Pipecleaners
103 000027 F495      nop
104 000028 F073      b _outerLoop      ;Reinintialize the block repeate count
000029 0000+
105                ;; == Should never get here
106 00002a FC00      ret

```

No Errors, No Warnings

The listing file gives the section relative addresses in the second column, as well as the program data bus contents in the third. Labels begin in column 22, and the assembly mnemonics start in column 30.

As with the 56k listing parser, a bash shell script was created to iterate through all of the program listings in a given directory tree, and pipe each `.lst` file to stdin of the parsing program. The output of the parsing program (stdout) is piped to a text file, named `c54results.txt`

```
find . -type f -name \*.lst -exec parsec54.exe '{} ' ';' > c54results.txt
```

The C5402 assembly file parsing program source code is show in appendix B.2. As with the 56k parser, the program looks for the beginning and end of the loop.

The loop is assumed to begin when the tagged comment "The loop body begins here" is found. The loop end is marked by the `b Loop`, `b _outerLoop` mnemonics, or alternatively the `";; == Should never get here"` comment. As would be expected, much of the parse program logic was derived from the 56k parser. Aside from the loop boundary detection, a number of other source code modifications were required, as the instruction words are only 16 bits wide and the mnemonics are quite different.

The output of the parsing program after missing fields are manually entered appear as follows:

```

File: Ebar3
Loop Lines: 34
Cycles: 48
Current 1: 259.46
Current 2: 259.31
Current 3: 259.40
Delta PA Bits: 112
Total PD Bits: 544
Delta PD Bits: 126
Delta RA Bits: 28
Delta WA Bits: 0
Total RD Bits: 256
Total WD Bits: 0
Delta RWD Bits: 256
ALU Src Bits: 0
ALU Dst Bits: 0
Delta ALU Bits: 0
ALU Paths: 0

```

When this is complete, the results from parsing all of the test runs will be combined into a single file suitable for post-processing as described in subsection 5.4.2.

6.2.3 Analysis of C5402 results

The results for this device were interesting, in that there were a greater number of significant terms to the model.

6.2.3.1 Full C5402 regression

The values of ΔDA , $TDRE$, ΔDRI , ΔDWE , ΔDRE , TAS , ΔDWI , $TDWE$, $TDRI$, and $TDWI$. It is interesting to note that internal memory reads were statistically significant, whereas internal memory write cycles were not. Internal bus activity may be more significant with this chip, possibly because it is quite

complex, requiring longer busses. It was also seen that unlike the 56303, ALU activity contributes a significant amount towards overall power consumption on this DSP. When external bus activity was deleted from the model, the statistically significant terms were ΔDA , ΔDRI , TAD , ΔPA , ΔDWI , $TDRI$, $TDWI$. Unlike the 56303, internal bus activity remains a statistically significant effect. The final model (#14) in table D.50 shows that program space alignment and bit transitions are unimportant. ΔAS has been removed from the model as well, probably because of its high correlation with TAS as revealed in the Pearson correlation table D.41.

There were a number of unwanted, strong dependent variable correlations seen in table D.41. ΔPA was very strongly correlated with TPD and ΔPD . While it was not a perfect correlation, it was strong enough that only one of the three should have been included in the regression. The same problem happened with the 56k regression, except in that case, the correlation happened to be perfect. While ΔPA and TPD have an inherent strong correlation, it would be possible to devise experiments to separate them further, using techniques as described in section 6.1.3.1. The linkage between TPD and ΔPD could be broken by adding more tests with sequences of identical instructions, so that the same 16 bytes of data is fetched repeatedly by the program counter. Table D.41 also shows unwanted correlations between TAS and TAD . It would have made sense to eliminate one of these terms from the model.

An examination of the coefficients of table D.50 does reveal something that was unexpected - some of the coefficient values for internal and external memory access were close. For example, $TDWE$ and $TDWI$ only differed by 36%. The difference for read operations was much greater, 218% for $TDRE$ vs. $TDRI$. Without detailed knowledge of how the CPU and on chip memory are designed and laid out, it is difficult to speculate on why this should be so. Unlike the 56303, the C5402 consumes less power doing a memory write cycle, as opposed to a read. As compared with the 56303 regression, there were no surprises in terms of negative coefficients in the final model. Some negative coefficients did appear in the intermediate steps, while the regression was destabilized by including highly colinear variables.

Iterations 8-12 of table D.50 show what happens when highly correlated variables are added to the regression. Coefficients with high VIF statistics radically change from one iteration to the next, as correlated terms are added. Fortunately, the final model (iteration #14) is stable, with no abnormally high VIF statistics. Table D.56 also shows the associated condition indices rising to very high levels during these

iterations.

The last iteration in table D.52 shows that none of the five excluded terms have statistical significance when the final model is complete. Three of them are highly correlated with each other. What is interesting to note, is that their partial correlations are quite low. It would have been interesting to re-run the regression with only one of the correlated terms present. TAD is highly correlated with TAS, which was included. ΔAS seems to have no measurable effect on energy, so it doesn't matter if the bit patterns passing through the ALU are changing state, so long as the ALU is active.

6.2.3.2 Internal busses only C5402 regression

As with the 56303 DSP, the data was re-examined, discarding all data using external memory accesses. The stepwise regression was restarted under these conditions. When the regression stopped after nine iterations, there were two changes of interest: TAD was used instead of TAS, (they are highly correlated so they are almost interchangeable) and ΔPA was added to the final model.

When the regression was complete, there were no negative coefficients. The coefficient for the newly added term ΔPA , is very small, but does show a high statistical significance. Of the four excluded terms in the final iteration of table D.71, two are correlated with ΔPA , so they could have been used instead. This analysis reinforces the conclusion made earlier, that ΔAS seems to have no measurable effect on energy.

The last iteration in table D.72, as well as the VIF statistics of table D.69, shows that the final regression model is stable, with no excessive interaction between terms.

6.2.3.3 Derived C5402 Power Consumption Model

The power consumption model parameters may be read directly from the $\hat{B}$ values of the final iteration listed in table D.50:

$$\begin{aligned}
 E_{loop} = & .982 + (8.669 * 10^{-3})\Delta DA + (2.944 * 10^{-3})TDRE + \\
 & (1.079 * 10^{-3})\Delta DRI + (1.196 * 10^{-3})\Delta DWE + (9.560 * 10^{-4})\Delta DRE + \\
 & (1.299 * 10^{-2})TAS + (5.056 * 10^{-4})\Delta DWI + (1.487 * 10^{-3})TDWE + \\
 & (1.349 * 10^{-3})TDRI + (1.095 * 10^{-3})TDWI
 \end{aligned} \tag{6.3}$$

where E_{loop} is specified in units of nanojoules per loop.

This may be converted from E_{loop} to $E_{instruction}$ as described in section 5.4.3, by dividing the constant term by the average number of instructions per loop, which was manually computed to be 11.165 for the C5402 test programs. This yields:

$$\begin{aligned}
 E_{instruction} = & 8.795 * 10^{-2} + (8.669 * 10^{-3})\Delta DA + (2.944 * 10^{-3})TDRE + \\
 & (1.079 * 10^{-3})\Delta DRI + (1.196 * 10^{-3})\Delta DWE + \\
 & (9.560 * 10^{-4})\Delta DRE + (1.299 * 10^{-2})TAS + \\
 & (5.056 * 10^{-4})\Delta DWI + (1.487 * 10^{-3})TDWE + \\
 & (1.349 * 10^{-3})TDRI + (1.095 * 10^{-3})TDWI
 \end{aligned} \tag{6.4}$$

where $E_{instruction}$ is specified in units of nanojoules per instruction.

6.3 C6211 Testing

The final DSP chosen for study is the Texas Instruments C6211. As with the others, the generic power consumption model must be mapped to this device, and a set of assembly language test programs developed to exercise each model parameter. The target board is connected as described in section 5.3. As before, results are processed using the methods of section 4.4, and the results are shown in section D.3. The final best case model resulting from the regression is equation 6.6.

6.3.1 C6211 Model

The TMS320C621 is a very popular, high performance VLIW DSP from Texas Instruments. The processor block diagram is shown below:

Although program instructions are nominally 32 bits long, they are fetched in groups of 8. This raises an interesting question - should program space be treated as being 32 bits wide or 256? Since internal memory is fetched as 256 bit wide words, it makes more sense to model this DSP using a 256 bit program data space. It is impossible to disable the L1 program cache with this processor. However, the hit rate is certain to be 100% for short loops. Thus, the model only considers the 256 bit wide program space bus between the core and the L1 cache.

Figure 6.7: C6211 Block diagram[33]

On this chip, figure 6.7 shows that there are two independent data paths that can function simultaneously. In each path, the ALU block is broken into two specialized parts that can also operate simultaneously. These are termed .L1 and .M1 for the A side. The B side equivalents are .L2 and .M2. Each block is 32 bits wide. As a first approximation, AS of equation 4.2 may be considered as the aggregate sum of all active blocks, multiplied by the width of their operands. There are always two 32 bit operands, and a single 32 bit result for each block. Thus if the test program consisted of a single instruction group where all four blocks were active, AS would be $4 \times 32 \times 2 = 256$. AD would be $4 \times 32 = 128$. If there were n instruction groups in the loop, AS would be $256n$, and AD would be $128n$. ΔAS would refer to the total number of operand bits that changed state whenever operands were sent to the .L1, .M1, .L2 or .M2 blocks.

The S and D blocks are used for branch computation and pointer operations, and are not explicitly modeled.

TPD of equation 4.2 is obtained by counting the number of 32 bit instructions in the test loop, and rounding up to the nearest modulo 256, since the entire fetch packet is assumed to move between L1 and the CPU, even if it is only partially used. TPD will always be evenly divisible by 256. ΔPD can be obtained by looking at

the number of bits that change state from one fetch packet to the next, since data moves between L1 and the CPU core 256 bits at a time.

This DSP operates at a very high clock frequency, so the wait state penalties for any off chip memory access are prohibitively large. For this reason, there are no external memory terms in the model. All tests were run exclusively from on chip memory.

Since there are no external data tests done, equation 4.2 is simplified for this CPU. DRI and DWI will always be multiples of 32, since the data bus is 32 bits wide. Data can be simultaneously fetched from two banks of on chip memory using $.D1$ and $.D2$ references.

The following terms from the generic model equation are included in the C6211 energy consumption analysis:

ΔPA is the number program space address line transitions that occur between the CPU and L1 cache while the test loop is running. The program address line only changes once for every fetch packet, so this number is quite small.

TPD is the total number of fetch packets in the test loop, multiplied by the fetch packet width.

ΔPD is the number of bit transitions between fetch packets that occur during the test loop. If the test loop occupies a single fetch packet, this will be zero.

ΔDA is the number of 32 bit data space address line transitions that occur during the loop. If $.D1$ and $.D2$ are both used during the test loop, then the number of transitions on both sides must be counted separately, and summed together to obtain ΔDA from equation 4.2.

$TDRI$ is the total number of data words read from on-chip memory during the test loop through $.D1$ plus $.D2$, multiplied by 32 bits per word.

ΔDRI is the number of bit transitions that occur while reading memory through $.D1$, plus the number of bit transitions that occur while reading a second address through $.D2$ during the test loop.

$TDWI$ is the total number of data words written to on-chip memory during the test loop through $.D1$ plus $.D2$, multiplied by 32 bits per word.

ΔDWI is the number of bit transitions that occur while writing memory through .D1, plus the number of bit transitions that occur while writing a second address through .D2 during the test loop.

TAS is the total number of operand bits used in operations involving the .L1, .M1, .L2 or .M2 blocks during the test loop. This will always be an even multiple of 64, since there are always two 32 bit operands.

ΔAS is the number of state changes that occur in .L1, .M1, .L2 or .M2 operand bits during the test loop.

TAD is the number of .L1, .M1, .L2 or .M2 results per test loop, multiplied by 32 since results are always 32 bits wide.

Energy is the normalized nanojoules consumed per loop

6.3.2 Tabulation of C6211 observations

Once the C6211 test programs have been run and all current measurements have been logged, the data must be tabulated for regression analysis. The C6211 test programs are invoked from the C language environment, which handles all of the uninteresting details involved in setting up the memory controller interface, as well as the interrupt control and status registers. This program shell is compiled as a separate file common to all test programs, and is discussed in appendix A.3. One of the C6211 test programs has been reproduced on the following page:

6.3.2.1 Parsing C6211 assembly listings

```

;*****
;*
;* File: G_Rd16_AL16.asm
;*
;* This is used to test power consumption of the TMS320C6211 DSK in the
;* state where it is busy reading an alternating bit pattern from 16 gra
;* coded addresses in an unrolled loop.
;*
;* Regs Modified      : A1,A2
;* Regs Used          : A1,A2
;* Local Frame Size   : 0 Args + 0 Auto + 0 Save = 0 byte
;*
;*****
;***** ASSEMBLY CODE: *****
.sect "asm_txt"
.align 0x800

.global _G_Rd16_AL16
.global EXT0_BASE

Pat55 .set 55555555h

_G_Rd16_AL16:
mvkl .s1 EXT0_BASE,a1 ;Load 32 bit add
|| mvkl .s2 Pat55,b1

mvkh .s1 EXT0_BASE,a1
|| mvkh .s2 Pat55,b1

nop 4

        mv        .s1 b1,a2

; Write the data to be read back in the test loop

```

; This code has not been optimized, so that it is as
; identical as possible for all tests.

```
stw .d1 a2,*a1  
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[1]  
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[3]  
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[2]  
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[6]  
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[7]  
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[5]  
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[4]  
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[12]  
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[13]  
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[15]  
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[14]
|| not .l1 a2,a2

stw .d1 a2,*a1[10]
|| not .l1 a2,a2

stw .d1 a2,*a1[11]
|| not .l1 a2,a2

stw .d1 a2,*a1[9]
|| not .l1 a2,a2

stw .d1 a2,*a1[8]
|| not .l1 a2,a2

;===== The actual reads start here =====

Lp:
ldw .d1 *a1,a2
|| nop

ldw .d1 *a1[1],a2
|| nop

ldw .d1 *a1[3],a2
|| nop

ldw .d1 *a1[2],a2
|| nop

ldw .d1 *a1[6],a2
|| nop

ldw .d1 *a1[7],a2
|| nop
```

```
ldw .d1 *a1[5],a2
|| nop

ldw .d1 *a1[4],a2
|| nop

ldw .d1 *a1[12],a2
|| nop

ldw .d1 *a1[13],a2
|| nop

ldw .d1 *a1[15],a2
|| nop
|| b .s1 Lp

ldw .d1 *a1[14],a2
|| nop

ldw .d1 *a1[10],a2
|| nop

ldw .d1 *a1[11],a2
|| nop

ldw .d1 *a1[9],a2
|| nop

        ldw .d1 *a1[8],a2
|| nop

.text
```

The first few instructions in the function loads the 32 bit wide alternating bit test pattern into `a2` and sets up a pointer to the base of the test memory block in `a1`. The `nop 4` instruction adds a delay equivalent to four `nop` instructions.

```
_G_Rd16_AL16:
mvkl .s1 EXT0_BASE,a1 ;Load 32 bit address and data
|| mvkl .s2 Pat55,b1

mvkh .s1 EXT0_BASE,a1
|| mvkh .s2 Pat55,b1

nop 4

        mv        .s1 b1,a2
```

Next, the test pattern is written out to memory. Because this DSP is capable of generating new register contents while the previous value is being written out, it makes sense to generate the inverted bit pattern on the fly. This portion of the code is common to all test functions.

```
; Write the data to be read back in the test loop
; This code has not been optimized, so that it is as
; identical as possible for all tests.
```

```
stw .d1 a2,*a1
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[1]
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[3]
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[2]
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[6]
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[7]
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[5]
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[4]
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[12]
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[13]
```

```
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[15]
```

```
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[14]
```

```
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[10]
```

```
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[11]
```

```
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[9]
```

```
|| not .l1 a2,a2
```

```
stw .d1 a2,*a1[8]
```

```
|| not .l1 a2,a2
```


The loop code itself is shown next. This particular test reads the alternating bit pattern in gray coded fashion, in an attempt to minimize ΔDA . This can be seen by examining the word offsets from register `a1`. Although the DSP is capable of executing up to eight 32 bit instructions in parallel, only two parallel operations are happening each cycle in the loop, and one of them is a `nop`. Once eight instructions have been executed, another 256 byte wide group is fetched. Instructions were grouped in pairs, in order to keep things more consistent between test functions, and make it easier to compute instruction cycles. This also makes them evenly divisible by eight, so an instruction group is always fetched every four cycles.

In this case, only one parallel instruction is not a `nop`. Unlike the other DSPs in this study, it does not contain dedicated registers or instructions for looping. Instead, looping is done by adding a parallel branch instruction to the pipeline. Because the instruction pipe is very deep with this processor, the branch is executed five cycles after it is fetched.

```
;===== The actual reads start here =====
```

```
Lp:
ldw .d1 *a1,a2
||  nop

ldw .d1 *a1[1],a2
||  nop

ldw .d1 *a1[3],a2
||  nop

ldw .d1 *a1[2],a2
||  nop

ldw .d1 *a1[6],a2
||  nop

ldw .d1 *a1[7],a2
||  nop
```

```
ldw .d1 *a1[5],a2
|| nop

ldw .d1 *a1[4],a2
|| nop

ldw .d1 *a1[12],a2
|| nop

ldw .d1 *a1[13],a2
|| nop

ldw .d1 *a1[15],a2
|| nop
|| b .s1 Lp

ldw .d1 *a1[14],a2
|| nop

ldw .d1 *a1[10],a2
|| nop

ldw .d1 *a1[11],a2
|| nop

ldw .d1 *a1[9],a2
|| nop

ldw .d1 *a1[8],a2
|| nop
```

There is no need for pipe-cleaner instructions here, since the five pairs of instructions occurring after the branch are actually executed.

As with the C5402 and 56303 DSP, the test program is assembled, and a program listing is generated with the `.lst` extension. The `.lst` file for this particular program is shown on the following page:

```

1          ;*****
2          ;*
3          ;* File: G_Rd16_AL16.asm
4          ;*
5          ;* This is used to test power consumption of the TMS320C6211 DSK in
6          ;* state where it is busy reading an alternating bit pattern from 16 gr
7          ;* coded addresses in an unrolled loop.
8          ;*
9          ;* Regs Modified      : A1,A2
10         ;* Regs Used         : A1,A2
11         ;* Local Frame Size  : 0 Args + 0 Auto + 0 Save = 0 byte
12         ;*
13         ;*****
14         ;***** ASSEMBLY CODE: *****
15 00000000      .sect "asm_txt"
16              .align 0x800
17
18              .global _G_Rd16_AL16
19              .global EXT0_BASE
20
21          55555555 Pat55      .set      55555555h
22
23 00000000      _G_Rd16_AL16:
24 00000000 00800029!          mvkl      .s1      EXT0_BASE,a1          ;Load 32 bit ad
25 00000004 00AAAAAA          ||      mvkl      .s2 Pat55,b1
26
27 00000008 00800069!          mvkh      .s1      EXT0_BASE,a1
28 0000000c 00AAAAEA          ||      mvkh      .s2 Pat55,b1
29
30 00000010 00006000          nop 4
31
32 00000014 010411A0          mv        .s1 b1,a2
33
34          ; Write the data to be read back in the test loop
35          ; This code has not been optimized, so that it is as
36          ; identical as possible for all tests.
37
38 00000018 01040275          stw        .d1 a2,*a1
39 0000001c 010BEDD8          ||      not      .l1 a2,a2
40
41 00000020 01042275          stw        .d1 a2,*a1[1]
42 00000024 010BEDD8          ||      not      .l1 a2,a2
43
44 00000028 01046275          stw        .d1 a2,*a1[3]
45 0000002c 010BEDD8          ||      not      .l1 a2,a2
46
47 00000030 01044275          stw        .d1 a2,*a1[2]
48 00000034 010BEDD8          ||      not      .l1 a2,a2
49
50 00000038 0104C275          stw        .d1 a2,*a1[6]
51 0000003c 010BEDD8          ||      not      .l1 a2,a2

```

```

52
53 00000040 0104E275          stw          .d1 a2,*a1[7]
54 00000044 010BEDD8      || not          .l1 a2,a2
55
56 00000048 0104A275          stw          .d1 a2,*a1[5]
57 0000004c 010BEDD8      || not          .l1 a2,a2
58
59 00000050 01048275          stw          .d1 a2,*a1[4]
60 00000054 010BEDD8      || not          .l1 a2,a2
61
62 00000058 01058275          stw          .d1 a2,*a1[12]
63 0000005c 010BEDD8      || not          .l1 a2,a2
64
65 00000060 0105A275          stw          .d1 a2,*a1[13]
66 00000064 010BEDD8      || not          .l1 a2,a2
67
68 00000068 0105E275          stw          .d1 a2,*a1[15]
69 0000006c 010BEDD8      || not          .l1 a2,a2
70
71 00000070 0105C275          stw          .d1 a2,*a1[14]
72 00000074 010BEDD8      || not          .l1 a2,a2
73
74 00000078 01054275          stw          .d1 a2,*a1[10]
75 0000007c 010BEDD8      || not          .l1 a2,a2
76
77 00000080 01056275          stw          .d1 a2,*a1[11]
78 00000084 010BEDD8      || not          .l1 a2,a2
79
80 00000088 01052275          stw          .d1 a2,*a1[9]
81 0000008c 010BEDD8      || not          .l1 a2,a2
82
83 00000090 01050275          stw          .d1 a2,*a1[8]
84 00000094 010BEDD8      || not          .l1 a2,a2
85
86          ;===== The actual reads start here =====
87
88 00000098          Lp:
89 00000098 01040265          ldw          .d1 *a1,a2
90 0000009c 00000000      || nop
91
92 000000a0 01042265          ldw          .d1 *a1[1],a2
93 000000a4 00000000      || nop
94
95 000000a8 01046265          ldw          .d1 *a1[3],a2
96 000000ac 00000000      || nop
97
98 000000b0 01044265          ldw          .d1 *a1[2],a2
99 000000b4 00000000      || nop
100
101 000000b8 0104C265          ldw          .d1 *a1[6],a2
102 000000bc 00000000      || nop

```

```

103
104 000000c0 0104E265          ldw          .d1 *a1[7],a2
105 000000c4 00000000      ||  nop
106
107 000000c8 0104A265          ldw          .d1 *a1[5],a2
108 000000cc 00000000      ||  nop
109
110 000000d0 01048265          ldw          .d1 *a1[4],a2
111 000000d4 00000000      ||  nop
112
113 000000d8 01058265          ldw          .d1 *a1[12],a2
114 000000dc 00000000      ||  nop
115
116 000000e0 0105A265          ldw          .d1 *a1[13],a2
117 000000e4 00000000      ||  nop
118
119 000000e8 0105E265          ldw          .d1 *a1[15],a2
120 000000ec 00000001      ||  nop
121 000000f0 0FFFF710      ||  b          .s1      Lp
122
123 000000f4 0105C265          ldw          .d1 *a1[14],a2
124 000000f8 00000000      ||  nop
125
126 00000100 01054265          ldw          .d1 *a1[10],a2
127 00000104 00000000      ||  nop
128
129 00000108 01056265          ldw          .d1 *a1[11],a2
130 0000010c 00000000      ||  nop
131
132 00000110 01052265          ldw          .d1 *a1[9],a2
133 00000114 00000000      ||  nop
134
135 00000118 01050265      ldw      .d1 *a1[8],a2
136 0000011c 00000000      ||  nop
137
138
139 00000000          .text

```

No Errors, No Warnings

The listing file gives the section relative addresses in the second column, as well as the program data bus contents in the third. Labels begin in column 28, and the assembly mnemonics start in column 36.

As with the 56k and C54 listing parser, a bash shell script was created to iterate through all of the program listings in a given directory tree, and pipe each `.lst` file to stdin of the parsing program. The output of the parsing program (stdout) is piped to a text file, named `c6xresults.txt`

```
find . -type f -name \*.lst -exec parsec6x.exe '{} ' ';' > c6xresults.txt
```

The C6211 assembly file parsing program source code is show in appendix B.3. As with the other parsers, the program looks for the beginning and end of the loop. The loop is assumed to begin just after the label `Lp:`, or `_loop:` is found. The loop end is marked by the `.text` linker directive, or the "`b .s1 _Idle_loop`" instruction. The basic parse program structure is superficially similar to the 56k and C5402 parser, but required substantial modification to correctly parse the C6211 assembly mnemonics and syntax. The address and data busses are considered to be 32 bits wide for the purposes of computing ΔPD , even though the internal instruction bus is 256 bits wide.

The output of the parsing program after missing fields are manually entered appear as shown below:

```

          File: G_Rd16_AL16
    Loop Lines: 40
          I Groups: 16
          Cycles: 16
    Current 1: 648.51
    Current 2: 648.47
    Current 3: 648.43
Total Fetch Packets: 5
    Delta PA Bits: 80
    Total PD Bits: 1280
    Delta PD Bits: 150
    Delta RA Bits: 16
    Delta WA Bits: 0
    Total RD Bits: 512
    Total WD Bits: 0
    Delta RWD Bits: 512
      ALU Src Bits: 0
      ALU Dst Bits: 0
    Delta ALU Bits: 0
      ALU Paths: 0
```

When this is complete, the results from parsing all of the test runs will be combined into a single file, as with the other test cases, to be post-processed as described in subsection 5.4.2.

6.3.3 Analysis of C6211 results

For the C6211 tests, there were no external memory tests done. The ΔAS term was omitted from regression as well, since variances in it were not studied.

6.3.3.1 Full C6211 regression

For the C6211, the significant terms were ΔPD , TDWI, TAS, ΔDWI , and ΔDRI . (There were no external memory access tests done) It is interesting to note that as was the case with the C5402, the TAS term meant ALU activity was statistically significant for this chip, although bit transitions through the ALU were not. This may be because the energy required to activate the ALU during a given cycle is much greater than the energy required to change states on the individual bus pins. Perhaps the ALU bus is tri-stated between cycles, in which case it will have no memory of any previous state. As with the other two DSPs, there is an undesirable, near perfect correlation between ΔPA and TPD. ΔPA , ΔPD , and ΔDA are also strongly correlated. In hindsight, it would have been desirable to devise additional experiments to better separate these three terms. TAS and TAD are perfectly correlated, so one of them could have been dropped from the model. Table D.77 is interesting, in that it shows TAS and TAD are strongly anti-correlated with energy.

ΔPD is significant, so it is worthwhile optimizing the code so as to minimize the number of bit transitions between adjacent code words. Because this is a VLIW processor with a 256 byte wide instruction word, the number of bits simultaneously changing state could potentially be very large. The tolerance and VIF columns of table D.86 show that even for the discarded terms, correlated variables did not pose a problem for the regression.

The excluded model terms in the final iteration of table D.88 were ΔPA , TPD, ΔDA , TDRI, and TAD. Since TAD and TAS are perfectly colinear, it is necessary to exclude one of them. Three of the other three excluded variables are strongly correlated with each other. The last term, TDRI is simply a very weak predictor of energy. TDWI is a significant factor, so it appears that memory writes require a lot

more energy than read cycles for this device. This is also reflected in the relative weighting of $\hat{\beta}$ for ΔDRI and ΔDWI in table D.86.

There were only 174 data points for this regression, so the F statistic acceptance threshold was higher than it was for the other two DSPs. This means that terms had to show a more pronounced effect on energy, in order to be included in the regression. It is easier to detect very weak and subtle relationships with large datasets. Despite this, interesting and significant results were obtained for the C6211 DSP as well.

All of the coefficients were positive, except for TAS. It seems nonsensical that sending data to one of the ALUs would have a fairly substantial, negative overall effect on energy consumption, as it suggests that an ALU operation consumes less power than a `nop`. Without detailed knowledge of the internal circuitry, one can only speculate. It may be possible that if there are no data memory accesses required during a given cycle, a lot of circuitry needed for driving the busses are idle. If this were so, it would appear as a stepwise relationship between TDRI or TDWI and energy, where any non-zero TDRI or TDWI value would result in an energy consumption jump. To eliminate this possible effect, the ALU data was re-examined with memory effects deleted, in the following section.

6.3.3.2 ALU only C6211 regression

In order to test if the negative ALU energy dependency was real, the data was re-examined using only the data points where TDRI and TDWI were zero. These results are presented in section D.4.3.2. There was only one dependent variable, (TAS) and 51 data points in this analysis. As it turns out, TAS was shown to have a small, positive effect on energy, which is the expected result. Since the results of the analysis changed, this suggests there is a substantial effect on energy consumption with data memory accesses. This is not something that was expected in the proposed model. Adding additional tests to explore multi-cycle ALU and multiplier activity in tandem with memory accesses would give a better handle on the contribution of ALU activity to the overall power consumption. The $\hat{\beta}$ value in table D.97 for TAS was only 0.001, as opposed to 0.404 for the baseline constant term. The TAS contribution is very small, but measurable.

6.3.3.3 Derived C6211 Power Consumption Model

Since table D.86 shows the last few regression iterations had negative value of $\hat{B}$ for TAS , the final TAS $\hat{B}$ value from table D.99 was used instead. The remaining $\hat{B}$ values are taken from the final row of table D.86.

$$E_{loop} = .872 + (1.380 * 10^{-3})\Delta PD + (1.328 * 10^{-3})TDWI + (10^{-2})TAS + (9.279 * 10^{-4})\Delta DWI + (4.610 * 10^{-4})\Delta DRI \quad (6.5)$$

where E_{loop} is specified in units of nanojoules per loop.

For this particular DSP, it makes the most sense to specify energy in terms of instruction groups. Energy may be converted from E_{loop} to E_{igroup} by dividing the constant term by the average number of instruction groups per loop, which was manually computed to be 10.828 for the C6211 test programs. This yields:

$$E_{igroup} = 8.05 * 10^{-2} + (1.380 * 10^{-3})\Delta PD + (1.328 * 10^{-3})TDWI + (10^{-2})TAS + (9.279 * 10^{-4})\Delta DWI + (4.610 * 10^{-4})\Delta DRI \quad (6.6)$$

where E_{igroup} is specified in units of nanojoules per instruction group.

Chapter 7

Conclusions

The increased use of DSPs and micro-controllers in feature packed battery operated devices such as cellular phones, has driven the demand for low power systems with greater computational complexity. Computationally intensive algorithms require high performance processors, which tend to consume a lot of power. This necessitates the use of larger, heavier, and more expensive batteries. If components dissipate enough power to require a heat sink, this drives up the cost and weight even further. But size, cost and weight are a primary concern with these devices.

Power consumption has not been a traditional concern with desktop, server and mainframe systems until recently. But as more functionality is being packed onto devices being driven at increasing clock rates. This has caused massive increases in power dissipation, with attendant power distribution and heat removal problems. This drives up both the up front system cost, as hefty power supplies and elaborate cooling solutions become increasingly necessary. The long term cost of operating them can be substantial.

To date, most of the focus has been on hardware optimization such as lower operating voltages, and switching CPU blocks on and off on the fly, as they are required. Techniques for optimizing software algorithms for reduced power consumption on an arbitrary platform has not been well researched. This thesis represents a step in this direction. The following contributions were made:

1. A simple, yet practical bus-oriented power consumption model for analyzing DSP software power consumption was created. In section 4.1, a simple, bus-oriented power consumption model was developed, as described in equa-

tion 4.1. The basic model was subsequently expanded upon in section 4.3.2. With these refinements, it was possible to apply the generic model to a completely arbitrary targets.

2. This model was subsequently expanded, to show how the effects of cache, multiple bus masters, multiple bank memories, multiple bus masters, and variable clock rates could be added to the basic generic model. Since these features have a significant effect on power consumption and are present on most controllers and DSPs, it was important to develop a generic model for them.
3. A technique for obtaining accurate board level measurements of power consumption and isolating the contributions from software on an arbitrary target was crucial to this research. Methods for obtaining measurements using inexpensive and commonly available laboratory tools, were described in chapter 5. The variance observed between replicates showed that the measurement technique was relatively noise free. The very small SEE (Standard Error of the Estimate) terms found in the analysis, (tables D.16, D.48, and D.84) could not have been achieved, unless very accurate measurements had been done, even if the system was perfectly modeled.
4. A practical application of these techniques were shown when the generic power model was applied to the three DSP processor boards selected for this experiment. The mapping techniques were discussed at length in chapter 6. Equation 4.1 was mapped to each processor by carefully examining the architectural block diagrams and instruction sets. In each case, the key program and data buses could be identified, while superfluous architectural details were ignored.
5. The use of statistical regression techniques to validate the model and remove insignificant model terms was shown. In each case, it was shown that the proposed model could accurately predict per instruction power. It was possible to extract statistically significant terms from the generic model, to a 95% confidence level and compute the weighting factors for each model term. This showed that it was possible to use statistical regression techniques to validate the model. The high adjusted residuals value for the last model iteration in tables D.16, D.48, and D.84 demonstrate that generic model can be used as a basis for a very accurate power model on a variety of target boards. The

high statistical significance of the results indicate that there are untapped opportunities for significant power optimization, achievable through software optimization. It is possible to use the derived per-instruction power consumption models of section 6.2, 6.4, and 6.6 as the basis for a power optimizing compiler strategy.

7.1 Directions for future work

Examination of figure 6.4 suggests that internal bus tests would have yielded better results if they were treated as missing data whenever external busses were active. It would be desirable to re-analyze the data with the ΔDA term into ΔDAI and ΔDAE components. When the external memory tests were deleted, the ΔDA term was no longer significant in both the 56303 and C5602 analysis, but became significant when external memory access was included. This suggests that the ΔDA data points for the internal memory tests skewed the importance of this effect, when external memory is being accessed. The models can be rounded out by including model terms for data and instruction fetches from external RAM and ROM, although these are less common scenarios. Since many processors split the ALU into separate blocks for addition, multiplication and logical operations, it would be interesting to try and model these separately. This would show if strength reduction optimizations had any measurable effect on power.

Since it was possible to accurately predict per-instruction energy consumption using a generic model for a diverse variety of DSP architectures, the information gained through this exercise could be used as a guideline for automated program optimization techniques. A very generalized energy model with CPU specific parameters, can be used as the basis for a power optimization strategy, within a retargetable compiler, to see if alternative instruction sequences or arrangements of data would result in more power efficient compiled code. It would be possible to apply this technique to measure the power effects of activity on DMA busses as well, although it was beyond the scope of this work. This analysis hints at how DSP hardware developers could reduce power consumption as well - since program code tend to be sequential, converting program address busses to a gray code would reduce power consumption. Likewise, commonly grouped instructions could be arranged so as to minimize the number of bit transitions between them.

Developing test programs and employing regression analysis is a powerful technique for evaluating models. As long as test programs can be developed, any power consumption model can be tested. If the model is correct then dependencies will remain; otherwise they will be shown as statistically insignificant, and terms will vanish in the data reduction. As was seen in the analysis, care must be taken when interpreting the results, and one needs to keep an open mind towards revising the proposed model in light of experimental results. Although the model does not predict it, there may exist non-linear relationships between the more significant model terms and the per-loop energy. It may prove worthwhile to re-examine some of the data to look for higher order terms, and try adjusting the model to account for such relationships, if any are found.

This work might even be extended to hardware such as FPGAs as well, as long as it is possible to model their on chip busses. If the power consumption costs of I/O activity on an FPGA or ASIC can be accurately modeled, the tradeoffs involved with using different interface options can be explored at design time.

There is another practical use for this work - one special case where the object is to reverse the objectives of this exercise. It is often important to stress system hardware under absolute worst case conditions, forcing the software to maximize power consumption, in order to check the power distribution and heat dissipation is adequate under the worst imaginable conditions. These stress test diagnostics are done both at a chip and system level.

7.2 56301 specifics

It would highly desirable to devise more tests to remove the unwanted statistical correlation between ΔPD and $TDRE$ and $TDWE$. This could be accomplished by taking more measurements using one or more `nop` instructions inserted before the loop, to better randomize the location where the reads or writes occur. By taking better measurements, and more of them, it may be possible to detect other dependencies that were not detectable with this setup, at the confidence levels chosen. But even if subtle dependencies can be extracted, it would not usually be worth complicating the model with additional terms, just to try optimizing the last fraction of a percent. The correlation between ΔPA and ΔDA could be reduced if more experiments were added with long loops that acted on only one or two data loca-

tions. This would also reduce the unwanted correlation between TPD and ΔDA . Adding `nop` instructions prior to the loop to alter the instruction alignment might also help reduce unwanted correlations between the dependent variables. This would also help expose more subtle dependencies.

Since the on chip memory can be reallocated as cache, it would be interesting to see how this would affect power consumption. Adding the effect of external memory program fetches to the model, would then allow us to predict the cost of a cache miss. Breaking down ALU operations into separate addition, logical, and multiplier paths would help to test the effect of any strength reduction.

7.3 C5402 specifics

There are many internal busses that could be added to the model for future refinements. On the other hand, if there are already a lot of significant terms, introducing more complexity might not be desirable. Testing for external program memory access effects would round out the model.

7.4 C6211 specifics

It would also be interesting to add a model term to account for bitwise transitions within the 256 bit instruction group, ΔG . Since this is a VLIW processor, at some point the program instruction bus becomes 256 bits wide. The effect of ΔPA could be better explored by creating long test loops that require many instruction groups.

It would also be desirable to model the multiplier blocks separately from the adder and logical blocks, since they can all operate in tandem.

For this processor, it would be helpful to model external memory accesses, as well as the effect of the two level cache. Since the L1 cache cannot be disabled, it would be necessary to construct “cache buster” test programs which jump between two locations that access memory in such a way as to force a miss on the L1 cache. It would be also desirable to repeat some of the measurements with the L1D cache enabled for comparison. The compiler could then try to adjust the power consumption weighting based on whether or not a cache hit was predicted. It would make sense to try modeling the multiplier and logical blocks separately in order to evalu-

ate the benefits of any strength reduction. Finally, it would also be worth repeating this work on the newer C64x, as well as the floating point variants of this DSP.

References

- [1] Dr. Eddy Borghers. Descriptive statistics - simple linear regression - significance test - significance test for r -1.
<http://www.xycoon.com/testingr1.htm>, 2000-2004.
- [2] Dr. Eddy Borghers. Descriptive statistics - simple linear regression - significance test - significance test for r -1.
http://www.xycoon.com/covariance_1.htm, 2000-2004.
- [3] Dr. Eddy Borghers. Descriptive statistics - simple linear regression - significance test - significance test for r -1.
http://www.xycoon.com/correlation_coefficient_2.htm, 2000-2004.
- [4] Michael T. Brannick. Partial and semipartial correlation: Collinearity.
<http://luna.cas.usf.edu/~mbrannic/files/regression/Collinearity.html>, 2004.
- [5] Michael T. Brannick. Partial and semipartial correlation: Working with residuals.
[//luna.cas.usf.edu/~mbrannic/files/regression/19%20Partial%20and%20Semi.ppt](http://luna.cas.usf.edu/~mbrannic/files/regression/19%20Partial%20and%20Semi.ppt), 2004.
- [6] James Dean Brown. Description of standard error of estimate. *Shiken*, 3(1), April 1999. University of Hawaii.
- [7] Chaitali Chakrabarti and Dinesh Gaitonde. Instruction level power model of microcontrollers.
- [8] Moses Charikar, Rajeev Motwani, Prabhakar Raghavan, and Craig Silverstein. Constrained TSP and low-power computing. In *Workshop on Algorithms and Data Structures*, pages 104–115, 1997.

- [9] Gerald E. Dallal. The little handbook of statistical practice: Partial correlation coefficients.
<http://www.tufts.edu/gdallal/partial.htm>, 2001-2003.
- [10] Philip Ender. Applied categorical & nonnormal data analysis, collinearity issues. <http://www.gseis.ucla.edu/courses/ed231c/notes2/collin.html>, Jan 2004.
- [11] David Garson. The little handbook of statistical practice: Partial correlation coefficients. <http://www2.chass.ncsu.edu/garson/pa765/partialr.htm>, Jan 2004.
- [12] David Garson. The little handbook of statistical practice: Spss regression output. <http://www2.chass.ncsu.edu/garson/pa765/regressa.htm>, Jan 2004.
- [13] Inki Hong, Gang Qu, Miodrag Potkonjak, and Mani B. Srivastava. Synthesis techniques for low-power hard real-time systems on variable voltage processors. In *RTSS*, pages 178–187, 1998.
- [14] Cheng-Ta Hsieh, Massoud Pedram, Gaurav Mehta, and Fred Rastgar. Profile-driven program synthesis for evaluation of system power dissipation. In *Design Automation Conference*, pages 576–581, 1997.
- [15] Milind B. Kamble and Kanad Ghose. Analytical energy dissipation models for low power caches, 1997.
- [16] Michelle Lacey. Anova for regression.
<http://www.stat.yale.edu/Courses/1997-98/101/anovareg.htm>, 1997-1998.
Yale University.
- [17] David M. Lane. Hyperstat online textbook.
<http://www.davidlane.com/hyperstat/A34739.html>, 2004. Rice university.
- [18] Bret Larget. Statistics 333, applied regression analysis: Polynomial regression in r.
<http://www.stat.wisc.edu/p/stat/course/st333-larget/public/html/poly.pdf>, 2003. University of Wisconsin-Madison.
- [19] M. Lee. Power analysis and low-power scheduling techniques for embedded dsp software, 1995.

- [20] M. Lee, V. Tiwari, S. Malik, and M. Fujita. Power analysis and minimization techniques for embedded dsp software, 1997.
- [21] Sheayun Lee, Andreas Ermedahl, Sang Lyul Min, and Naehyuck Chang. An accurate instruction-level energy consumption model for embedded RISC processors. In *LCTES/OM*, pages 1–10, 2001.
- [22] S. Li and A. Hopper. A framework to integrate synchronous and asynchronous collaboration. In *Proc. 7th IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 96–103, Stanford, CA, 1998. IEEE Computer Society Press.
- [23] David Meko. Time series course lesson 11: Multiple linear regression. http://www.ltrr.arizona.edu/dmeko/notes_11.pdf, 2003.
- [24] Motorola Inc. *Technical Data DSP56303/D 24-Bit Digital Signal Processor*, January 2002.
- [25] Marc R. Rosenblum. Poli 6002: Quantitative political research methods, model selection. <http://www.uno.edu/mrosenbl/classes/6002-10.ppt>, Jan 2004.
- [26] Kaushik Roy and Mark C. Johnson. Software design for low power.
- [27] J. Russell and M. Jacome. Software power estimation and optimization for high performance, 32-bit embedded processors.
- [28] Sap help portal. http://help.sap.com/saphelp_470/helpdata/en/35/e1e6b9e2df0c04e10000009b38f983/content.htm.
- [29] Tajana Simunic, Luca Benini, and Giovanni De Micheli. Cycle-accurate simulation of energy consumption in embedded systems. In *Design Automation Conference*, pages 867–872, 1999.
- [30] Tajana Simunic, Luca Benini, Giovanni De Micheli, and Mat Hans. Source code optimization and profiling of energy consumption in embedded systems.
- [31] Scott M. Smith. *PC-MDS Multidimensional Statistics Package*. Brigham Young University, Dept. Of Marketing, Provo, Utah 84602, 1999.

- [32] SPSS Inc., 233 South Wacker Drive, 11th floor, Chicago IL. 60606-6307. *SPSS 10.0 Syntax Reference Guide for SPSS Base, SPSS Regression Models, SPSS Advanced Models*, 1999.
- [33] Texas Instruments. *TMS320C6211, TMS320C6211B Fixed-Point Digital Signal Processors*, August 1998.
- [34] Texas Instruments. *Volume 1: CPU and Peripherals*, March 2001.
- [35] V. Tiwari, S. Malik, and A. Wolfe. Compilation techniques for low energy: An overview, 1994.
- [36] V. Tiwari, S. Malik, and A. Wolfe. Power analysis of embedded software: a first step towards software power minimization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2(4):437–445, 1994.
- [37] V. Tiwari, S. Malik, A. Wolfe, and M. Lee. Instruction level power analysis and optimization of software, 1996.
- [38] M. Toburen, T. Conte, and M. Reilly. Instruction scheduling for low power dissipation in high performance microprocessors, 1998.
- [39] Narayanan Vijaykrishnan, Mahmut T. Kandemir, Mary Jane Irwin, H. S. Kim, and W. Ye. Energy-driven integrated hardware-software optimizations using simplepower. In *ISCA*, pages 95–106, 2000.
- [40] N. Wehn and M. Mnch. Minimizing power consumption in digital circuits and systems: an overview.
- [41] Eric W. Weisstein. Description of f statistic.
<http://mathworld.wolfram.com/F-Distribution.html>.
- [42] Eric W. Weisstein. Gamma function.
<http://mathworld.wolfram.com/GammaFunction.html>.
- [43] Eric W. Weisstein. Student's t-distribution.
<http://mathworld.wolfram.com/Studentst-Distribution.html>.
- [44] Richard Williams. Sociology 592: Sociology graduate statistics i, standardized coefficients. <http://www.nd.edu/~rwilliam/stats1/x92.pdf>, Sep. 2003.

- [45] Ted Zhihong Yu, Fei Chen, and Edwin H.-M. Sha. Loop scheduling algorithms for power reduction.

Appendix A

Shared Build Files

A.1 56303 shared build files

A.1.1 Link map file

This is the linker memory map configuration file that was used for all 56303 test programs.

```
ident Shared 0 0
start Start
;*****
;
; File: test.mem
;
; This file contains all of the memory map configuration.
; Note that all of the on chip memory has been configured for use as RAM
; and not as cache.
;
;=====
; Authours
; Dec 27 2000 Gary Cameron
; Creation
;
;*****

reserve P:$000002..$0000FF
```

region LI

base L:\$000000

memory L:\$000008

base X:\$000009

memory X:\$00003F

base Y:\$000009

memory Y:\$00003F

endr

region MI

base P:\$0000100

memory P:\$00007FF

base L:\$000040

memory L:\$0007FF

base X:\$000040

memory X:\$0007FF

base Y:\$000040

memory Y:\$0007FF

endr

;X and Y memory are aliased with offset 0 in this space

region Unified

base X:\$008000

memory X:\$00FFFF

```
base    Y:$008000
```

```
memory  Y:$00FFFF
```

```
endr
```

```
region ROM
```

```
base P:$FF0000
```

```
memory P:$FFFFFF
```

```
endr
```

A.1.2 Build file

The 56303 tests did not require makefile - instead a DOS batch file was used to build each program. One example is reproduced below; Only the name of the test file being compiled (in this case EBLAR3) would change between tests.

REM Linker script

asm56300 -b EBLAR3.cln -g -I.../shared -l EBLAR3.lst -OCC,CL,MD,MEX EBLAR3.s | more

dsplnk -A -B EBLAR3.cld -g -m -r.../shared/test.mem EBLAR3.cln | more

A.2 C5402 shared build files

A.2.1 Linker map file

This is the link map file that was used for the C54 tests.

```

/*****
* FILENAME: $Source: /db/sds/syseng/rtdx/tutorial/c54x/sect_1/less_2/c5402.
* VERSION : $Revision: 1.1 $
* DATE    : $Date: 2000/01/26 18:41:37 $
* Copyright (c) 1997 Texas Instruments Incorporated
*
* COMMAND FILE FOR LINKING TMS5402 Targets
*
* Usage: lnk500 <obj files...> -o <out file> -m <map file> lcf.cmd
*        cl500  <src files...> -z -o <out file> -m <map file> lcf.cmd
*
* Description: This file is a sample command file that can be used
*              for linking programs built with the TMS54x C Compiler.
*              Use it as a guideline; you may want to change
*              the allocation scheme according to the size of your
*              program and the memory layout of your target system.
*
* Notes: (1)  You must specify the directory in which rts.lib is
*              located. Either add a "-i<directory>" line to this
*              file, or use the system environment variable C_DIR to
*              specify a search path for libraries.
*
*          (2)  If the run-time support library you are using is not
*              named rts.lib, be sure to use the correct name here.
*
*****/
-stack 0x0256
-heap 0x0256

/*
* This symbol defines the interrupt mask to be applied to IMR inside
* RTDX Critical Code sections.
*
* This example masks the timer interrupt.
*/

```

```

{
GROUP : > DARAM /* group sects in overlay for cont. addresses */
{
    .text          /* User code */
    .cinit          /* initialization tables */
    .pinit          /* initialization functions */
    .switch         /* for C-switch tables */

    .cio            /* C-IO Buffer */
    .bss            /* global & static vars */
    .const          /* C constant tables */

    .systemem       : fill = ODEADh /* dynamic heap */

```

```
        .stack          : fill = 0BEEFh /* system stack      */
    }

GROUP : > DARAM2
{
    asm_text            /* Test code */
    asm_data            /* Test data */
}

GROUP : > DARAM3
{
    asm_text2           /* Test code */
    asm_data2           /* Test data */
}

asm_extern      : > EXT0          /* Data in external RAM */

.intvecs        : > VECS  PAGE 0    /* interrupt vector table */
.data           : > SPAD  PAGE 1    /* asm data area          */
}
```

A.2.2 Project file

This is the Code Composer Studio project file used for C5402 test program development. The project root, paths, and .out filenames will vary, depending on the name and directory path of the test program.

```
/***** Code Composer V1 Project Data *****/
The following section contains data generated by Code Composer
to store project information like build options, source filenames
and dependencies.

[command filename]
..\..\Shared\c5402.cmd 1

[Project Root]
C:\ti\myprojects\C54\Eb\Ebar4

[build options]
3
Linker = "-e _main -o Ebar4.out -x "
Assembler = "-ls "
Compiler = "-g -as -frC:\ti\myprojects\C54\Eb\Ebar4 "

[source files]
Ebar4.asm 983647844 1
18213

[dependencies]
0 -802

[version]
2.0
*/
-e _main -o Ebar4.out -x
"Ebar4.obj"
```

```
"..\..\Shared\c5402.cmd"
```

```
/***** End of Project Data Generated by Code Composer *****/
```

A.3 C6211 shared build files

A.3.1 Linker map file

This is the link map file that was used for the C6211 tests.

```
MEMORY
{
    INT_RAM:    origin = 0x00000000 length = 0x00010000 /* Internal RAM (L2) */
    EXT_BANK0:  origin = 0x80000000 length = 0x00200000 /* CE0, SDRAM, 8 MBytes */
    EXT_BANK1:  origin = 0x80200000 length = 0x00200000 /* CE0, SDRAM, 8 MBytes */
    EXT_FLASH:  origin = 0x90000000 length = 0x00020000 /* CE1, 8-bit ROM, 128 KBytes*/
}

SECTIONS
{
    .text:      > INT_RAM
    asm_txt:    > INT_RAM
    .const:     > INT_RAM
    .data       > INT_RAM
    .bss        > INT_RAM
    .cinit      > INT_RAM
    .pinit      > INT_RAM
    .stack      > INT_RAM
    .far        > INT_RAM

    /* Define external memory addresses */
    EXT_BANK0
    {
        EXT0_BASE = .;
    } > EXT_BANK0

    EXT_BANK1
    {
        EXT1_BASE = .;
    } > EXT_BANK1
}
```

A.3.2 Project file

This is the Code Composer Studio project file used for C6211 test program development. The project root, paths, and .out filenames will vary, depending on the name and directory path of the test program.

```

/***** Code Composer V1 Project Data *****/
  The following section contains data generated by Code Composer
  to store project information like build options, source filenames
  and dependencies.
[command filename]
link.cmd 8

[Project Root]
C:\ti\myprojects\G_Rd16_AL16

[build options]
3
Linker = "-q -c -m G_Rd16_AL16.map -o G_Rd16_AL16.out -x "
Assembler = "-glqs "
Compiler = "-gkq -o3 -frC:\ti\myprojects\Branch_Lp -ic:\ti\c6000\dsk\include "

[source files]
main.c 976511684 1
stdio.h 947508744 0
c6x.h 949765756 0
c6211dsk.h 953176586 0
linkage.h 947508744 0
cstdio 950200896 0
..\..\c6000\cgtools\lib\rts6200.lib 0 1
G_Rd16_AL16.asm 0 1
26368

[dependencies]
3 1:9 2:10 3:11 -31709
2 4:8 5:15 1053
0 -802
0 -802
0 -802
1 4:8 -28365
0 -802
0 -802

[version]
2.0
*/
-q -c -m G_Rd16_AL16.map -o G_Rd16_AL16.out -x
"G_Rd16_AL16.obj"
"..Branch_Lp\main.obj"
"....\c6000\cgtools\lib\rts6200.lib"
"link.cmd"
/***** End of Project Data Generated by Code Composer *****/

```

A.3.3 C main file

This is the main.c file that was common to C6211 test programs. The only thing that changed here between tests, was the name of the assembly language test file that was called from main.c, immediately after the “Call the test code” comment block.

```

/*****
 * FILENAME
 *   main.c
 *
 * DESCRIPTION
 *   Startup shell for test programs
 *
 *****/
#include <stdio.h>
#include <c6x.h>
#include "c6211dsk.h"

/*****
 * Function prototypes
 *****/
extern void G_Rd16_AL16(); /* Declare external test function */

/*****
 * FUNCTION : main
 *
 * ARGUMENTS :
 *   VOID
 *
 * DESCRIPTION :
 *   The code turns off the LEDs and calls the test function.
 *
 *****/
int main()
{
    /* DSP initialization */
    CSR=0x100; /* Disable all interrupts */
    IER=1; /* Disable all interrupts except NMI */
    ICR=0xffff; /* Clear all pending interrupts */

```



```

/*****
* Standard 6211/6711 DSK includes 2 MT48LC1M16A1-7 devices = 4MB SDRAM *
* For these devices use the following parameter: *
* EMIF_SDCTRL=0x07126000 *
*****/

*(unsigned volatile int *)EMIF_GCR = 0x3300; /* EMIF global control */
*(unsigned volatile int *)EMIF_CE1 = CE1_32; /* EMIF CE1 control, 32bit */
*(unsigned volatile int *)EMIF_SDCTRL = 0x07126000; /* EMIF SDRAM control */
*(unsigned volatile int *)EMIF_CEO = 0x30; /* EMIF CEO control */
*(unsigned volatile int *)EMIF_SDRP = 0x61a; /* EMIF SDRAM refresh period */
*(unsigned volatile int *)EMIF_SDEXT= 0x54529; /* EMIF SDRAM extension */

/*****
* Start state
*****/
*(unsigned volatile int *)IO_PORT = 0x07000000; /* Turn off all user LEDs*/

/*****
* Call the test code
*****/
G_Rd16_AL16();
}

/* END OF FILE */

```

Appendix B

List Parsing Programs

The following C++ programs were used to parse the DSP assembly language listing files to extract statistical information, and generate the raw results. Some fields were entered manually. The parsing programs were all compiled using the GNU gcc compiler using the command:

g++ -O2 -Wall -ofilename.exe

B.1 56300 Assembly Listfile Parsing Program

```

/*****
*
* FILENAME
*   parse56k.cc
*
* DESCRIPTION
*   Parses the 56k listing file to obtain the required information.
*   Output format:
*
*   File           = 8 characters for filename
*   Loop Lines     = nbr of 32 bit instructions per loop
*   Cycles         = cycles per loop (left blank, to be filled manuell
*   Current1       = Current measurement 1 (left blank, to be filled m
*   Current2       = Current measurement 2 (left blank, to be filled m
*   Current3       = Current measurement 3 (left blank, to be filled m
*   Delta PA Bits  = nbr of program space address bit changes per loop
*   Total PD Bits  = Total number of program space bits processed per
*   Delta PD Bits  = Number of program space data bit transitions per
*   Delta RA Bits  = Number of data space read address bit transitions
*   Delta WA Bits  = Number of data space write address bit transition
*   Total RD Bits  = Number of data space read bits per lp
*   Total WD Bits  = Number of data space write bits per lp
*
*   The following fields are all entered manually:
*   Delta RWD Bits = Number of data space data bit transitions per loo
*   ALU Src Bits   = Number of ALU source bits per loop
*   ALU Dst Bits   = Number of ALU destination bits per loop
*   Delta ALU Bits = Number of ALU bit transitions per loop
*   ALU Paths      = Number of ALU paths active
*
*=====
* History:
* May 12 2001    Gary Cameron
*   - Creation
* Oct 26 2001    Gary Cameron
*   - Fixed error in result generation, pipecleaners are not fetch

```

```

* subsequent passes.
*
*****/

#include <iostream.h>
#include <fstream.h>
#include <string.h>
#include <ctype.h>
#include <strstream.h>

// #define DEBUGGING

enum parseStage { FIND_LP, IN_LP, PARSED_LP};

const int FALSE(0);

const unsigned LINE_BUFFER_LENGTH(255U);

const unsigned MAX_LOOP_WORDS(16U);
const unsigned PIPELINE_FLUSH_WORDS(2U);
const unsigned LOOP_WORD_BUFFER_SIZE
    (MAX_LOOP_WORDS * PIPELINE_FLUSH_WORDS);

const unsigned BYTES_PER_DSP_WORD(3U);
const unsigned HEX_CHARS_PER_DSP_WORD(6U);
const unsigned BITS_PER_DSP_WORD(24U);

const unsigned long NOP_WORD(0UL);          // Used for loop filler

const char *SLASH_STRING ("/");             // Used to tokenize paths
const char *DOT_STRING (".");

const char *PATTERN          ("PATTERN");
const char END_PATTERN_MARKER ('$');

const char *LP_MARKER ("Lp:");              // Marks beginning of the test loop

```

```

const char *DOR_MARKER ("dor");          // Skip this, never executed

const char *END_LP_MARKER_1 ("LpEnd:");  // Marks the end of test lp
const char *END_LP_MARKER_2 ("bra");     // Alternate end of test lp

const char *X_MEMORY_REFERENCE ("x:(r"); // Detect X vs L mem reference
const char *L_MEMORY_REFERENCE ("l:(r");

const char LEFT_BRACKET ('(');
const char RIGHT_BRACKET (')');
const char OFFSET_MARKER ('+');

const char *MOVE_INSTRUCTION("move");
const char *READ_DATA_INSTRUCTION (""),");
const char *WRITE_DATA_INSTRUCTION ("1,");

const char *MODULO_REGISTER_LOAD(",m0");
const char *MODULO_OFFSET_DELIMITER(")+");
const char *IMMEDIATE_LOAD("#");

const unsigned ADDRESS_FIELD_BEGINS = 12;
const unsigned ADDRESS_FIELD_ENDS = 17;

const unsigned DATA_FIELD_BEGINS = 19;
const unsigned DATA_FIELD_ENDS = 24;

// Return a line of text from the file
// Returns number of characters read, including the end of line
unsigned getLine(ifstream &file,
                char *line)
{
    unsigned pos;                // Position within the string
    char ch;

    pos = 0U;

```

```

while (file.get(ch))                // Exit loop if EOF
{
    if (ch == '\n')
    {
        break;                    // Done if read the EOL char
    }
    line[pos++] = ch;
}
line[pos] = '\0';
return(pos);
}

// Checks if the entire string range is numeric, and if so returns the
// unsigned long value in the specified argument.
int parseInt32(char *line,
               const unsigned startPos,
               const unsigned endPos,
               unsigned long &longword)
{
    unsigned pos;
    char *tokenPos;                //Pointer to within line
    if (startPos >= endPos)
    {
        cerr << "Error: parseInt32, start =" << startPos <<
            " end =" << endPos << endl;
        return(0);
    }

    for (pos = startPos; pos <= endPos; pos++)
    {
        if (!isdigit(line[pos]))
        {
            return(0);            // String range not numeric
        }
    }
    tokenPos = &line[startPos];
}

```

```

    istrstream(tokenPos, (endPos - startPos + 1)) >> hex >> longword;
    return(1);
}

// This counts the number of 1 bits found in a longword
unsigned Count1Bits(unsigned long word)
{
    unsigned OneBits = 0U;
    static const unsigned long bitMask(0x01UL);

    do
    {
        OneBits += word & bitMask;
        word >>= 1;
    } while (word != 0UL);
    return(OneBits);
}

// This counts the number of bit transitions within a loop
unsigned CountLoopDeltaBits(const unsigned long *buffer,
                           const unsigned bufferSize,
                           const unsigned deltaSize)
{
    unsigned index;
    unsigned transitions(0U);

    for (index = 0U; index < bufferSize; index++)
    {
        transitions += Count1Bits(buffer[index] ^
                                   buffer[(index + deltaSize) % bufferSize]);
    }
    return(transitions);
}

// The main line program
int main (int argc, char *argv[])

```

```

{
    parseStage stage(FIND_LP);           // Parsing stage
    char *inputFileNameAndPath;
    char *shortFileName;                 // Filename
    char *leftMarker;                    // Pointer into parsed line
    char *rightMarker;

    char line[LINE_BUFFER_LENGTH];       // Space for a line of text
    unsigned loopLines(OU);               // Lines of loop text
    unsigned modulo;                       // Modulo value
    unsigned currentModulo(OU);

    unsigned long programAddressSpace[LOOP_WORD_BUFFER_SIZE];
    unsigned long deltaProgramAddrSpace(OUL);
    unsigned long programDataSpace[LOOP_WORD_BUFFER_SIZE];
    unsigned long totalProgramDataSpace(OUL);
    unsigned long deltaProgramDataSpace(OUL);
    unsigned long dataSpaceAddressOffsets[MAX_LOOP_WORDS];
    unsigned long deltaDataSpaceAddressOffsets(OUL);
    unsigned long deltaRdDataAddressOffsets;
    unsigned long deltaWrDataAddressOffsets;
    unsigned long readBits(OUL);
    unsigned long readBitsIncrement(OUL);
    unsigned long writeBits(OUL);
    unsigned long writeBitsIncrement(OUL);

    #ifdef DEBUGGING
        unsigned long streamBase;         // Used for iostream mode set
    #endif

    inputFileNameAndPath = argv[1];

    cout.precision(4);
    if (argc != 2)
    {
        cout << "argc is: " << argc << endl;
    }
}

```



```
    return(argc);
}

#ifdef DEBUGGING
    cerr << "Filename is: " << inputFileNameAndPath << endl;
#endif

// Open the file for reading
ifstream inFile(inputFileNameAndPath, ios::in);
if (!inFile)
{
    cerr << "Unable to open " << inputFileNameAndPath << endl;
    return(-1);
}

// Find the short filename, first trim the path
shortFileName = strtok(inputFileNameAndPath, SLASH_STRING);
do
{
    leftMarker = shortFileName;
    shortFileName = strtok(NULL, SLASH_STRING);
} while (shortFileName);

// And finally, the extension

shortFileName = strtok(leftMarker, DOT_STRING);

// Read data from the file and echo
while (getLine(inFile,line))
{
    switch (stage)
    {
        case FIND_LP:
            leftMarker = strstr(line, MODULO_REGISTER_LOAD);
            if (leftMarker)
            {
```

```

        // Record the modulus register load value for later ref.
        leftMarker = strstr(line, IMMEDIATE_LOAD);
        if (leftMarker)
        {
            istrstream(++leftMarker) >> modulo;
            modulo++;
        } else {
            cerr << "Unable to parse modulo load, file: " <<
                shortFileName << endl;
            return(-1);
        }
    }
    if (strstr(line, LP_MARKER))
    {
        stage = IN_LP;

        #ifdef DEBUGGING
            cout << "Got " << LP_MARKER << endl;
        #endif
    }
    break;

case IN_LP:
    if (strstr(line, DOR_MARKER))
    {
        // Ignore any line actually containing the dor statement,
        // since it is not a part of the loop proper.  we must
        // also ignore the following line, which is also part of
        // the dor instruction
        getLine(inFile, line);
        break;
    }
    if (strstr(line, END_LP_MARKER_1) ||
        (strstr(line, END_LP_MARKER_2)))
    {
        stage = PARSED_LP;
    }

```

```

#ifdef DEBUGGING
    cout << "Got " << END_LP_MARKER << endl;
#endif
} else {
    // Still processing loop lines

    // First, check if line is long enough
    if (strlen(line) >= DATA_FIELD_ENDS)
    {
        // If the line is long enough, we have a candidate, so
        // check the data fields first - some lines just skip the
        // address fields if the instruction requires three data
        // words.
        if (parseInt32(line,
                        DATA_FIELD_BEGINS,
                        DATA_FIELD_ENDS,
                        programDataSpace[loopLines]))
        {
            // Ok, we have a valid loop line. Let's see if
            // it has an address field, if not we will have to
            // kludge one up.
            if (!parseInt32(line,
                            ADDRESS_FIELD_BEGINS,
                            ADDRESS_FIELD_ENDS,
                            programAddressSpace[loopLines]))
            {
                // We have to kludge an address field
                programAddressSpace[loopLines] =
                    programAddressSpace[loopLines-1]+1;
            }
        }
#ifdef DEBUGGING
        // Set output in hexadecimal notation
        streamBase = cout.setf(ios::hex, ios::basefield);
        cout.setf(ios::uppercase);
        cout << "PA: ";

```

```

        cout.fill('0');
        cout.width(6);
        cout << programAddressSpace[loopLines];
        cout << " PD: ";
        cout.fill('0');
        cout.width(6);
        cout << programDataSpace[loopLines] << endl;
        cout.setf(streamBase, ios::basefield);
    #endif
    // Now, check for data space address references
    // Assumption: always have exactly one MOVE_INSTRUCTION
    // per iGroup
    // leftMarker = strstr(line, MOVE_INSTRUCTION);
    leftMarker = strstr(line, MOVE_INSTRUCTION);

    if (leftMarker)
    {
        // We have found a move reference, now try to
        // parse an offset
        rightMarker = strstr(leftMarker, L_MEMORY_REFERENCE);
        if (rightMarker)
        {
            // L memory offsets are handled differently
            rightMarker += sizeof(L_MEMORY_REFERENCE);

            istrstream(rightMarker, 1 ) >>
                dataSpaceAddressOffsets[loopLines];
        } else {
            // Default: zero address offset
            dataSpaceAddressOffsets[loopLines] = 0UL;

            leftMarker = strrchr(leftMarker, OFFSET_MARKER);
            if (leftMarker)
            {
                // We have found the last offset marker, first
                // check for modulo address reference

```

```

        if (strstr(line, MODULO_OFFSET_DELIMITER))
        {
            // Modulo addressing, increment is 1
            dataSpaceAddressOffsets[loopLines] = currentModulo;
            currentModulo = (currentModulo + 1) % modulo;
        } else {
            // Read the offset, in decimal notation
            istrstream(++leftMarker) >>
                dataSpaceAddressOffsets[loopLines];
        }
    }
} // !L memory reference

readBitsIncrement = 0UL;
writeBitsIncrement = 0UL;

// Read/write detection
if (strstr(line, READ_DATA_INSTRUCTION))
{
    readBitsIncrement = BITS_PER_DSP_WORD;
}

if (strstr(line, WRITE_DATA_INSTRUCTION))
{
    writeBitsIncrement = BITS_PER_DSP_WORD;
}

if (strstr(line, L_MEMORY_REFERENCE))
{
    // The word size doubles for L memory reference
    readBitsIncrement *= 2UL;
    writeBitsIncrement *= 2UL;
}

readBits += readBitsIncrement;
writeBits += writeBitsIncrement;

```

```

        #ifdef DEBUGGING
            streamBase = cout.setf(ios::dec, ios::basefield);
            cout.setf(ios::uppercase);
            cout << "loopLines: ";
            cout.fill(' ');
            cout.width(3);
            cout << loopLines;
            streamBase = cout.setf(ios::hex, ios::basefield);
            cout << " DA: ";
            cout.fill('0');
            cout.width(8);
            cout << dataSpaceAddressOffsets[loopLines] << endl;
        #endif
    } // Found left marker, move

    #ifdef DEBUGGING
        cout.setf(streamBase, ios::basefield);
    #endif

    loopLines++;
} // Found valid loop line

} // Still processing loop lines
}
break;

case PARSED_LP:
    // Just fall through, do nothing
    break;

default:
    cerr << "Error: Enumeration fault!" << endl;
    // Nothing more to do
    break;
}

```

```

}

// Report results
switch (stage)
{
    case FIND_LP:
        cerr << "Unable to find " << LP_MARKER << " File:" <<
            inputFileNameAndPath << endl;
        break;

    case IN_LP:
        cerr << "Unable to find " << END_LP_MARKER_1 << " or " <<
            END_LP_MARKER_2 << " File:" << inputFileNameAndPath << endl;
        break;

    case PARSED_LP:
        // Compute the remaining results
        totalProgramDataSpace = loopLines * BITS_PER_DSP_WORD;

        if (loopLines > 1)
        {
            deltaProgramAddrSpace = CountLoopDeltaBits(programAddressSpace,
                loopLines,
                1UL);
            deltaProgramDataSpace = CountLoopDeltaBits(programDataSpace,
                loopLines,
                1UL);
        } else {
            deltaProgramAddrSpace = 0UL;
            deltaProgramDataSpace = 0UL;
        }

        // Compute data space address delta bits
        deltaDataSpaceAddressOffsets =
            CountLoopDeltaBits(dataSpaceAddressOffsets,

```

```

                                loopLines,
                                1U);

if (readBits)
{
    deltaRdDataAddressOffsets = deltaDataSpaceAddressOffsets;
} else {
    deltaRdDataAddressOffsets = 0U;
}

if (writeBits)
{
    deltaWrDataAddressOffsets = deltaDataSpaceAddressOffsets;
} else {
    deltaWrDataAddressOffsets = 0U;
}

// Now we display the results
cout << "                File: " << shortFileName << endl;
cout << "            Loop Lines: " << loopLines << endl;
cout << "            Cycles: " << endl;
cout << "            Current 1: " << endl;
cout << "            Current 2: " << endl;
cout << "            Current 3: " << endl;
cout << "        Delta PA Bits: " << deltaProgramAddrSpace
    << endl;
cout << "        Total PD Bits: " << totalProgramDataSpace
    << endl;
cout << "        Delta PD Bits: " << deltaProgramDataSpace
    << endl;

cout << "        Delta RA Bits: " << deltaRdDataAddressOffsets
    << endl;
cout << "        Delta WA Bits: " << deltaWrDataAddressOffsets
    << endl;

```



```
    cout << "        Total RD Bits: " << readBits << endl;
    cout << "        Total WD Bits: " << writeBits << endl;
    // Cannot automate this easily, so just insert 0,
    // fill manually later
    cout << "        Delta RWD Bits: 0 " << endl;

    // Cannot automate this easily, so just insert 0,
    // fill manually later
    cout << "        ALU Src Bits: 0 " << endl;
    cout << "        ALU Dst Bits: 0 " << endl;
    // Bits changing in->out
    cout << "        Delta ALU Bits: 0 " << endl;
    // Only different for C6x
    cout << "        ALU Paths: 0 " << endl;
    cout << endl;
    break;

default:
    cerr << "Error: Enumeration fault!" << endl;
    break;
}

// Close the file and exit
inFile.close();
return(0);
}
```

B.2 C5402 Assembly Listfile Parsing Program

```

/*****
*
* FILENAME
*   parsec54.cc
*
* DESCRIPTION
*   Parses the c54 listing file to obtain the required information.
*   Output format:
*
* File           = 8 characters for filename
* Loop Lines     = nbr of instruction groups per loop
* Cycles         = cycles per loop (left blank, to be filled manually)
* Current1       = Current measurement 1 (left blank, to be filled manual
* Current2       = Current measurement 2 (left blank, to be filled manual
* Current3       = Current measurement 3 (left blank, to be filled manual
* Delta PA Bits  = nbr of program space address bit changes per loop
* Total PD Bits  = Total number of program space bits processed per loop
* Delta PD Bits  = Number of program space data bit transitions per loop
* Delta RA Bits  = Number of data space read address bit transitions per
* Delta WA Bits  = Number of data space write address bit transitions per
* Total RD Bits  = Number of data space read bits per lp
* Total WD Bits  = Number of data space write bits per lp
*
*   The following fields are all entered manually:
* Delta RWD Bits = Number of data space data bit transitions per loop
* ALU Src Bits   = Number of ALU source bits per loop
* ALU Dst Bits   = Number of ALU destination bits per loop
* Delta ALU Bits = Number of ALU bit transitions per loop
* ALU Paths      = Number of ALU paths active
*
*=====
* History:
* Jun 10 2001    Gary Cameron
*   - Creation
*
*****/

```

```
#include <iostream.h>
#include <fstream.h>
#include <string.h>
#include <ctype.h>
#include <strstream.h>

// #define DEBUGGING

enum parseStage { FIND_LP, IN_LP, PARSED_LP};

const unsigned LINE_BUFFER_LENGTH(255U);

const unsigned MAX_LOOP_WORDS(16U);
const unsigned PIPECLEANERS(2U);
const unsigned LOOP_WORD_BUFFER_SIZE(MAX_LOOP_WORDS * PIPECLEANERS);

const unsigned BYTES_PER_WORD(2U);
const unsigned BITS_PER_WORD(16U);

const char *SLASH_STRING ("/"); // Used to tokenize paths
const char *DOT_STRING (".");

// Marks beginning of the test loop
const char *LP_MARKER ("The loop body begins here");

// Marks the end of the test loop
const char *END_LP_MARKER ("b Loop");
const char *ALT_END_LP_MARKER_1 ("b _outerLoop");
const char *ALT_END_LP_MARKER_2 (";; == Should never get here");

const char *AR5_REFERENCE ("*ar5");
const char LEFT_BRACKET ('(');
const char RIGHT_BRACKET (')');
const char PLUS_SIGN ('+');

const char *MODULO_REGISTER_LOAD(",BK");
```

```
const char *IMMEDIATE_LOAD("#");
const char *ABSOLUTE_STORE_MODULO("ASM,");

const char *READ_DATA_INSTRUCTION ("ld");
const char *WRITE_DATA_INSTRUCTION ("stl");

const unsigned ADDRESS_FIELD_BEGINS = 9;
const unsigned ADDRESS_FIELD_ENDS = 14;

const unsigned DATA_FIELD_BEGINS = 16;
const unsigned DATA_FIELD_ENDS = 19;

// Return a line of text from the file
// Returns number of characters read, including the end of line
unsigned getLine(istream &file,
                 char *line)
{
    unsigned pos;                // Position within the string
    char ch;

    pos = 0U;

    while (file.get(ch))         // Exit loop if nothing to read (EOF)
    {
        if (ch == '\n')
        {
            break;               // Done if we have read the EOL char
        }
        line[pos++] = ch;
    }
    line[pos] = '\0';
    return(pos);
}
```

```

// Checks if the entire string range is numeric, and if so
// returns the unsigned long value in the specified argument.
int parseInt32(char *line,
               const unsigned startPos,
               const unsigned endPos,
               unsigned long &longword)
{
    unsigned pos;
    char *tokenPos;           // Pointer to within line
    if (startPos >= endPos)
    {
        cerr << "Error: parseInt32, start =" << startPos << " end ="
              << endPos << endl;
        return(0);
    }

    for (pos = startPos; pos <= endPos; pos++)
    {
        if (!isxdigit(line[pos]))
        {
            return(0);           // String range is not numeric
        }
    }
    tokenPos = &line[startPos];
    istringstream(tokenPos, (endPos - startPos + 1)) >> hex >> longword;
    return(1);
}

// This counts the number of 1 bits found in a longword
unsigned Count1Bits(unsigned long word)
{
    unsigned OneBits = 0U;
    static const unsigned long bitMask(0x01UL);

    do
    {

```

```

    OneBits += word & bitMask;
    word >>= 1;
} while (word != 0UL);
return(OneBits);
}

// This counts the number of bit transitions within a loop
unsigned CountLoopDeltaBits(const unsigned long *buffer,
                           const unsigned bufferSize,
                           const unsigned deltaSize)
{
    unsigned index;
    unsigned transitions(0U);

    for (index = 0U; index < bufferSize; index++)
    {
        transitions += Count1Bits(buffer[index] ^
                                   buffer[(index + deltaSize) % bufferSize]);
    }
    return(transitions);
}

// The main line program
int main (int argc, char *argv[])
{
    parseStage stage(FIND_LP);           // Parsing stage
    char *inputFileNameAndPath;
    char *shortFileName;                 // Filename
    char *lineMarker;                    // Pointer into parsed line
    char line[LINE_BUFFER_LENGTH];       // Space for a line of text
    unsigned loopLines(0U);               // Lines of loop text
    unsigned modulo;                       // Modulo value
    unsigned currentModulo(0U);

    unsigned long programAddressSpace[LOOP_WORD_BUFFER_SIZE];

```

```

unsigned long deltaProgramAddrSpace(OUL);
unsigned long programDataSpace[LOOP_WORD_BUFFER_SIZE];
unsigned long totalProgramDataSpace(OUL);
unsigned long deltaProgramDataSpace(OUL);
unsigned long dataSpaceAddressOffsets[MAX_LOOP_WORDS];
unsigned long deltaDataSpaceAddressOffsets(OUL);
unsigned long deltaRdDataAddressOffsets;
unsigned long deltaWrDataAddressOffsets;
unsigned long readBits(OUL);
unsigned long writeBits(OUL);

#ifdef DEBUGGING
    unsigned long streamBase;          // iostream mode setting
#endif

inputFileNameAndPath = argv[1];

cout.precision(4);
if (argc != 2)
{
    cout << "argc is: " << argc << endl;
    return(argc);
}

#ifdef DEBUGGING
    cout << "Filename is: " << inputFileNameAndPath << endl;
#endif

// Open the file for reading
ifstream inFile(inputFileNameAndPath, ios::in);
if (!inFile)
{
    cerr << "Unable to open " << inputFileNameAndPath << endl;
    return(-1);
}

```

```

// Find the short filename, first trim the path
shortFileName = strtok(inputFileNameAndPath, SLASH_STRING);
do
{
    lineMarker = shortFileName;
    shortFileName = strtok(NULL, SLASH_STRING);
} while (shortFileName);

// And finally, the extension

shortFileName = strtok(lineMarker, DOT_STRING);

// Read data from the file and echo
while (getLine(inFile, line))
{
    switch (stage)
    {
        case FIND_LP:
            lineMarker = strstr(line, MODULO_REGISTER_LOAD);
            if (lineMarker)
            {
                // Record modulus register load value for later reference
                lineMarker = strstr(line, IMMEDIATE_LOAD);
                if (lineMarker)
                {
                    istrstream(++lineMarker) >> modulo;
                } else {
                    cerr << "Unable to parse modulo load, file: " <<
                        shortFileName << endl;
                    return(-1);
                }
            }
            if (strstr(line, LP_MARKER))
            {
                stage = IN_LP;
            }
        }
    }

```



```

        #ifdef DEBUGGING
            cout << "Got " << LP_MARKER << endl;
        #endif
    }
    break;

case IN_LP:
    if (strstr(line, END_LP_MARKER) ||
        strstr(line, ALT_END_LP_MARKER_1) ||
        strstr(line, ALT_END_LP_MARKER_2))
    {
        stage = PARSED_LP;

        #ifdef DEBUGGING
            cout << "Got " << END_LP_MARKER << endl;
        #endif
        break;
    }

    if (parseInt32(line,
        ADDRESS_FIELD_BEGINS,
        ADDRESS_FIELD_ENDS,
        programAddressSpace[loopLines]) &&
        parseInt32(line,
        DATA_FIELD_BEGINS,
        DATA_FIELD_ENDS,
        programDataSpace[loopLines]))
    {

        #ifdef DEBUGGING
            // Set output in hexadecimal notation
            streamBase = cout.setf(ios::hex, ios::basefield);
            cout.setf(ios::uppercase);
            cout << "PA: ";
            cout.fill('0');
            cout.width(8);

```

```

    cout << programAddressSpace[loopLines];
    cout << " PD: ";
    cout.fill('0');
    cout.width(8);
    cout << programDataSpace[loopLines] << endl;
    cout.setf(streamBase, ios::basefield);
#endif

// Now, check for data space address references
lineMarker = strstr(line, AR5_REFERENCE);
if (lineMarker)
{
    // We have found an A1 reference, now parse the offset
    lineMarker += strlen(AR5_REFERENCE);

    dataSpaceAddressOffsets[loopLines] = 0UL; // Default
    switch (*lineMarker)
    {
        case LEFT_BRACKET:
            istrstream(++lineMarker) >>
                dataSpaceAddressOffsets[loopLines];
            break;

        case PLUS_SIGN:
            // Modulo addressing, increment is 1
            dataSpaceAddressOffsets[loopLines] = currentModulo;
            currentModulo = (currentModulo + 1) % modulo;
            break;

        default:
            // Might still be a modulo store
            if (strstr(line, WRITE_DATA_INSTRUCTION) &&
                strstr(line, ABSOLUTE_STORE_MODULO))
            {
                // Yes, we have a modulo store
                dataSpaceAddressOffsets[loopLines] = currentModulo;
            }
    }
}

```

```

        currentModulo = (currentModulo + 1) % modulo;
    }
    break;
}

if (*lineMarker == LEFT_BRACKET)
{
    istrstream(++lineMarker) >>
        dataSpaceAddressOffsets[loopLines];
} else {
    dataSpaceAddressOffsets[loopLines] = 0UL;
}

// Read/write detection
if (strstr(line, READ_DATA_INSTRUCTION))
{
    readBits += BITS_PER_WORD;
}

if (strstr(line, WRITE_DATA_INSTRUCTION))
{
    writeBits += BITS_PER_WORD;
}

#ifdef DEBUGGING
    streamBase = cout.setf(ios::dec, ios::basefield);
    cout.setf(ios::uppercase);
    streamBase = cout.setf(ios::hex, ios::basefield);
    cout << " DA: ";
    cout.fill('0');
    cout.width(8);
    cout << dataSpaceAddressOffsets[loopLines] << endl;
#endif
}

#ifdef DEBUGGING

```

```

        cout.setf(streamBase, ios::basefield);
    #endif

    // Record the valid line!
    loopLines++;
}

break;

case PARSED_LP:
    // Just fall through, do nothing
    break;

default:
    cerr << "Error: Enumeration fault!" << endl;
    // Nothing more to do
    break;
}
}

// Report results
switch (stage)
{
    case FIND_LP:
        cerr << "Unable to find " << LP_MARKER << endl;
        break;

    case IN_LP:
        cerr << "Unable to find " << END_LP_MARKER << endl;
        break;

    case PARSED_LP:
        totalProgramDataSpace = loopLines * BITS_PER_WORD;

        if (loopLines > 1)

```

```

{
    deltaProgramAddrSpace = CountLoopDeltaBits(programAddressSpace,
        loopLines,
        1UL);
    deltaProgramDataSpace = CountLoopDeltaBits(programDataSpace,
        loopLines,
        1UL);
} else {
    deltaProgramAddrSpace = 0UL;
    deltaProgramDataSpace = 0UL;
}

// Compute data space address delta bits
deltaDataSpaceAddressOffsets =
    CountLoopDeltaBits(dataSpaceAddressOffsets,
        loopLines,
        1U);

if (readBits)
{
    deltaRdDataAddressOffsets = deltaDataSpaceAddressOffsets;
} else {
    deltaRdDataAddressOffsets = 0U;
}

if (writeBits)
{
    deltaWrDataAddressOffsets = deltaDataSpaceAddressOffsets;
} else {
    deltaWrDataAddressOffsets = 0U;
}

// Now we display the results
cout << "                File: " << shortFileName << endl;
cout << "                Loop Lines: " << loopLines << endl;
cout << "                Cycles: " << endl;

```

```

    cout << "          Current 1: " << endl;
    cout << "          Current 2: " << endl;
    cout << "          Current 3: " << endl;
    cout << "      Delta PA Bits: " << deltaProgramAddrSpace << endl;
    cout << "      Total PD Bits: " << totalProgramDataSpace << endl;
    cout << "      Delta PD Bits: " << deltaProgramDataSpace << endl;

    cout << "      Delta RA Bits: " << deltaRdDataAddressOffsets <<
        endl;
    cout << "      Delta WA Bits: " << deltaWrDataAddressOffsets <<
        endl;
    cout << "      Total RD Bits: " << readBits << endl;
    cout << "      Total WD Bits: " << writeBits << endl;
    // Cannot automate this easily, so insert 0, fill manually later
    cout << "      Delta RWD Bits: 0 " << endl;

    // Cannot automate this easily, so insert 0, fill manually later
    cout << "      ALU Src Bits: 0 " << endl;
    cout << "      ALU Dst Bits: 0 " << endl;
    // Bits changing in->out
    cout << "      Delta ALU Bits: 0 " << endl;
    cout << "      ALU Paths: 0 " << endl;
    cout << endl;
    break;

default:
    cerr << "Error: Enumeration fault!" << endl;
    break;
}

// Close the file and exit
inFile.close();
return(0);
}

```

B.3 C6211 Assembly Listfile Parsing Program

```

/*****
*
* FILENAME
*   parsec6x.cc
*
* DESCRIPTION
*   Parses the c6x listing file to obtain the required information.
*   Output format:
*
* File           = 8 characters for filename
* Loop Lines     = nbr of 32 bit instructions per loop
* I Groups       = Nbr of fetch packet instruction groups per loop
* Cycles         = cycles per loop (left blank, to be filled manually)
* Current1       = Current measurement 1 (left blank, to be filled manually)
* Current2       = Current measurement 2 (left blank, to be filled manually)
* Current3       = Current measurement 3 (left blank, to be filled manually)
* Delta PA Bits  = nbr of program space address bit changes per loop
* Total PD Bits  = Total number of program space bits processed per loop
* Delta PD Bits  = Number of program space data bit transitions per loop
* Delta RA Bits  = Number of data space read address bit transitions per
* Delta WA Bits  = Number of data space write address bit transitions per
* Total RD Bits  = Number of data space read bits per loop
* Total WD Bits  = Number of data space write bits per loop
*
*   The following fields are all entered manually:
* Delta RWD Bits = Number of data space data bit transitions per loop
* ALU Src Bits   = Number of ALU source bits per loop
* ALU Dst Bits   = Number of ALU destination bits per loop
* Delta ALU Bits = Number of ALU bit transitions per loop
* ALU Paths      = Number of ALU paths active
*
*=====
* History:
* Apr 28 2001   Gary Cameron
*   - Creation
*

```

```

*****/
#include <iostream.h>
#include <fstream.h>
#include <string.h>
#include <ctype.h>
#include <strstream.h>

// #define DEBUGGING

enum parseStage { FIND_LP, IN_LP, PARSED_LP};

const unsigned LINE_BUFFER_LENGTH(255U);

const unsigned MAX_LOOP_WORDS(16U);
const unsigned MAX_WORDS_PER_FETCH_PACKET(8U);
const unsigned LOOP_WORD_BUFFER_SIZE
    (MAX_LOOP_WORDS * MAX_WORDS_PER_FETCH_PACKET);

const unsigned BYTES_PER_WORD(4U);
const unsigned BITS_PER_WORD(32U);

const unsigned long NOP_WORD(0UL);           // Used to align fetch packet

const char *SLASH_STRING ("/");              // Used to tokenize paths
const char *DOT_STRING (".");

const char *LP_MARKER ("Lp:");               // Marks beginning of test loop
const char *ALT_LP_MARKER ("_loop:");

const char *END_LP_MARKER (".text");         // Marks the end of test loop
const char *ALT_END_LP_MARKER ("b          .s1      _Idle_loop");

const char *PARALLEL_INSTRUCTION_BARS ("||");
const char *A1_REFERENCE ("*a1");
const char LEFT_SQUARE_BRACE ('[');
const char RIGHT_SQUARE_BRACE (']');

```



```

const char *READ_DATA_INSTRUCTION ("ldw");
const char *WRITE_DATA_INSTRUCTION ("stw");

const unsigned ADDRESS_FIELD_BEGINS = 9;
const unsigned ADDRESS_FIELD_ENDS = 16;

const unsigned DATA_FIELD_BEGINS = 18;
const unsigned DATA_FIELD_ENDS = 25;

// Return a line of text from the file
// Returns number of characters read, including the end of line
unsigned getLine(ifstream &file,
                char *line)
{
    unsigned pos;                                // Position within the string
    char ch;

    pos = 0U;

    while (file.get(ch))                          // Exit loop if (EOF)
    {
        if (ch == '\n')
        {
            break;                                // Done if read the EOL char
        }
        line[pos++] = ch;
    }
    line[pos] = '\0';
    return(pos);
}

// Checks if the entire string range is numeric, and if so returns the
// unsigned long value in the specified argument.
int parseInt32(char *line,
               const unsigned startPos,

```

```

        const unsigned endPos,
        unsigned long &longword)
{
    unsigned pos;
    char *tokenPos;                //Pointer to within line
    if (startPos >= endPos)
    {
        cerr << "Error: parseInt32, start =" << startPos << " end =" <<
            endPos << endl;
        return(0);
    }

    for (pos = startPos; pos <= endPos; pos++)
    {
        if (!isxdigit(line[pos]))
        {
            return(0);                // String range is not numeric
        }
    }
    tokenPos = &line[startPos];
    istringstream(tokenPos, (endPos - startPos + 1)) >> hex >> longword;
    return(1);
}

// This counts the number of 1 bits found in a longword
unsigned Count1Bits(unsigned long word)
{
    unsigned OneBits = 0U;
    static const unsigned long bitMask(0x01UL);

    do
    {
        OneBits += word & bitMask;
        word >>= 1;
    } while (word != 0UL);
    return(OneBits);
}

```

```

}

// This counts the number of bit transitions within a loop
unsigned CountLoopDeltaBits(const unsigned long *buffer,
                           const unsigned bufferSize,
                           const unsigned deltaSize)
{
    unsigned index;
    unsigned transitions(0U);

    for (index = 0U; index < bufferSize; index++)
    {
        transitions += Count1Bits(buffer[index] ^
                                   buffer[(index + deltaSize) % bufferSize]);
    }
    return(transitions);
}

// The main line program
int main (int argc, char *argv[])
{
    parseStage stage(FIND_LP);           // Parsing stage
    char *inputFileNameAndPath;
    char *shortFileName;                 // Filename
    char *lineMarker;                    // Pointer into parsed line
    char line[LINE_BUFFER_LENGTH];       // Space for a line of text
    unsigned loopLines(0U);               // Lines of loop text

    unsigned long programAddressSpace[LOOP_WORD_BUFFER_SIZE];
    unsigned long totalFetchPackets(0UL);
    unsigned long deltaProgramAddrSpace(0UL);
    unsigned long programDataSpace[LOOP_WORD_BUFFER_SIZE];
    unsigned long totalProgramDataSpace(0UL);
    unsigned long deltaProgramDataSpace(0UL);
    unsigned long dataSpaceAddressOffsets[MAX_LOOP_WORDS];

```

```

unsigned long deltaDataSpaceAddressOffsets(0UL);
unsigned long deltaRdDataAddressOffsets;
unsigned long deltaWrDataAddressOffsets;
unsigned long readBits(0UL);
unsigned long writeBits(0UL);

unsigned iGroups(0U);                // Instruction groups per loop

#ifdef DEBUGGING
    unsigned long streamBase;        // iostream mode setting
#endif

inputFileNameAndPath = argv[1];

cout.precision(4);
if (argc != 2)
{
    cout << "argc is: " << argc << endl;
    return(argc);
}

#ifdef DEBUGGING
    cout << "Filename is: " << inputFileNameAndPath << endl;
#endif

// Open the file for reading
ifstream inFile(inputFileNameAndPath, ios::in);
if (!inFile)
{
    cerr << "Unable to open " << inputFileNameAndPath << endl;
    return(-1);
}

// Find the short filename, first trim the path
shortFileName = strtok(inputFileNameAndPath, SLASH_STRING);
do

```

```
{
    lineMarker = shortFileName;
    shortFileName = strtok(NULL, SLASH_STRING);
} while (shortFileName);

// And finally, the extension

shortFileName = strtok(lineMarker, DOT_STRING);

// Read data from the file and echo
while (getLine(inFile,line))
{
    switch (stage)
    {
        case FIND_LP:
            if (strstr(line, LP_MARKER) || strstr(line, ALT_LP_MARKER))
            {
                stage = IN_LP;

                #ifdef DEBUGGING
                    cout << "Got " << LP_MARKER << endl;
                #endif
            }
            break;

        case IN_LP:
            if (strstr(line, END_LP_MARKER) ||
                strstr(line, ALT_END_LP_MARKER))
            {
                stage = PARSED_LP;

                #ifdef DEBUGGING
                    cout << "Got " << END_LP_MARKER << endl;
                #endif
            } else {
                if (parseInt32(line,
```

```

        ADDRESS_FIELD_BEGINS,
        ADDRESS_FIELD_ENDS,
        programAddressSpace[loopLines]) &&
    parseInt32(line,
        DATA_FIELD_BEGINS,
        DATA_FIELD_ENDS,
        programDataSpace[loopLines]))
{
    // Kludge to fix fetch packet alignment issue
    if (loopLines)
    {
        if ((programAddressSpace[loopLines] -
            programAddressSpace[loopLines-1]) != BYTES_PER_WORD)
        {
            // Found alignment issue, add the missing NOP
            programAddressSpace[loopLines+1] =
                programAddressSpace[loopLines];
            programDataSpace[loopLines+1] =
                programDataSpace[loopLines];
            programAddressSpace[loopLines] -= BYTES_PER_WORD;
            programDataSpace[loopLines] = NOP_WORD;
#ifdef DEBUGGING
            // Set output in hexadecimal notation
            streamBase = cout.setf(ios::hex, ios::basefield);
            cout.setf(ios::uppercase);
            cout << "PA: ";
            cout.fill('0');
            cout.width(8);
            cout << programAddressSpace[loopLines];
            cout << " PD: ";
            cout.fill('0');
            cout.width(8);
            cout << programDataSpace[loopLines] << endl;
            cout.setf(streamBase, ios::basefield);
#endif
            loopLines++;

```

```

    }
}

#ifdef DEBUGGING
    // Set output in hexadecimal notation
    streamBase = cout.setf(ios::hex, ios::basefield);
    cout.setf(ios::uppercase);
    cout << "PA: ";
    cout.fill('0');
    cout.width(8);
    cout << programAddressSpace[loopLines];
    cout << " PD: ";
    cout.fill('0');
    cout.width(8);
    cout << programDataSpace[loopLines] << endl;
    cout.setf(streamBase, ios::basefield);
#endif

// Now, check for data space address references
// Assumption: always have exactly one A1_REFERENCE
// per iGroup
lineMarker = strstr(line, A1_REFERENCE);
if (lineMarker)
{
    // We have found an A1 reference, now parse the offset
    lineMarker += strlen(A1_REFERENCE);
    if (*lineMarker == LEFT_SQUARE_BRACE)
    {
        istrstream(++lineMarker) >>
            dataSpaceAddressOffsets[iGroups];
    } else {
        dataSpaceAddressOffsets[iGroups] = 0UL;
    }

    // Read/write detection

```

```

    if (strstr(line, READ_DATA_INSTRUCTION))
    {
        readBits += BITS_PER_WORD;
    }

    if (strstr(line, WRITE_DATA_INSTRUCTION))
    {
        writeBits += BITS_PER_WORD;
    }

#ifdef DEBUGGING
    streamBase = cout.setf(ios::dec, ios::basefield);
    cout.setf(ios::uppercase);
    cout << "iGroups: ";
    cout.fill(' ');
    cout.width(3);
    cout << iGroups;
    streamBase = cout.setf(ios::hex, ios::basefield);
    cout << " DA: ";
    cout.fill('0');
    cout.width(8);
    cout << dataSpaceAddressOffsets[iGroups] << endl;
#endif
}

#ifdef DEBUGGING
    cout.setf(streamBase, ios::basefield);
#endif

// Look for lack of parallel bars,
// denoting new instruction group
if (!strstr(line, PARALLEL_INSTRUCTION_BARS))
{
    iGroups++;
}

```



```
        // Record the valid line!
        loopLines++;
    }

    // Here is where we collect the data

}
break;

case PARSED_LP:
    // Just fall through, do nothing
    break;

default:
    cerr << "Error: Enumeration fault!" << endl;
    // Nothing more to do
    break;
}
}

// Report results
switch (stage)
{
    case FIND_LP:
        cerr << "Unable to find " << LP_MARKER << endl;
        break;

    case IN_LP:
        cerr << "Unable to find " << END_LP_MARKER << endl;
        break;

    case PARSED_LP:
        // Pad the last missing fetch packet with NOPs
        while (loopLines % MAX_WORDS_PER_FETCH_PACKET)
```

```

{
    programAddressSpace[loopLines] =
        programAddressSpace[loopLines-1] + BYTES_PER_WORD;
    programDataSpace[loopLines] = NOP_WORD;
#ifdef DEBUGGING
        cout << "Pad ";
        // Set output in hexadecimal notation
        streamBase = cout.setf(ios::hex, ios::basefield);
        cout.setf(ios::uppercase);
        cout << "PA: ";
        cout.fill('0');
        cout.width(8);
        cout << programAddressSpace[loopLines];
        cout << " PD: ";
        cout.fill('0');
        cout.width(8);
        cout << programDataSpace[loopLines] << endl;
        cout.setf(streamBase, ios::basefield);
    #endif
    loopLines++;
}

// Compute the remaining results
totalFetchPackets = loopLines / MAX_WORDS_PER_FETCH_PACKET;
totalProgramDataSpace = loopLines * BITS_PER_WORD;

if (totalFetchPackets > 1)
{
    deltaProgramAddrSpace = CountLoopDeltaBits(programAddressSpace,
        loopLines,
        MAX_WORDS_PER_FETCH_PACKET);
    deltaProgramDataSpace = CountLoopDeltaBits(programDataSpace,
        loopLines,
        MAX_WORDS_PER_FETCH_PACKET);
} else {
    deltaProgramAddrSpace = OUL;

```

```

    deltaProgramDataSpace = 0UL;
}

// Compute data space address delta bits
deltaDataSpaceAddressOffsets =
    CountLoopDeltaBits(dataSpaceAddressOffsets,
                        iGroups,
                        1U);

if (readBits)
{
    deltaRdDataAddressOffsets = deltaDataSpaceAddressOffsets;
} else {
    deltaRdDataAddressOffsets = 0U;
}

if (writeBits)
{
    deltaWrDataAddressOffsets = deltaDataSpaceAddressOffsets;
} else {
    deltaWrDataAddressOffsets = 0U;
}

// Now we display the results
cout << "           File: " << shortFileName << endl;
cout << "       Loop Lines: " << loopLines << endl;
cout << "         I Groups: " << iGroups << endl;
cout << "         Cycles: " << endl;
cout << "       Current 1: " << endl;
cout << "       Current 2: " << endl;
cout << "       Current 3: " << endl;
cout << "Total Fetch Packets: " << totalFetchPackets << endl;
cout << "       Delta PA Bits: " << deltaProgramAddrSpace <<
    endl;
cout << "       Total PD Bits: " << totalProgramDataSpace <<
    endl;

```

```

        cout << "          Delta PD Bits: " << deltaProgramDataSpace <<
            endl;

        cout << "          Delta RA Bits: " << deltaRdDataAddressOffsets <<
            endl;
        cout << "          Delta WA Bits: " << deltaWrDataAddressOffsets <<
            endl;

        cout << "          Total RD Bits: " << readBits << endl;
        cout << "          Total WD Bits: " << writeBits << endl;
        // Cannot automate this easily, so just insert 0,
        // fill manually later
        cout << "          Delta RWD Bits: 0 " << endl;

        // Cannot automate this easily, so insert 0, fill manually later
        cout << "          ALU Src Bits: 0 " << endl;
        cout << "          ALU Dst Bits: 0 " << endl;
        // Bits changing in->out
        cout << "          Delta ALU Bits: 0 " << endl;
        cout << "          ALU Paths: 0 " << endl;
        cout << endl;
        break;

    default:
        cerr << "Error: Enumeration fault!" << endl;
        break;
}

// Close the file and exit
inFile.close();
return(0);
}

```

Appendix C

Octave Post Processing Files

These files were used to further process the raw results into tabular form, suitable for analysis using SPSS.

C.1 56301 Results Processing

C.1.1 Raw Result to Tabular Data Conversion

```
## [raw] = read_power (filename)
##
## Reads the parsed text format data generated from parse56k.exe or
## parsec6x.exe or parsec54.exe, with some manually added fields.
##
## [raw] = read_power(filename, raw_file)
##
## if raw_file exists and is non zero, an output file is written
## with the same name as filename, appended with _raw with the .txt
## extension.
##
## If the first character of the "Filename:" field is an E then the
## data space is assumed to reside in external memory. Since there
## are three regressions for each test, there will be three rows of
## data generated for every test. The columns of the output matrix
## are arranged as follows:
```

```
##
## Column      Name                      Description
##   1         L.                      Loop Lines
##   2         C.                      Loop Cycles
##   3         Delta_P_A                Program space address line transitions
##   4         Total_P_D                Program space data lines (loop lines)
##   5         Delta_P_D                Program space data line transitions
##   6         Delta_D_A                Data space address line transitions
##
##   7         Total_D_R_I              Data space internal memory read bits
##   8         Delta_D_R_I              Data space internal memory read bits
##   9         Total_D_W_I              Data space internal memory write bits
##  10         Delta_D_W_I              Data space internal memory write bits
##
##  11         Total_D_R_E              Data space external memory read bits
##  12         Delta_D_R_E              Data space external memory read bits
##  13         Total_D_W_E              Data space external memory write bits
##  14         Delta_D_W_E              Data space external memory write bits
##  15         Total_A_S                Total ALU source bits
##  16         Delta_A_S                Delta ALU source bits
##  17         Total_A_D                Total ALU destination bits
##  18         Current                  Power consumption
##
## Author: Gary Cameron
## Date: Oct. 3 2001
## Keywords: DSP 56k c62 c54 Thesis scaled read power
```

```
function raw = read_power(filename,...)
```

```
    if (nargin == 1)
        raw_file = 0;
    else
        raw_file = va_arg();
    endif
```

```

file_id = fopen(filename, "r", "native");

while (feof(file_id(1)) == 0) && (ferror(file_id(1)) == "")

    # while debugging == 0
    original_filename = fscanf(file_id(1), '%s %s', 1);

    if (feof(file_id(1)) ~= 0) || (ferror(file_id(1)) ~= "")
        break;
    end

    Lines = fscanf(file_id(1), '%s %s %d', 1);
    Cycles = fscanf(file_id(1), '%s %d', 1);
    Current = fscanf(file_id(1), '%s %s %f', 3);
    Current = Current';
    Delta_P_A = fscanf(file_id(1), '%s %s %s %d', 1);
    Total_P_D = fscanf(file_id(1), '%s %s %s %d', 1);
    Delta_P_D = fscanf(file_id(1), '%s %s %s %d', 1);
    Delta_D_R_A = fscanf(file_id(1), '%s %s %s %d', 1);
    Delta_D_W_A = fscanf(file_id(1), '%s %s %s %d', 1);

    if (Delta_D_R_A == 0)
        Delta_D_A = Delta_D_W_A;
    else
        Delta_D_A = Delta_D_R_A;
    end

    # Default as zero

    Total_D_R_I = 0;
    Total_D_W_I = 0;
    Delta_D_R_I = 0;
    Delta_D_W_I = 0;

```

```

Total_D_R_E = 0;
Total_D_W_E = 0;
Delta_D_R_E = 0;
Delta_D_W_E = 0;

if (original_filename(1) == "E")
    Total_D_R_E = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    Total_D_W_E = fscanf(file_id(1), '%*s %*s %*s %d', 1);

    if (Delta_D_W_A == 0)
        Delta_D_R_E = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    else
        Delta_D_W_E = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    end
else
    Total_D_R_I = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    Total_D_W_I = fscanf(file_id(1), '%*s %*s %*s %d', 1);

    if (Delta_D_W_A == 0)
        Delta_D_R_I = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    else
        Delta_D_W_I = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    end
endif

Total_A_S = fscanf(file_id(1), '%*s %*s %*s %d', 1);
Total_A_D = fscanf(file_id(1), '%*s %*s %*s %d', 1);
Delta_A_S = fscanf(file_id(1), '%*s %*s %*s %d', 1);

# Add each replicant to matrix
for count = 1:length(Current),
    # Create a new matrix row
    clear current_row;

```



```

    current_row = [Lines, Cycles, Delta_P_A, Total_P_D, \
        Delta_P_D, Delta_D_A, Total_D_R_I, Delta_D_R_I, \
        Total_D_W_I, Delta_D_W_I, Total_D_R_E, \
        Delta_D_R_E, Total_D_W_E, Delta_D_W_E, \
        Total_A_S, Delta_A_S, Total_A_D, \
        Current(count)];

    if exist("raw") == 0
        raw = current_row;
    else
        raw = [raw;current_row];
    end
end

                                # This is not used
ALU_Paths = fscanf(file_id(1), '%*s %*s %s',1);
end
fclose(file_id);

if (raw_file)

    filename_split = split(filename, ".");
    raw_filename = strcat(filename_split(1,:), "_raw.txt");
    raw_file_id = fopen(raw_filename, "w", "native");

    fputs(raw_file_id, "L. Cyc. D_PA T_PD D_PD D_DA ");
    fputs(raw_file_id, "T_DRI D_DRI T_DWI D_DWI ");
    fputs(raw_file_id, "T_DRE D_DRE T_DWE D_DWE T_AS D_AS T_AD ");
    fputs(raw_file_id, "Current\n");

    fputs(raw_file_id, "-- ---- ---- ---- ---- ");
    fputs(raw_file_id, "----- ----- ----- ");
    fputs(raw_file_id, "----- ----- ----- ---- ---- ");
    fputs(raw_file_id, "-----\n");

```

```
        fprintf(raw_file_id,  
"%2.0f %3.0f %4.0f %4.0f %4.0f %4.0f %4.0f %5.0f %5.0f %5.0f  
%5.0f %5.0f %5.0f %5.0f %5.0f %4.0f %4.0f %8.3f\n", raw');  
        fclose(raw_file_id);  
    endif  
endfunction
```

C.1.2 Power Conversion Scaling

```
## [scaled] = scale_power (raw, voltage, mhz_clk_speed, scaled_file)
##
## Deletes the first column of the raw matrix, and converts the
## last column from average current to scaled power. The zero
## reference reading (lowest reading in data) is subtracted off.
##
## The columns of the output matrix are arranged as follows:
##
## Column      Name      Description
## 1      Delta_P_A      Prog. space address line transitions
## 2      Total_P_D      Program space data lines (loop lines)
## 3      Delta_P_D      Program space data line transitions
## 4      Delta_D_A      Data space address line transitions
##
## 5      Total_D_R_I      Data space internal memory read bits
## 6      Delta_D_R_I      Data space internal memory read bits
## 7      Total_D_W_I      Data space internal memory write bits
## 8      Delta_D_W_I      Data space internal memory write bits
##
## 9      Total_D_R_E      Data space external memory read bits
## 10     Delta_D_R_E      Data space external memory read bits
## 11     Total_D_W_E      Data space external memory write bits
## 12     Delta_D_W_E      Data space external memory write bits
## 13     Total_A_S      Total ALU source bits
## 14     Delta_A_S      Delta ALU source bits
## 15     Total_A_D      Total ALU destination bits
## 16     Energy      microjoules
## 17     Current      Power consumption
##
##
## Author: Gary Cameron
## Date: Oct. 25 2001
```

```

## Keywords: DSP 56k c62 c54 Thesis scaled read power

function scaled = scale_power(raw, voltage, mhz_clk_speed,...)

    scaled = zeros(rows(raw),columns(raw)-2);

    if (nargin == 3)
        scaled_file = 0;
    else
        scaled_file = 1;
        scaled_filename = va_arg();
    endif

    for count = 3:columns(raw),
        scaled(:,count-2) = raw(:,count);
    end

    size(raw)
    size(scaled)

    sorted = sort(raw(:,columns(raw)));
    baseline = sorted(1)

    scaled(:,columns(scaled)) = voltage .* raw(:,2) .* \
        (raw(:,columns(raw)) .- baseline) ./ \
        (min(raw(:,2),raw(:,1)) .* \
        mhz_clk_speed);

    if (scaled_file)
        scaled_file_id = fopen(scaled_filename, "w", "native");

        fputs(scaled_file_id, "D_PA T_PD D_PD D_DA ");
        fputs(scaled_file_id,"T_DRI D_DRI T_DWI D_DWI ");
        fputs(scaled_file_id,"T_DRE D_DRE T_DWE D_DWE T_AS D_AS T_AD ");
    end

```

```
fputs(scaled_file_id, " Energy\n");

fputs(scaled_file_id, "---- ---- ---- ");
fputs(scaled_file_id, "----- ----- ");
fputs(scaled_file_id, "----- ----- ---- ");
fputs(scaled_file_id, " -----\n");

fprintf(scaled_file_id,
"%3.0f %4.0f %4.0f %4.0f %5.0f %4.0f %5.0f %5.0f %5.0f
%5.0f %5.0f %5.0f %4.0f %4.0f %8.4f\n", scaled');
fclose(scaled_file_id);
endif
endfunction
```

C.2 C5401 Results Processing

C.2.1 Raw Result to Tabular Data Conversion

```
## [raw] = read_power (filename)
##
## Reads the parsed text format data generatedd from parse56k.exe
## or parsec6x.exe or parsec54.exe, with some manually added
## fields.
##
## [raw] = read_power(filename, raw_file)
##
## if raw_file exists and is non zero, an output file is written
## with the same name as filename, appended with _raw with the
## .txt extension.
##
## If the first character if the "Filename:" field is an E then the
## data space is assumed to reside in external memory. Since there
## three regressions for each test, there will be three rows of data
## are generated for every test. The columns of the output matrix
## are arranged as follows:
##
## Column      Name                      Description
## 1           L.                        Loop Lines
## 2           C.                        Loop Cycles
## 3           Delta_P_A                 Prog. space address line transitions
## 4           Total_P_D                 Prog. space data lines (loop lines)
## 5           Delta_P_D                 Prog. space data line transitions
## 6           Delta_D_A                 Data space address line transitions
##
## 7           Total_D_R_I               Data space internal memory read bits
## 8           Delta_D_R_I               Data space internal memory read bits
## 9           Total_D_W_I               Data space internal memory write bits
## 10          Delta_D_W_I               Data space internal memory write bits
```

```
##
## 11      Total_D_R_E      Data space external memory read bits
## 12      Delta_D_R_E      Data space external memory read bits
## 13      Total_D_W_E      Data space external memory write bits
## 14      Delta_D_W_E      Data space external memory write bits
## 15      Total_A_S        Total ALU source bits
## 16      Delta_A_S        Delta ALU source bits
## 17      Total_A_D        Total ALU destination bits
## 18      Current          Power consumption
##
```

```
## Author: Gary Cameron
## Date: Oct. 3 2001
## Keywords: DSP 56k c62 c54 Thesis scaled read power
```

```
function raw = read_power(filename,...)

    if (nargin == 1)
        raw_file = 0;
    else
        raw_file = va_arg();
    endif

    file_id = fopen(filename, "r", "native");

    while (feof(file_id(1)) == 0) && (ferror(file_id(1)) == "")

        # while debugging == 0
        original_filename = fscanf(file_id(1), '%*s %s', 1);

        if (feof(file_id(1)) ~= 0) || (ferror(file_id(1)) ~= "")
            break;
        end
    end
```

```

Lines = fscanf(file_id(1), '%*s %*s %d', 1);
Cycles = fscanf(file_id(1), '%*s %d', 1);
Current = fscanf(file_id(1), '%*s %*s %f', 3);
Current = Current';
Delta_P_A = fscanf(file_id(1), '%*s %*s %*s %d', 1);
Total_P_D = fscanf(file_id(1), '%*s %*s %*s %d', 1);
Delta_P_D = fscanf(file_id(1), '%*s %*s %*s %d', 1);
Delta_D_R_A = fscanf(file_id(1), '%*s %*s %*s %d', 1);
Delta_D_W_A = fscanf(file_id(1), '%*s %*s %*s %d', 1);

if (Delta_D_R_A == 0)
    Delta_D_A = Delta_D_W_A;
else
    Delta_D_A = Delta_D_R_A;
end

                                # Default as zero

Total_D_R_I = 0;
Total_D_W_I = 0;
Delta_D_R_I = 0;
Delta_D_W_I = 0;

Total_D_R_E = 0;
Total_D_W_E = 0;
Delta_D_R_E = 0;
Delta_D_W_E = 0;

if (original_filename(1) == "E")
    Total_D_R_E = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    Total_D_W_E = fscanf(file_id(1), '%*s %*s %*s %d', 1);

    if (Delta_D_W_A == 0)
        Delta_D_R_E = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    else

```



```

        Delta_D_W_E = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    end
else
    Total_D_R_I = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    Total_D_W_I = fscanf(file_id(1), '%*s %*s %*s %d', 1);

    if (Delta_D_W_A == 0)
        Delta_D_R_I = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    else
        Delta_D_W_I = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    end
endif

Total_A_S = fscanf(file_id(1), '%*s %*s %*s %d', 1);
Total_A_D = fscanf(file_id(1), '%*s %*s %*s %d', 1);
Delta_A_S = fscanf(file_id(1), '%*s %*s %*s %d', 1);

# Add each replicant to matrix
for count = 1:length(Current),
    # Create a new matrix row
    clear current_row;

    current_row = [Lines, Cycles, Delta_P_A, Total_P_D, \
        Delta_P_D, Delta_D_A, Total_D_R_I, Delta_D_R_I, \
        Total_D_W_I, Delta_D_W_I, Total_D_R_E, \
        Delta_D_R_E, Total_D_W_E, Delta_D_W_E, \
        Total_A_S, Delta_A_S, Total_A_D, \
        Current(count)];

    if exist("raw") == 0
        raw = current_row;
    else
        raw = [raw;current_row];
    end
end

```

```

end

                                # This is not used
    ALU_Paths = fscanf(file_id(1), '%*s %*s %s',1);
end
fclose(file_id);

if (raw_file)

    filename_split = split(filename, ".");
    raw_filename = strcat(filename_split(1,:), "_raw.txt");
    raw_file_id = fopen(raw_filename, "w", "native");

    fputs(raw_file_id, "L. Cyc. D_PA T_PD D_PD D_DA ");
    fputs(raw_file_id, "T_DRI D_DRI T_DWI D_DWI ");
    fputs(raw_file_id, "T_DRE D_DRE T_DWE D_DWE T_AS D_AS T_AD ");
    fputs(raw_file_id, "Current\n");

    fputs(raw_file_id, "-- ---- ---- ---- ---- ");
    fputs(raw_file_id, "----- ");
    fputs(raw_file_id, "----- ");
    fputs(raw_file_id, "-----\n");

    fprintf(raw_file_id,
        "%2.0f %3.0f %4.0f %4.0f %4.0f %4.0f %4.0f %5.0f %5.0f\n",
        raw');
    fclose(raw_file_id);
endif
endfunction

```

C.2.2 Power Conversion Scaling

```
## [scaled] = scale_power (raw, voltage, mhz_clk_speed, scaled_file)
##
## Deletes the first column of the raw matrix, and converts the
## last column from average current to scaled power. The zero
## reference reading (lowest reading in data) is subtracted off.
##
## The columns of the output matrix are arranged as follows:
##
## Column      Name                      Description
##  1          Delta_P_A                  Prog. space address line transitions
##  2          Total_P_D                   Prog. space data lines (loop lines)
##  3          Delta_P_D                   Prog. space data line transitions
##  4          Delta_D_A                   Data space address line transitions
##
##  5          Total_D_R_I                 Data space internal memory read bits
##  6          Delta_D_R_I                 Data space internal memory read bits
##  7          Total_D_W_I                 Data space internal memory write bits
##  8          Delta_D_W_I                 Data space internal memory write bits
##
##  9          Total_D_R_E                 Data space external memory read bits
## 10          Delta_D_R_E                 Data space external memory read bits
## 11          Total_D_W_E                 Data space external memory write bits
## 12          Delta_D_W_E                 Data space external memory write bits
## 13          Total_A_S                   Total ALU source bits
## 14          Delta_A_S                   Delta ALU source bits
## 15          Total_A_D                   Total ALU destination bits
## 16          Energy                      microjoules
## 17          Current                     Power consumption
##
##
## Author: Gary Cameron
## Date: Oct. 25 2001
```

```

## Keywords: DSP 56k c62 c54 Thesis scaled read power

function scaled = scale_power(raw, voltage, mhz_clk_speed,...)

    scaled = zeros(rows(raw),columns(raw)-2);

    if (nargin == 3)
        scaled_file = 0;
    else
        scaled_file = 1;
        scaled_filename = va_arg();
    endif

    for count = 3:columns(raw),
        scaled(:,count-2) = raw(:,count);
    end

    size(raw)
    size(scaled)

    sorted = sort(raw(:,columns(raw)));
    baseline = sorted(1)

    scaled(:,columns(scaled)) = voltage .* raw(:,2) .* \
        (raw(:,columns(raw)) .- baseline) ./ \
        (min(raw(:,2),raw(:,1)) .* mhz_clk_speed);

    if (scaled_file)
        scaled_file_id = fopen(scaled_filename, "w", "native");

        fputs(scaled_file_id, "D_PA T_PD D_PD D_DA ");
        fputs(scaled_file_id,"T_DRI D_DRI T_DWI D_DWI ");
        fputs(scaled_file_id,"T_DRE D_DRE T_DWE D_DWE T_AS D_AS T_AD ");
        fputs(scaled_file_id," Energy\n");
    end

```

```
fputs(scaled_file_id, "---- ---- ---- ");
fputs(scaled_file_id, "----- ");
fputs(scaled_file_id, "----- ----- ---- ");
fputs(scaled_file_id, " -----\\n");

fprintf(scaled_file_id,
        "%3.0f %4.0f %4.0f %4.0f %5.0f %4.0f %5.0f %5.0f %5.0f %5.0f
        %5.0f %5.0f %5.0f %4.0f %4.0f %8.4f\\n", scaled');
fclose(scaled_file_id);
endif
endfunction
```

C.3 C6211 Results Processing

C.3.1 Raw Result to Tabular Data Conversion

```
## [raw] = read_c6x_power (filename)
##
## Reads the parsed text format data generatedd from parsec6x.exe
## with some manually added fields.
##
## [raw] = read_c6x_power(filename, raw_file)
##
## if raw_file exists and is non zero, an output file is written
## with the same name as filename, appended with _raw with the .txt
## extension.
##
## If the first character if the "Filename:" field is an E then the
## data space is assumed to reside in external memory. Since there
## are three regressions for each test, there will be three rows of
## data generated for every test. The columns of the output matrix
## are arranged as follows:
##
## Column      Name                      Description
## 1           L.                        Loop Lines
## 2           C.                        Loop Cycles
## 3           Delta_P_A                 Prog. space address line transitions
## 4           Total_P_D                 Prog. space data lines (loop lines)
## 5           Delta_P_D                 Prog. space data line transitions
## 6           Delta_D_A                 Data space address line transitions
##
## 7           Total_D_R_I               Data space internal memory read bits
## 8           Delta_D_R_I               Data space internal memory read bits
## 9           Total_D_W_I               Data space internal memory write bits
## 10          Delta_D_W_I               Data space internal memory write bits
##
```

```
## 11      Total_D_R_E      Data space external memory read bits
## 12      Delta_D_R_E      Data space external memory read bits
## 13      Total_D_W_E      Data space external memory write bits
## 14      Delta_D_W_E      Data space external memory write bits
## 15      Total_A_S        Total ALU source bits
## 16      Delta_A_S        Delta ALU source bits
## 17      Total_A_D        Total ALU destination bits
## 18      Current          Power consumption
##
```

```
## Author: Gary Cameron
## Date: Oct. 3 2001
## Modified: Nov 9 2001
## Keywords: DSP c62 c6x Thesis scaled read power
```

```
function raw = read_c6x_power(filename,...)

    if (nargin == 1)
        raw_file = 0;
    else
        raw_file = va_arg();
    endif

    file_id = fopen(filename, "r", "native");

    while (feof(file_id(1)) == 0) && (ferror(file_id(1)) == "")

        # while debugging == 0
        original_filename = fscanf(file_id(1), '%*s %s', 1);

        if (feof(file_id(1)) ~= 0) || (ferror(file_id(1)) ~= "")
            break;
        end
```

```

Lines = fscanf(file_id(1), '%*s %*s %d', 1);
I_Groups = fscanf(file_id(1), '%*s %*s %*s %d', 1);
Cycles = fscanf(file_id(1), '%*s %d', 1);
Current = fscanf(file_id(1), '%*s %*s %f', 3);
Current = Current';
Total_F_P = fscanf(file_id(1), '%*s %*s %*s %d', 1);
Delta_P_A = fscanf(file_id(1), '%*s %*s %*s %d', 1);
Total_P_D = fscanf(file_id(1), '%*s %*s %*s %d', 1);
Delta_P_D = fscanf(file_id(1), '%*s %*s %*s %d', 1);
Delta_D_R_A = fscanf(file_id(1), '%*s %*s %*s %d', 1);
Delta_D_W_A = fscanf(file_id(1), '%*s %*s %*s %d', 1);

if (Delta_D_R_A == 0)
    Delta_D_A = Delta_D_W_A;
else
    Delta_D_A = Delta_D_R_A;
end

                                # Default as zero

Total_D_R_I = 0;
Total_D_W_I = 0;
Delta_D_R_I = 0;
Delta_D_W_I = 0;

Total_D_R_E = 0;
Total_D_W_E = 0;
Delta_D_R_E = 0;
Delta_D_W_E = 0;

if (original_filename(1) == "E")
    Total_D_R_E = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    Total_D_W_E = fscanf(file_id(1), '%*s %*s %*s %d', 1);

    if (Delta_D_W_A == 0)

```



```

        Delta_D_R_E = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    else
        Delta_D_W_E = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    end
else
    Total_D_R_I = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    Total_D_W_I = fscanf(file_id(1), '%*s %*s %*s %d', 1);

    if (Delta_D_W_A == 0)
        Delta_D_R_I = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    else
        Delta_D_W_I = fscanf(file_id(1), '%*s %*s %*s %d', 1);
    end
endif

Total_A_S = fscanf(file_id(1), '%*s %*s %*s %d', 1);
Total_A_D = fscanf(file_id(1), '%*s %*s %*s %d', 1);
Delta_A_S = fscanf(file_id(1), '%*s %*s %*s %d', 1);

# Add each replicant to matrix
for count = 1:length(Current),
    # Create a new matrix row
    clear current_row;

    current_row = [Lines, Cycles, Delta_P_A, Total_P_D, \
        Delta_P_D, Delta_D_A, Total_D_R_I, Delta_D_R_I, \
        Total_D_W_I, Delta_D_W_I, Total_D_R_E, Delta_D_R_E, \
        Total_D_W_E, Delta_D_W_E, Total_A_S, Delta_A_S, \
        Total_A_D, Current(count)];

    if exist("raw") == 0
        raw = current_row;
    else
        raw = [raw;current_row];
    end
endfor

```

```

        end
    end

        # This is not used
    ALU_Paths = fscanf(file_id(1), '%*s %*s %s',1);
end
fclose(file_id);

if (raw_file)

    filename_split = split(filename, ".");
    raw_filename = strcat(filename_split(1,:), "_raw.txt");
    raw_file_id = fopen(raw_filename, "w", "native");

    fputs(raw_file_id, "L. Cyc. D_PA T_PD D_PD D_DA ");
    fputs(raw_file_id, "T_DRI D_DRI T_DWI D_DWI ");
    fputs(raw_file_id, "T_DRE D_DRE T_DWE D_DWE T_AS D_AS T_AD ");
    fputs(raw_file_id, "Current\n");

    fputs(raw_file_id, "-- ---- ---- ---- ---- ");
    fputs(raw_file_id, "----- ----- ----- ");
    fputs(raw_file_id, "----- ----- ----- ---- ");
    fputs(raw_file_id, "-----\n");

    fprintf(raw_file_id,
"%2.0f %3.0f %4.0f %4.0f %4.0f %4.0f %4.0f %5.0f %5.0f %5.0f
%5.0f %5.0f %5.0f %5.0f %5.0f %4.0f %4.0f %8.3f\n", raw');
    fclose(raw_file_id);
endif
endfunction

```

C.3.2 Power Conversion Scaling

```
## [scaled] = scale_power (raw, voltage, mhz_clk_speed, scaled_file)
##
## Deletes the first column of the raw matrix, and converts the
## last column from average current to scaled power. The zero
## reference reading (lowest reading in data) is subtracted off.
##
## The columns of the output matrix are arranged as follows:
##
## Column      Name                      Description
##  1          Delta_P_A                  Prog. space address line transitions
##  2          Total_P_D                  Prog. space data lines (loop lines)
##  3          Delta_P_D                  Prog. space data line transitions
##  4          Delta_D_A                  Data space address line transitions
##
##  5          Total_D_R_I                Data space internal memory read bits
##  6          Delta_D_R_I                Data space internal memory read bits
##  7          Total_D_W_I                Data space internal memory write bits
##  8          Delta_D_W_I                Data space internal memory write bits
##
##  9          Total_D_R_E                Data space external memory read bits
## 10          Delta_D_R_E                Data space external memory read bits
## 11          Total_D_W_E                Data space external memory write bits
## 12          Delta_D_W_E                Data space external memory write bits
## 13          Total_A_S                  Total ALU source bits
## 14          Delta_A_S                  Delta ALU source bits
## 15          Total_A_D                  Total ALU destination bits
## 16          Energy                     microjoules
## 17          Current                    Power consumption
##
##
## Author: Gary Cameron
## Date: Oct. 25 2001
```

```

## Keywords: DSP 56k c62 c54 Thesis scaled read power

function scaled = scale_power(raw, voltage, mhz_clk_speed,...)

    scaled = zeros(rows(raw),columns(raw)-2);

    if (nargin == 3)
        scaled_file = 0;
    else
        scaled_file = 1;
        scaled_filename = va_arg();
    endif

    for count = 3:columns(raw),
        scaled(:,count-2) = raw(:,count);
    end

    size(raw)
    size(scaled)

    sorted = sort(raw(:,columns(raw)));
    baseline = sorted(1)

    scaled(:,columns(scaled)) = voltage .* raw(:,2) .* \
        (raw(:,columns(raw)) .- baseline) ./ \
        (min(raw(:,2),raw(:,1)) .* mhz_clk_speed);

    if (scaled_file)
        scaled_file_id = fopen(scaled_filename, "w", "native");

        fputs(scaled_file_id, "D_PA T_PD D_PD D_DA ");
        fputs(scaled_file_id,"T_DRI D_DRI T_DWI D_DWI ");
        fputs(scaled_file_id,"T_DRE D_DRE T_DWE D_DWE T_AS D_AS T_AD ");
        fputs(scaled_file_id," Energy\n");
    end

```

```
fputs(scaled_file_id, "---- ---- ---- ");
fputs(scaled_file_id, "----- ----- ");
fputs(scaled_file_id, "----- ----- ---- ");
fputs(scaled_file_id, " -----\n");

fprintf(scaled_file_id,
"%3.0f %4.0f %4.0f %4.0f %5.0f %4.0f %5.0f %5.0f %5.0f %5.0f
    %5.0f %5.0f %5.0f %4.0f %4.0f %8.4f\n", scaled');
fclose(scaled_file_id);
endif
endfunction
```

Appendix D

Processed Results

D.1 56303 Results

Table D.1: 56303 Results Table Legend

ΔPA	Program space address line transitions
TPD	Total program space data lines (loop lines)
ΔPD	Program space data line transitions
ΔDA	Data space address line transitions
TDRI	Total data space internal memory read bits
ΔDRI	Δ data space internal memory read bits
TDWI	Total data space internal memory write bits
ΔDWI	Δ data space internal memory write bits
TDRE	Total data space external memory read bits
ΔDRE	Δ data space external memory read bits
TDWE	Total data space external memory write bits
ΔDWE	Δ data space external memory write bits
TAS	Total ALU source bits
ΔAS	Δ ALU source bits
TAD	Total ALU destination bits
Energy	Normalized nanojoules per cycle

Table D.2: 56303 Results Table

Δ P A	T P D	Δ P D	Δ D A	T D R I	Δ D R I	T D W I	Δ D W I	T D R E	Δ D R E	T D W E	Δ D W E	T A S	Δ A S	T A D	Energy
6	96	8	0	0	0	0	0	0	0	0	0	56	0	56	12.4814
6	96	8	0	0	0	0	0	0	0	0	0	56	0	56	12.4800
6	96	8	0	0	0	0	0	0	0	0	0	56	0	56	12.4936
6	96	8	0	0	0	0	0	0	0	0	0	56	2	56	12.6191
6	96	8	0	0	0	0	0	0	0	0	0	56	2	56	12.6082
6	96	8	0	0	0	0	0	0	0	0	0	56	2	56	12.6000
6	96	8	0	0	0	0	0	0	0	0	0	56	56	56	13.6855
6	96	8	0	0	0	0	0	0	0	0	0	56	56	56	13.6705
6	96	8	0	0	0	0	0	0	0	0	0	56	56	56	13.6814
6	96	12	0	0	0	0	0	0	0	0	0	48	0	56	14.6073
6	96	12	0	0	0	0	0	0	0	0	0	48	0	56	14.5909
6	96	12	0	0	0	0	0	0	0	0	0	48	0	56	14.6005
6	96	12	0	0	0	0	0	0	0	0	0	48	2	56	14.7764
6	96	12	0	0	0	0	0	0	0	0	0	48	2	56	14.7968
6	96	12	0	0	0	0	0	0	0	0	0	48	2	56	14.7764
6	96	12	0	0	0	0	0	0	0	0	0	48	48	56	16.0036
6	96	12	0	0	0	0	0	0	0	0	0	48	48	56	15.9886
6	96	12	0	0	0	0	0	0	0	0	0	48	48	56	15.9818
6	96	12	0	0	0	0	0	0	0	0	0	57	0	56	12.7255
6	96	12	0	0	0	0	0	0	0	0	0	57	0	56	12.7350
6	96	12	0	0	0	0	0	0	0	0	0	57	0	56	12.7255
6	96	12	0	0	0	0	0	0	0	0	0	57	2	56	12.8877
6	96	12	0	0	0	0	0	0	0	0	0	57	2	56	12.8782
6	96	12	0	0	0	0	0	0	0	0	0	57	2	56	12.8809
6	96	12	0	0	0	0	0	0	0	0	0	57	56	56	14.4900
6	96	12	0	0	0	0	0	0	0	0	0	57	56	56	14.4655
6	96	12	0	0	0	0	0	0	0	0	0	57	56	56	14.4709
6	96	8	0	0	0	0	0	0	0	0	0	24	0	56	12.4568
6	96	8	0	0	0	0	0	0	0	0	0	24	0	56	12.4445
6	96	8	0	0	0	0	0	0	0	0	0	24	0	56	12.4595
6	96	8	0	0	0	0	0	0	0	0	0	24	2	56	12.5973
6	96	8	0	0	0	0	0	0	0	0	0	24	2	56	12.5986
6	96	8	0	0	0	0	0	0	0	0	0	24	2	56	12.5932
6	96	8	0	0	0	0	0	0	0	0	0	24	24	56	13.1577
6	96	8	0	0	0	0	0	0	0	0	0	24	24	56	13.1591

6	96	8	0	0	0	0	0	0	0	0	0	0	24	24	56	13.1468
6	96	12	0	0	0	0	0	0	0	0	0	0	25	0	24	13.2668
6	96	12	0	0	0	0	0	0	0	0	0	0	25	0	24	13.2873
6	96	12	0	0	0	0	0	0	0	0	0	0	25	0	24	13.2586
6	96	12	0	0	0	0	0	0	0	0	0	0	25	2	24	13.3377
6	96	12	0	0	0	0	0	0	0	0	0	0	25	2	24	13.3500
6	96	12	0	0	0	0	0	0	0	0	0	0	25	2	24	13.3377
6	96	12	0	0	0	0	0	0	0	0	0	0	25	24	24	13.2941
6	96	12	0	0	0	0	0	0	0	0	0	0	25	24	24	13.2955
6	96	12	0	0	0	0	0	0	0	0	0	0	25	24	24	13.2818
6	96	6	0	0	0	0	0	0	0	0	0	0	0	0	0	12.0518
6	96	6	0	0	0	0	0	0	0	0	0	0	0	0	0	12.0709
6	96	6	0	0	0	0	0	0	0	0	0	0	0	0	0	12.0545
6	96	6	0	0	0	0	0	0	0	0	0	0	0	0	0	12.0355
6	96	6	0	0	0	0	0	0	0	0	0	0	0	0	0	12.0327
6	96	6	0	0	0	0	0	0	0	0	0	0	0	0	0	12.0436
6	96	6	0	0	0	0	0	0	0	0	0	0	0	0	0	12.0791
6	96	6	0	0	0	0	0	0	0	0	0	0	0	0	0	12.0668
6	96	6	0	0	0	0	0	0	0	0	0	0	0	0	0	12.0682
6	96	6	0	0	0	0	0	0	0	0	0	0	0	0	0	11.9945
6	96	6	0	0	0	0	0	0	0	0	0	0	0	0	0	11.9945
6	96	6	0	0	0	0	0	0	0	0	0	0	0	0	0	11.9959
34	432	10	0	0	0	0	0	768	0	0	0	0	0	0	0	90.1139
34	432	10	0	0	0	0	0	768	0	0	0	0	0	0	0	90.0655
34	432	10	0	0	0	0	0	768	0	0	0	0	0	0	0	90.0509
34	432	26	16	0	0	0	0	768	768	0	0	0	0	0	0	91.6024
34	432	26	16	0	0	0	0	768	768	0	0	0	0	0	0	91.5006
34	432	26	16	0	0	0	0	768	768	0	0	0	0	0	0	91.4424
34	432	34	24	0	0	0	0	768	0	0	0	0	0	0	0	92.2473
34	432	34	24	0	0	0	0	768	0	0	0	0	0	0	0	92.2376
34	432	34	24	0	0	0	0	768	0	0	0	0	0	0	0	92.2424
34	432	38	28	0	0	0	0	768	0	0	0	0	0	0	0	92.7224
34	432	38	28	0	0	0	0	768	0	0	0	0	0	0	0	92.6400
34	432	38	28	0	0	0	0	768	0	0	0	0	0	0	0	92.5576
34	432	8	0	0	0	0	0	0	0	768	0	0	0	0	0	73.8424
34	432	8	0	0	0	0	0	0	0	768	0	0	0	0	0	73.7697
34	432	8	0	0	0	0	0	0	0	768	0	0	0	0	0	73.7697
34	432	24	16	0	0	0	0	0	0	768	0	0	0	0	0	75.4570
34	432	24	16	0	0	0	0	0	0	768	0	0	0	0	0	75.4521
34	432	24	16	0	0	0	0	0	0	768	0	0	0	0	0	75.3358
34	432	32	24	0	0	0	0	0	0	768	0	0	0	0	0	76.1939

34	432	32	24	0	0	0	0	0	0	768	0	0	0	0	76.1648
34	432	32	24	0	0	0	0	0	0	768	0	0	0	0	76.1358
34	432	36	28	0	0	0	0	0	0	768	0	0	0	0	76.5430
34	432	36	28	0	0	0	0	0	0	768	0	0	0	0	76.4994
34	432	36	28	0	0	0	0	0	0	768	0	0	0	0	76.4800
34	432	26	16	0	0	0	0	768	768	0	0	0	0	0	102.1624
34	432	26	16	0	0	0	0	768	768	0	0	0	0	0	102.1673
34	432	26	16	0	0	0	0	768	768	0	0	0	0	0	102.0606
34	432	34	24	0	0	0	0	768	768	0	0	0	0	0	102.8461
34	432	34	24	0	0	0	0	768	768	0	0	0	0	0	102.8703
34	432	34	24	0	0	0	0	768	768	0	0	0	0	0	102.8024
34	432	38	28	0	0	0	0	768	768	0	0	0	0	0	103.0448
34	432	38	28	0	0	0	0	768	768	0	0	0	0	0	103.0158
34	432	38	28	0	0	0	0	768	768	0	0	0	0	0	103.0448
34	432	40	16	0	0	0	0	0	0	768	768	0	0	0	90.8558
34	432	40	16	0	0	0	0	0	0	768	768	0	0	0	90.8509
34	432	40	16	0	0	0	0	0	0	768	768	0	0	0	90.8848
34	432	48	24	0	0	0	0	0	0	768	768	0	0	0	91.5976
34	432	48	24	0	0	0	0	0	0	768	768	0	0	0	91.5830
34	432	48	24	0	0	0	0	0	0	768	768	0	0	0	91.6073
34	432	52	28	0	0	0	0	0	0	768	768	0	0	0	91.8836
34	432	52	28	0	0	0	0	0	0	768	768	0	0	0	91.9418
34	432	52	28	0	0	0	0	0	0	768	768	0	0	0	91.9370
34	432	32	16	0	0	0	0	0	0	768	384	0	0	0	82.8267
34	432	32	16	0	0	0	0	0	0	768	384	0	0	0	82.7830
34	432	32	16	0	0	0	0	0	0	768	384	0	0	0	82.7830
34	432	34	24	0	0	0	0	0	0	768	192	0	0	0	79.8448
34	432	34	24	0	0	0	0	0	0	768	192	0	0	0	79.8352
34	432	34	24	0	0	0	0	0	0	768	192	0	0	0	79.8642
34	432	38	28	0	0	0	0	0	0	768	96	0	0	0	78.3176
34	432	38	28	0	0	0	0	0	0	768	96	0	0	0	78.2109
34	432	38	28	0	0	0	0	0	0	768	96	0	0	0	78.2933
4	72	8	0	0	0	0	0	0	0	24	0	0	0	0	27.6509
4	72	8	0	0	0	0	0	0	0	24	0	0	0	0	27.6600
4	72	8	0	0	0	0	0	0	0	24	0	0	0	0	27.6418
6	96	8	0	0	0	0	0	0	0	48	0	0	0	0	41.6345
6	96	8	0	0	0	0	0	0	0	48	0	0	0	0	41.6045
6	96	8	0	0	0	0	0	0	0	48	0	0	0	0	41.6155
18	240	8	0	0	0	0	0	0	0	192	0	0	0	0	66.4975
18	240	8	0	0	0	0	0	0	0	192	0	0	0	0	66.4713
18	240	8	0	0	0	0	0	0	0	192	0	0	0	0	66.4800

34	432	12	12	0	0	0	0	384	0	0	0	0	0	0	44.4873
34	432	12	12	0	0	0	0	384	0	0	0	0	0	0	44.4703
34	432	12	12	0	0	0	0	384	0	0	0	0	0	0	44.4752
34	432	28	16	0	0	0	0	384	0	0	0	0	0	0	45.6097
34	432	28	16	0	0	0	0	384	0	0	0	0	0	0	45.6145
34	432	28	16	0	0	0	0	384	0	0	0	0	0	0	45.5709
34	432	36	24	0	0	0	0	384	0	0	0	0	0	0	46.1867
34	432	36	24	0	0	0	0	384	0	0	0	0	0	0	46.1648
34	432	36	24	0	0	0	0	384	0	0	0	0	0	0	46.1915
34	432	40	28	0	0	0	0	384	0	0	0	0	0	0	46.3636
34	432	40	28	0	0	0	0	384	0	0	0	0	0	0	46.3952
34	432	40	28	0	0	0	0	384	0	0	0	0	0	0	46.3636
34	432	42	30	0	0	0	0	384	0	0	0	0	0	0	46.4727
34	432	42	30	0	0	0	0	384	0	0	0	0	0	0	46.4824
34	432	42	30	0	0	0	0	384	0	0	0	0	0	0	46.4727
34	432	10	0	0	0	0	0	0	0	384	0	0	0	0	39.2218
34	432	10	0	0	0	0	0	0	0	384	0	0	0	0	39.2145
34	432	10	0	0	0	0	0	0	0	384	0	0	0	0	39.2194
34	432	28	16	0	0	0	0	0	0	0	0	0	0	0	40.3491
34	432	28	16	0	0	0	0	0	0	0	0	0	0	0	40.3588
34	432	28	16	0	0	0	0	0	0	0	0	0	0	0	40.3394
34	432	34	24	0	0	0	0	0	0	384	0	0	0	0	40.9067
34	432	34	24	0	0	0	0	0	0	384	0	0	0	0	40.9042
34	432	34	24	0	0	0	0	0	0	384	0	0	0	0	40.8800
34	432	38	28	0	0	0	0	0	0	384	0	0	0	0	41.1442
34	432	38	28	0	0	0	0	0	0	384	0	0	0	0	41.1467
34	432	38	28	0	0	0	0	0	0	384	0	0	0	0	41.1248
34	432	40	30	0	0	0	0	0	0	384	0	0	0	0	41.2170
34	432	40	30	0	0	0	0	0	0	384	0	0	0	0	41.2194
34	432	40	30	0	0	0	0	0	0	384	0	0	0	0	41.1855
34	432	28	16	0	0	0	0	384	384	0	0	0	0	0	51.8085
34	432	28	16	0	0	0	0	384	384	0	0	0	0	0	51.7842
34	432	28	16	0	0	0	0	384	384	0	0	0	0	0	51.8109
34	432	36	24	0	0	0	0	384	384	0	0	0	0	0	52.3830
34	432	36	24	0	0	0	0	384	384	0	0	0	0	0	52.3661
34	432	36	24	0	0	0	0	384	384	0	0	0	0	0	52.3855
34	432	40	28	0	0	0	0	384	384	0	0	0	0	0	52.5430
34	432	40	28	0	0	0	0	384	384	0	0	0	0	0	52.5576
34	432	40	28	0	0	0	0	384	384	0	0	0	0	0	52.5406
34	432	42	30	0	0	0	0	384	384	0	0	0	0	0	52.6642
34	432	42	30	0	0	0	0	384	384	0	0	0	0	0	52.6691

34	432	42	30	0	0	0	0	384	384	0	0	0	0	0	52.6327
34	432	26	0	0	0	0	0	0	384	384	0	0	0	0	48.5988
34	432	26	0	0	0	0	0	0	384	384	0	0	0	0	48.5745
34	432	26	0	0	0	0	0	0	384	384	0	0	0	0	48.5964
34	432	42	16	0	0	0	0	0	0	384	384	0	0	0	50.0291
34	432	42	16	0	0	0	0	0	0	384	384	0	0	0	50.0097
34	432	42	16	0	0	0	0	0	0	384	384	0	0	0	50.0242
34	432	50	24	0	0	0	0	0	0	384	384	0	0	0	50.6012
34	432	50	24	0	0	0	0	0	0	384	384	0	0	0	50.5842
34	432	50	24	0	0	0	0	0	0	384	384	0	0	0	50.5673
34	432	54	28	0	0	0	0	0	0	384	384	0	0	0	50.7879
34	432	54	28	0	0	0	0	0	0	384	384	0	0	0	50.7855
34	432	54	28	0	0	0	0	0	0	384	384	0	0	0	50.7612
34	432	56	32	0	0	0	0	0	0	384	384	0	0	0	50.8436
34	432	56	32	0	0	0	0	0	0	384	384	0	0	0	50.7855
34	432	56	32	0	0	0	0	0	0	384	384	0	0	0	50.7636
34	432	34	16	0	0	0	0	0	0	384	192	0	0	0	44.9600
34	432	34	16	0	0	0	0	0	0	384	192	0	0	0	44.9624
34	432	34	16	0	0	0	0	0	0	384	192	0	0	0	44.9552
34	432	38	24	0	0	0	0	0	0	384	96	0	0	0	43.0885
34	432	38	24	0	0	0	0	0	0	384	96	0	0	0	43.0861
34	432	38	24	0	0	0	0	0	0	384	96	0	0	0	43.1200
34	432	40	28	0	0	0	0	0	0	384	48	0	0	0	42.0848
34	432	40	28	0	0	0	0	0	0	384	48	0	0	0	42.0873
34	432	40	28	0	0	0	0	0	0	384	48	0	0	0	42.0752
34	432	12	0	0	0	0	0	384	0	0	0	0	0	0	48.3079
34	432	12	0	0	0	0	0	384	0	0	0	0	0	0	48.3055
34	432	12	0	0	0	0	0	384	0	0	0	0	0	0	48.3248
34	432	192	16	0	0	0	0	384	0	0	0	0	0	0	48.0412
34	432	192	16	0	0	0	0	384	0	0	0	0	0	0	48.0485
34	432	192	16	0	0	0	0	384	0	0	0	0	0	0	48.0388
34	432	112	24	0	0	0	0	384	0	0	0	0	0	0	47.2921
34	432	112	24	0	0	0	0	384	0	0	0	0	0	0	47.2994
34	432	112	24	0	0	0	0	384	0	0	0	0	0	0	47.2945
34	432	72	28	0	0	0	0	384	0	0	0	0	0	0	46.9091
34	432	72	28	0	0	0	0	384	0	0	0	0	0	0	46.8945
34	432	72	28	0	0	0	0	384	0	0	0	0	0	0	46.8897
34	432	52	32	0	0	0	0	384	0	0	0	0	0	0	46.6376
34	432	52	32	0	0	0	0	384	0	0	0	0	0	0	46.6327
34	432	52	32	0	0	0	0	384	0	0	0	0	0	0	46.6497
34	432	10	0	0	0	0	0	0	0	384	0	0	0	0	40.2570

34	432	10	0	0	0	0	0	0	0	384	0	0	0	0	40.2206
34	432	10	0	0	0	0	0	0	0	384	0	0	0	0	40.2352
34	432	12	16	0	0	0	0	0	0	384	0	0	0	0	42.0388
34	432	12	16	0	0	0	0	0	0	384	0	0	0	0	42.0145
34	432	12	16	0	0	0	0	0	0	384	0	0	0	0	42.0267
34	432	160	16	0	0	0	0	0	0	384	0	0	0	0	41.6461
34	432	160	16	0	0	0	0	0	0	384	0	0	0	0	41.6558
34	432	160	16	0	0	0	0	0	0	384	0	0	0	0	41.6170
34	432	26	16	0	0	0	0	0	0	384	0	0	0	0	41.8352
34	432	26	16	0	0	0	0	0	0	384	0	0	0	0	41.7794
34	432	26	16	0	0	0	0	0	0	384	0	0	0	0	41.7915
34	432	12	24	0	0	0	0	0	0	384	0	0	0	0	42.6739
34	432	12	24	0	0	0	0	0	0	384	0	0	0	0	42.6642
34	432	12	24	0	0	0	0	0	0	384	0	0	0	0	42.6594
34	432	96	24	0	0	0	0	0	0	384	0	0	0	0	41.4788
34	432	96	24	0	0	0	0	0	0	384	0	0	0	0	41.4909
34	432	96	24	0	0	0	0	0	0	384	0	0	0	0	41.4618
34	432	34	24	0	0	0	0	0	0	384	0	0	0	0	42.5503
34	432	34	24	0	0	0	0	0	0	384	0	0	0	0	42.5624
34	432	34	24	0	0	0	0	0	0	384	0	0	0	0	42.5842
34	432	12	28	0	0	0	0	0	0	384	0	0	0	0	42.9527
34	432	12	28	0	0	0	0	0	0	384	0	0	0	0	42.9430
34	432	12	28	0	0	0	0	0	0	384	0	0	0	0	42.9261
34	432	64	28	0	0	0	0	0	0	384	0	0	0	0	41.3576
34	432	64	28	0	0	0	0	0	0	384	0	0	0	0	41.3673
34	432	64	28	0	0	0	0	0	0	384	0	0	0	0	41.3527
34	432	38	28	0	0	0	0	0	0	384	0	0	0	0	42.8364
34	432	38	28	0	0	0	0	0	0	384	0	0	0	0	42.8339
34	432	38	28	0	0	0	0	0	0	384	0	0	0	0	42.8291
34	432	12	32	0	0	0	0	0	0	384	0	0	0	0	43.0012
34	432	12	32	0	0	0	0	0	0	384	0	0	0	0	43.0158
34	432	12	32	0	0	0	0	0	0	384	0	0	0	0	42.9964
34	432	48	32	0	0	0	0	0	0	384	0	0	0	0	41.2242
34	432	48	32	0	0	0	0	0	0	384	0	0	0	0	41.2339
34	432	48	32	0	0	0	0	0	0	384	0	0	0	0	41.2315
4	72	10	0	0	0	0	0	0	0	24	0	0	0	0	22.0923
4	72	10	0	0	0	0	0	0	0	24	0	0	0	0	22.0705
4	72	10	0	0	0	0	0	0	0	24	0	0	0	0	22.0923
6	96	10	2	0	0	0	0	0	0	48	0	0	0	0	22.0636
6	96	10	2	0	0	0	0	0	0	48	0	0	0	0	22.0650
6	96	10	2	0	0	0	0	0	0	48	0	0	0	0	22.0459

10	144	10	4	0	0	0	0	0	0	96	0	0	0	0	29.4345
10	144	10	4	0	0	0	0	0	0	96	0	0	0	0	29.4291
10	144	10	4	0	0	0	0	0	0	96	0	0	0	0	29.4200
18	240	10	14	0	0	0	0	0	0	192	0	0	0	0	36.2269
18	240	10	14	0	0	0	0	0	0	192	0	0	0	0	36.2487
18	240	10	14	0	0	0	0	0	0	192	0	0	0	0	36.2356
4	72	12	0	0	0	0	0	0	0	0	0	0	0	0	22.5682
4	72	12	0	0	0	0	0	0	0	0	0	0	0	0	22.5655
4	72	12	0	0	0	0	0	0	0	0	0	0	0	0	22.5586
6	96	12	0	0	0	0	0	0	0	0	0	0	0	0	22.7018
6	96	12	0	0	0	0	0	0	0	0	0	0	0	0	22.6909
6	96	12	0	0	0	0	0	0	0	0	0	0	0	0	22.6950
10	144	12	0	0	0	0	0	0	0	0	0	0	0	0	30.2200
10	144	12	0	0	0	0	0	0	0	0	0	0	0	0	30.2200
10	144	12	0	0	0	0	0	0	0	0	0	0	0	0	30.2164
18	240	12	0	0	0	0	0	0	0	0	0	0	0	0	36.2269
18	240	12	0	0	0	0	0	0	0	0	0	0	0	0	36.2487
18	240	12	0	0	0	0	0	0	0	0	0	0	0	0	36.2356
34	432	26	16	0	0	0	0	768	0	0	0	0	0	0	82.5164
34	432	26	16	0	0	0	0	768	0	0	0	0	0	0	82.5018
34	432	26	16	0	0	0	0	768	0	0	0	0	0	0	82.4921
34	432	26	16	0	0	0	0	768	0	0	0	0	0	0	91.5442
34	432	26	16	0	0	0	0	768	0	0	0	0	0	0	91.4909
34	432	26	16	0	0	0	0	768	0	0	0	0	0	0	91.5539
34	432	24	16	0	0	0	0	0	0	0	0	0	0	0	75.4085
34	432	24	16	0	0	0	0	0	0	0	0	0	0	0	75.4230
34	432	24	16	0	0	0	0	0	0	0	0	0	0	0	75.4715
34	432	24	16	0	0	0	0	0	0	0	0	0	0	0	75.4182
34	432	24	16	0	0	0	0	0	0	0	0	0	0	0	75.3648
34	432	24	16	0	0	0	0	0	0	0	0	0	0	0	75.3939
34	432	26	16	0	0	0	0	768	768	0	0	0	0	0	102.0558
34	432	26	16	0	0	0	0	768	768	0	0	0	0	0	102.0267
34	432	26	16	0	0	0	0	768	768	0	0	0	0	0	102.0800
34	432	26	16	0	0	0	0	768	768	0	0	0	0	0	102.0218
34	432	26	16	0	0	0	0	768	768	0	0	0	0	0	101.9733
34	432	26	16	0	0	0	0	768	768	0	0	0	0	0	101.9394
34	432	40	16	0	0	0	0	0	0	768	768	0	0	0	90.8606
34	432	40	16	0	0	0	0	0	0	768	768	0	0	0	90.8509
34	432	40	16	0	0	0	0	0	0	768	768	0	0	0	90.8267
34	432	42	16	0	0	0	0	0	0	768	768	0	0	0	90.9964
34	432	42	16	0	0	0	0	0	0	768	768	0	0	0	90.9721

34	432	42	16	0	0	0	0	0	0	768	768	0	0	0	90.9479
34	432	28	16	0	0	0	0	0	0	768	192	0	0	0	79.1903
34	432	28	16	0	0	0	0	0	0	768	192	0	0	0	79.1467
34	432	28	16	0	0	0	0	0	0	768	192	0	0	0	79.0836
34	432	28	16	0	0	0	0	0	0	768	96	0	0	0	78.9091
34	432	28	16	0	0	0	0	0	0	768	96	0	0	0	78.9479
34	432	28	16	0	0	0	0	0	0	768	96	0	0	0	78.9042
34	432	28	30	0	0	0	0	384	0	0	0	0	0	0	45.6364
34	432	28	30	0	0	0	0	384	0	0	0	0	0	0	45.6000
34	432	28	30	0	0	0	0	384	0	0	0	0	0	0	45.6145
34	432	28	28	0	0	0	0	384	0	0	0	0	0	0	45.5952
34	432	28	28	0	0	0	0	384	0	0	0	0	0	0	45.5855
34	432	28	28	0	0	0	0	384	0	0	0	0	0	0	45.5830
34	432	28	28	0	0	0	0	384	0	0	0	0	0	0	45.5855
34	432	28	28	0	0	0	0	384	0	0	0	0	0	0	45.5952
34	432	28	28	0	0	0	0	384	0	0	0	0	0	0	45.5976
34	432	26	16	0	0	0	0	0	0	384	0	0	0	0	40.3830
34	432	26	16	0	0	0	0	0	0	384	0	0	0	0	40.3515
34	432	26	16	0	0	0	0	0	0	384	0	0	0	0	40.3806
34	432	26	16	0	0	0	0	0	0	384	0	0	0	0	40.3297
34	432	26	16	0	0	0	0	0	0	384	0	0	0	0	40.3321
34	432	26	16	0	0	0	0	0	0	384	0	0	0	0	40.2812
34	432	26	16	0	0	0	0	0	0	384	0	0	0	0	40.3636
34	432	26	16	0	0	0	0	0	0	384	0	0	0	0	40.3297
34	432	26	16	0	0	0	0	0	0	384	0	0	0	0	40.3224
34	432	28	16	0	0	0	0	384	384	0	0	0	0	0	51.8594
34	432	28	16	0	0	0	0	384	384	0	0	0	0	0	51.8400
34	432	28	16	0	0	0	0	384	384	0	0	0	0	0	51.8376
34	432	28	16	0	0	0	0	384	384	0	0	0	0	0	51.7624
34	432	28	16	0	0	0	0	384	384	0	0	0	0	0	51.7576
34	432	28	16	0	0	0	0	384	384	0	0	0	0	0	51.7576
34	432	28	16	0	0	0	0	384	384	0	0	0	0	0	51.7697
34	432	28	16	0	0	0	0	384	384	0	0	0	0	0	51.7842
34	432	28	16	0	0	0	0	384	384	0	0	0	0	0	51.7527
34	432	42	16	0	0	0	0	0	0	384	384	0	0	0	50.0873
34	432	42	16	0	0	0	0	0	0	384	384	0	0	0	50.0606
34	432	42	16	0	0	0	0	0	0	384	384	0	0	0	50.0485
34	432	42	16	0	0	0	0	0	0	384	384	0	0	0	49.9612
34	432	42	16	0	0	0	0	0	0	384	384	0	0	0	49.9806
34	432	42	16	0	0	0	0	0	0	384	384	0	0	0	49.9661
34	432	42	16	0	0	0	0	0	0	384	384	0	0	0	50.0533

[illegible]

34	432	10	0	768	0	0	0	0	0	0	0	0	0	0	12.9532
34	432	26	16	768	0	0	0	0	0	0	0	0	0	0	13.2232
34	432	26	16	768	0	0	0	0	0	0	0	0	0	0	13.2382
34	432	26	16	768	0	0	0	0	0	0	0	0	0	0	13.2395
34	432	34	24	768	0	0	0	0	0	0	0	0	0	0	13.3364
34	432	34	24	768	0	0	0	0	0	0	0	0	0	0	13.3500
34	432	34	24	768	0	0	0	0	0	0	0	0	0	0	13.3527
34	432	38	28	768	0	0	0	0	0	0	0	0	0	0	13.4209
34	432	38	28	768	0	0	0	0	0	0	0	0	0	0	13.4182
34	432	38	28	768	0	0	0	0	0	0	0	0	0	0	13.4345
34	432	8	0	0	0	768	0	0	0	0	0	0	0	0	13.6732
34	432	8	0	0	0	768	0	0	0	0	0	0	0	0	13.6732
34	432	8	0	0	0	768	0	0	0	0	0	0	0	0	13.6923
34	432	24	16	0	0	768	0	0	0	0	0	0	0	0	13.9432
34	432	24	16	0	0	768	0	0	0	0	0	0	0	0	13.9527
34	432	24	16	0	0	768	0	0	0	0	0	0	0	0	13.9323
34	432	32	16	0	0	768	0	0	0	0	0	0	0	0	14.0523
34	432	32	16	0	0	768	0	0	0	0	0	0	0	0	14.0523
34	432	32	16	0	0	768	0	0	0	0	0	0	0	0	14.0536
34	432	36	16	0	0	768	0	0	0	0	0	0	0	0	14.1641
34	432	36	16	0	0	768	0	0	0	0	0	0	0	0	14.1695
34	432	36	16	0	0	768	0	0	0	0	0	0	0	0	14.1614
34	432	26	16	768	0	0	0	0	0	0	0	0	0	0	15.0873
34	432	26	16	768	0	0	0	0	0	0	0	0	0	0	15.1064
34	432	26	16	768	0	0	0	0	0	0	0	0	0	0	15.0941
34	432	34	24	768	0	0	0	0	0	0	0	0	0	0	15.2086
34	432	34	24	768	0	0	0	0	0	0	0	0	0	0	15.2168
34	432	34	24	768	0	0	0	0	0	0	0	0	0	0	15.2073
34	432	38	28	768	0	0	0	0	0	0	0	0	0	0	15.3150
34	432	38	28	768	0	0	0	0	0	0	0	0	0	0	15.3150
34	432	38	28	768	0	0	0	0	0	0	0	0	0	0	15.3136
34	432	40	16	0	0	768	768	0	0	0	0	0	0	0	16.3582
34	432	40	16	0	0	768	768	0	0	0	0	0	0	0	16.3500
34	432	40	16	0	0	768	768	0	0	0	0	0	0	0	16.3418
34	432	48	24	0	0	768	768	0	0	0	0	0	0	0	16.4782
34	432	48	24	0	0	768	768	0	0	0	0	0	0	0	16.4700
34	432	48	24	0	0	768	768	0	0	0	0	0	0	0	16.4373
34	432	52	28	0	0	768	768	0	0	0	0	0	0	0	16.5423
34	432	52	28	0	0	768	768	0	0	0	0	0	0	0	16.5614
34	432	52	28	0	0	768	768	0	0	0	0	0	0	0	16.5450
34	432	32	16	0	0	768	384	0	0	0	0	0	0	0	15.0723

34	432	32	16	0	0	768	384	0	0	0	0	0	0	0	15.0668
34	432	32	16	0	0	768	384	0	0	0	0	0	0	0	15.0791
34	432	36	24	0	0	768	192	0	0	0	0	0	0	0	14.5527
34	432	36	24	0	0	768	192	0	0	0	0	0	0	0	14.5405
34	432	36	24	0	0	768	192	0	0	0	0	0	0	0	14.5418
34	432	38	28	0	0	768	96	0	0	0	0	0	0	0	14.3209
34	432	38	28	0	0	768	96	0	0	0	0	0	0	0	14.3250
34	432	38	28	0	0	768	96	0	0	0	0	0	0	0	14.3168
4	72	8	0	0	0	48	0	0	0	0	0	0	0	0	13.5259
4	72	8	0	0	0	48	0	0	0	0	0	0	0	0	13.5382
4	72	8	0	0	0	48	0	0	0	0	0	0	0	0	13.5382
6	96	8	2	0	0	96	0	0	0	0	0	0	0	0	13.8273
6	96	8	2	0	0	96	0	0	0	0	0	0	0	0	13.8150
6	96	8	2	0	0	96	0	0	0	0	0	0	0	0	13.8109
10	144	8	6	0	0	192	0	0	0	0	0	0	0	0	13.7441
10	144	8	6	0	0	192	0	0	0	0	0	0	0	0	13.7427
10	144	8	6	0	0	192	0	0	0	0	0	0	0	0	13.7455
18	240	8	14	0	0	384	0	0	0	0	0	0	0	0	13.7018
18	240	8	14	0	0	384	0	0	0	0	0	0	0	0	13.7005
18	240	8	14	0	0	384	0	0	0	0	0	0	0	0	13.6923
34	432	12	0	384	0	0	0	0	0	0	0	0	0	0	11.7586
34	432	12	0	384	0	0	0	0	0	0	0	0	0	0	11.7505
34	432	12	0	384	0	0	0	0	0	0	0	0	0	0	11.7450
34	432	14	16	384	0	0	0	0	0	0	0	0	0	0	12.1895
34	432	14	16	384	0	0	0	0	0	0	0	0	0	0	12.1759
34	432	14	16	384	0	0	0	0	0	0	0	0	0	0	12.1650
34	432	192	16	384	0	0	0	0	0	0	0	0	0	0	13.3814
34	432	192	16	384	0	0	0	0	0	0	0	0	0	0	13.3759
34	432	192	16	384	0	0	0	0	0	0	0	0	0	0	13.3827
34	432	14	24	384	0	0	0	0	0	0	0	0	0	0	12.1650
34	432	14	24	384	0	0	0	0	0	0	0	0	0	0	12.1677
34	432	14	24	384	0	0	0	0	0	0	0	0	0	0	12.1677
34	432	112	24	384	0	0	0	0	0	0	0	0	0	0	13.1209
34	432	112	24	384	0	0	0	0	0	0	0	0	0	0	13.1291
34	432	112	24	384	0	0	0	0	0	0	0	0	0	0	13.1345
34	432	14	28	384	0	0	0	0	0	0	0	0	0	0	12.1950
34	432	14	28	384	0	0	0	0	0	0	0	0	0	0	12.2073
34	432	14	28	384	0	0	0	0	0	0	0	0	0	0	12.1936
34	432	72	28	384	0	0	0	0	0	0	0	0	0	0	13.0132
34	432	72	28	384	0	0	0	0	0	0	0	0	0	0	13.0200
34	432	72	28	384	0	0	0	0	0	0	0	0	0	0	13.0200

34	432	14	30	384	0	0	0	0	0	0	0	0	0	0	12.2059
34	432	14	30	384	0	0	0	0	0	0	0	0	0	0	12.2086
34	432	14	30	384	0	0	0	0	0	0	0	0	0	0	12.2073
34	432	52	30	384	0	0	0	0	0	0	0	0	0	0	12.9695
34	432	52	30	384	0	0	0	0	0	0	0	0	0	0	12.9791
34	432	52	30	384	0	0	0	0	0	0	0	0	0	0	12.9818
34	432	10	0	0	0	384	0	0	0	0	0	0	0	0	12.1105
34	432	10	0	0	0	384	0	0	0	0	0	0	0	0	12.0982
34	432	10	0	0	0	384	0	0	0	0	0	0	0	0	12.1036
34	432	160	16	0	0	384	0	0	0	0	0	0	0	0	12.1541
34	432	160	16	0	0	384	0	0	0	0	0	0	0	0	12.1473
34	432	160	16	0	0	384	0	0	0	0	0	0	0	0	12.1541
34	432	96	24	0	0	384	0	0	0	0	0	0	0	0	13.3023
34	432	96	24	0	0	384	0	0	0	0	0	0	0	0	13.3118
34	432	96	24	0	0	384	0	0	0	0	0	0	0	0	13.2995
34	432	64	28	0	0	384	0	0	0	0	0	0	0	0	13.1918
34	432	64	28	0	0	384	0	0	0	0	0	0	0	0	13.1918
34	432	64	28	0	0	384	0	0	0	0	0	0	0	0	13.2000
34	432	48	30	0	0	384	0	0	0	0	0	0	0	0	13.1618
34	432	48	30	0	0	384	0	0	0	0	0	0	0	0	13.1536
34	432	48	30	0	0	384	0	0	0	0	0	0	0	0	13.1591
34	432	28	16	384	0	0	0	0	0	0	0	0	0	0	12.8318
34	432	28	16	384	0	0	0	0	0	0	0	0	0	0	12.8264
34	432	28	16	384	0	0	0	0	0	0	0	0	0	0	12.8209
34	432	36	24	384	0	0	0	0	0	0	0	0	0	0	12.9068
34	432	36	24	384	0	0	0	0	0	0	0	0	0	0	12.9136
34	432	36	24	384	0	0	0	0	0	0	0	0	0	0	12.9068
34	432	40	28	384	0	0	0	0	0	0	0	0	0	0	12.9068
34	432	40	28	384	0	0	0	0	0	0	0	0	0	0	12.9273
34	432	40	28	384	0	0	0	0	0	0	0	0	0	0	12.9191
34	432	42	30	384	0	0	0	0	0	0	0	0	0	0	12.9573
34	432	42	30	384	0	0	0	0	0	0	0	0	0	0	12.9559
34	432	42	30	384	0	0	0	0	0	0	0	0	0	0	12.9600
34	432	10	0	0	0	384	0	0	0	0	0	0	0	0	12.8659
34	432	10	0	0	0	384	0	0	0	0	0	0	0	0	12.8536
34	432	10	0	0	0	384	0	0	0	0	0	0	0	0	12.8714
34	432	28	16	0	0	384	0	0	0	0	0	0	0	0	13.0473
34	432	28	16	0	0	384	0	0	0	0	0	0	0	0	13.0527
34	432	28	16	0	0	384	0	0	0	0	0	0	0	0	13.0336
34	432	34	24	0	0	384	0	0	0	0	0	0	0	0	13.0923
34	432	34	24	0	0	384	0	0	0	0	0	0	0	0	13.0991

34	432	34	24	0	0	384	0	0	0	0	0	0	0	0	13.0786
34	432	38	28	0	0	384	0	0	0	0	0	0	0	0	13.1386
34	432	38	28	0	0	384	0	0	0	0	0	0	0	0	13.1373
34	432	38	28	0	0	384	0	0	0	0	0	0	0	0	13.1468
34	432	40	30	0	0	384	0	0	0	0	0	0	0	0	13.1727
34	432	40	30	0	0	384	0	0	0	0	0	0	0	0	13.1905
34	432	40	30	0	0	384	0	0	0	0	0	0	0	0	13.1659
34	432	28	16	384	384	0	0	0	0	0	0	0	0	0	13.2764
34	432	28	16	384	384	0	0	0	0	0	0	0	0	0	13.2777
34	432	28	16	384	384	0	0	0	0	0	0	0	0	0	13.2859
34	432	36	24	384	384	0	0	0	0	0	0	0	0	0	13.3377
34	432	36	24	384	384	0	0	0	0	0	0	0	0	0	13.3418
34	432	36	24	384	384	0	0	0	0	0	0	0	0	0	13.3391
34	432	40	28	384	384	0	0	0	0	0	0	0	0	0	13.3841
34	432	40	28	384	384	0	0	0	0	0	0	0	0	0	13.3773
34	432	40	28	384	384	0	0	0	0	0	0	0	0	0	13.4005
34	432	42	30	384	384	0	0	0	0	0	0	0	0	0	13.4018
34	432	42	30	384	384	0	0	0	0	0	0	0	0	0	13.4182
34	432	42	30	384	384	0	0	0	0	0	0	0	0	0	13.4155
34	432	26	0	0	384	384	0	0	0	0	0	0	0	0	13.5232
34	432	26	0	0	384	384	0	0	0	0	0	0	0	0	13.5218
34	432	26	0	0	384	384	0	0	0	0	0	0	0	0	13.5205
34	432	42	16	0	0	384	384	0	0	0	0	0	0	0	13.6500
34	432	42	16	0	0	384	384	0	0	0	0	0	0	0	13.6555
34	432	42	16	0	0	384	384	0	0	0	0	0	0	0	13.6336
34	432	50	24	0	0	384	384	0	0	0	0	0	0	0	13.7345
34	432	50	24	0	0	384	384	0	0	0	0	0	0	0	13.7414
34	432	50	24	0	0	384	384	0	0	0	0	0	0	0	13.7414
34	432	54	28	0	0	384	384	0	0	0	0	0	0	0	13.7577
34	432	54	28	0	0	384	384	0	0	0	0	0	0	0	13.7659
34	432	54	28	0	0	384	384	0	0	0	0	0	0	0	20.5868
34	432	56	30	0	0	384	384	0	0	0	0	0	0	0	13.8014
34	432	56	30	0	0	384	384	0	0	0	0	0	0	0	13.8055
34	432	56	30	0	0	384	384	0	0	0	0	0	0	0	13.8123
34	432	34	16	0	0	384	192	0	0	0	0	0	0	0	13.3268
34	432	34	16	0	0	384	192	0	0	0	0	0	0	0	13.2927
34	432	34	16	0	0	384	192	0	0	0	0	0	0	0	13.2968
34	432	38	24	0	0	384	96	0	0	0	0	0	0	0	13.2314
34	432	38	24	0	0	384	96	0	0	0	0	0	0	0	13.2477
34	432	38	24	0	0	384	96	0	0	0	0	0	0	0	13.2341
34	432	40	28	0	0	384	48	0	0	0	0	0	0	0	13.1727

[illegible]

[illegible]

D.2 C5402 Results

Table D.3: C5402 Results Table Legend

ΔPA	Program space address line transitions
TPD	Total program space data lines (loop lines)
ΔPD	Program space data line transitions
ΔDA	Data space address line transitions
TDRI	Total data space internal memory read bits
ΔDRI	Δ data space internal memory read bits
TDWI	Total data space internal memory write bits
ΔDWI	Δ data space internal memory write bits
TDRE	Total data space external memory read bits
ΔDRE	Δ data space external memory read bits
TDWE	Total data space external memory write bits
ΔDWE	Δ data space external memory write bits
TAS	Total ALU source bits
ΔAS	Δ ALU source bits
TAD	Total ALU destination bits
Energy	Normalized nanojoules per cycle

Table D.4: C5402 Results Table

Δ P A	T P D	Δ P D	Δ D A	T D R I	Δ D R I	T D W I	Δ D W I	T D R E	Δ D R E	T D W E	Δ D W E	T A S	Δ A S	T A D	Energy
112	544	114	0	0	0	0	0	256	0	0	0	0	0	0	1.2962
112	544	114	0	0	0	0	0	256	0	0	0	0	0	0	1.3774
112	544	114	0	0	0	0	0	256	0	0	0	0	0	0	1.4003
112	544	112	16	0	0	0	0	256	0	0	0	0	0	0	1.8362
112	544	112	16	0	0	0	0	256	0	0	0	0	0	0	1.8238
112	544	112	16	0	0	0	0	256	0	0	0	0	0	0	1.8353
112	544	128	24	0	0	0	0	256	0	0	0	0	0	0	1.8997
112	544	128	24	0	0	0	0	256	0	0	0	0	0	0	1.8971
112	544	128	24	0	0	0	0	256	0	0	0	0	0	0	1.8838
112	544	126	28	0	0	0	0	256	0	0	0	0	0	0	1.9121
112	544	126	28	0	0	0	0	256	0	0	0	0	0	0	1.9059
112	544	126	28	0	0	0	0	256	0	0	0	0	0	0	1.9218
112	544	142	30	0	0	0	0	256	0	0	0	0	0	0	1.8900
112	544	142	30	0	0	0	0	256	0	0	0	0	0	0	1.9112
112	544	142	30	0	0	0	0	256	0	0	0	0	0	0	1.9006
112	544	114	0	0	0	0	0	0	0	256	0	0	0	0	1.3631
112	544	114	0	0	0	0	0	0	0	256	0	0	0	0	1.3650
112	544	114	0	0	0	0	0	0	0	256	0	0	0	0	1.3656
112	544	112	16	0	0	0	0	0	0	256	0	0	0	0	1.5563
112	544	112	16	0	0	0	0	0	0	256	0	0	0	0	1.5706
112	544	112	16	0	0	0	0	0	0	256	0	0	0	0	1.5663
112	544	128	24	0	0	0	0	0	0	256	0	0	0	0	1.5869
112	544	128	24	0	0	0	0	0	0	256	0	0	0	0	1.5988
112	544	128	24	0	0	0	0	0	0	256	0	0	0	0	1.6250
112	544	126	28	0	0	0	0	0	0	256	0	0	0	0	1.6369
112	544	126	28	0	0	0	0	0	0	256	0	0	0	0	1.6225
112	544	126	28	0	0	0	0	0	0	256	0	0	0	0	1.6231
112	544	142	30	0	0	0	0	0	0	256	0	0	0	0	1.6306
112	544	142	30	0	0	0	0	0	0	256	0	0	0	0	1.6144
112	544	142	30	0	0	0	0	0	0	256	0	0	0	0	1.6306
112	544	112	16	0	0	0	0	256	256	0	0	0	0	0	2.0268
112	544	112	16	0	0	0	0	256	256	0	0	0	0	0	2.0488
112	544	112	16	0	0	0	0	256	256	0	0	0	0	0	2.0153
112	544	128	24	0	0	0	0	256	256	0	0	0	0	0	2.0841
112	544	128	24	0	0	0	0	256	256	0	0	0	0	0	2.0850

112	544	128	24	0	0	0	0	256	256	0	0	0	0	0	2.1009
112	544	126	28	0	0	0	0	256	256	0	0	0	0	0	2.1106
112	544	126	28	0	0	0	0	256	256	0	0	0	0	0	2.0974
112	544	126	28	0	0	0	0	256	256	0	0	0	0	0	2.1053
112	544	142	30	0	0	0	0	256	256	0	0	0	0	0	2.1141
112	544	142	30	0	0	0	0	256	256	0	0	0	0	0	2.1168
112	544	142	30	0	0	0	0	256	256	0	0	0	0	0	2.0868
112	544	122	0	0	0	0	0	0	256	256	0	0	0	0	1.6331
112	544	122	0	0	0	0	0	0	256	256	0	0	0	0	1.6288
112	544	122	0	0	0	0	0	0	256	256	0	0	0	0	1.6306
112	544	120	16	0	0	0	0	0	0	256	256	0	0	0	1.8563
112	544	120	16	0	0	0	0	0	0	256	256	0	0	0	1.8400
112	544	120	16	0	0	0	0	0	0	256	256	0	0	0	1.8506
112	544	136	24	0	0	0	0	0	0	256	256	0	0	0	1.8812
112	544	136	24	0	0	0	0	0	0	256	256	0	0	0	1.8850
112	544	136	24	0	0	0	0	0	0	256	256	0	0	0	1.8731
112	544	134	28	0	0	0	0	0	0	256	256	0	0	0	1.8913
112	544	134	28	0	0	0	0	0	0	256	256	0	0	0	1.8856
112	544	134	28	0	0	0	0	0	0	256	256	0	0	0	1.9012
112	544	150	30	0	0	0	0	0	0	256	256	0	0	0	1.9113
112	544	150	30	0	0	0	0	0	0	256	256	0	0	0	1.9113
112	544	150	30	0	0	0	0	0	0	256	256	0	0	0	1.9094
112	544	130	16	0	0	0	0	256	0	0	0	0	0	0	1.8212
112	544	130	16	0	0	0	0	256	0	0	0	0	0	0	1.8318
112	544	130	16	0	0	0	0	256	0	0	0	0	0	0	1.8035
112	544	128	16	0	0	0	0	256	0	0	0	0	0	0	1.8185
112	544	128	16	0	0	0	0	256	0	0	0	0	0	0	1.8106
112	544	128	16	0	0	0	0	256	0	0	0	0	0	0	1.8026
112	544	144	16	0	0	0	0	256	0	0	0	0	0	0	1.8088
112	544	144	16	0	0	0	0	256	0	0	0	0	0	0	1.7788
112	544	144	16	0	0	0	0	256	0	0	0	0	0	0	1.7894
112	544	130	16	0	0	0	0	0	0	256	0	0	0	0	1.5831
112	544	130	16	0	0	0	0	0	0	256	0	0	0	0	1.5600
112	544	130	16	0	0	0	0	0	0	256	0	0	0	0	1.5544
112	544	128	16	0	0	0	0	0	0	256	0	0	0	0	1.5331
112	544	128	16	0	0	0	0	0	0	256	0	0	0	0	1.5494
112	544	128	16	0	0	0	0	0	0	256	0	0	0	0	1.5388
112	544	144	16	0	0	0	0	0	0	256	0	0	0	0	1.5663
112	544	144	16	0	0	0	0	0	0	256	0	0	0	0	1.5575
112	544	144	16	0	0	0	0	0	0	256	0	0	0	0	1.5450
112	544	130	16	0	0	0	0	256	256	0	0	0	0	0	2.0488

112	544	130	16	0	0	0	0	256	256	0	0	0	0	0	2.0347
112	544	130	16	0	0	0	0	256	256	0	0	0	0	0	2.0197
112	544	128	16	0	0	0	0	256	256	0	0	0	0	0	2.0188
112	544	128	16	0	0	0	0	256	256	0	0	0	0	0	2.0135
112	544	128	16	0	0	0	0	256	256	0	0	0	0	0	2.0162
112	544	144	16	0	0	0	0	256	256	0	0	0	0	0	1.9941
112	544	144	16	0	0	0	0	256	256	0	0	0	0	0	1.9950
112	544	144	16	0	0	0	0	256	256	0	0	0	0	0	2.0206
112	544	138	16	0	0	0	0	0	0	256	256	0	0	0	1.8769
112	544	138	16	0	0	0	0	0	0	256	256	0	0	0	1.8744
112	544	138	16	0	0	0	0	0	0	256	256	0	0	0	1.8662
112	544	136	16	0	0	0	0	0	0	256	256	0	0	0	1.8406
112	544	136	16	0	0	0	0	0	0	256	256	0	0	0	1.8494
112	544	136	16	0	0	0	0	0	0	256	256	0	0	0	1.8344
112	544	152	16	0	0	0	0	0	0	256	256	0	0	0	1.8481
112	544	152	16	0	0	0	0	0	0	256	256	0	0	0	1.8644
112	544	152	16	0	0	0	0	0	0	256	256	0	0	0	1.8462
112	544	114	0	256	0	0	0	0	0	0	0	0	0	0	1.2688
112	544	114	0	256	0	0	0	0	0	0	0	0	0	0	1.2844
112	544	114	0	256	0	0	0	0	0	0	0	0	0	0	1.2663
112	544	112	16	256	0	0	0	0	0	0	0	0	0	0	1.7700
112	544	112	16	256	0	0	0	0	0	0	0	0	0	0	1.8159
112	544	112	16	256	0	0	0	0	0	0	0	0	0	0	1.8088
112	544	128	24	256	0	0	0	0	0	0	0	0	0	0	1.8097
112	544	128	24	256	0	0	0	0	0	0	0	0	0	0	1.8318
112	544	128	24	256	0	0	0	0	0	0	0	0	0	0	1.8362
112	544	126	28	256	0	0	0	0	0	0	0	0	0	0	1.8388
112	544	126	28	256	0	0	0	0	0	0	0	0	0	0	1.8238
112	544	126	28	256	0	0	0	0	0	0	0	0	0	0	1.8282
112	544	142	30	256	0	0	0	0	0	0	0	0	0	0	1.8079
112	544	142	30	256	0	0	0	0	0	0	0	0	0	0	1.8097
112	544	142	30	256	0	0	0	0	0	0	0	0	0	0	1.8229
112	544	114	0	0	0	256	0	0	0	0	0	0	0	0	1.3344
112	544	114	0	0	0	256	0	0	0	0	0	0	0	0	1.3400
112	544	114	0	0	0	256	0	0	0	0	0	0	0	0	1.3462
112	544	112	16	0	0	256	0	0	0	0	0	0	0	0	1.3706
112	544	112	16	0	0	256	0	0	0	0	0	0	0	0	1.3750
112	544	112	16	0	0	256	0	0	0	0	0	0	0	0	1.3775
112	544	128	24	0	0	256	0	0	0	0	0	0	0	0	1.3744
112	544	128	24	0	0	256	0	0	0	0	0	0	0	0	1.3625
112	544	128	24	0	0	256	0	0	0	0	0	0	0	0	1.3906

112	544	126	28	0	0	256	0	0	0	0	0	0	0	0	1.3563
112	544	126	28	0	0	256	0	0	0	0	0	0	0	0	1.3838
112	544	126	28	0	0	256	0	0	0	0	0	0	0	0	1.3662
112	544	142	30	0	0	256	0	0	0	0	0	0	0	0	1.3763
112	544	142	30	0	0	256	0	0	0	0	0	0	0	0	1.3644
112	544	142	30	0	0	256	0	0	0	0	0	0	0	0	1.3856
112	544	112	16	256	256	0	0	0	0	0	0	0	0	0	1.9729
112	544	112	16	256	256	0	0	0	0	0	0	0	0	0	1.9968
112	544	112	16	256	256	0	0	0	0	0	0	0	0	0	1.9668
112	544	128	24	256	256	0	0	0	0	0	0	0	0	0	2.0047
112	544	128	24	256	256	0	0	0	0	0	0	0	0	0	2.0100
112	544	128	24	256	256	0	0	0	0	0	0	0	0	0	2.0153
112	544	126	28	256	256	0	0	0	0	0	0	0	0	0	2.0082
112	544	126	28	256	256	0	0	0	0	0	0	0	0	0	2.0109
112	544	126	28	256	256	0	0	0	0	0	0	0	0	0	1.9765
112	544	142	30	256	256	0	0	0	0	0	0	0	0	0	1.9871
112	544	142	30	256	256	0	0	0	0	0	0	0	0	0	1.9994
112	544	142	30	256	256	0	0	0	0	0	0	0	0	0	2.0171
112	544	122	0	0	256	256	0	0	0	0	0	0	0	0	1.4787
112	544	122	0	0	256	256	0	0	0	0	0	0	0	0	1.4625
112	544	122	0	0	256	256	0	0	0	0	0	0	0	0	1.4981
112	544	120	16	0	0	256	256	0	0	0	0	0	0	0	1.4813
112	544	120	16	0	0	256	256	0	0	0	0	0	0	0	1.4963
112	544	120	16	0	0	256	256	0	0	0	0	0	0	0	1.4981
112	544	136	24	0	0	256	256	0	0	0	0	0	0	0	1.4981
112	544	136	24	0	0	256	256	0	0	0	0	0	0	0	1.4681
112	544	136	24	0	0	256	256	0	0	0	0	0	0	0	1.5119
112	544	134	28	0	0	256	256	0	0	0	0	0	0	0	1.4938
112	544	134	28	0	0	256	256	0	0	0	0	0	0	0	1.4500
112	544	134	28	0	0	256	256	0	0	0	0	0	0	0	1.4869
112	544	150	30	0	0	256	256	0	0	0	0	0	0	0	1.5006
112	544	150	30	0	0	256	256	0	0	0	0	0	0	0	1.5200
112	544	150	30	0	0	256	256	0	0	0	0	0	0	0	1.4894
112	544	130	24	256	0	0	0	0	0	0	0	0	0	0	1.8344
112	544	130	24	256	0	0	0	0	0	0	0	0	0	0	1.8706
112	544	130	24	256	0	0	0	0	0	0	0	0	0	0	1.8618
112	544	128	28	256	0	0	0	0	0	0	0	0	0	0	1.8379
112	544	128	28	256	0	0	0	0	0	0	0	0	0	0	1.8424
112	544	128	28	256	0	0	0	0	0	0	0	0	0	0	1.8353
112	544	144	30	256	0	0	0	0	0	0	0	0	0	0	1.8185
112	544	144	30	256	0	0	0	0	0	0	0	0	0	0	1.8503

[illegible]

34	288	10	30	256	0	0	0	0	0	0	0	0	0	0	1.3281
34	288	10	30	256	0	0	0	0	0	0	0	0	0	0	1.3475
34	288	10	16	256	256	0	0	0	0	0	0	0	0	0	1.5200
34	288	10	16	256	256	0	0	0	0	0	0	0	0	0	1.4969
34	288	10	16	256	256	0	0	0	0	0	0	0	0	0	1.5156
34	288	10	24	256	256	0	0	0	0	0	0	0	0	0	1.5269
34	288	10	24	256	256	0	0	0	0	0	0	0	0	0	1.5412
34	288	10	24	256	256	0	0	0	0	0	0	0	0	0	1.5038
34	288	10	28	256	0	0	0	0	0	0	0	0	0	0	1.5106
34	288	10	28	256	0	0	0	0	0	0	0	0	0	0	1.5200
34	288	10	28	256	0	0	0	0	0	0	0	0	0	0	1.5431
34	288	10	30	256	256	0	0	0	0	0	0	0	0	0	1.5412
34	288	10	30	256	256	0	0	0	0	0	0	0	0	0	1.5213
34	288	10	30	256	256	0	0	0	0	0	0	0	0	0	1.5213
10	64	20	0	16	0	0	0	0	0	0	0	32	0	40	1.3406
10	64	20	0	16	0	0	0	0	0	0	0	32	0	40	1.3231
10	64	20	0	16	0	0	0	0	0	0	0	32	0	40	1.3219
10	64	20	0	16	0	0	0	0	0	0	0	32	2	40	1.3469
10	64	20	0	16	0	0	0	0	0	0	0	32	2	40	1.3413
10	64	20	0	16	0	0	0	0	0	0	0	32	2	40	1.3363
10	64	20	0	16	0	0	0	0	0	0	0	32	33	40	1.3763
10	64	20	0	16	0	0	0	0	0	0	0	32	33	40	1.3812
10	64	20	0	16	0	0	0	0	0	0	0	32	33	40	1.3669
10	64	14	0	16	0	0	0	0	0	0	0	32	0	40	1.4162
10	64	14	0	16	0	0	0	0	0	0	0	32	0	40	1.4331
10	64	14	0	16	0	0	0	0	0	0	0	32	0	40	1.4313
10	64	14	0	16	0	0	0	0	0	0	0	32	2	40	1.4256
10	64	14	0	16	0	0	0	0	0	0	0	32	2	40	1.4181
10	64	14	0	16	0	0	0	0	0	0	0	32	2	40	1.4181
10	64	14	0	16	0	0	0	0	0	0	0	32	33	40	1.4806
10	64	14	0	16	0	0	0	0	0	0	0	32	33	40	1.4819
10	64	14	0	16	0	0	0	0	0	0	0	32	33	40	1.4781
10	64	12	0	16	0	0	0	0	0	0	0	16	0	33	1.3006
10	64	12	0	16	0	0	0	0	0	0	0	16	0	33	1.2975
10	64	12	0	16	0	0	0	0	0	0	0	16	0	33	1.2881
10	64	12	0	16	0	0	0	0	0	0	0	16	2	33	1.2925
10	64	12	0	16	0	0	0	0	0	0	0	16	2	33	1.3169
10	64	12	0	16	0	0	0	0	0	0	0	16	2	33	1.3112
10	64	12	0	16	0	0	0	0	0	0	0	16	33	33	1.3144
10	64	12	0	16	0	0	0	0	0	0	0	16	33	33	1.3294
10	64	12	0	16	0	0	0	0	0	0	0	16	33	33	1.3219

34	288	10	0	0	0	0	0	256	0	0	0	0	0	0	1.1600
34	288	10	0	0	0	0	0	256	0	0	0	0	0	0	1.0500
34	288	10	0	0	0	0	0	256	0	0	0	0	0	0	1.0456
34	288	10	16	0	0	0	0	256	0	0	0	0	0	0	2.1144
34	288	10	16	0	0	0	0	256	0	0	0	0	0	0	2.1833
34	288	10	16	0	0	0	0	256	0	0	0	0	0	0	2.0433
34	288	10	24	0	0	0	0	256	0	0	0	0	0	0	2.1967
34	288	10	24	0	0	0	0	256	0	0	0	0	0	0	2.1733
34	288	10	24	0	0	0	0	256	0	0	0	0	0	0	2.2600
34	288	10	28	0	0	0	0	256	0	0	0	0	0	0	2.2922
34	288	10	28	0	0	0	0	256	0	0	0	0	0	0	2.3300
34	288	10	28	0	0	0	0	256	0	0	0	0	0	0	2.3256
34	288	10	30	0	0	0	0	256	0	0	0	0	0	0	2.3100
34	288	10	30	0	0	0	0	256	0	0	0	0	0	0	2.3322
34	288	10	30	0	0	0	0	256	0	0	0	0	0	0	2.2911
34	288	10	0	0	0	0	0	0	0	256	0	0	0	0	1.2550
34	288	10	0	0	0	0	0	0	0	256	0	0	0	0	1.2325
34	288	10	0	0	0	0	0	0	0	256	0	0	0	0	1.2575
34	288	10	16	0	0	0	0	0	0	256	0	0	0	0	1.4806
34	288	10	16	0	0	0	0	0	0	256	0	0	0	0	1.4456
34	288	10	16	0	0	0	0	0	0	256	0	0	0	0	1.5100
34	288	10	24	0	0	0	0	0	0	256	0	0	0	0	1.5588
34	288	10	24	0	0	0	0	0	0	256	0	0	0	0	1.5313
34	288	10	24	0	0	0	0	0	0	256	0	0	0	0	1.5375
34	288	10	28	0	0	0	0	0	0	256	0	0	0	0	1.5544
34	288	10	28	0	0	0	0	0	0	256	0	0	0	0	1.5706
34	288	10	28	0	0	0	0	0	0	256	0	0	0	0	1.5588
34	288	10	30	0	0	0	0	0	0	256	0	0	0	0	1.5750
34	288	10	30	0	0	0	0	0	0	256	0	0	0	0	1.5788
34	288	10	30	0	0	0	0	0	0	256	0	0	0	0	1.5700
34	288	10	16	0	0	0	0	256	256	0	0	0	0	0	2.4222
34	288	10	16	0	0	0	0	256	256	0	0	0	0	0	2.4244
34	288	10	16	0	0	0	0	256	256	0	0	0	0	0	2.4411
34	288	10	24	0	0	0	0	256	256	0	0	0	0	0	2.5267
34	288	10	24	0	0	0	0	256	256	0	0	0	0	0	2.4967
34	288	10	24	0	0	0	0	256	256	0	0	0	0	0	2.5544
34	288	10	28	0	0	0	0	256	256	0	0	0	0	0	2.5989
34	288	10	28	0	0	0	0	256	256	0	0	0	0	0	2.5633
34	288	10	28	0	0	0	0	256	256	0	0	0	0	0	2.5367
34	288	10	30	0	0	0	0	256	256	0	0	0	0	0	2.3378
34	288	10	30	0	0	0	0	256	256	0	0	0	0	0	2.3389

34	288	10	30	0	0	0	0	256	256	0	0	0	0	2.3011
34	288	26	0	0	0	0	0	0	256	256	0	0	0	1.2544
34	288	26	0	0	0	0	0	0	256	256	0	0	0	1.2637
34	288	26	0	0	0	0	0	0	256	256	0	0	0	1.2513
34	288	26	16	0	0	0	0	0	0	256	0	0	0	1.8219
34	288	26	16	0	0	0	0	0	0	256	0	0	0	1.8269
34	288	26	16	0	0	0	0	0	0	256	0	0	0	1.7894
34	288	26	24	0	0	0	0	0	0	256	256	0	0	1.8338
34	288	26	24	0	0	0	0	0	0	256	256	0	0	1.8619
34	288	26	24	0	0	0	0	0	0	256	256	0	0	1.8088
34	288	26	28	0	0	0	0	0	0	256	256	0	0	1.8763
34	288	26	28	0	0	0	0	0	0	256	256	0	0	1.8700
34	288	26	28	0	0	0	0	0	0	256	256	0	0	1.8850
34	288	26	30	0	0	0	0	0	0	256	256	0	0	1.8644
34	288	26	30	0	0	0	0	0	0	256	256	0	0	1.8725
34	288	26	30	0	0	0	0	0	0	256	256	0	0	1.8419
34	288	10	0	256	0	0	0	0	0	0	0	0	0	1.3556
34	288	10	0	256	0	0	0	0	0	0	0	0	0	1.3819
34	288	10	0	256	0	0	0	0	0	0	0	0	0	1.3556
34	288	10	16	256	0	0	0	0	0	0	0	0	0	1.4006
34	288	10	16	256	0	0	0	0	0	0	0	0	0	1.3825
34	288	10	16	256	0	0	0	0	0	0	0	0	0	1.3975
34	288	10	24	256	0	0	0	0	0	0	0	0	0	1.4050
34	288	10	24	256	0	0	0	0	0	0	0	0	0	1.3838
34	288	10	24	256	0	0	0	0	0	0	0	0	0	1.3988
34	288	10	28	256	0	0	0	0	0	0	0	0	0	1.4006
34	288	10	28	256	0	0	0	0	0	0	0	0	0	1.4150
34	288	10	28	256	0	0	0	0	0	0	0	0	0	1.3931
34	288	10	30	256	0	0	0	0	0	0	0	0	0	1.4019
34	288	10	30	256	0	0	0	0	0	0	0	0	0	1.3894
34	288	10	30	256	0	0	0	0	0	0	0	0	0	1.4025
34	288	10	0	0	0	256	0	0	0	0	0	0	0	1.5062
34	288	10	0	0	0	256	0	0	0	0	0	0	0	1.4775
34	288	10	0	0	0	256	0	0	0	0	0	0	0	1.4925
34	288	10	16	0	0	256	0	0	0	0	0	0	0	1.5113
34	288	10	16	0	0	256	0	0	0	0	0	0	0	1.5269
34	288	10	16	0	0	256	0	0	0	0	0	0	0	1.4769
34	288	10	24	0	0	256	0	0	0	0	0	0	0	1.5137
34	288	10	24	0	0	256	0	0	0	0	0	0	0	1.5144
34	288	10	24	0	0	256	0	0	0	0	0	0	0	1.5319
34	288	10	28	0	0	256	0	0	0	0	0	0	0	1.5356

[illegible]

[illegible]

[illegible]

[illegible]

D.3 C6211 Results

Table D.5: C6211 Results Table Legend

ΔPA	Program space address line transitions
TPD	Total program space data lines (loop lines)
ΔPD	Program space data line transitions
ΔDA	Data space address line transitions
TDRI	Total data space internal memory read bits
ΔDRI	Δ data space internal memory read bits
TDWI	Total data space internal memory write bits
ΔDWI	Δ data space internal memory write bits
TDRE	Total data space external memory read bits
ΔDRE	Δ data space external memory read bits
TDWE	Total data space external memory write bits
ΔDWE	Δ data space external memory write bits
TAS	Total ALU source bits
ΔAS	Δ ALU source bits
TAD	Total ALU destination bits
Energy	Normalized nanojoules per cycle

Table D.6: C6211 Results Table

Δ P A	T P D	Δ P D	Δ D A	T D R I	Δ D R I	T D W I	Δ D W I	T D R E	Δ D R E	T D W E	Δ D W E	T A S	Δ A S	T A D	Energy
0	256	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0023
0	256	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0053
0	256	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
80	1280	122	16	512	0	0	0	0	0	0	0	0	0	0	1.0253
80	1280	122	16	512	0	0	0	0	0	0	0	0	0	0	1.0313
80	1280	122	16	512	0	0	0	0	0	0	0	0	0	0	1.0243
80	1280	122	16	512	0	0	0	0	0	0	0	0	0	0	1.3130
80	1280	122	16	512	0	0	0	0	0	0	0	0	0	0	1.3123
80	1280	122	16	512	0	0	0	0	0	0	0	0	0	0	1.3137
0	256	0	0	0	0	0	0	0	0	0	0	64	0	32	0.5217
0	256	0	0	0	0	0	0	0	0	0	0	64	0	32	0.5243
0	256	0	0	0	0	0	0	0	0	0	0	64	0	32	0.5283
0	256	0	0	0	0	0	0	0	0	0	0	128	0	64	0.5263
0	256	0	0	0	0	0	0	0	0	0	0	128	0	64	0.5280
0	256	0	0	0	0	0	0	0	0	0	0	128	0	64	0.5303
0	256	0	0	0	0	0	0	0	0	0	0	64	0	32	0.5220
0	256	0	0	0	0	0	0	0	0	0	0	64	0	32	0.5207
0	256	0	0	0	0	0	0	0	0	0	0	64	0	32	0.5237
0	256	0	0	0	0	0	0	0	0	0	0	128	0	64	0.5277
0	256	0	0	0	0	0	0	0	0	0	0	128	0	64	0.5270
0	256	0	0	0	0	0	0	0	0	0	0	128	0	64	0.5270
0	256	0	0	0	0	0	0	0	0	0	0	64	0	32	0.5227
0	256	0	0	0	0	0	0	0	0	0	0	64	0	32	0.5217
0	256	0	0	0	0	0	0	0	0	0	0	64	0	32	0.5183
0	256	0	0	0	0	0	0	0	0	0	0	128	0	64	0.5297
0	256	0	0	0	0	0	0	0	0	0	0	128	0	64	0.5260
0	256	0	0	0	0	0	0	0	0	0	0	128	0	64	0.5280
64	1280	122	0	0	0	512	0	0	0	0	0	0	0	0	1.6717
64	1280	122	0	0	0	512	0	0	0	0	0	0	0	0	1.6717
64	1280	122	0	0	0	512	0	0	0	0	0	0	0	0	1.6743
64	1280	266	0	0	0	512	0	0	0	0	0	0	0	0	2.3703
64	1280	266	0	0	0	512	0	0	0	0	0	0	0	0	2.3703
64	1280	266	0	0	0	512	0	0	0	0	0	0	0	0	2.3690
0	256	0	0	0	0	0	0	0	0	0	0	64	0	32	0.5217
0	256	0	0	0	0	0	0	0	0	0	0	64	0	32	0.5220

[illegible]

80	1280	150	16	512	512	0	0	0	0	0	0	0	0	0	1.3180
80	1280	150	16	512	512	0	0	0	0	0	0	0	0	0	1.3167
80	1280	124	16	512	0	0	0	0	0	0	0	0	0	0	1.0120
80	1280	124	16	512	0	0	0	0	0	0	0	0	0	0	1.0183
80	1280	124	16	512	0	0	0	0	0	0	0	0	0	0	1.0110
80	1280	124	16	512	512	0	0	0	0	0	0	0	0	0	1.3230
80	1280	124	16	512	512	0	0	0	0	0	0	0	0	0	1.3207
80	1280	124	16	512	512	0	0	0	0	0	0	0	0	0	1.3217
80	1280	144	16	512	512	0	0	0	0	0	0	0	0	0	1.0110
80	1280	144	16	512	512	0	0	0	0	0	0	0	0	0	1.0093
80	1280	144	16	512	512	0	0	0	0	0	0	0	0	0	1.0090
80	1280	144	16	512	512	0	0	0	0	0	0	0	0	0	1.3187
80	1280	144	16	512	512	0	0	0	0	0	0	0	0	0	1.3170
80	1280	144	16	512	512	0	0	0	0	0	0	0	0	0	1.3177
0	256	0	0	0	0	32	0	0	0	0	0	0	0	0	1.6770
0	256	0	0	0	0	32	0	0	0	0	0	0	0	0	1.6773
0	256	0	0	0	0	32	0	0	0	0	0	0	0	0	1.6783
0	256	0	0	0	32	32	0	0	0	0	0	0	0	0	2.3723
0	256	0	0	0	32	32	0	0	0	0	0	0	0	0	2.3737
0	256	0	0	0	32	32	0	0	0	0	0	0	0	0	2.3737
0	256	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5120
0	256	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5127
0	256	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5147
0	256	0	0	32	0	0	0	0	0	0	0	0	0	0	1.0097
0	256	0	0	32	0	0	0	0	0	0	0	0	0	0	1.0097
0	256	0	0	32	0	0	0	0	0	0	0	0	0	0	1.0107
80	1280	150	30	512	0	0	0	0	0	0	0	0	0	0	1.0103
80	1280	150	30	512	0	0	0	0	0	0	0	0	0	0	1.0097
80	1280	150	30	512	0	0	0	0	0	0	0	0	0	0	1.0073
80	1280	150	30	512	512	0	0	0	0	0	0	0	0	0	1.3520
80	1280	150	30	512	512	0	0	0	0	0	0	0	0	0	1.3527
80	1280	150	30	512	512	0	0	0	0	0	0	0	0	0	1.3510
64	1280	132	16	0	0	512	0	0	0	0	0	0	0	0	1.7120
64	1280	132	16	0	0	512	0	0	0	0	0	0	0	0	1.7127
64	1280	132	16	0	0	512	0	0	0	0	0	0	0	0	1.7133
64	1280	270	16	0	0	512	512	0	0	0	0	0	0	0	2.4097
64	1280	270	16	0	0	512	512	0	0	0	0	0	0	0	2.4080
64	1280	270	16	0	0	512	512	0	0	0	0	0	0	0	2.4120
64	1280	194	16	0	0	512	128	0	0	0	0	0	0	0	1.9060
64	1280	194	16	0	0	512	128	0	0	0	0	0	0	0	1.9050
64	1280	194	16	0	0	512	128	0	0	0	0	0	0	0	1.9050

64	1280	152	16	0	0	512	0	0	0	0	0	0	0	0	1.7127
64	1280	152	16	0	0	512	0	0	0	0	0	0	0	0	1.7097
64	1280	152	16	0	0	512	0	0	0	0	0	0	0	0	1.7150
64	1280	290	16	0	0	512	512	0	0	0	0	0	0	0	2.4103
64	1280	290	16	0	0	512	512	0	0	0	0	0	0	0	2.4070
64	1280	290	16	0	0	512	512	0	0	0	0	0	0	0	2.4093
64	1280	200	16	0	0	512	64	0	0	0	0	0	0	0	1.7950
64	1280	200	16	0	0	512	64	0	0	0	0	0	0	0	1.7960
64	1280	200	16	0	0	512	64	0	0	0	0	0	0	0	1.7943
64	1280	158	16	0	0	512	0	0	0	0	0	0	0	0	1.7113
64	1280	158	16	0	0	512	0	0	0	0	0	0	0	0	1.7150
64	1280	158	16	0	0	512	0	0	0	0	0	0	0	0	1.7167
64	1280	284	16	0	0	512	512	0	0	0	0	0	0	0	2.4103
64	1280	284	16	0	0	512	512	0	0	0	0	0	0	0	2.4107
64	1280	284	16	0	0	512	512	0	0	0	0	0	0	0	2.4113
64	1280	186	16	0	0	512	32	0	0	0	0	0	0	0	1.7473
64	1280	186	16	0	0	512	32	0	0	0	0	0	0	0	1.7457
64	1280	186	16	0	0	512	32	0	0	0	0	0	0	0	1.7460
64	1280	158	30	0	0	512	0	0	0	0	0	0	0	0	1.7467
64	1280	158	30	0	0	512	0	0	0	0	0	0	0	0	1.7460
64	1280	158	30	0	0	512	0	0	0	0	0	0	0	0	1.7437
64	1280	280	30	0	0	512	512	0	0	0	0	0	0	0	2.4413
64	1280	280	30	0	0	512	512	0	0	0	0	0	0	0	2.4410
64	1280	280	30	0	0	512	512	0	0	0	0	0	0	0	2.4417
64	1280	186	30	0	0	512	32	0	0	0	0	0	0	0	1.7773
64	1280	186	30	0	0	512	32	0	0	0	0	0	0	0	1.7770
64	1280	186	30	0	0	512	32	0	0	0	0	0	0	0	1.7767
64	1280	130	16	0	0	512	0	0	0	0	0	0	0	0	1.7227
64	1280	130	16	0	0	512	0	0	0	0	0	0	0	0	1.7170
64	1280	130	16	0	0	512	0	0	0	0	0	0	0	0	1.7160
64	1280	268	16	0	0	512	512	0	0	0	0	0	0	0	2.4083
64	1280	268	16	0	0	512	512	0	0	0	0	0	0	0	2.4067
64	1280	268	16	0	0	512	512	0	0	0	0	0	0	0	2.4090
64	1280	248	16	0	0	512	256	0	0	0	0	0	0	0	2.0727
64	1280	248	16	0	0	512	256	0	0	0	0	0	0	0	2.0720
64	1280	248	16	0	0	512	256	0	0	0	0	0	0	0	2.0710
64	1280	136	24	0	0	512	0	0	0	0	0	0	0	0	1.7260
64	1280	136	24	0	0	512	0	0	0	0	0	0	0	0	1.7263
64	1280	136	24	0	0	512	0	0	0	0	0	0	0	0	1.7253
64	1280	268	24	0	0	512	512	0	0	0	0	0	0	0	2.4280
64	1280	268	24	0	0	512	512	0	0	0	0	0	0	0	2.4267

[illegible]

D.4 Data Reduction

D.4.1 56303 Data Reduction

D.4.1.1 Internal and External Memory Access

Table D.7: 56303 Data Reduction SPSS Output Notes

Output Created	18-FEB-2002 23:34:47	
Comments		
Input	Data	M:\school\thesis\code\56k\56kData_2.sav
	Filter	<none>
	Weight	<none>
	Split File	<none>
	Rows in Working Data File	657
Missing Value Handling	Definition of Missing	User-defined missing values are treated as missing.
	Cases Used	No missing values for any variable used.
Syntax		REGRESSION /DESCRIPTIVES MEAN STDDEV CORR SIG N /MISSING LISTWISE /STATISTICS COEFF OUTS CI BCOV R ANOVA COLLIN TOL CHANGE ZPP /CRITERIA=PIN(.05) POUT(.10) /NOORIGIN /DEPENDENT energy /METHOD=STEPWISE ΔPA TPD ΔPD ΔDA $TDRI$ ΔDRI $TDWI$ ΔDWI $TDRE$ ΔDRE $TDWE$ ΔDWE TAS ΔAS TAD .

Table D.8: Descriptive Statistics

	Mean	Std. Deviation	N
ENERGY	31.029116	25.669952	657
ΔPA	27.23	11.67	657
TPD	350.79	140.03	657
ΔPD	30.01	27.01	657
ΔDA	15.28	10.55	657
TDRI	82.30	198.23	657
ΔDRI	30.68	126.82	657
TDWI	109.92	217.85	657
ΔDWI	28.05	114.82	657
TDRE	73.64	194.23	657
ΔDRE	35.07	142.64	657
TDWE	108.05	219.21	657
ΔDWE	36.38	135.10	657
TAS	2.88	11.29	657
ΔAS	1.00	6.60	657
TAD	3.40	12.98	657

Table D.9: Pearson Correlation, part A

	ENERGY	ΔPA	TPD	ΔPD	ΔDA	TDRI	ΔDRI	TDWI
ENERGY	1.000	.351	.351	.132	.243	-.285	-.157	-.338
ΔPA	.351	1.000	1.000	.438	.712	.226	.141	.226
TPD	.351	1.000	1.000	.438	.712	.226	.141	.226
ΔPD	.132	.438	.438	1.000	.458	.063	.021	.095
ΔDA	.243	.712	.712	.458	1.000	.172	.040	.130
TDRI	-.285	.226	.226	.063	.172	1.000	.302	-.210
ΔDRI	-.157	.141	.141	.021	.040	.302	1.000	.171
TDWI	-.338	.226	.226	.095	.130	-.210	.171	1.000
ΔDWI	-.153	.142	.142	.130	.139	-.102	-.059	.517
TDRE	.623	.220	.220	.091	.172	-.158	-.092	-.192
ΔDRE	.480	.143	.143	.004	.085	-.102	-.060	-.124
TDWE	.585	.264	.264	.112	.189	-.205	-.119	-.249
ΔDWE	.444	.156	.156	.131	.125	-.112	-.065	-.136
TAS	-.174	-.464	-.464	-.185	-.370	-.106	-.062	-.129
ΔAS	-.099	-.275	-.275	-.109	-.219	-.063	-.037	-.076
TAD	-.179	-.477	-.477	-.192	-.380	-.109	-.063	-.132

Table D.10: Pearson Correlation, part B

	ΔDWI	TDRE	DRE	TDWE	ΔDWE	TAS	ΔAS	TAD
ENERGY	-.153	.623	.480	.585	.444	-.174	-.099	-.179
ΔPA	.142	.220	.143	.264	.156	-.464	-.275	-.477
TPD	.142	.220	.143	.264	.156	-.464	-.275	-.477
ΔPD	.130	.091	.004	.112	.131	-.185	-.109	-.192
ΔDA	.139	.172	.085	.189	.125	-.370	-.219	-.380
TDRI	-.102	-.158	-.102	-.205	-.112	-.106	-.063	-.109
ΔDRI	-.059	-.092	-.060	-.119	-.065	-.062	-.037	-.063
TDWI	.517	-.192	-.124	-.249	-.136	-.129	-.076	-.132
ΔDWI	1.000	-.093	-.060	-.121	-.066	-.062	-.037	-.064
TDRE	-.093	1.000	.661	-.187	-.102	-.097	-.057	-.099
ΔDRE	-.060	.661	1.000	-.100	-.066	-.063	-.037	-.064
TDWE	-.121	-.187	-.100	1.000	.624	-.126	-.074	-.129
ΔDWE	-.066	-.102	-.066	.624	1.000	-.069	-.041	-.071
TAS	-.062	-.097	-.063	-.126	-.069	1.000	.589	.960
ΔAS	-.037	-.057	-.037	-.074	-.041	.589	1.000	.568
TAD	-.064	-.099	-.064	-.129	-.071	.960	.568	1.000

Table D.11: Significance: 1-tailed, part A

	ENERGY	ΔPA	TPD	ΔPD	ΔDA	TDRI	ΔDRI	TDWI
ENERGY	.	.000	.000	.000	.000	.000	.000	.000
ΔPA	.000	.	.000	.000	.000	.000	.000	.000
TPD	.000	.000	.	.000	.000	.000	.000	.000
ΔPD	.000	.000	.000	.	.000	.055	.296	.007
ΔDA	.000	.000	.000	.000	.	.000	.152	.000
TDRI	.000	.000	.000	.055	.000	.	.000	.000
ΔDRI	.000	.000	.000	.296	.152	.000	.	.000
TDWI	.000	.000	.000	.007	.000	.000	.000	.
ΔDWI	.000	.000	.000	.000	.000	.005	.065	.000
TDRE	.000	.000	.000	.010	.000	.000	.009	.000
ΔDRE	.000	.000	.000	.464	.015	.004	.064	.001
TDWE	.000	.000	.000	.002	.000	.000	.001	.000
ΔDWE	.000	.000	.000	.000	.001	.002	.047	.000
TAS	.000	.000	.000	.000	.000	.003	.057	.000
ΔAS	.006	.000	.000	.003	.000	.054	.175	.025
TAD	.000	.000	.000	.000	.000	.003	.052	.000

Table D.12: Significance: 1-tailed, part B

	ΔDWI	TDRE	ΔDRE	TDWE	ΔDWE	TAS	ΔAS	TAD
ENERGY	.000	.000	.000	.000	.000	.000	.006	.000
ΔPA	.000	.000	.000	.000	.000	.000	.000	.000
TPD	.000	.000	.000	.000	.000	.000	.000	.000
ΔPD	.000	.010	.464	.002	.000	.000	.003	.000
ΔDA	.000	.000	.015	.000	.001	.000	.000	.000
TDRI	.005	.000	.004	.000	.002	.003	.054	.003
ΔDRI	.065	.009	.064	.001	.047	.057	.175	.052
TDWI	.000	.000	.001	.000	.000	.000	.025	.000
ΔDWI	.	.009	.062	.001	.046	.055	.173	.050
TDRE	.009	.	.000	.000	.004	.007	.071	.005
ΔDRE	.062	.000	.	.005	.045	.054	.171	.049
TDWE	.001	.000	.005	.	.000	.001	.028	.000
ΔDWE	.046	.004	.045	.000	.	.039	.149	.035
TAS	.055	.007	.054	.001	.039	.	.000	.000
ΔAS	.173	.071	.171	.028	.149	.000	.	.000
TAD	.050	.005	.049	.000	.035	.000	.000	.

Table D.13: Number of data points, part A

[illegible]

Table D.14: Number of data points, part B

[illegible]

Table D.15: Variables Entered or Removed^a

Model	Variables		Method
	Entered	Removed	
1	TDRE	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
2	TDWE	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
3	ΔDWE	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
4	ΔDRE	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
5	ΔDA	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).

a. Dependent Variable: ENERGY

Table D.16: Model Summary

Model	R	R^2	Adjusted R^2	Std. Error of the Estimate	R^2 Change	Change Statistics			
						F Change	d of f 1	2	Sig. F Change
1	.623 ^a	.389	.388	20.088529	.389	416.168	1	655	.000
2	.948 ^b	.898	.898	8.195480	.510	3281.399	1	654	.000
3	.952 ^c	.906	.906	7.890951	.008	52.453	1	653	.000
4	.954 ^d	.911	.910	7.696062	.005	34.491	1	652	.000
5	.955 ^e	.911	.911	7.678827	.001	3.930	1	651	.048

a. Predictors: (Constant), TDRE

b. Predictors: (Constant), TDRE, TDWE

c. Predictors: (Constant), TDRE, TDWE, ΔDWE

d. Predictors: (Constant), TDRE, TDWE, ΔDWE , ΔDRE

e. Predictors: (Constant), TDRE, TDWE, ΔDWE , ΔDRE , ΔDA

Table D.17: ANOVA^f

	Model	Σ of Squares	df	Mean Square	F	Sig.
1	Model	167944.254	1	167944.254	416.168	.000 ^a
	Residual	264324.598	655	403.549		
	Total	432268.852	656			
2	Model	388342.355	2	194171.178	2890.919	.000 ^b
	Residual	43926.497	654	67.166		
	Total	432268.852	656			
3	Model	391608.428	3	130536.143	2096.390	.000 ^c
	Residual	40660.424	653	62.267		
	Total	432268.852	656			
4	Model	393651.301	4	98412.825	1661.554	.000 ^d
	Residual	38617.551	652	59.229		
	Total	432268.852	656			
5	Model	393883.039	5	78776.608	1336.003	.000 ^e
	Residual	38385.814	651	58.964		
	Total	432268.852	656			

a. Predictors: (Constant), TDRE

b. Predictors: (Constant), TDRE, TDWE

c. Predictors: (Constant), TDRE, TDWE, ΔDWE

d. Predictors: (Constant), TDRE, TDWE, ΔDWE , ΔDRE

e. Predictors: (Constant), TDRE, TDWE, ΔDWE , ΔDRE , ΔDA

f. Dependent Variable: ENERGY

Table D.18: Coefficients, part A^a

Model		Unstandardized Coefficients		Standardized Coefficients β	t	Sig.
		$\hat{B}$	Std. Error			
1	(Constant)	24.963	.838		29.779	.000
	TDRE	8.238E-02	.004	.623	20.400	.000
2	(Constant)	14.441	.388		37.200	.000
	TDRE	.100	.002	.759	59.843	.000
	TDWE	8.512E-02	.001	.727	57.283	.000
3	(Constant)	14.571	.374		38.939	.000
	TDRE	.100	.002	.758	62.004	.000
	TDWE	7.695E-02	.002	.657	42.235	.000
	ΔDWE	2.114E-02	.003	.111	7.242	.000
4	(Constant)	14.617	.365		40.043	.000
	TDRE	9.206E-02	.002	.697	44.036	.000
	TDWE	7.653E-02	.002	.654	43.035	.000
	ΔDWE	2.154E-02	.003	.113	7.561	.000
	ΔDRE	1.650E-02	.003	.092	5.873	.000
5	(Constant)	15.395	.535		28.760	.000
	TDRE	9.288E-02	.002	.703	43.685	.000
	TDWE	7.718E-02	.002	.659	42.784	.000
	ΔDWE	2.156E-02	.003	.113	7.587	.000
	ΔDRE	1.624E-02	.003	.090	5.785	.000
	ΔDA	-5.883E-02	.030	-.024	-1.982	.048

a. Dependent Variable: ENERGY

Table D.19: Coefficients, part B^a

Model		95% Confidence Interval for $\hat{B}$		Correlations			Colinearity Statistics	
		Lower Bound	Upper Bound	Zero Order	Partial	Part	Tolerance	VIF
1	(Constant)	23.317	26.608					
	TDRE	.074	.090	.623	.623	.623	1.000	1.000
2	(Constant)	13.678	15.203					
	TDRE	.097	.104	.623	.920	.746	.965	1.036
	TDWE	.082	.088	.585	.913	.714	.965	1.036
3	(Constant)	13.836	15.305					
	TDRE	.097	.103	.623	.925	.744	.965	1.037
	TDWE	.073	.081	.585	.856	.507	.595	1.681
	ΔDWE	.015	.027	.444	.273	.087	.610	1.639
4	(Constant)	13.900	15.334					
	TDRE	.088	.096	.623	.865	.515	.548	1.826
	TDWE	.073	.080	.585	.860	.504	.594	1.683
	ΔDWE	.016	.027	.444	.284	.089	.610	1.640
	ΔDRE	.011	.022	.480	.224	.069	.562	1.779
5	(Constant)	14.344	16.446					
	TDRE	.089	.097	.623	.864	.510	.527	1.897
	TDWE	.074	.081	.585	.859	.500	.575	1.740
	ΔDWE	.016	.027	.444	.285	.089	.610	1.640
	ΔDRE	.011	.022	.480	.221	.068	.561	1.784
	ΔDA	-.117	-.001	.243	-.077	-.023	.918	1.090

a. Dependent Variable: ENERGY

Table D.20: Excluded Variables^f

Model		β in	t	Sig.	Partial Correlation	Colinearity Statistics		
						Tolerance	VIF	Minimum Tolerance
1	ΔPA	.225 ^a	7.472	.000	.280	.952	1.051	.952
	TPD	.225 ^a	7.472	.000	.280	.952	1.051	.952
	ΔPD	.076 ^a	2.474	.014	.096	.992	1.008	.992
	ΔDA	.140 ^a	4.580	.000	.176	.971	1.030	.971
	TDRI	-.191 ^a	-6.361	.000	-.241	.975	1.025	.975
	ΔDRI	-.100 ^a	-3.299	.001	-.128	.992	1.009	.992
	TDWI	-.227 ^a	-7.592	.000	-.285	.963	1.038	.963
	ΔDWI	-.096 ^a	-3.154	.002	-.122	.991	1.009	.991
	ΔDRE	.120 ^a	2.958	.003	.115	.563	1.777	.563
	TDWE	.727 ^a	57.283	.000	.913	.965	1.036	.965
	ΔDWE	.513 ^a	22.029	.000	.653	.990	1.011	.990
	TAS	-.114 ^a	-3.763	.000	-.146	.991	1.009	.991
	ΔAS	-.063 ^a	-2.067	.039	-.081	.997	1.003	.997
	TAD	-.118 ^a	-3.887	.000	-.150	.990	1.010	.990
2	ΔPA	-.009 ^b	-.676	.500	-.026	.855	1.169	.855
	TPD	-.009 ^b	-.676	.500	-.026	.855	1.169	.855
	ΔPD	-.019 ^b	-1.527	.127	-.060	.974	1.026	.948
	ΔDA	-.027 ^b	-2.087	.037	-.081	.920	1.087	.915
	TDRI	-.017 ^b	-1.336	.182	-.052	.918	1.089	.909
	ΔDRI	.000 ^b	-.023	.981	-.001	.972	1.029	.946
	TDWI	-.013 ^b	-.967	.334	-.038	.879	1.137	.879
	ΔDWI	.005 ^b	.406	.685	.016	.972	1.029	.946
	ΔDRE	.089 ^b	5.467	.000	.209	.562	1.778	.548
	ΔDWE	.111 ^b	7.242	.000	.273	.610	1.639	.595
	TAS	-.009 ^b	-.706	.481	-.028	.969	1.032	.944
	ΔAS	-.001 ^b	-.080	.936	-.003	.989	1.011	.958
	TAD	-.010 ^b	-.772	.440	-.030	.967	1.034	.943
	ΔPA	-.008 ^c	-.578	.563	-.023	.855	1.170	.553
	TPD	-.008 ^c	-.578	.563	-.023	.855	1.170	.553
	ΔPD	-.026 ^c	-2.150	.032	-.084	.969	1.032	.593
	ΔDA	-.028 ^c	-2.207	.028	-.086	.920	1.087	.576

3	TDRI	-.020 ^c	-1.571	.117	-.061	.918	1.090	.570
	ΔDRI	-.002 ^c	-.128	.898	-.005	.972	1.029	.587
	TDWI	-.016 ^c	-1.232	.218	-.048	.878	1.139	.557
	ΔDWI	.004 ^c	.317	.752	.012	.971	1.029	.586
	ΔDRE	.092 ^c	5.873	.000	.224	.562	1.779	.548
	TAS	-.010 ^c	-.843	.400	-.033	.969	1.032	.586
	ΔAS	-.002 ^c	-.147	.883	-.006	.989	1.011	.592
	TAD	-.011 ^c	-.914	.361	-.036	.967	1.034	.585
4	ΔPA	-.006 ^d	-.503	.615	-.020	.855	1.170	.519
	TPD	-.006 ^d	-.503	.615	-.020	.855	1.170	.519
	ΔPD	-.021 ^d	-1.742	.082	-.068	.963	1.039	.537
	ΔDA	-.024 ^d	-1.982	.048	-.077	.918	1.090	.527
	TDRI	-.020 ^d	-1.679	.094	-.066	.917	1.090	.533
	ΔDRI	-.002 ^d	-.169	.866	-.007	.972	1.029	.543
	TDWI	-.017 ^d	-1.348	.178	-.053	.878	1.139	.526
	ΔDWI	.003 ^d	.286	.775	.011	.971	1.029	.543
	TAS	-.011 ^d	-.904	.366	-.035	.969	1.032	.542
	ΔAS	-.002 ^d	-.174	.862	-.007	.989	1.011	.546
	TAD	-.012 ^d	-.979	.328	-.038	.967	1.034	.542
5	ΔPA	.020 ^e	1.157	.248	.045	.458	2.185	.458
	TPD	.020 ^e	1.157	.248	.045	.458	2.185	.458
	ΔPD	-.013 ^e	-.977	.329	-.038	.780	1.282	.525
	TDRI	-.015 ^e	-1.180	.238	-.046	.848	1.180	.502
	ΔDRI	.000 ^e	.014	.989	.001	.964	1.038	.521
	TDWI	-.011 ^e	-.874	.382	-.034	.821	1.218	.494
	ΔDWI	.008 ^e	.693	.488	.027	.933	1.071	.518
	TAS	-.021 ^e	-1.681	.093	-.066	.857	1.166	.526
	ΔAS	-.007 ^e	-.580	.562	-.023	.950	1.053	.527
	TAD	-.023 ^e	-1.788	.074	-.070	.850	1.177	.526

- a. Predictors in the Model: (Constant), TDRE
- b. Predictors in the Model: (Constant), TDRE, TDWE
- c. Predictors in the Model: (Constant), TDRE, TDWE, ΔDWE
- d. Predictors in the Model: (Constant), TDRE, TDWE, ΔDWE , ΔDRE

e. Predictors in the Model: (Constant), TDRE, TDWE, ΔDWE , ΔDRE , ΔDA

f. Dependent Variable: ENERGY

Table D.21: Coefficient Correlations^a

Model			TDRE	TDWE	ΔDWE	ΔDRE	ΔDA
1	Correlations	TDRE	1.000				
	Covariances	TDRE	1.631E-05				
2	Correlations	TDRE	1.000	.187			
		TDWE	.187	1.000			
	Covariances	TDRE	2.812E-06	4.664E-07			
		TDWE	4.664E-07	2.208E-06			
3	Correlations	TDRE	1.000	.159	-.019		
		TDWE	.159	1.000	-.619		
		ΔDWE	-.019	-.619	1.000		
	Covariances	TDRE	2.608E-06	4.670E-07	-8.946E-08		
		TDWE	4.670E-07	3.319E-06	-3.293E-06		
		ΔDWE	-8.946E-08	-3.293E-06	8.523E-06		
4	Correlations	TDRE	1.000	.146	-.030	-.658	
		TDWE	.146	1.000	-.619	-.040	
		ΔDWE	-.030	-.619	1.000	.023	
		ΔDRE	-.658	-.040	.023	1.000	
	Covariances	TDRE	4.371E-06	5.420E-07	-1.768E-07	-3.863E-06	
		TDWE	5.420E-07	3.163E-06	-3.137E-06	-1.999E-07	
		ΔDWE	-1.768E-07	-3.137E-06	8.112E-06	1.874E-07	
		ΔDRE	-3.863E-06	-1.999E-07	1.874E-07	7.896E-06	
5	Correlations	TDRE	1.000	.175	-.028	-.654	-.193
		TDWE	.175	1.000	-.609	-.048	-.180
		ΔDWE	-.028	-.609	1.000	.023	-.004
		ΔDRE	-.654	-.048	.023	1.000	.048
		ΔDA	-.193	-.180	-.004	.048	1.000
	Covariances	TDRE	4.520E-06	6.731E-07	-1.710E-07	-3.901E-06	-1.220E-05
		TDWE	6.731E-07	3.254E-06	-3.119E-06	-2.426E-07	-9.630E-06
		ΔDWE	-1.710E-07	-3.119E-06	8.075E-06	1.849E-07	-3.590E-07
		ΔDRE	-3.901E-06	-2.426E-07	1.849E-07	7.879E-06	3.983E-06
		ΔDA	-1.220E-05	-9.630E-06	-3.590E-07	3.983E-06	8.805E-04

a. Dependent Variable: ENERGY

Table D.22: Colinearity Diagnostics^a

M o d e l	D i m	Eigenvalue	Condition Index	Variance Properties					
				C o n s t	T D R E	T D W E	Δ <i>D</i> <i>W</i> <i>E</i>	Δ <i>D</i> <i>R</i> <i>E</i>	Δ <i>D</i> <i>A</i>
1	1	1.355	1.000	.32	.32				
	2	.645	1.449	.68	.68				
2	1	1.567	1.000	.22	.11	.15			
	2	1.000	1.252	.00	.51	.30			
	3	.433	1.903	.78	.38	.55			
3	1	1.958	1.000	.09	.02	.09	.09		
	2	1.183	1.286	.10	.45	.03	.06		
	3	.563	1.866	.54	.45	.00	.25		
	4	.296	2.571	.27	.09	.87	.60		
4	1	2.076	1.000	.09	.04	.05	.05	.04	
	2	1.663	1.117	.00	.08	.06	.08	.09	
	3	.656	1.779	.64	.01	.00	.21	.11	
	4	.337	2.483	.02	.31	.40	.41	.39	
	5	.268	2.782	.26	.56	.48	.26	.37	
5	1	2.710	1.000	.03	.02	.03	.02	.02	.03
	2	1.664	1.276	.00	.08	.06	.07	.09	.00
	3	.844	1.792	.08	.02	.03	.20	.12	.07
	4	.339	2.828	.00	.27	.42	.45	.35	.01
	5	.269	3.173	.07	.58	.46	.26	.41	.01
	6	.174	3.943	.82	.03	.00	.00	.01	.89

a. Dependent Variable: ENERGY

D.4.1.2 Internal Memory Access Only

Table D.23: Data Reduction SPSS Output Notes

Output Created	18-FEB-2002 23:50:46	
Comments		
Input	Data	M:\school\thesis\code\56k\56kDataNoE_1.sav
	Filter	<none>
	Weight	<none>
	Split File	<none>
	Rows in Working Data File	405
Missing Value Handling	Definition of Missing	User-defined missing values are treated as missing.
	Cases Used	No missing values for any variable used.
Syntax		REGRESSION /DESCRIPTIVES MEAN STDDEV CORR SIG N /MISSING LISTWISE /STATISTICS COEFF OUTS CI BCOV R ANOVA COLLIN TOL CHANGE ZPP /CRITERIA=PIN(.05) POUT(.10) /NOORIGIN /DEPENDENT energy /METHOD=STEPWISE ΔPA TPD ΔPD ΔDA $TDRI$ ΔDRI $TDWI$ ΔDWI TAS ΔAS TAD .

Table D.24: Descriptive Statistics

	Mean	Std. Deviation	N
ENERGY	14.595015	8.515326	405
ΔPA	24.30	12.96	405
TPD	315.56	155.58	405
ΔPD	26.34	25.59	405
ΔDA	13.11	10.69	405
$TDRI$	133.51	238.65	405
ΔDRI	49.78	158.63	405
$TDWI$	178.31	254.63	405
ΔDWI	45.51	143.56	405
TAS	4.67	14.09	405
ΔAS	1.61	8.35	405
TAD	5.51	16.18	405

Table D.25: Pearson Correlation, part A

	ENERGY	ΔPA	TPD	ΔPD	ΔDA	TDRI	ΔDRI	TDWI
ENERGY	1.000	.119	.119	.008	.037	-.075	-.007	-.060
ΔPA	.119	1.000	1.000	.488	.754	.402	.235	.442
TPD	.119	1.000	1.000	.488	.754	.402	.235	.442
ΔPD	.008	.488	.488	1.000	.501	.169	.074	.240
ΔDA	.037	.754	.754	.501	1.000	.343	.115	.320
TDRI	-.075	.402	.402	.169	.343	1.000	.258	-.393
ΔDRI	-.007	.235	.235	.074	.115	.258	1.000	.105
TDWI	-.060	.442	.442	.240	.320	-.393	.105	1.000
ΔDWI	.014	.238	.238	.224	.243	-.178	-.100	.489
TAS	-.040	-.469	-.469	-.206	-.407	-.186	-.104	-.233
ΔAS	-.007	-.273	-.273	-.120	-.238	-.108	-.061	-.136
TAD	-.043	-.482	-.482	-.215	-.419	-.191	-.107	-.239

Table D.26: Pearson Correlation, part B

	ΔDWI	TAS	ΔAS	TAD
ENERGY	.014	-.040	-.007	-.043
ΔPA	.238	-.469	-.273	-.482
TPD	.238	-.469	-.273	-.482
ΔPD	.224	-.206	-.120	-.215
ΔDA	.243	-.407	-.238	-.419
TDRI	-.178	-.186	-.108	-.191
ΔDRI	-.100	-.104	-.061	-.107
TDWI	.489	-.233	-.136	-.239
ΔDWI	1.000	-.105	-.061	-.108
TAS	-.105	1.000	.581	.958
ΔAS	-.061	.581	1.000	.559
TAD	-.108	.958	.559	1.000

Table D.27: Significance: 1-tailed, part A

	ENERGY	ΔPA	TPD	ΔPD	ΔDA	TDRI	ΔDRI	TDWI
ENERGY	.	.009	.009	.433	.230	.066	.446	.114
ΔPA	.009	.	.000	.000	.000	.000	.000	.000
TPD	.009	.000	.	.000	.000	.000	.000	.000
ΔPD	.433	.000	.000	.	.000	.000	.069	.000
ΔDA	.230	.000	.000	.000	.	.000	.010	.000
TDRI	.066	.000	.000	.000	.000	.	.000	.000
ΔDRI	.446	.000	.000	.069	.010	.000	.	.017
TDWI	.114	.000	.000	.000	.000	.000	.017	.
ΔDWI	.391	.000	.000	.000	.000	.000	.022	.000
TAS	.211	.000	.000	.000	.000	.000	.018	.000
ΔAS	.441	.000	.000	.008	.000	.015	.111	.003
TAD	.192	.000	.000	.000	.000	.000	.016	.000

Table D.28: Significance: 1-tailed, part B

	ΔDWI	TAS	ΔAS	TAD
ENERGY	.391	.211	.441	.192
ΔPA	.000	.000	.000	.000
TPD	.000	.000	.000	.000
ΔPD	.000	.000	.008	.000
ΔDA	.000	.000	.000	.000
TDRI	.000	.000	.015	.000
ΔDRI	.022	.018	.111	.016
TDWI	.000	.000	.003	.000
ΔDWI	.	.017	.109	.015
TAS	.017	.	.000	.000
ΔAS	.109	.000	.	.000
TAD	.015	.000	.000	.

Table D.29: Number of data points, part A

	ENERGY	ΔPA	TPD	ΔPD	ΔDA	TDRI	ΔDRI	TDWI
ENERGY	405	405	405	405	405	405	405	405
ΔPA	405	405	405	405	405	405	405	405
TPD	405	405	405	405	405	405	405	405
ΔPD	405	405	405	405	405	405	405	405
ΔDA	405	405	405	405	405	405	405	405
TDRI	405	405	405	405	405	405	405	405
ΔDRI	405	405	405	405	405	405	405	405
TDWI	405	405	405	405	405	405	405	405
ΔDWI	405	405	405	405	405	405	405	405
TAS	405	405	405	405	405	405	405	405
ΔAS	405	405	405	405	405	405	405	405
TAD	405	405	405	405	405	405	405	405

Table D.30: Number of data points, part B

	ΔDWI	TAS	ΔAS	TAD
ENERGY	405	405	405	405
ΔPA	405	405	405	405
TPD	405	405	405	405
ΔPD	405	405	405	405
ΔDA	405	405	405	405
TDRI	405	405	405	405
ΔDRI	405	405	405	405
TDWI	405	405	405	405
ΔDWI	405	405	405	405
TAS	405	405	405	405
ΔAS	405	405	405	405
TAD	405	405	405	405

Table D.31: Variables Entered or Removed^a

Model	Variables		Method
	Entered	Removed	
1	ΔPA	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
2	TDRI	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
3	TDWI	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).

a. Dependent Variable: ENERGY

Table D.32: Model Summary

Model	R	R^2	Adjusted R^2	Std. Error of the Estimate	Change Statistics				
					R^2 Change	F Change	d of f		Sig. F Change
1	.119 ^a	.014	.012	8.4657664	.014	5.744	1	403	.017
2	.179 ^b	.032	.027	8.3988375	.018	7.448	1	402	.007
3	.352 ^c	.124	.117	7.9999702	.092	42.086	1	401	.000

a. Predictors: (Constant), ΔPA

b. Predictors: (Constant), ΔPA , TDRI

c. Predictors: (Constant), ΔPA , TDRI, TDWI

Table D.33: ANOVA^f

Model		Σ of Squares	df	Mean Square	F	Sig.
1	Model	411.667	1	411.667	5.744	.017 ^a
	Residual	28882.688	403	71.669		
	Total	29294.355	404			
2	Model	937.085	2	468.543	6.642	.001 ^b
	Residual	28357.269	402	70.540		
	Total	29294.355	404			
3	Model	3630.546	3	1210.182	18.909	.000 ^c
	Residual	25663.809	401	64.000		
	Total	29294.355	404			

- a. Predictors: (Constant), ΔPA
- b. Predictors: (Constant), ΔPA , TDRI
- c. Predictors: (Constant), ΔPA , TDRI, TDWI
- d. Dependent Variable: ENERGY

Table D.34: Coefficients, part A^a

Model		Unstandardized Coefficients		Standardized Coefficients β	t	Sig.
		$\hat{B}$	Std. Error			
1	(Constant)	12.703	.894		14.203	.000
	ΔPA	.708	.032	.119	2.397	.017
2	(Constant)	12.462	.892		13.975	.000
	ΔPA	.116	.035	.177	3.309	.001
	TDRI	-.005	.002	-.146	-2.729	.007
3	(Constant)	11.425	.864		13.219	.000
	ΔPA	.337	.048	.513	7.058	.000
	TDRI	-.017	.003	-.466	-6.566	.000
	TDWI	-.016	.002	-.470	-6.487	.000

a. Dependent Variable: ENERGY

Table D.35: Coefficients, part B^a

Model		95% Confidence Interval for $\hat{B}$		Correlations			Colinearity Statistics	
		Lower Bound	Upper Bound	Zero Order	Partial	Part	Tolerance	VIF
1	(Constant)	10.945	14.462					
	ΔPA	.014	.142	.119	.119	.119	1.000	1.000
2	(Constant)	10.709	14.215					
	ΔPA	.047	.186	.119	.162	.162	.838	1.193
	TDRI	-.009	-.001	-.075	-.134	-.134	.838	1.193
3	(Constant)	9.726	13.124					
	ΔPA	.243	.431	.119	.330	.330	.413	2.419
	TDRI	-.022	-.012	-.075	-.307	-.307	.434	2.302
	TDWI	-.020	-.011	-.060	-.303	-.303	.417	2.398

a. Dependent Variable: ENERGY

Table D.36: Excluded Variables^f

Model		β in	t	Sig.	Partial Correlation	Colinearity Statistics		
						Tolerance	VIF	Minimum Tolerance
1	TPD	. ^a	.	.	.	.000	.	.000
	ΔPD	-.065 ^a	-1.145	.253	-.057	.762	1.313	.762
	ΔDA	-.122 ^a	-1.619	.106	-.080	.432	2.315	.432
	TDRI	-.146 ^a	-2.729	.007	-.135	.838	1.193	.838
	ΔDRI	-.037 ^a	-.722	.471	-.036	.945	1.059	.945
	TDWI	-.140 ^a	-2.550	.011	-.126	.805	1.242	.805
	ΔDWI	-.015 ^a	-.330	.764	-.015	.943	1.060	.943
	TAS	.020 ^a	.356	.722	.018	.780	1.281	.780
	ΔAS	.027 ^a	.527	.599	.026	.925	1.081	.925
	TAD	.018 ^a	.317	.751	.016	.768	1.302	.768
2	TPD	. ^b	.	.	.	.000	.	.000
	ΔPD	-.070 ^b	-1.247	.213	-.062	.761	1.314	.657
	ΔDA	-.109 ^b	-1.453	.147	-.072	.430	2.325	.409
	ΔDRI	-.012 ^b	-.231	.817	-.012	.913	1.096	.810
	TDWI	-.470 ^b	-6.487	.000	-.308	.417	2.398	.413
	ΔDWI	-.064 ^b	-1.200	.231	-.060	.854	1.171	.740
	TAS	.020 ^b	.367	.714	.018	.780	1.281	.678
	ΔAS	.027 ^b	.535	.593	.027	.925	1.081	.785
	TAD	.018 ^b	.329	.743	-.016	.768	1.302	.668
3	TPD	. ^c	.	.	.	.000	.	.000
	ΔPD	-.066 ^c	-1.236	.217	-.062	.761	1.314	.366
	ΔDA	-.093 ^c	-1.305	.193	-.065	.430	2.328	.278
	ΔDRI	.047 ^c	.951	.342	.047	.883	1.133	.403
	ΔDWI	.051 ^c	.944	.346	.047	.760	1.315	.371
	TAS	-.006 ^c	.115	.908	.006	.779	1.284	.377
	ΔAS	-.020 ^c	.417	.677	.021	.925	1.081	.402
	TAD	-.003 ^c	.065	.948	.003	.766	1.305	.374

a. Predictors: (Constant), ΔPA b. Predictors: (Constant), ΔPA , TDRIc. Predictors: (Constant), ΔPA , TDRI, TDWI

d. Dependent Variable: ENERGY

Table D.37: Coefficient Correlations^a

Model			ΔPA	TDRI	TDWI
1	Correlations	ΔPA	1.000		
	Covariances	ΔPA	.001		
2	Correlations	ΔPA	1.000	-.402	
		TDRI	-.402	1.000	
	Covariances	ΔPA	.001	-2.705E-05	
		TDRI	-2.705E-05	3.656E-06	
3	Correlations	ΔPA	1.000	-.697	-.712
		TDRI	-.697	1.000	.694
		TDWI	-.712	.694	1.000
	Covariances	ΔPA	.002	-8.425E-05	-8.228E-05
		TDRI	-8.425E-05	6.403E-06	4.252E-06
		TDWI	-8.228E-05	4.252E-06	5.859E-06

a. Dependent Variable: ENERGY

Table D.38: Colinearity Diagnostics^a

Model	Dim.	Eigenvalue	Condition Index	Variance Properties			
				Const.	ΔPA	TDRI	TDWI
1	1	1.882	1.000	.06	.06		
	2	.118	4.002	.94	.94		
2	1	2.327	1.000	.03	.03	.07	
	2	.566	2.028	.08	.02	.82	
	3	.107	4.659	.89	.95	.11	
3	1	2.694	1.000	.03	.01	.02	.02
	2	1.000	1.641	.00	.00	.19	.12
	3	.247	3.304	.46	.00	.26	.30
	4	.059	6.735	.51	.99	.53	.56

a. Dependent Variable: ENERGY

D.4.2 C5402 Data Reduction

D.4.2.1 Internal and External Memory Access

Table D.39: C54 Data Reduction SPSS Output Notes

Output Created	19-FEB-2002 21:57:28	
Comments		
Input	Data	M:\school\thesis\code\c54\c54Scaled.sav
	Filter	<none>
	Weight	<none>
	Split File	<none>
	Rows in Working Data File	465
Missing Value Handling	Definition of Missing	User-defined missing values are treated as missing.
	Cases Used	No missing values for any variable used.
Syntax		REGRESSION /DESCRIPTIVES MEAN STDDEV CORR SIG N /MISSING LISTWISE /STATISTICS COEFF OUTS CI BCOV R ANOVA COLLIN TOL CHANGE ZPP /CRITERIA=PIN(.05) POUT(.10) /NOORIGIN /DEPENDENT energy /METHOD=STEPWISE ΔPA TPD ΔPD ΔDA $TDRI$ ΔDRI $TDWI$ ΔDWI $TDRE$ ΔDRE $TDWE$ ΔDWE TAS ΔAS TAD .

Table D.40: Descriptive Statistics

	Mean	Std. Deviation	N
ENERGY	1.575742	.384501	465
ΔPA	61.41	42.21	465
TPD	356.23	171.13	465
ΔPD	59.52	58.76	465
ΔDA	17.26	11.73	465
TDRI	70.30	113.40	465
ΔDRI	28.08	80.08	465
TDWI	59.46	108.22	465
ΔDWI	24.77	75.77	465
TDRE	39.64	92.71	465
ΔDRE	21.47	71.04	465
TDWE	42.94	95.75	465
ΔDWE	16.52	62.96	465
TAS	1.55	6.50	465
ΔAS	.68	4.55	465
TAD	2.19	8.85	465

Table D.41: Pearson Correlation, part A

	ENERGY	ΔPA	TPD	ΔPD	ΔDA	TDRI	ΔDRI	TDWI
ENERGY	1.000	.385	.461	.332	.521	.058	.177	-.079
ΔPA	.385	1.000	.967	.972	.371	-.003	.077	.095
TPD	.461	.967	1.000	.894	.487	.069	.107	.147
ΔPD	.332	.972	.894	1.000	.300	-.084	.049	.080
ΔDA	.521	.371	.487	.300	1.000	.265	.096	.170
TDRI	.058	-.003	.069	-.084	.265	1.000	.435	-.341
ΔDRI	.177	.077	.107	.049	.096	.435	1.000	-.046
TDWI	-.079	.095	.147	.080	.170	-.341	-.046	1.000
ΔDWI	.029	.070	.098	.102	.218	-.203	-.115	.595
TDRE	.500	.217	.230	.181	.112	-.266	-.150	-.235
ΔDRE	.402	.148	.158	.129	.039	-.188	-.106	-.166
TDWE	.101	.219	.235	.219	.058	-.279	-.158	-.247
ΔDWE	.199	.169	.170	.201	.124	-.163	-.092	-.144
TAS	-.123	-.291	-.407	-.177	-.351	-.114	-.084	-.131
ΔAS	-.072	-.182	-.255	-.112	-.219	-.071	-.052	-.082
TAD	-.133	-.301	-.423	-.185	-.364	-.119	-.087	-.136

Table D.42: Pearson Correlation, part B

	ΔDWI	TDRE	ΔDRE	TDWE	ΔDWE	TAS	ΔAS	TAD
ENERGY	.029	.500	.402	.101	.199	-.123	-.072	-.133
ΔPA	.070	.217	.148	.219	.169	-.291	-.182	-.301
TPD	.098	.230	.158	.235	.170	-.407	-.255	-.423
ΔPD	.102	.181	.129	.219	.201	-.177	-.112	-.185
ΔDA	.218	.112	.039	.058	.124	-.351	-.219	-.364
TDRI	-.203	-.266	-.188	-.279	-.163	-.114	-.071	-.119
ΔDRI	-.115	-.150	-.106	-.158	-.092	-.084	-.052	-.087
TDWI	.595	-.235	-.166	-.247	-.144	-.131	-.082	-.136
ΔDWI	1.000	-.140	-.099	-.147	-.086	-.078	-.049	-.081
TDRE	-.140	1.000	.578	-.192	-.112	-.102	-.064	-.106
ΔDRE	-.099	.578	1.000	-.011	-.079	-.072	-.045	-.075
TDWE	-.147	-.192	-.011	1.000	.585	-.107	-.067	-.111
ΔDWE	-.086	-.112	-.079	.585	1.000	-.063	-.039	-.065
TAS	-.078	-.102	-.072	-.107	-.063	1.000	.576	.981
ΔAS	-.049	-.064	-.045	-.067	-.039	.576	1.000	.597
TAD	-.081	-.106	-.075	-.111	-.065	.981	.597	1.000

Table D.43: Significance: 1-tailed, part A

	ENERGY	ΔPA	TPD	ΔPD	ΔDA	TDRI	ΔDRI	TDWI
ENERGY	.	.000	.000	.000	.000	.104	.000	.045
ΔPA	.000	.	.000	.000	.000	.470	.048	.021
TPD	.000	.000	.	.000	.000	.068	.010	.001
ΔPD	.000	.000	.000	.	.000	.036	.144	.041
ΔDA	.000	.000	.000	.000	.	.000	.019	.000
TDRI	.104	.470	.068	.036	.000	.	.000	.000
ΔDRI	.000	.048	.010	.144	.019	.000	.	.159
TDWI	.045	.021	.001	.041	.000	.000	.159	.
ΔDWI	.268	.066	.017	.014	.000	.000	.007	.000
TDRE	.000	.000	.000	.000	.008	.000	.001	.000
ΔDRE	.000	.001	.000	.003	.202	.000	.011	.000
TDWE	.014	.000	.000	.000	.107	.000	.000	.000
ΔDWE	.000	.000	.000	.000	.004	.000	.023	.001
TAS	.004	.000	.000	.000	.000	.007	.036	.002
ΔAS	.061	.000	.000	.008	.000	.062	.130	.039
TAD	.002	.000	.000	.000	.000	.005	.031	.002

Table D.44: Significance: 1-tailed, part B

	ΔDWI	TDRE	ΔDRE	TDWE	ΔDWE	TAS	ΔAS	TAD
ENERGY	.268	.000	.000	.014	.000	.004	.061	.002
ΔPA	.066	.000	.001	.000	.000	.000	.000	.000
TPD	.017	.000	.000	.000	.000	.000	.000	.000
ΔPD	.014	.000	.003	.000	.000	.000	.008	.000
ΔDA	.000	.008	.202	.107	.004	.000	.000	.000
TDRI	.000	.000	.000	.000	.000	.007	.062	.005
ΔDRI	.007	.001	.011	.000	.023	.036	.130	.031
TDWI	.000	.000	.000	.000	.001	.002	.039	.002
ΔDWI	.	.001	.016	.001	.032	.046	.147	.041
TDRE	.001	.	.000	.000	.008	.014	.085	.011
ΔDRE	.016	.000	.	.404	.043	.060	.166	.054
TDWE	.001	.000	.404	.	.000	.010	.075	.008
ΔDWE	.032	.008	.043	.000	.	.089	.200	.081
TAS	.046	.014	.060	.010	.089	.	.000	.000
ΔAS	.147	.085	.166	.075	.200	.000	.	.000
TAD	.041	.011	.054	.008	.081	.000	.000	.

Table D.45: Number of data points, part A

[illegible]

Table D.46: Number of data points, part B

[illegible]

Table D.47: Variables Entered or Removed^a

Model	Variables		Method
	Entered	Removed	
1	ΔDA	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
2	TDRE	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
3	ΔDRI	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
4	ΔDWE	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
5	ΔDRE	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
6	TAS	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
7	TPD	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
8	ΔPA	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
9	ΔDWI	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
10	TDWE	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
11	TDRI	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
12	TDWI	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
13	.	TPD	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
14	.	ΔPA	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).

a. Dependent Variable: ENERGY

Table D.48: Model Summary

Model	R	R^2	Adjusted R^2	Std. Error of the Estimate	R^2 Change	Change Statistics			Sig. F Change
						F Change	d of f 1	2	
1	.521 ^a	.271	.270	.328622	.271	172.214	1	463	.000
2	.685 ^b	.469	.466	.280886	.198	171.744	1	462	.000
3	.714 ^c	.510	.507	.270063	.041	38.773	1	461	.000
4	.748 ^d	.559	.555	.256378	.049	51.525	1	460	.000
5	.765 ^e	.585	.581	.248970	.026	28.783	1	459	.000
6	.775 ^f	.600	.595	.244759	.015	16.931	1	458	.000
7	.784 ^g	.615	.609	.240507	.015	17.337	1	457	.000
8	.796 ^h	.633	.627	.234826	.019	23.378	1	456	.000
9	.798 ⁱ	.637	.630	.234037	.003	4.080	1	455	.044
10	.800 ^j	.640	.633	.233068	.004	4.792	1	454	.029
11	.804 ^k	.646	.637	.231623	.005	6.682	1	453	.010
12	.809 ^l	.655	.646	.228838	.009	12.092	1	452	.001
13	.809 ^m	.655	.647	.228589	.000	.012	1	454	.913
14	.809 ⁿ	.655	.647	.228348	.000	.044	1	455	.833

- a. Predictors: (Constant), ΔDA
- b. Predictors: (Constant), ΔDA , TDRE
- c. Predictors: (Constant), ΔDA , TDRE, ΔDRI
- d. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE
- e. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE
- f. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS
- g. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD
- h. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD, ΔPA
- i. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD, ΔPA , ΔDWI

- j. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD, ΔPA , ΔDWI , TDWE
- k. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD, ΔPA , ΔDWI , TDWE, TDRI
- l. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD, ΔPA , ΔDWI , TDWE, TDRI, TDWI
- m. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, ΔPA , ΔDWI , TDWE, TDRI, TDWI
- n. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, ΔDWI , TDWE, TDRI, TDWI

Table D.49: ANOVA^f

Model		Σ of Squares	df	Mean Square	F	Sig.
1	Model	18.598	1	18.598	172.214	.000 ^a
	Residual	50.000	463	.108		
	Total	68.598	464			
2	Model	32.148	2	16.074	203.733	.000 ^b
	Residual	36.450	462	7.890E-02		
	Total	68.598	464			
3	Model	34.976	3	11.659	159.851	.000 ^c
	Residual	33.622	461	7.293E-02		
	Total	68.598	464			
4	Model	38.362	4	9.591	145.909	.000 ^d
	Residual	30.236	460	6.573E-02		
	Total	68.598	464			
5	Model	40.147	5	8.029	129.534	.000 ^e
	Residual	28.452	459	6.199E-02		
	Total	68.598	464			
6	Model	41.161	6	6.860	114.514	.000 ^f
	Residual	27.437	458	5.991E-02		
	Total	68.598	464			
7	Model	42.164	7	6.023	104.132	.000 ^g
	Residual	26.434	457	5.784E-02		
	Total	68.598	464			
8	Model	43.453	8	5.432	98.500	.000 ^h
	Residual	25.145	456	5.514E-02		
	Total	68.598	464			
9	Model	43.676	9	4.853	88.600	.000 ⁱ
	Residual	24.922	455	5.477E-02		
	Total	68.598	464			
10	Model	43.937	10	4.394	80.884	.000 ^j
	Residual	24.662	454	5.432E-02		
	Total	68.598	464			
	Model	44.295	11	4.027	75.058	.000 ^k

11	Residual	24.303	453	5.365E-02		
	Total	68.598	464			
12	Model	44.928	12	3.744	71.496	.000 ^l
	Residual	23.670	452	5.237E-02		
	Total	68.598	464			
13	Model	44.928	11	4.084	78.165	.000 ^m
	Residual	23.671	453	5.225E-02		
	Total	68.598	464			
14	Model	44.925	10	4.493	86.158	.000 ⁿ
	Residual	23.673	454	5.214E-02		
	Total	68.598	464			

- a. Predictors: (Constant), ΔDA
- b. Predictors: (Constant), ΔDA , TDRE
- c. Predictors: (Constant), ΔDA , TDRE, ΔDRI
- d. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE
- e. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE
- f. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS
- g. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD
- h. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD, ΔPA
- i. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD, ΔPA , ΔDWI
- j. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD, ΔPA , ΔDWI , TDWE
- k. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD, ΔPA , ΔDWI , TDWE, TDRI
- l. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD, ΔPA , ΔDWI , TDWE, TDRI, TDWI

- m. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, ΔPA , ΔDWI , TDWE, TDRI, TDWI
- n. Predictors: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, ΔDWI , TDWE, TDRI, TDWI
- o. Dependent Variable: ENERGY

Table D.50: Coefficients^a

Model		Unstandardized Coefficients		Standardized Coefficients β	t	Sig.
		$\hat{B}$	Std. Error			
1	(Constant)	1.281	.027		47.190	.000
	ΔDA	1.707E-02	.001	.521	13.123	.000
2	(Constant)	1.236	.023		52.688	.000
	ΔDA	1.543E-02	.001	.471	13.787	.000
	TDRE	1.855E-03	.000	.447	13.105	.000
3	(Constant)	1.216	.023		53.371	.000
	ΔDA	1.465E-02	.001	.447	13.529	.000
	TDRE	1.995E-03	.000	.481	14.462	.000
	ΔDRI	9.926E-04	.000	.207	6.227	.000
4	(Constant)	1.203	.022		55.443	.000
	ΔDA	1.351E-02	.001	.412	12.983	.000
	TDRE	2.136E-03	.000	.515	16.129	.000
	ΔDRI	1.134E-03	.000	.236	7.430	.000
	ΔDWE	1.391E-03	.000	.228	7.178	.000
5	(Constant)	1.196	.021		56.612	.000
	ΔDA	1.365E-02	.001	.416	13.507	.000
	TDRE	1.663E-03	.000	.401	10.665	.000
	ΔDRI	1.152E-03	.000	.240	7.770	.000
	ΔDWE	1.407E-03	.000	.230	7.478	.000
	ΔDRE	1.071E-03	.000	.198	5.365	.000
6	(Constant)	1.156	.023		50.405	.000
	ΔDA	1.508E-02	.001	.460	14.329	.000
	TDRE	1.694E-03	.000	.409	11.042	.000
	ΔDRI	1.194E-03	.000	.249	8.174	.000
	ΔDWE	1.437E-03	.000	.235	7.761	.000
	ΔDRE	1.096E-03	.000	.202	5.582	.000
	TAS	7.723E-03	.002	.131	4.115	.000
	(Constant)	1.071	.030		35.322	.000
	ΔDA	1.334E-02	.001	.407	11.955	.000
	TDRE	1.591E-03	.000	.384	10.415	.000

7	ΔDRI	1.125E-03	.000	.234	7.782	.000
	ΔDWE	1.305E-03	.000	.214	7.071	.000
	ΔDRE	1.053E-03	.000	.195	5.451	.000
	TAS	9.907E-03	.002	.168	5.167	.000
	TPD	3.378E-04	.000	.150	4.164	.000
8	(Constant)	.913	.044		20.751	.000
	ΔDA	1.092E-02	.001	.333	9.107	.000
	TDRE	1.585E-03	.000	.382	10.627	.000
	ΔDRI	1.066E-03	.000	.222	7.524	.000
	ΔDWE	1.326E-03	.000	.217	7.357	.000
	ΔDRE	1.019E-03	.000	.188	5.402	.000
	TAS	1.394E-02	.002	.236	6.802	.000
	TPD	1.930E-03	.000	.859	5.698	.000
	ΔPA	-6.056E-03	.001	-.665	-4.835	.000
9	(Constant)	.915	.044		20.849	.000
	ΔDA	1.046E-02	.001	.319	8.606	.000
	TDRE	1.641E-03	.000	.396	10.852	.000
	ΔDRI	1.125E-03	.000	.234	7.803	.000
	ΔDWE	1.392E-03	.000	.228	7.623	.000
	ΔDRE	1.030E-03	.000	.190	5.474	.000
	TAS	1.394E-02	.002	.236	6.823	.000
	TPD	1.900E-03	.000	.846	5.622	.000
	ΔPA	-5.984E-03	.001	-.657	-4.792	.000
	ΔDWI	3.113E-04	.000	.061	2.020	.044
10	(Constant)	.921	.044		21.032	.000
	ΔDA	1.076E-02	.001	.328	8.830	.000
	TDRE	1.757E-03	.000	.424	11.007	.000
	ΔDRI	1.208E-03	.000	.252	8.134	.000
	ΔDWE	1.138E-03	.000	.186	5.280	.000
	ΔDRE	9.689E-04	.000	.179	5.114	.000
	TAS	1.392E-02	.002	.235	6.843	.000
	TPD	1.753E-03	.000	.780	5.108	.000
	ΔPA	-5.606E-03	.001	-.615	-4.465	.000
	ΔDWI	3.884E-04	.000	.077	2.467	.014

	TDWE	3.408E-04	.000	.085	2.189	.029
11	(Constant)	.930	.044		21.303	.000
	ΔDA	9.922E-03	.001	.303	7.921	.000
	TDRE	1.965E-03	.000	.474	11.046	.000
	ΔDRI	1.115E-03	.000	.232	7.336	.000
	ΔDWE	1.158E-03	.000	.190	5.399	.000
	ΔDRE	9.638E-04	.000	.178	5.118	.000
	TAS	1.368E-02	.002	.231	6.761	.000
	TPD	1.480E-03	.000	.659	4.147	.000
	ΔPA	-4.672E-03	.001	-.513	-3.596	.000
	ΔDWI	6.116E-04	.000	.121	3.422	.001
	TDWE	5.347E-04	.000	.133	3.110	.002
	TDRI	3.660E-04	.000	.108	2.585	.010
12	(Constant)	.985	.046		21.437	.000
	ΔDA	8.658E-03	.001	.264	6.712	.000
	TDRE	2.954E-03	.000	.712	8.836	.000
	ΔDRI	1.076E-03	.000	.224	7.149	.000
	ΔDWE	1.194E-03	.000	.195	5.629	.000
	ΔDRE	9.560E-04	.000	.177	5.138	.000
	TAS	1.292E-02	.002	.219	6.424	.000
	TPD	-6.177E-05	.001	-.027	-.109	.913
	ΔPA	2.709E-04	.002	.030	.141	.888
	ΔDWI	5.046E-04	.000	.099	2.816	.005
	TDWE	1.499E-03	.000	.373	4.610	.000
	TDRI	1.366E-03	.000	.403	4.272	.000
	TDWI	1.110E-03	.000	.312	3.477	.001
13	(Constant)	.981	.031		31.785	.000
	ΔDA	8.659E-03	.001	.264	6.720	.000
	TDRE	2.927E-03	.000	.706	13.059	.000
	ΔDRI	1.076E-03	.000	.224	7.158	.000
	ΔDWE	1.194E-03	.000	.196	5.636	.000
	ΔDRE	9.560E-04	.000	.177	5.144	.000
	TAS	1.300E-02	.002	.220	6.926	.000
	ΔPA	6.478E-05	.000	.007	.210	.833

	ΔDWI	5.047E-04	.000	.099	2.819	.005
	TDWE	1.472E-03	.000	.366	7.022	.000
	TDRI	1.338E-03	.000	.395	6.754	.000
	TDWI	1.083E-03	.000	.305	5.455	.000
	(Constant)	.982	.031		32.123	.000
	ΔDA	8.669E-03	.001	.264	6.740	.000
	TDRE	2.944E-03	.000	.710	14.068	.000
	ΔDRI	1.079E-03	.000	.225	7.216	.000
	ΔDWE	1.196E-03	.000	.196	5.662	.000
14	ΔDRE	9.560E-04	.000	.177	5.149	.000
	TAS	1.299E-02	.002	.220	6.930	.000
	ΔDWI	5.056E-04	.000	.100	2.828	.005
	TDWE	1.487E-03	.000	.370	7.605	.000
	TDRI	1.349E-03	.000	.398	7.039	.000
	TDWI	1.095E-03	.000	.308	5.795	.000

a. Dependent Variable: ENERGY

Table D.51: Coefficients^a

Model		95% Confidence Interval for $\hat{B}$		Correlations			Colinearity Statistics	
		Lower Bound	Upper Bound	Zero Order	Partial	Part	Tolerance	VIF
1	(Constant)	1.228	1.334					
	ΔDA	.015	.020	.521	.521	.521	1.000	1.000
2	(Constant)	1.190	1.282					
	ΔDA	.013	.018	.521	.540	.468	.987	1.013
	TDRE	.002	.002	.500	.521	.444	.987	1.013
3	(Constant)	1.171	1.261					
	ΔDA	.013	.017	.521	.533	.441	.974	1.026
	TDRE	.002	.002	.500	.559	.472	.961	1.040
	ΔDRI	.001	.001	.177	.279	.203	.965	1.037
4	(Constant)	1.160	1.246					
	ΔDA	.011	.016	.521	.518	.402	.952	1.051
	TDRE	.002	.002	.500	.601	.499	.940	1.064
	ΔDRI	.001	.001	.177	.327	.230	.948	1.054
	ΔDWE	.001	.002	.199	.317	.222	.952	1.050
5	(Constant)	1.154	1.237					
	ΔDA	.012	.016	.521	.533	.406	.951	1.052
	TDRE	.001	.002	.500	.446	.321	.640	1.564
	ΔDRI	.001	.001	.177	.341	.234	.948	1.055
	ΔDWE	.001	.002	.199	.330	.225	.952	1.050
	ΔDRE	.001	.001	.402	.243	.161	.665	1.505
6	(Constant)	1.110	1.201					
	ΔDA	.013	.017	.521	.556	.423	.848	1.179
	TDRE	.001	.002	.500	.459	.326	.638	1.567
	ΔDRI	.001	.001	.177	.357	.242	.943	1.060
	ΔDWE	.001	.002	.199	.341	.229	.951	1.052
	ΔDRE	.001	.001	.402	.252	.165	.664	1.506
	TAS	.004	.011	-.123	.189	.122	.867	1.154
	(Constant)	1.011	1.131					

7	ΔDA	.011	.016	.521	.488	.347	.729	1.372
	TDRE	.001	.002	.500	.438	.302	.621	1.610
	ΔDRI	.001	.001	.177	.342	.226	.931	1.075
	ΔDWE	.001	.002	.199	.314	.205	.923	1.083
	ΔDRE	.001	.001	.402	.247	.158	.662	1.511
	TAS	.006	.014	-.123	.235	.150	.802	1.247
	TPD	.000	.000	.461	.191	.121	.647	1.546
8	(Constant)	.827	1.000					
	ΔDA	.009	.013	.521	.392	.258	.602	1.662
	TDRE	.001	.002	.500	.446	.301	.621	1.610
	ΔDRI	.001	.001	.177	.332	.213	.924	1.083
	ΔDWE	.001	.002	.199	.326	.209	.922	1.084
	ΔDRE	.001	.001	.402	.245	.153	.661	1.513
	TAS	.010	.018	-.123	.303	.193	.669	1.495
	TPD	.001	.003	.461	.258	.162	.035	28.269
	ΔPA	-.009	-.004	.385	-.221	-.137	.043	23.516
9	(Constant)	.829	1.001					
	ΔDA	.008	.013	.521	.374	.243	.581	1.721
	TDRE	.001	.002	.500	.453	.307	.600	1.666
	ΔDRI	.001	.001	.177	.344	.220	.885	1.130
	ΔDWE	.001	.002	.199	.337	.215	.893	1.120
	ΔDRE	.001	.001	.402	.249	.155	.661	1.514
	TAS	.010	.018	-.123	.305	.193	.669	1.495
	TPD	.001	.003	.461	.255	.159	.035	28.324
	ΔPA	-.008	-.004	.385	-.219	-.135	.042	23.535
	ΔDWI	.000	.001	.029	.094	.057	.866	1.155
10	(Constant)	.835	1.007					
	ΔDA	.008	.013	.521	.383	.248	.574	1.742
	TDRE	.001	.002	.500	.459	.310	.535	1.871
	ΔDRI	.001	.001	.177	.357	.229	.828	1.208
	ΔDWE	.001	.002	.199	.241	.149	.635	1.574
	ΔDRE	.001	.001	.402	.233	.144	.646	1.547
	TAS	.010	.018	-.123	.306	.193	.669	1.495
	TPD	.001	.002	.461	.233	.144	.034	29.451

	ΔPA	-.008	-.003	.385	-.205	-.126	.042	23.988
	ΔDWI	.000	.001	.029	.115	.069	.822	1.216
	TDWE	.000	.001	.101	.102	.062	.527	1.898
11	(Constant)	.844	1.016					
	ΔDA	.007	.012	.521	.349	.222	.536	1.866
	TDRE	.002	.002	.500	.461	.309	.425	2.353
	ΔDRI	.001	.001	.177	.326	.205	.781	1.280
	ΔDWE	.001	.002	.199	.246	.151	.634	1.576
	ΔDRE	.001	.001	.402	.234	.143	.646	1.548
	TAS	.010	.018	-.123	.303	.189	.668	1.498
	TPD	.001	.002	.461	.191	.116	.031	32.271
	ΔPA	-.007	-.002	.385	-.167	-.101	.038	26.002
	ΔDWI	.000	.001	.029	.159	.096	.631	1.586
	TDWE	.000	.001	.101	.145	.087	.427	2.345
	TDRI	.000	.001	.058	.121	.072	.448	2.230
12	(Constant)	.895	1.076					
	ΔDA	.006	.011	.521	.301	.185	.493	2.027
	TDRE	.002	.004	.500	.384	.244	.117	8.513
	ΔDRI	.001	.001	.177	.319	.198	.777	1.287
	ΔDWE	.001	.002	.199	.256	.156	.633	1.580
	ΔDRE	.001	.001	.402	.235	.142	.646	1.548
	TAS	.009	.017	-.123	.289	.177	.660	1.516
	TPD	-.001	.001	.461	-.005	-.003	.012	83.289
	ΔPA	-.003	.004	.385	.007	.004	.017	57.894
	ΔDWI	.000	.001	.029	.131	.078	.612	1.634
	TDWE	.001	.002	.101	.212	.127	.116	8.589
	TDRI	.001	.002	.058	.197	.118	.086	11.646
	TDWI	.000	.002	-.079	.161	.096	.095	10.575
13	(Constant)	.921	1.042					
	ΔDA	.006	.011	.521	.301	.185	.493	2.027
	TDRE	.002	.003	.500	.523	.360	.261	3.835
	ΔDRI	.001	.001	.177	.319	.198	.777	1.287
	ΔDWE	.001	.002	.199	.256	.156	.633	1.580
	ΔDRE	.001	.001	.402	.235	.142	.646	1.548

	TAS	.009	.017	-.123	.309	.191	.756	1.323
	ΔPA	-.001	.001	.385	.010	.006	.667	1.499
	ΔDWI	.000	.001	.029	.131	.078	.612	1.634
	TDWE	.001	.002	.101	.313	.194	.280	3.576
	TDRI	.001	.002	.058	.302	.186	.223	4.483
	TDWI	.001	.001	-.079	.248	.151	.244	4.097
14	(Constant)	.922	1.042					
	ΔDA	.006	.011	.521	.302	.186	.494	2.024
	TDRE	.003	.003	.500	.551	.388	.299	3.349
	ΔDRI	.001	.001	.177	.321	.199	.784	1.276
	ΔDWE	.001	.002	.199	.257	.156	.635	1.575
	ΔDRE	.001	.001	.402	.235	.142	.646	1.548
	TAS	.009	.017	-.123	.309	.191	.757	1.322
	ΔDWI	.000	.001	.029	.132	.078	.612	1.633
	TDWE	.001	.002	.101	.336	.210	.320	3.121
	TDRI	.001	.002	.058	.314	.194	.238	4.201
	TDWI	.001	.001	-.079	.262	.160	.269	3.724

a. Dependent Variable: ENERGY

Table D.52: Excluded Variables^o

Model		β in	t	Sig.	Partial Correlation	Colinearity Statistics		
						Tolerance	VIF	Minimum Tolerance
1	ΔPA	.223 ^a	5.376	.000	.243	.863	1.159	.863
	TPD	.271 ^a	6.212	.000	.278	.763	1.311	.763
	ΔPD	.193 ^a	4.748	.000	.216	.910	1.099	.910
	TDRI	-.086 ^a	-2.093	.037	-.097	.930	1.076	.930
	ΔDRI	.129 ^a	3.259	.001	.150	.991	1.009	.991
	TDWI	-.172 ^a	-4.361	.000	-.199	.971	1.030	.971
	ΔDWI	-.089 ^a	-2.198	.028	-.102	.952	1.050	.952
	TDRE	.447 ^a	13.105	.000	.521	.987	1.013	.987
	ΔDRE	.382 ^a	10.757	.000	.448	.998	1.002	.998
	TDWE	.072 ^a	1.805	.072	.084	.997	1.003	.997
	ΔDWE	.137 ^a	3.458	.001	.159	.985	1.016	.985
	TAS	.069 ^a	1.623	.105	.075	.877	1.141	.877
	ΔAS	.045 ^a	1.095	.274	.051	.952	1.051	.952
	TAD	.066 ^a	1.552	.121	.072	.867	1.153	.867
2	ΔPA	.137 ^b	3.740	.000	.172	.832	1.203	.832
	TPD	.176 ^b	4.529	.000	.206	.731	1.367	.731
	ΔPD	.124 ^b	3.477	.001	.160	.888	1.126	.888
	TDRI	.062 ^b	1.686	.093	.078	.841	1.189	.841
	ΔDRI	.207 ^b	6.227	.000	.279	.965	1.037	.961
	TDWI	-.059 ^b	-1.660	.098	-.077	.905	1.104	.905
	ΔDWI	-.012 ^b	-.342	.733	-.016	.925	1.081	.925
	ΔDRE	.188 ^b	4.623	.000	.211	.665	1.504	.658
	TDWE	.167 ^b	4.950	.000	.225	.957	1.045	.948
	ΔDWE	.197 ^b	5.933	.000	.266	.968	1.033	.968
	TAS	.101 ^b	2.805	.005	.130	.873	1.146	.871
	ΔAS	.063 ^b	1.815	.070	.084	.950	1.052	.942
	TAD	.100 ^b	2.760	.006	.127	.863	1.159	.862
	ΔPA	.120 ^c	3.396	.001	.156	.826	1.210	.826
	TPD	.152 ^c	4.042	.000	.185	.723	1.383	.723
	ΔPD	.114 ^c	3.314	.001	.153	.886	1.129	.886
	TDRI	-.032 ^c	-.820	.413	-.038	.706	1.417	.706

3	TDWI	-.036 ^c	-1.035	.301	-.048	.894	1.118	.889
	ΔDWI	.025 ^c	.727	.468	.034	.898	1.114	.898
	ΔDRE	.193 ^c	4.956	.000	.225	.665	1.504	.648
	TDWE	.219 ^c	6.724	.000	.299	.917	1.090	.909
	ΔDWE	.228 ^c	7.178	.000	.317	.952	1.050	.940
	TAS	.116 ^c	3.347	.001	.154	.869	1.151	.864
	ΔAS	.071 ^c	2.138	.033	.099	.949	1.054	.932
	TAD	.116 ^c	3.319	.001	.153	.859	1.164	.856
4	ΔPA	.080 ^d	2.330	.020	.108	.801	1.248	.801
	TPD	.110 ^d	3.015	.003	.139	.702	1.425	.702
	ΔPD	.068 ^d	2.033	.043	.094	.849	1.178	.849
	TDRI	.030 ^d	.794	.428	.037	.669	1.495	.669
	TDWI	.019 ^d	.571	.569	.027	.847	1.180	.847
	ΔDWI	.066 ^d	2.005	.046	.093	.872	1.146	.872
	ΔDRE	.198 ^d	5.365	.000	.243	.665	1.505	.640
	TDWE	.129 ^d	3.327	.001	.153	.625	1.599	.625
	TAS	.125 ^d	3.820	.000	.176	.868	1.153	.849
	ΔAS	.077 ^d	2.420	.016	.112	.948	1.054	.912
	TAD	.125 ^d	3.802	.000	.175	.857	1.166	.841
5	ΔPA	.072 ^e	2.142	.033	.100	.799	1.251	.623
	TPD	.099 ^e	2.783	.006	.129	.699	1.431	.621
	ΔPD	.060 ^e	1.846	.066	.086	.847	1.181	.628
	TDRI	.037 ^e	1.002	.317	.047	.668	1.497	.604
	TDWI	.026 ^e	.801	.423	.037	.846	1.182	.603
	ΔDWI	.070 ^e	2.186	.029	.102	.872	1.147	.621
	TDWE	.098 ^e	2.558	.011	.119	.608	1.643	.602
	TAS	.131 ^e	4.115	.000	.189	.867	1.154	.638
	ΔAS	.080 ^e	2.595	.010	.120	.948	1.055	.639
	TAD	.131 ^e	4.104	.000	.188	.857	1.167	.638
6	ΔPA	.097 ^f	2.917	.004	.135	.777	1.286	.623
	TPD	.150 ^f	4.164	.000	.191	.647	1.546	.621
	ΔPD	.068 ^f	2.130	.034	.099	.844	1.185	.627
	TDRI	.042 ^f	1.155	.249	.054	.667	1.498	.602
	TDWI	.043 ^f	1.320	.188	.062	.834	1.200	.599
	ΔDWI	.074 ^f	2.361	.019	.110	.871	1.148	.619
	TDWE	.119 ^f	3.141	.002	.145	.599	1.668	.598

	ΔAS	.017 ^f	.461	.645	.022	.668	1.497	.611
	TAD	.059 ^f	.380	.704	.018	3.651E-02	27.386	3.651E-02
7	ΔPA	-.665 ^g	-4.835	.000	-.221	4.252E-02	23.516	3.538E-02
	ΔPD	-.311 ^g	-4.087	.000	-.188	.141	7.100	.108
	TDRI	.046 ^g	1.288	.198	.060	.667	1.500	.588
	TDWI	.020 ^g	.631	.528	.030	.809	1.236	.575
	ΔDWI	.065 ^g	2.107	.036	.098	.867	1.154	.601
	TDWE	.086 ^g	2.228	.026	.104	.564	1.774	.564
	ΔAS	.019 ^g	.546	.585	.026	.668	1.497	.580
	TAD	.116 ^g	.762	.446	.036	3.622E-02	27.608	3.622E-02
8	ΔPD	.186 ^h	.937	.349	.044	2.037E-02	49.096	6.150E-03
	TDRI	.017 ^h	.483	.629	.023	.647	1.546	3.443E-02
	TDWI	-.006 ^h	-.172	.864	-.008	.786	1.273	3.390E-02
	ΔDWI	.061 ^h	2.020	.044	.094	.866	1.155	3.531E-02
	TDWE	.063 ^h	1.670	.096	.078	.555	1.803	3.421E-02
	ΔAS	.024 ^h	.703	.482	.033	.667	1.499	3.534E-02
	TAD	.227 ^h	1.508	.132	.071	3.544E-02	28.214	3.443E-02
9	ΔPD	.037 ⁱ	.174	.862	.008	1.733E-02	57.695	5.306E-03
	TDRI	.051 ⁱ	1.352	.177	.063	.554	1.805	3.399E-02
	TDWI	-.057 ⁱ	-1.501	.134	-.070	.557	1.795	3.369E-02
	TDWE	.085 ⁱ	2.189	.029	.102	.527	1.898	3.395E-02
	ΔAS	.024 ⁱ	.706	.481	.033	.667	1.499	3.527E-02
	TAD	.227 ⁱ	1.513	.131	.071	3.544E-02	28.214	3.437E-02
10	ΔPD	-.010 ^j	-.048	.961	-.002	1.715E-02	58.299	5.197E-03
	TDRI	.108 ^j	2.585	.010	.121	.448	2.230	3.099E-02
	TDWI	-.033 ^j	-.816	.415	-.038	.494	2.025	3.119E-02
	ΔAS	.024 ^j	.708	.479	.033	.667	1.499	3.392E-02
	TAD	.226 ^j	1.516	.130	.071	3.544E-02	28.214	3.308E-02
11	ΔPD	.056 ^k	.260	.795	.012	1.691E-02	59.134	5.197E-03
	TDWI	.312 ^k	3.477	.001	.161	9.456E-02	10.575	1.201E-02
	ΔAS	.024 ^k	.704	.482	.033	.667	1.499	3.096E-02
	TAD	.220 ^k	1.483	.139	.070	3.543E-02	28.222	3.021E-02
12	ΔPD	.112 ^l	.526	.599	.025	1.682E-02	59.469	4.663E-03
	ΔAS	.023 ^l	.684	.494	.032	.667	1.499	1.200E-02
	TAD	.200 ^l	1.362	.174	.064	3.538E-02	28.267	1.180E-02
	TPD	-.027 ^m	-.109	.913	-.005	1.201E-02	83.289	1.201E-02

13	ΔPD	.094 ^m	.510	.610	.024	2.242E-02	44.612	1.935E-02
	ΔAS	.023 ^m	.688	.492	.032	.668	1.498	.223
	TAD	.199 ^m	1.367	.172	.064	3.598E-02	27.791	3.598E-02
14	ΔPA	.007 ⁿ	.210	.833	.010	.667	1.499	.223
	TPD	.008 ⁿ	.190	.849	.009	.464	2.157	.196
	ΔPD	.009 ⁿ	.294	.769	.014	.773	1.294	.235
	ΔAS	.023 ⁿ	.688	.492	.032	.668	1.498	.238
	TAD	.198 ⁿ	1.366	.173	.064	3.599E-02	27.788	3.599E-02

- a. Predictors in the Model: (Constant), ΔDA
- b. Predictors in the Model: (Constant), ΔDA , TDRE
- c. Predictors in the Model: (Constant), ΔDA , TDRE, ΔDRI
- d. Predictors in the Model: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE
- e. Predictors in the Model: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE
- f. Predictors in the Model: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS
- g. Predictors in the Model: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD
- h. Predictors in the Model: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD, ΔPA
- i. Predictors in the Model: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD, ΔPA , ΔDWI
- j. Predictors in the Model: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD, ΔPA , ΔDWI , TDWE
- k. Predictors in the Model: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD, ΔPA , ΔDWI , TDWE, TDRI
- l. Predictors in the Model: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, TPD, ΔPA , ΔDWI , TDWE, TDRI, TDWI

- m. Predictors in the Model: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, ΔPA , ΔDWI , TDWE, TDRI, TDWI
- n. Predictors in the Model: (Constant), ΔDA , TDRE, ΔDRI , ΔDWE , ΔDRE , TAS, ΔDWI , TDWE, TDRI, TDWI
- o. Dependent Variable: ENERGY

Table D.53: Coefficient Correlations, part A^a

Model			ΔDA	TDRE	ΔDRI	ΔDWE
1	Correlations	ΔDA	1.000			
	Covariances	ΔDA	1.693E-06			
2	Correlations	ΔDA	1.000	-.112		
		TDRE	-.112	1.000		
	Covariances	ΔDA	1.253E-06	-1.777E-08		
		TDRE	-1.777E-08	2.004E-08		
3	Correlations	ΔDA	1.000	-.129	-.115	
		TDRE	-.129	1.000	.163	
		ΔDRI	-.115	.163	1.000	
	Covariances	ΔDA	1.173E-06	-1.922E-08	-1.985E-08	
		TDRE	-1.922E-08	1.903E-08	3.579E-09	
		ΔDRI	-1.985E-08	3.579E-09	2.541E-08	
4	Correlations	ΔDA	1.000	-.148	-.132	-.153
		TDRE	-.148	1.000	.179	.148
		ΔDRI	-.132	.179	1.000	.129
		ΔDWE	-.153	.148	.129	1.000
	Covariances	ΔDA	1.083E-06	-2.044E-08	-2.103E-08	-3.088E-08
		TDRE	-2.044E-08	1.753E-08	3.612E-09	3.798E-09
		ΔDRI	-2.103E-08	3.612E-09	2.329E-08	3.814E-09
		ΔDWE	-3.088E-08	3.798E-09	3.814E-09	3.753E-08
5	Correlations	ΔDA	1.000	-.137	-.132	-.153
		TDRE	-.137	1.000	.135	.113
		ΔDRI	-.132	.135	1.000	.129
		ΔDWE	-.153	.113	.129	1.000
		ΔDRE	.027	-.565	.022	.016
	Covariances	ΔDA	1.022E-06	-2.165E-08	-1.974E-08	-2.904E-08
		TDRE	-2.165E-08	2.430E-08	3.114E-09	3.311E-09
		ΔDRI	-1.974E-08	3.114E-09	2.197E-08	3.607E-09
		ΔDWE	-2.904E-08	3.311E-09	3.607E-09	3.540E-08
		ΔDRE	5.376E-09	-1.759E-08	6.617E-10	6.128E-10
	Correlations	ΔDA	1.000	-.113	-.101	-.131
		TDRE	-.113	1.000	.138	.115
		ΔDRI	-.101	.138	1.000	.132

6		ΔDWE	-.131	.115	.132	1.000
		ΔDRE	.035	-.563	.024	.018
		TAS	.329	.050	.071	.039
	Covariances	ΔDA	1.107E-06	-1.826E-08	-1.551E-08	-2.558E-08
		TDRE	-1.826E-08	2.355E-08	3.089E-09	3.255E-09
		ΔDRI	-1.551E-08	3.089E-09	2.134E-08	3.560E-09
		ΔDWE	-2.558E-08	3.255E-09	3.560E-09	3.426E-08
		ΔDRE	7.303E-09	-1.696E-08	7.024E-10	6.359E-10
		TAS	6.492E-07	1.448E-08	1.937E-08	1.346E-08
7	Correlations	ΔDA	1.000	-.043	-.050	-.056
		TDRE	-.043	1.000	.154	.139
		ΔDRI	-.050	.154	1.000	.149
		ΔDWE	-.056	.139	.149	1.000
		ΔDRE	.053	-.546	.030	.026
		TAS	.190	.003	.036	-.010
		TPD	-.375	-.162	-.116	-.171
	Covariances	ΔDA	1.244E-06	-7.283E-09	-7.985E-09	-1.152E-08
		TDRE	-7.283E-09	2.335E-08	3.395E-09	3.922E-09
		ΔDRI	-7.985E-09	3.395E-09	2.089E-08	3.964E-09
		ΔDWE	-1.152E-08	3.922E-09	3.964E-09	3.408E-08
		ΔDRE	1.136E-08	-1.612E-08	8.502E-10	9.386E-10
		TAS	4.074E-07	1.009E-09	9.945E-09	-3.523E-09
		TPD	-3.395E-08	-2.006E-09	-1.355E-09	-2.556E-09
8	Correlations	ΔDA	1.000	-.035	-.009	-.061
		TDRE	-.035	1.000	.154	.139
		ΔDRI	-.009	.154	1.000	.146
		ΔDWE	-.061	.139	.146	1.000
		ΔDRE	.063	-.545	.033	.025
		TAS	-.012	.000	-.002	.001
		TPD	-.486	-.046	-.111	-.016
		ΔPA	.418	.008	.086	-.024
	Covariances	ΔDA	1.437E-06	-6.327E-09	-1.499E-09	-1.315E-08
		TDRE	-6.327E-09	2.226E-08	3.252E-09	3.733E-09
		ΔDRI	-1.499E-09	3.252E-09	2.006E-08	3.725E-09
		ΔDWE	-1.315E-08	3.733E-09	3.725E-09	3.250E-08
		ΔDRE	1.430E-08	-1.536E-08	8.951E-10	8.647E-10

		TAS	-2.921E-08	-6.549E-11	-7.111E-10	2.628E-10
		TPD	-1.972E-07	-2.318E-09	-5.314E-09	-1.007E-09
		ΔPA	6.268E-07	1.543E-09	1.530E-08	-5.437E-09
9	Correlations	ΔDA	1.000	-.068	-.046	-.092
		TDRE	-.068	1.000	.186	.167
		ΔDRI	-.046	.186	1.000	.177
		ΔDWE	-.092	.167	.177	1.000
		ΔDRE	.057	-.531	.038	.030
		TAS	-.012	.000	-.003	.001
		TPD	-.469	-.053	-.117	-.024
		ΔPA	.405	.013	.090	-.019
		ΔDWI	-.185	.183	.205	.178
	Covariances	ΔDA	1.478E-06	-1.252E-08	-8.144E-09	-2.040E-08
		TDRE	-1.252E-08	2.288E-08	4.048E-09	4.610E-09
		ΔDRI	-8.144E-09	4.048E-09	2.080E-08	4.663E-09
		ΔDWE	-2.040E-08	4.610E-09	4.663E-09	3.335E-08
		ΔDRE	1.302E-08	-1.511E-08	1.044E-09	1.030E-09
		TAS	-2.864E-08	-1.104E-10	-7.548E-10	2.077E-10
		TPD	-1.925E-07	-2.717E-09	-5.721E-09	-1.488E-09
		ΔPA	6.146E-07	2.514E-09	1.624E-08	-4.245E-09
		ΔDWI	-3.469E-08	4.265E-09	4.555E-09	5.018E-09
10	Correlations	ΔDA	1.000	-.027	-.017	-.136
		TDRE	-.027	1.000	.253	-.045
		ΔDRI	-.017	.253	1.000	.008
		ΔDWE	-.136	-.045	.008	1.000
		ΔDRE	.040	-.544	-.001	.104
		TAS	-.012	-.002	-.003	.003
		TPD	-.478	-.114	-.161	.085
		ΔPA	.414	.058	.121	-.089
		ΔDWI	-.155	.242	.250	.026
	Covariances	TDWE	.111	.331	.254	-.537
		ΔDA	1.484E-06	-5.292E-09	-2.995E-09	-3.584E-08
		TDRE	-5.292E-09	2.548E-08	6.008E-09	-1.551E-09
		ΔDRI	-2.995E-09	6.008E-09	2.205E-08	2.548E-10
		ΔDWE	-3.584E-08	-1.551E-09	2.548E-10	4.649E-08
		ΔDRE	9.147E-09	-1.646E-08	-1.846E-11	4.259E-09

		TAS	-2.948E-08	-5.322E-10	-1.050E-09	1.133E-09
		TPD	-1.999E-07	-6.242E-09	-8.205E-09	6.302E-09
		ΔPA	6.328E-07	1.162E-08	2.262E-08	-2.421E-08
		ΔDWI	-2.966E-08	6.092E-09	5.847E-09	8.940E-10
		TDWE	2.098E-08	8.226E-09	5.872E-09	-1.804E-08
11	Correlations	ΔDA	1.000	-.140	.046	-.141
		TDRE	-.140	1.000	.112	-.025
		ΔDRI	.046	.112	1.000	.000
		ΔDWE	-.141	-.025	.000	1.000
		ΔDRE	.041	-.490	.002	.104
		TAS	.000	-.022	.007	.001
		TPD	-.365	-.231	-.079	.071
		ΔPA	.312	.176	.047	-.076
		ΔDWI	-.255	.408	.098	.040
		TDWE	-.016	.463	.118	-.468
		TDRI	-.257	.453	-.238	.034
	Covariances	ΔDA	1.569E-06	-3.120E-08	8.681E-09	-3.777E-08
		TDRE	-3.120E-08	3.165E-08	3.027E-09	-9.383E-10
		ΔDRI	8.681E-09	3.027E-09	2.308E-08	-1.409E-11
		ΔDWE	-3.777E-08	-9.383E-10	-1.409E-11	4.597E-08
		ΔDRE	9.676E-09	-1.642E-08	5.352E-11	4.192E-09
		TAS	5.545E-10	-7.936E-09	2.283E-09	4.413E-10
		TPD	-1.634E-07	-1.466E-08	-4.297E-09	5.447E-09
		ΔPA	5.084E-07	4.059E-08	9.296E-09	-2.125E-08
		ΔDWI	-5.713E-08	1.297E-08	2.659E-09	1.519E-09
		TDWE	-3.478E-09	1.417E-08	3.091E-09	-1.726E-08
		TDRI	-4.567E-08	1.140E-08	-5.110E-09	1.043E-09
	Correlations	ΔDA	1.000	-.310	.064	-.149
		TDRE	-.310	1.000	-.004	.029
		ΔDRI	.064	-.004	1.000	-.004
		ΔDWE	-.149	.029	-.004	1.000
		ΔDRE	.043	-.268	.003	.103
		TAS	.031	-.104	.015	-.004
		TPD	.002	-.741	.008	.006
		ΔPA	-.008	.693	-.023	-.015
		ΔDWI	-.193	.065	.109	.031

12		TDWE	-.248	.853	-.001	-.202
		TDRI	-.361	.869	-.170	.059
		TDWI	-.282	.851	-.074	.049
	Covariances	ΔDA	1.664E-06	-1.338E-07	1.250E-08	-4.066E-08
		TDRE	-1.338E-07	1.118E-07	-1.993E-10	2.053E-09
		ΔDRI	1.250E-08	-1.993E-10	2.265E-08	-1.295E-10
		ΔDWE	-4.066E-08	2.053E-09	-1.295E-10	4.498E-08
		ΔDRE	1.026E-08	-1.666E-08	7.709E-11	4.069E-09
		TAS	8.007E-08	-6.997E-08	4.654E-09	-1.853E-09
		TPD	1.651E-09	-1.404E-07	7.224E-10	6.885E-10
		ΔPA	-2.044E-08	4.439E-07	-6.686E-09	-5.906E-09
		ΔDWI	-4.459E-08	3.913E-09	2.936E-09	1.161E-09
		TDWE	-1.042E-07	9.268E-08	-5.632E-11	-1.396E-08
		TDRI	-1.491E-07	9.290E-08	-8.175E-09	4.019E-09
		TDWI	-1.160E-07	9.080E-08	-3.540E-09	3.332E-09
13	Correlations	ΔDA	1.000	-.460	.064	-.149
		TDRE	-.460	1.000	.003	.049
		ΔDRI	.064	.003	1.000	-.004
		ΔDWE	-.149	.049	-.004	1.000
		ΔDRE	.043	-.399	.003	.103
		TAS	.032	.256	.013	-.007
		ΔPA	-.038	-.356	-.092	-.055
		ΔDWI	-.193	.100	.109	.031
		TDWE	-.382	.661	.008	-.307
		TDRI	-.580	.691	-.263	.103
		TDWI	-.450	.647	-.108	.086
	Covariances	ΔDA	1.660E-06	-1.328E-07	1.247E-08	-4.058E-08
		TDRE	-1.328E-07	5.024E-08	1.165E-10	2.349E-09
		ΔDRI	1.247E-08	1.165E-10	2.260E-08	-1.308E-10
		ΔDWE	-4.058E-08	2.349E-09	-1.308E-10	4.488E-08
		ΔDRE	1.024E-08	-1.662E-08	7.690E-11	4.060E-09
		TAS	7.781E-08	1.076E-07	3.731E-09	-2.719E-09
		ΔPA	-1.490E-08	-2.455E-08	-4.267E-09	-3.601E-09
		ΔDWI	-4.449E-08	4.018E-09	2.929E-09	1.158E-09
		TDWE	-1.032E-07	3.104E-08	2.598E-10	-1.362E-08
		TDRI	-1.480E-07	3.068E-08	-7.839E-09	4.314E-09

		TDWI	-1.151E-07	2.881E-08	-3.214E-09	3.628E-09
14	Correlations	ΔDA	1.000	-.507	.061	-.151
		TDRE	-.507	1.000	-.032	.032
		ΔDRI	.061	-.032	1.000	-.009
		ΔDWE	-.151	.032	-.009	1.000
		ΔDRE	.043	-.428	.003	.103
		TAS	.033	.286	.016	-.005
		ΔDWI	-.194	.098	.107	.029
		TDWE	-.424	.612	-.026	-.350
		TDRI	-.609	.665	-.297	.092
		TDWI	-.484	.606	-.143	.073
	Covariances	ΔDA	1.654E-06	-1.364E-07	1.178E-08	-4.106E-08
		TDRE	-1.364E-07	4.379E-08	-9.866E-10	1.414E-09
		ΔDRI	1.178E-08	-9.866E-10	2.236E-08	-2.923E-10
		ΔDWE	-4.106E-08	1.414E-09	-2.923E-10	4.465E-08
		ΔDRE	1.020E-08	-1.661E-08	7.261E-11	4.048E-09
		TAS	8.059E-08	1.123E-07	4.569E-09	-2.000E-09
		ΔDWI	-4.460E-08	3.675E-09	2.865E-09	1.107E-09
		TDWE	-1.066E-07	2.503E-08	-7.745E-10	-1.447E-08
		TDRI	-1.501E-07	2.666E-08	-8.509E-09	3.725E-09
		TDWI	-1.177E-07	2.398E-08	-4.036E-09	2.921E-09

a. Dependent Variable: ENERGY

Table D.54: Coefficient Correlations, part B^a

Model			ΔDRE	TAS	TPD	TPA
1	Correlations	ΔDA				
	Covariances	ΔDA				
2	Correlations	ΔDA TDRE				
	Covariances	ΔDA TDRE				
3	Correlations	ΔDA TDRE ΔDRI				
	Covariances	ΔDA TDRE ΔDRI				
4	Correlations	ΔDA TDRE ΔDRI ΔDWE				
	Covariances	ΔDA TDRE ΔDRI ΔDWE				
5	Correlations	ΔDA TDRE ΔDRI ΔDWE ΔDRE	.027 -.565 .022 .016 1.000			
	Covariances	ΔDA TDRE ΔDRI ΔDWE ΔDRE	5.376E-09 -1.759E-08 6.617E-10 6.128E-10 3.984E-08			
	Correlations	ΔDA TDRE ΔDRI	.035 -.563 .024	.329 .050 .071		

6		ΔDWE	.018	.039		
		ΔDRE	1.000	.031		
		TAS	.031	1.000		
	Covariances	ΔDA	7.303E-09	6.492E-07		
		TDRE	-1.696E-08	1.448E-08		
		ΔDRI	7.024E-10	1.937E-08		
		ΔDWE	6.359E-10	1.346E-08		
		ΔDRE	3.854E-08	1.143E-08		
		TAS	1.143E-08	3.523E-06		
7	Correlations	ΔDA	.053	.190	-.375	
		TDRE	-.546	.003	-.162	
		ΔDRI	.030	.036	-.116	
		ΔDWE	.026	-.010	-.171	
		ΔDRE	1.000	.015	-.053	
		TAS	.015	1.000	.274	
		TPD	-.053	.274	1.000	
	Covariances	ΔDA	1.136E-08	4.074E-07	-3.395E-08	
		TDRE	-1.612E-08	1.009E-09	-2.006E-09	
		ΔDRI	8.502E-10	9.945E-09	-1.355E-09	
		ΔDWE	9.386E-10	-3.523E-09	-2.556E-09	
		ΔDRE	3.732E-08	5.637E-09	-8.359E-10	
		TAS	5.637E-09	3.676E-06	4.255E-08	
		TPD	-8.359E-10	4.255E-08	6.582E-09	
8	Correlations	ΔDA	.063	-.012	-.486	.418
		TDRE	-.545	.000	-.046	.008
		ΔDRI	.033	-.002	-.111	.086
		ΔDWE	.025	.001	-.016	-.024
		ΔDRE	1.000	-.001	-.048	.037
		TAS	-.001	1.000	.454	-.407
		TPD	-.048	.454	1.000	-.972
		ΔPA	.037	-.407	-.972	1.000
	Covariances	ΔDA	1.430E-08	-2.921E-08	-1.972E-07	6.268E-07
		TDRE	-1.536E-08	-6.549E-11	-2.318E-09	1.543E-09
		ΔDRI	8.951E-10	-7.111E-10	-5.314E-09	1.530E-08
		ΔDWE	8.647E-10	2.628E-10	-1.007E-09	-5.437E-09
		ΔDRE	3.562E-08	-4.028E-10	-3.077E-09	8.671E-09

		TAS	-4.028E-10	4.201E-06	3.153E-07	-1.045E-06
		TPD	-3.077E-09	3.153E-07	1.147E-07	-4.124E-07
		ΔPA	8.671E-09	-1.045E-06	-4.124E-07	1.569E-06
9	Correlations	ΔDA	.057	-.012	-.469	.405
		TDRE	-.531	.000	-.053	.013
		ΔDRI	.038	-.003	-.117	.090
		ΔDWE	.030	.001	-.024	-.019
		ΔDRE	1.000	-.001	-.049	.037
		TAS	-.001	1.000	.454	-.407
		TPD	-.049	.454	1.000	-.972
		ΔPA	.037	-.407	-.972	1.000
		ΔDWI	.028	-.001	-.044	.028
	Covariances	ΔDA	1.302E-08	-2.864E-08	-1.925E-07	6.146E-07
		TDRE	-1.511E-08	-1.104E-10	-2.717E-09	2.514E-09
		ΔDRI	1.044E-09	-7.548E-10	-5.721E-09	1.624E-08
		ΔDWE	1.030E-09	2.077E-10	-1.488E-09	-4.245E-09
		ΔDRE	3.541E-08	-4.087E-10	-3.135E-09	8.799E-09
		TAS	-4.087E-10	4.173E-06	3.132E-07	-1.038E-06
		TPD	-3.135E-09	3.132E-07	1.142E-07	-4.102E-07
		ΔPA	8.799E-09	-1.038E-06	-4.102E-07	1.559E-06
		ΔDWI	8.087E-10	-2.526E-10	-2.308E-09	5.465E-09
10	Correlations	ΔDA	.040	-.012	-.478	.414
		TDRE	-.544	-.002	-.114	.058
		ΔDRI	-.001	-.003	-.161	.121
		ΔDWE	.104	.003	.085	-.089
		ΔDRE	1.000	.000	-.019	.016
		TAS	.000	1.000	.446	-.404
		TPD	-.019	.446	1.000	-.971
		ΔPA	.016	-.404	-.971	1.000
		ΔDWI	-.006	-.002	-.086	.058
	Covariances	TDWE	-.148	-.004	-.196	.138
		ΔDA	9.147E-09	-2.948E-08	-1.999E-07	6.328E-07
		TDRE	-1.646E-08	-5.322E-10	-6.242E-09	1.162E-08
		ΔDRI	-1.846E-11	-1.050E-09	-8.205E-09	2.262E-08
		ΔDWE	4.259E-09	1.133E-09	6.302E-09	-2.421E-08
		ΔDRE	3.590E-08	-1.818E-10	-1.233E-09	3.901E-09

		TAS	-1.818E-10	4.138E-06	3.112E-07	-1.031E-06
		TPD	-1.233E-09	3.112E-07	1.177E-07	-4.184E-07
		ΔPA	3.901E-09	-1.031E-06	-4.184E-07	1.576E-06
		ΔDWI	-1.829E-10	-5.324E-10	-4.654E-09	1.150E-08
		TDWE	-4.351E-09	-1.245E-09	-1.045E-08	2.688E-08
11	Correlations	ΔDA	.041	.000	-.365	.312
		TDRE	-.490	-.022	-.231	.176
		ΔDRI	.002	.007	-.079	.047
		ΔDWE	.104	.001	.071	-.076
		ΔDRE	1.000	.000	-.015	.013
		TAS	.000	1.000	.439	-.400
		TPD	-.015	.439	1.000	-.973
		ΔPA	.013	-.400	-.973	1.000
		ΔDWI	-.010	-.023	-.215	.183
		TDWE	-.137	-.023	-.297	.240
		TDRI	-.011	-.045	-.296	.278
	Covariances	ΔDA	9.676E-09	5.545E-10	-1.634E-07	5.084E-07
		TDRE	-1.642E-08	-7.936E-09	-1.466E-08	4.059E-08
		ΔDRI	5.352E-11	2.283E-09	-4.297E-09	9.296E-09
		ΔDWE	4.192E-09	4.413E-10	5.447E-09	-2.125E-08
		ΔDRE	3.546E-08	3.419E-12	-1.008E-09	3.134E-09
		TAS	3.419E-12	4.096E-06	3.170E-07	-1.051E-06
		TPD	-1.008E-09	3.170E-07	1.274E-07	-4.514E-07
		ΔPA	3.134E-09	-1.051E-06	-4.514E-07	1.688E-06
		ΔDWI	-3.523E-10	-8.469E-09	-1.370E-08	4.257E-08
		TDWE	-4.446E-09	-8.133E-09	-1.824E-08	5.367E-08
		TDRI	-2.816E-10	-1.303E-08	-1.494E-08	5.119E-08
	Correlations	ΔDA	.043	.031	.002	-.008
		TDRE	-.268	-.104	-.741	.693
		ΔDRI	.003	.015	.008	-.023
		ΔDWE	.103	-.004	.006	-.015
		ΔDRE	1.000	.001	.000	.000
		TAS	.001	1.000	.357	-.347
		TPD	.000	.357	1.000	-.987
		ΔPA	.000	-.347	-.987	1.000
		ΔDWI	-.008	-.004	.003	-.006

12		TDWE	-.082	-.105	-.764	.717
		TDRI	-.015	-.118	-.784	.749
		TDWI	-.012	-.109	-.783	.742
	Covariances	ΔDA	1.026E-08	8.007E-08	1.651E-09	-2.044E-08
		TDRE	-1.666E-08	-6.997E-08	-1.404E-07	4.439E-07
		ΔDRI	7.709E-11	4.654E-09	7.224E-10	-6.686E-09
		ΔDWE	4.069E-09	-1.853E-09	6.885E-10	-5.906E-09
		ΔDRE	3.462E-08	4.937E-10	1.018E-11	-1.260E-10
		TAS	4.937E-10	4.046E-06	4.065E-07	-1.337E-06
		TPD	1.018E-11	4.065E-07	3.210E-07	-1.071E-06
		ΔPA	-1.260E-10	-1.337E-06	-1.071E-06	3.668E-06
		ΔDWI	-2.749E-10	-1.538E-09	2.591E-10	-2.160E-09
		TDWE	-4.961E-09	-6.859E-08	-1.407E-07	4.464E-07
		TDRI	-9.191E-10	-7.561E-08	-1.421E-07	4.586E-07
		TDWI	-7.154E-10	-6.984E-08	-1.415E-07	4.538E-07
13	Correlations	ΔDA	.043	.032		-.038
		TDRE	-.399	.256		-.356
		ΔDRI	.003	.013		-.092
		ΔDWE	.103	-.007		-.055
		ΔDRE	1.000	.001		-.002
		TAS	.001	1.000		.033
		ΔPA	-.002	.033		1.000
		ΔDWI	-.008	-.006		-.023
		TDWE	-.127	.278		-.357
		TDRI	-.025	.280		-.251
		TDWI	-.019	.293		-.302
	Covariances	ΔDA	1.024E-08	7.781E-08		-1.490E-08
		TDRE	-1.662E-08	1.076E-07		-2.455E-08
		ΔDRI	7.690E-11	3.731E-09		-4.267E-09
		ΔDWE	4.060E-09	-2.719E-09		-3.601E-09
		ΔDRE	3.454E-08	4.797E-10		-9.185E-11
		TAS	4.797E-10	3.523E-06		1.882E-08
		ΔPA	-9.185E-11	1.882E-08		9.478E-08
		ΔDWI	-2.743E-10	-1.862E-09		-1.293E-09
		TDWE	-4.946E-09	1.094E-07		-2.301E-08
		TDRI	-9.126E-10	1.040E-07		-1.530E-08

		TDWI	-7.094E-10	1.092E-07		-1.845E-08
14	Correlations	ΔDA	.043	.033		
		TDRE	-.428	.286		
		ΔDRI	.003	.016		
		ΔDWE	.103	-.005		
		ΔDRE	1.000	.001		
		TAS	.001	1.000		
		ΔDWI	-.008	-.005		
		TDWE	-.137	.310		
		TDRI	-.026	.298		
		TDWI	-.021	.318		
	Covariances	ΔDA	1.020E-08	8.059E-08		
		TDRE	-1.661E-08	1.123E-07		
		ΔDRI	7.261E-11	4.569E-09		
		ΔDWE	4.048E-09	-2.000E-09		
		ΔDRE	3.447E-08	4.969E-10		
		TAS	4.969E-10	3.512E-06		
		ΔDWI	-2.750E-10	-1.602E-09		
		TDWE	-4.958E-09	1.137E-07		
		TDRI	-9.255E-10	1.069E-07		
		TDWI	-7.258E-10	1.126E-07		

a. Dependent Variable: ENERGY

Table D.55: Coefficient Correlations, part C^a

Model			ΔDWI	TDWE	TDRI	TDWI
1	Correlations	ΔDA				
	Covariances	ΔDA				
2	Correlations	ΔDA TDRE				
	Covariances	ΔDA TDRE				
3	Correlations	ΔDA TDRE ΔDRI				
	Covariances	ΔDA TDRE ΔDRI				
4	Correlations	ΔDA TDRE ΔDRI ΔDWE				
	Covariances	ΔDA TDRE ΔDRI ΔDWE				
5	Correlations	ΔDA TDRE ΔDRI ΔDWE ΔDRE				
	Covariances	ΔDA TDRE ΔDRI ΔDWE ΔDRE				
	Correlations	ΔDA TDRE ΔDRI				

6		ΔDWE ΔDRE TAS				
	Covariances	ΔDA TDRE ΔDRI ΔDWE ΔDRE TAS				
7	Correlations	ΔDA TDRE ΔDRI ΔDWE ΔDRE TAS TPD				
	Covariances	ΔDA TDRE ΔDRI ΔDWE ΔDRE TAS TPD				
8	Correlations	ΔDA TDRE ΔDRI ΔDWE ΔDRE TAS TPD ΔPA				
	Covariances	ΔDA TDRE ΔDRI ΔDWE ΔDRE				

		TAS TPD ΔPA				
9	Correlations	ΔDA TDRE ΔDRI ΔDWE ΔDRE TAS TPD ΔPA ΔDWI	-0.185 .183 .205 .178 .028 -0.001 -0.044 .028 1.000			
	Covariances	ΔDA TDRE ΔDRI ΔDWE ΔDRE TAS TPD ΔPA ΔDWI	-3.469E-08 4.265E-09 4.555E-09 5.018E-09 8.087E-10 -2.526E-10 -2.308E-09 5.465E-09 2.375E-08			
10	Correlations	ΔDA TDRE ΔDRI ΔDWE ΔDRE TAS TPD ΔPA ΔDWI TDWE	-0.155 .242 .250 .026 -0.006 -0.002 -0.086 .058 1.000 .224	.111 .331 .254 -0.537 -0.148 -0.004 -0.196 .138 .224 1.000		
	Covariances	ΔDA TDRE ΔDRI ΔDWE ΔDRE	-2.966E-08 6.092E-09 5.847E-09 8.940E-10 -1.829E-10	2.098E-08 8.226E-09 5.872E-09 -1.804E-08 -4.351E-09		

		TAS	-5.324E-10	-1.245E-09		
		TPD	-4.654E-09	-1.045E-08		
		ΔPA	1.150E-08	2.688E-08		
		ΔDWI	2.479E-08	5.486E-09		
		TDWE	5.486E-09	2.424E-08		
11	Correlations	ΔDA	-.255	-.016	-.257	
		TDRE	.408	.463	.453	
		ΔDRI	.098	.118	-.238	
		ΔDWE	.040	-.468	.034	
		ΔDRE	-.010	-.137	-.011	
		TAS	-.023	-.023	-.045	
		TPD	-.215	-.297	-.296	
		ΔPA	.183	.240	.278	
		ΔDWI	1.000	.387	.483	
		TDWE	.387	1.000	.436	
		TDRI	.483	.436	1.000	
	Covariances	ΔDA	-5.713E-08	-3.478E-09	-4.567E-08	
		TDRE	1.297E-08	1.417E-08	1.140E-08	
		ΔDRI	2.659E-09	3.091E-09	-5.110E-09	
		ΔDWE	1.519E-09	-1.726E-08	1.043E-09	
		ΔDRE	-3.523E-10	-4.446E-09	-2.816E-10	
		TAS	-8.469E-09	-8.133E-09	-1.303E-08	
		TPD	-1.370E-08	-1.824E-08	-1.494E-08	
		ΔPA	4.257E-08	5.367E-08	5.119E-08	
		ΔDWI	3.194E-08	1.190E-08	1.222E-08	
		TDWE	1.190E-08	2.957E-08	1.062E-08	
		TDRI	1.222E-08	1.062E-08	2.005E-08	
	Correlations	ΔDA	-.193	-.248	-.361	-.282
		TDRE	.065	.853	.869	.851
		ΔDRI	.109	-.001	-.170	-.074
		ΔDWE	.031	-.202	.059	.049
		ΔDRE	-.008	-.082	-.015	-.012
		TAS	-.004	-.105	-.118	-.109
		TPD	.003	-.764	-.784	-.783
		ΔPA	-.006	.717	.749	.742
		ΔDWI	1.000	.053	.054	-.172

12		TDWE	.053	1.000	.866	.853
		TDRI	.054	.866	1.000	.899
		TDWI	-.172	.853	.899	1.000
	Covariances	ΔDA	-4.459E-08	-1.042E-07	-1.491E-07	-1.160E-07
		TDRE	3.913E-09	9.268E-08	9.290E-08	9.080E-08
		ΔDRI	2.936E-09	-5.632E-11	-8.175E-09	-3.540E-09
		ΔDWE	1.161E-09	-1.396E-08	4.019E-09	3.332E-09
		ΔDRE	-2.749E-10	-4.961E-09	-9.191E-10	-7.154E-10
		TAS	-1.538E-09	-6.859E-08	-7.561E-08	-6.984E-08
		TPD	2.591E-10	-1.407E-07	-1.421E-07	-1.415E-07
		ΔPA	-2.160E-09	4.464E-07	4.586E-07	4.538E-07
		ΔDWI	3.212E-08	3.085E-09	3.091E-09	-9.817E-09
		TDWE	3.085E-09	1.057E-07	9.007E-08	8.850E-08
		TDRI	3.091E-09	9.007E-08	1.022E-07	9.177E-08
		TDWI	-9.817E-09	8.850E-08	9.177E-08	1.019E-07
13	Correlations	ΔDA	-.193	-.382	-.580	-.450
		TDRE	.100	.661	.691	.647
		ΔDRI	.109	.008	-.263	-.108
		ΔDWE	.031	-.307	.103	.086
		ΔDRE	-.008	-.127	-.025	-.019
		TAS	-.006	.278	.280	.293
		ΔPA	-.023	-.357	-.251	-.302
		ΔDWI	1.000	.085	.090	-.272
		TDWE	.085	1.000	.668	.634
		TDRI	.090	.668	1.000	.739
		TDWI	-.272	.634	.739	1.000
	Covariances	ΔDA	-4.449E-08	-1.032E-07	-1.480E-07	-1.151E-07
		TDRE	4.018E-09	3.104E-08	3.068E-08	2.881E-08
		ΔDRI	2.929E-09	2.598E-10	-7.839E-09	-3.214E-09
		ΔDWE	1.158E-09	-1.362E-08	4.314E-09	3.628E-09
		ΔDRE	-2.743E-10	-4.946E-09	-9.126E-10	-7.094E-10
		TAS	-1.862E-09	1.094E-07	1.040E-07	1.092E-07
		ΔPA	-1.293E-09	-2.301E-08	-1.530E-08	-1.845E-08
		ΔDWI	3.205E-08	3.192E-09	3.199E-09	-9.682E-09
		TDWE	3.192E-09	4.392E-08	2.773E-08	2.638E-08
		TDRI	3.199E-09	2.773E-08	3.926E-08	2.906E-08

		TDWI	-9.682E-09	2.638E-08	2.906E-08	3.940E-08
14	Correlations	ΔDA	-.194	-.424	-.609	-.484
		TDRE	.098	.612	.665	.606
		ΔDRI	.107	-.026	-.297	-.143
		ΔDWE	.029	-.350	.092	.073
		ΔDRE	-.008	-.137	-.026	-.021
		TAS	-.005	.310	.298	.318
		ΔDWI	1.000	.082	.087	-.293
		TDWE	.082	1.000	.639	.591
		TDRI	.087	.639	1.000	.719
		TDWI	-.293	.591	.719	1.000
	Covariances	ΔDA	-4.460E-08	-1.066E-07	-1.501E-07	-1.177E-07
		TDRE	3.675E-09	2.503E-08	2.666E-08	2.398E-08
		ΔDRI	2.865E-09	-7.745E-10	-8.509E-09	-4.036E-09
		ΔDWE	1.107E-09	-1.447E-08	3.725E-09	2.921E-09
		ΔDRE	-2.750E-10	-4.958E-09	-9.255E-10	-7.258E-10
		TAS	-1.602E-09	1.137E-07	1.069E-07	1.126E-07
		ΔDWI	3.197E-08	2.872E-09	2.984E-09	-9.913E-09
		TDWE	2.872E-09	3.825E-08	2.396E-08	2.186E-08
		TDRI	2.984E-09	2.396E-08	3.671E-08	2.603E-08
		TDWI	-9.913E-09	2.186E-08	2.603E-08	3.573E-08

a. Dependent Variable: ENERGY

Table D.56: Colinearity Diagnostics part A^a

Model	Dimension	Eigenvalue	Condition Index	Variance Properties					
				Constant	Δ	T	Δ	Δ	Δ
				D	D	R	R	W	R
				A	E	I	E	E	E
1	1	1.828	1.000	.09	.09				
	2	.172	3.255	.91	.91				
2	1	2.102	1.000	.06	.06	.08			
	2	.726	1.701	.04	.04	.92			
	3	.172	3.492	.90	.90	.00			
3	1	2.243	1.000	.05	.05	.05	.04		
	2	1.000	1.498	.00	.00	.34	.50		
	3	.584	1.960	.06	.07	.61	.46		
	4	.172	3.608	.89	.88	.00	.00		
4	1	2.325	1.000	.05	.05	.04	.03	.02	
	2	1.000	1.524	.00	.00	.12	.05	.70	
	3	1.000	1.525	.00	.00	.31	.52	.00	
	4	.503	2.150	.08	.08	.53	.40	.28	
	5	.172	3.676	.87	.87	.00	.00	.00	
5	1	2.543	1.000	.04	.03	.04	.02	.01	.03
	2	1.340	1.378	.01	.01	.10	.12	.09	.14
	3	1.000	1.594	.00	.00	.00	.33	.54	.00
	4	.591	2.075	.06	.07	.00	.48	.33	.19
	5	.355	2.676	.02	.01	.85	.04	.03	.63
	6	.171	3.852	.87	.87	.00	.00	.00	.01
6	1	2.554	1.000	.03	.03	.04	.02	.01	.03
	2	1.347	1.377	.01	.01	.11	.11	.08	.14
	3	1.030	1.574	.00	.00	.00	.01	.05	.00
	4	1.000	1.598	.00	.00	.00	.35	.51	.00
	5	.577	2.105	.04	.08	.00	.45	.32	.20

	6	.353	2.689	.01	.02	.86	.05	.03	.62
	7	.139	4.283	.90	.86	.00	.00	.00	.01
7	1	3.396	1.000	.01	.02	.02	.01	.01	.01
	2	1.375	1.572	.00	.00	.12	.08	.06	.17
	3	1.035	1.811	.00	.00	.00	.01	.04	.00
	4	1.000	1.843	.00	.00	.00	.35	.49	.00
	5	.608	2.364	.01	.03	.00	.50	.36	.18
	6	.355	3.093	.00	.01	.84	.04	.03	.63
	7	.156	4.664	.18	.93	.00	.00	.00	.01
	8	7.470E-02	6.743	.79	.01	.01	.00	.02	.00
8	1	4.239	1.000	.00	.01	.01	.01	.01	.01
	2	1.385	1.749	.00	.00	.13	.07	.06	.17
	3	1.039	2.020	.00	.00	.00	.00	.04	.00
	4	1.001	2.058	.00	.00	.00	.36	.46	.00
	5	.625	2.604	.00	.01	.01	.51	.40	.17
	6	.356	3.452	.00	.00	.85	.04	.03	.64
	7	.223	4.356	.01	.44	.00	.00	.00	.00
	8	.128	5.758	.31	.32	.00	.00	.01	.00
	9	4.275E-03	31.488	.67	.21	.00	.01	.00	.00
9	1	4.342	1.000	.00	.01	.01	.01	.00	.01
	2	1.424	1.746	.00	.00	.13	.04	.03	.17
	3	1.040	2.043	.00	.00	.00	.00	.02	.00
	4	1.004	2.080	.00	.00	.00	.03	.34	.00
	5	1.000	2.084	.00	.00	.00	.47	.25	.00
	6	.503	2.938	.00	.01	.01	.33	.26	.35
	7	.338	3.584	.00	.00	.85	.11	.09	.46
	8	.218	4.466	.01	.45	.00	.01	.00	.01
	9	.127	5.846	.30	.34	.01	.00	.01	.00
	10	4.269E-03	31.891	.67	.20	.00	.01	.00	.00
	1	4.563	1.000	.00	.01	.01	.00	.00	.00
	2	1.569	1.705	.00	.00	.08	.00	.11	.08
	3	1.305	1.870	.00	.00	.04	.11	.06	.08
	4	1.039	2.096	.00	.00	.00	.00	.00	.00
	5	1.002	2.134	.00	.00	.00	.38	.00	.00

10	6	.503	3.011	.00	.01	.01	.31	.14	.37
	7	.433	3.248	.00	.00	.23	.00	.50	.18
	8	.245	4.312	.01	.13	.54	.16	.10	.26
	9	.210	4.662	.01	.30	.09	.01	.05	.02
	10	.126	6.023	.31	.34	.00	.00	.02	.00
	11	4.139E-03	33.207	.67	.21	.01	.02	.01	.00
11	1	4.815	1.000	.00	.01	.00	.00	.00	.00
	2	1.603	1.733	.00	.00	.02	.11	.04	.05
	3	1.569	1.751	.00	.00	.06	.00	.10	.08
	4	1.107	2.086	.00	.00	.01	.06	.01	.02
	5	1.039	2.153	.00	.00	.00	.00	.00	.00
	6	.550	2.960	.00	.01	.00	.71	.04	.10
	7	.434	3.333	.00	.00	.21	.01	.36	.30
	8	.410	3.428	.00	.01	.03	.07	.32	.32
	9	.212	4.767	.00	.16	.19	.03	.08	.06
	10	.135	5.974	.00	.59	.37	.00	.00	.05
	11	.124	6.219	.34	.10	.05	.00	.03	.00
	12	3.806E-03	35.568	.64	.12	.05	.00	.01	.00
12	1	5.034	1.000	.00	.01	.00	.00	.00	.00
	2	1.686	1.728	.00	.00	.01	.02	.04	.05
	3	1.569	1.791	.00	.00	.02	.00	.11	.08
	4	1.523	1.818	.00	.00	.00	.12	.00	.01
	5	1.041	2.199	.00	.00	.00	.01	.00	.00
	6	.569	2.975	.00	.01	.00	.73	.04	.09
	7	.442	3.376	.00	.00	.01	.00	.08	.32
	8	.433	3.409	.00	.00	.05	.01	.49	.18
	9	.326	3.927	.00	.00	.03	.08	.19	.24
	10	.202	4.996	.01	.24	.02	.02	.04	.01
	11	.128	6.281	.21	.44	.01	.00	.02	.01
	12	4.581E-02	10.483	.24	.29	.30	.01	.00	.01
	13	1.510E-03	57.741	.54	.00	.55	.00	.00	.00
	1	4.085	1.000	.01	.01	.00	.01	.00	.00
	2	1.685	1.557	.00	.00	.02	.02	.04	.05
	3	1.569	1.613	.00	.00	.04	.00	.11	.08

13	4	1.523	1.638	.00	.00	.00	.12	.00	.01
	5	1.040	1.982	.00	.00	.00	.01	.00	.00
	6	.565	2.689	.00	.01	.00	.72	.03	.07
	7	.435	3.066	.00	.00	.14	.01	.21	.42
	8	.422	3.112	.01	.00	.00	.00	.37	.08
	9	.326	3.540	.00	.00	.07	.08	.20	.25
	10	.177	4.810	.01	.25	.02	.02	.02	.00
	11	.128	5.660	.46	.43	.02	.00	.02	.01
	12	4.566E-02	9.459	.51	.30	.69	.01	.00	.01
14	1	3.289	1.000	.01	.01	.00	.01	.01	.01
	2	1.677	1.400	.00	.00	.02	.02	.05	.06
	3	1.569	1.448	.00	.00	.04	.00	.11	.08
	4	1.522	1.470	.00	.00	.00	.12	.00	.01
	5	1.038	1.780	.00	.00	.00	.01	.00	.00
	6	.564	2.415	.00	.02	.00	.72	.02	.06
	7	.434	2.753	.00	.00	.15	.01	.32	.34
	8	.401	2.863	.02	.00	.01	.00	.23	.13
	9	.324	3.187	.00	.00	.10	.09	.24	.29
	10	.133	4.967	.34	.65	.01	.01	.03	.00
	11	4.800E-02	8.278	.63	.32	.66	.02	.00	.01

a. Dependent Variable: ENERGY

Table D.57: Colinearity Diagnostics part B^a

M o d e l	D i m	Variance Properties						
		<i>T</i>	<i>T</i>	Δ	Δ	<i>T</i>	<i>T</i>	<i>T</i>
		<i>A</i>	<i>P</i>	<i>P</i>	<i>D</i>	<i>D</i>	<i>D</i>	<i>D</i>
		<i>S</i>	<i>D</i>	<i>A</i>	<i>W</i>	<i>W</i>	<i>R</i>	<i>W</i>
1	1							
	2							
2	1							
	2							
	3							
3	1							
	2							
	3							
	4							
4	1							
	2							
	3							
	4							
	5							
5	1							
	2							
	3							
	4							
	5							
	6							
6	1	.00						
	2	.01						
	3	.71						
	4	.01						
	5	.04						

	6	.01						
	7	.22						
7	1	.00	.01					
	2	.01	.00					
	3	.67	.00					
	4	.01	.00					
	5	.03	.01					
	6	.00	.00					
	7	.10	.11					
	8	.19	.87					
8	1	.00	.00	.00				
	2	.00	.00	.00				
	3	.55	.00	.00				
	4	.01	.00	.00				
	5	.02	.00	.00				
	6	.00	.00	.00				
	7	.00	.00	.02				
	8	.20	.00	.01				
	9	.21	1.00	.97				
9	1	.00	.00	.00	.01			
	2	.00	.00	.00	.04			
	3	.56	.00	.00	.01			
	4	.00	.00	.00	.43			
	5	.01	.00	.00	.06			
	6	.01	.00	.00	.31			
	7	.01	.00	.00	.10			
	8	.00	.00	.02	.04			
	9	.20	.00	.01	.01			
	10	.21	1.00	.97	.00			
	1	.00	.00	.00	.00	.01		
	2	.00	.00	.00	.00	.07		
	3	.01	.00	.00	.11	.04		
	4	.56	.00	.00	.03	.00		
	5	.01	.00	.00	.35	.00		

10	6	.01	.00	.00	.30	.00		
	7	.00	.00	.00	.00	.26		
	8	.01	.00	.00	.20	.41		
	9	.00	.00	.02	.00	.16		
	10	.20	.00	.01	.00	.02		
	11	.20	1.00	.97	.01	.03		
11	1	.00	.00	.00	.00	.00	.00	
	2	.00	.00	.00	.00	.03	.05	
	3	.00	.00	.00	.00	.05	.00	
	4	.01	.00	.00	.36	.01	.02	
	5	.56	.00	.00	.02	.00	.00	
	6	.01	.00	.00	.11	.00	.08	
	7	.00	.00	.00	.00	.19	.01	
	8	.00	.00	.00	.06	.05	.18	
	9	.00	.00	.02	.02	.26	.00	
	10	.00	.00	.00	.36	.23	.49	
	11	.22	.00	.01	.03	.10	.08	
	12	.20	1.00	.97	.04	.08	.08	
12	1	.00	.00	.00	.00	.00	.00	.00
	2	.00	.00	.00	.06	.01	.00	.01
	3	.00	.00	.00	.00	.01	.00	.00
	4	.00	.00	.00	.08	.00	.01	.01
	5	.56	.00	.00	.00	.00	.00	.00
	6	.01	.00	.00	.00	.00	.02	.01
	7	.00	.00	.00	.40	.00	.02	.02
	8	.00	.00	.00	.01	.06	.00	.00
	9	.00	.00	.00	.39	.03	.00	.05
	10	.00	.00	.01	.03	.03	.00	.01
	11	.16	.00	.00	.03	.00	.01	.00
	12	.13	.00	.00	.00	.27	.31	.28
	13	.12	1.00	.98	.00	.59	.62	.62
	1	.00		.01	.01	.00	.00	.00
	2	.00		.00	.06	.02	.00	.02
	3	.00		.00	.00	.03	.00	.00

13	4	.00		.00	.08	.00	.03	.02
	5	.65		.00	.00	.00	.00	.00
	6	.01		.00	.00	.00	.05	.01
	7	.00		.00	.05	.09	.01	.01
	8	.01		.03	.40	.05	.03	.05
	9	.00		.01	.36	.09	.01	.13
	10	.00		.74	.02	.05	.01	.01
	11	.18		.14	.03	.00	.02	.00
	12	.15		.07	.00	.66	.83	.75
14	1	.00			.01	.00	.00	.01
	2	.00			.06	.02	.00	.02
	3	.00			.00	.04	.00	.00
	4	.00			.08	.00	.04	.02
	5	.65			.00	.00	.00	.00
	6	.01			.00	.00	.06	.02
	7	.00			.01	.13	.00	.00
	8	.01			.52	.04	.03	.08
	9	.00			.29	.12	.01	.12
	10	.15			.04	.00	.03	.00
	11	.17			.00	.63	.82	.73

a. Dependent Variable: ENERGY

D.4.2.2 Internal Memory Access Only

Table D.58: C54 No External Memory Data Reduction SPSS Output Notes

Output Created	19-FEB-2002 22:04:00	
Comments		
Input	Filter	<none>
	Weight	<none>
	Split File	<none>
	Rows in Working Data File	315
Missing Value Handling	Definition of Missing	User-defined missing values are treated as missing.
	Cases Used	No missing values for any variable used.
Syntax		REGRESSION /DESCRIPTIVES MEAN STDDEV CORR SIG N /MISSING LISTWISE /STATISTICS COEFF OUTS CI BCOV R ANOVA COLLIN TOL CHANGE ZPP /CRITERIA=PIN(.05) POUT(.10) /NOORIGIN /DEPENDENT <i>energy</i> /METHOD=STEPWISE <i>ΔPA TPD ΔPD ΔDA TDRI ΔDRI TDWI ΔDWI TAS ΔAS TAD</i> .

Table D.59: Descriptive Statistics

	Mean	Std. Deviation	N
ENERGY	1.451711	.347516	315
<i>ΔPA</i>	51.43	40.48	315
TPD	313.14	173.57	315
<i>ΔPD</i>	46.76	55.03	315
<i>ΔDA</i>	16.19	12.65	315
TDRI	103.77	124.57	315
<i>ΔDRI</i>	41.45	94.45	315
TDWI	87.77	121.71	315
<i>ΔDWI</i>	36.57	89.72	315
TAS	2.29	7.80	315
<i>ΔAS</i>	1.00	5.51	315
TAD	3.23	10.61	315

Table D.60: Pearson Correlation, part A

	ENERGY	ΔPA	TPD	ΔPD	ΔDA	TDRI	ΔDRI	TDWI
ENERGY	1.000	.439	.509	.379	.514	.385	.403	.144
ΔPA	.439	1.000	.959	.962	.495	.201	.210	.308
TPD	.509	.959	1.000	.861	.616	.299	.242	.369
ΔPD	.379	.962	.861	1.000	.395	.074	.168	.281
ΔDA	.514	.495	.616	.395	1.000	.402	.149	.269
TDRI	.385	.201	.299	.074	.402	1.000	.378	-.603
ΔDRI	.403	.210	.242	.168	.149	.378	1.000	-.154
TDWI	.144	.308	.369	.281	.269	-.603	-.154	1.000
ΔDWI	.186	.192	.222	.230	.287	-.341	-.179	.565
TDRE	.	.	.	.	.	.	.	.
ΔDRE	.	.	.	.	.	.	.	.
TDWE	.	.	.	.	.	.	.	.
ΔDWE	.	.	.	.	.	.	.	.
TAS	-.062	-.301	-.422	-.164	-.376	-.207	-.129	-.212
ΔAS	-.032	-.186	-.261	-.104	-.233	-.128	-.080	-.131
TAD	-.072	-.313	-.438	-.173	-.391	-.215	-.134	-.220

Table D.61: Pearson Correlation, part B

	ΔDWI	TAS	ΔAS	TAD
<i>ENERGY</i>	.186	-.062	-.032	-.072
ΔPA	.192	-.301	-.186	-.313
<i>TPD</i>	.222	-.422	-.261	-.438
ΔPD	.230	-.164	-.104	-.173
ΔDA	.287	-.376	-.233	-.391
<i>TDRI</i>	-.341	-.207	-.128	-.215
ΔDRI	-.179	-.129	-.080	-.134
<i>TDWI</i>	.565	-.212	-.131	-.220
ΔDWI	1.000	-.120	-.074	-.124
<i>TDRE</i>	.	.	.	.
ΔDRE	.	.	.	.
<i>TDWE</i>	.	.	.	.
ΔDWE	.	.	.	.
<i>TAS</i>	-.120	1.000	.570	.981
ΔAS	-.074	.570	1.000	.592
<i>TAD</i>	-.124	.981	.592	1.000

Table D.62: Significance: 1-tailed, part A

	ENERGY	ΔPA	TPD	ΔPD	ΔDA	TDRI	ΔDRI	TDWI
<i>ENERGY</i>	.	.000	.000	.000	.000	.000	.000	.005
<i>ΔPA</i>	.000	.	.000	.000	.000	.000	.000	.000
<i>TPD</i>	.000	.000	.	.000	.000	.000	.000	.000
<i>ΔPD</i>	.000	.000	.000	.	.000	.096	.001	.000
<i>ΔDA</i>	.000	.000	.000	.000	.	.000	.004	.000
<i>TDRI</i>	.000	.000	.000	.096	.000	.	.000	.000
<i>ΔDRI</i>	.000	.000	.000	.001	.004	.000	.	.003
<i>TDWI</i>	.005	.000	.000	.000	.000	.000	.003	.
<i>ΔDWI</i>	.000	.000	.000	.000	.000	.000	.001	.000
<i>TAS</i>	.135	.000	.000	.002	.000	.000	.011	.000
<i>ΔAS</i>	.284	.000	.000	.033	.000	.011	.078	.010
<i>TAD</i>	.102	.000	.000	.001	.000	.000	.009	.000

Table D.63: Significance: 1-tailed, part B

	ΔDWI	TAS	ΔAS	TAD
<i>ENERGY</i>	.000	.135	.284	.102
ΔPA	.000	.000	.000	.000
<i>TPD</i>	.000	.000	.000	.000
ΔPD	.000	.002	.033	.001
ΔDA	.000	.000	.000	.000
<i>TDRI</i>	.000	.000	.011	.000
ΔDRI	.001	.011	.078	.009
<i>TDWI</i>	.000	.000	.010	.000
ΔDWI	.	.017	.094	.014
<i>TAS</i>	.017	.	.000	.000
ΔAS	.094	.000	.	.000
<i>TAD</i>	.014	.000	.000	.

Table D.64: Number of data points, part A

	ENERGY	ΔPA	TPD	ΔPD	ΔDA	TDRI	ΔDRI	TDWI
ENERGY	315	315	315	315	315	315	315	315
ΔPA	315	315	315	315	315	315	315	315
TPD	315	315	315	315	315	315	315	315
ΔPD	315	315	315	315	315	315	315	315
ΔDA	315	315	315	315	315	315	315	315
TDRI	315	315	315	315	315	315	315	315
ΔDRI	315	315	315	315	315	315	315	315
TDWI	315	315	315	315	315	315	315	315
ΔDWI	315	315	315	315	315	315	315	315
TAS	315	315	315	315	315	315	315	315
ΔAS	315	315	315	315	315	315	315	315
TAD	315	315	315	315	315	315	315	315

Table D.65: Number of data points, part B

	ΔDWI	TAS	ΔAS	TAD
ENERGY	315	315	315	315
ΔPA	315	315	315	315
TPD	315	315	315	315
ΔPD	315	315	315	315
ΔDA	315	315	315	315
TDRI	315	315	315	315
ΔDRI	315	315	315	315
TDWI	315	315	315	315
ΔDWI	315	315	315	315
TAS	315	315	315	315
ΔAS	315	315	315	315
TAD	315	315	315	315

Table D.66: Variables Entered or Removed^a

Model	Variables		Method
	Entered	Removed	
1	ΔDA	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
2	ΔDRI	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
3	TPD	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
4	TAD	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
5	ΔPA	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
6	ΔDWI	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
7	TDRI	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
8	TDWI	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
9	.	TPD	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).

a. Dependent Variable: ENERGY

Table D.67: Model Summary

Model	R	R^2	Adjusted R^2	Std. Error of the Estimate	R^2 Change	Change Statistics			
						F Change	d of f 1	2	Sig. F Change
1	.514 ^a	.264	.262	.298570	.264	112.390	1	313	.000
2	.611 ^b	.373	.369	.276009	.109	54.262	1	312	.000
3	.638 ^c	.407	.402	.268844	.034	17.851	1	311	.000
4	.676 ^d	.458	.451	.257575	.050	28.809	1	310	.000
5	.693 ^e	.480	.471	.252697	.022	13.082	1	309	.000
6	.700 ^f	.490	.480	.250663	.010	6.035	1	308	.015
7	.712 ^g	.507	.496	.246775	.017	10.783	1	307	.001
8	.728 ^h	.531	.518	.241181	.024	15.404	1	306	.000
9	.728 ⁱ	.531	.520	.240810	.000	.055	1	308	.815

a. Predictors: (Constant), ΔDA

b. Predictors: (Constant), ΔDA , ΔDRI

c. Predictors: (Constant), ΔDA , ΔDRI , TPD

d. Predictors: (Constant), ΔDA , ΔDRI , TPD, TAD

e. Predictors: (Constant), ΔDA , ΔDRI , TPD, TAD, ΔPA

f. Predictors: (Constant), ΔDA , ΔDRI , TPD, TAD, ΔPA , ΔDWI

g. Predictors: (Constant), ΔDA , ΔDRI , TPD, TAD, ΔPA , ΔDWI , TDRI

h. Predictors: (Constant), ΔDA , ΔDRI , TPD, TAD, ΔPA , ΔDWI , TDRI, TDWI

i. Predictors: (Constant), ΔDA , ΔDRI , TAD, ΔPA , ΔDWI , TDRI, TDWI

Table D.68: ANOVA^j

	Model	Σ of Squares	df	Mean Square	F	Sig.
1	Model	10.019	1	10.019	112.390	.000 ^a
	Residual	27.902	313	8.914E-02		
	Total	37.921	314			
2	Model	14.153	2	7.076	92.888	.000 ^b
	Residual	23.768	312	7.618E-02		
	Total	37.921	314			
3	Model	15.443	3	5.148	71.221	.000 ^c
	Residual	22.478	311	7.228E-02		
	Total	37.921	314			
4	Model	17.354	4	4.339	65.394	.000 ^d
	Residual	20.567	310	6.634E-02		
	Total	37.921	314			
5	Model	18.190	5	3.638	56.971	.000 ^e
	Residual	19.731	309	6.386E-02		
	Total	37.921	314			
6	Model	18.569	6	3.095	49.255	.000 ^f
	Residual	19.352	308	6.283E-02		
	Total	37.921	314			
8	Model	19.225	7	2.746	45.100	.000 ^g
	Residual	18.696	307	6.090E-02		
	Total	37.921	314			
9	Model	20.121	8	2.515	43.240	.000 ^h
	Residual	17.800	306	5.817E-02		
	Total	37.921	314			
10	Model	20.118	7	2.874	49.562	.000 ⁱ
	Residual	17.803	307	5.799E-02		
	Total	37.921	314			

a. Predictors: (Constant), ΔDA b. Predictors: (Constant), ΔDA , ΔDRI c. Predictors: (Constant), ΔDA , ΔDRI , TPD

- d. Predictors: (Constant), ΔDA , ΔDRI , TPD, TAD
- e. Predictors: (Constant), ΔDA , ΔDRI , TPD, TAD, ΔPA
- f. Predictors: (Constant), ΔDA , ΔDRI , TPD, TAD, ΔPA , ΔDWI
- g. Predictors: (Constant), ΔDA , ΔDRI , TPD, TAD, ΔPA , ΔDWI , TDRI
- h. Predictors: (Constant), ΔDA , ΔDRI , TPD, TAD, ΔPA , ΔDWI , TDRI, TDWI
- i. Predictors: (Constant), ΔDA , ΔDRI , TAD, ΔPA , ΔDWI , TDRI, TDWI
- j. Dependent Variable: ENERGY

Table D.69: Coefficients^a

Model		Unstandardized Coefficients		Standardized Coefficients β	t	Sig.
		$\hat{B}$	Std. Error			
1	(Constant)	1.223	.027		44.710	.000
	ΔDA	1.412E-02	.001	.514	10.601	.000
2	(Constant)	1.194	.026		46.672	.000
	ΔDA	1.276E-02	.001	.464	10.243	.000
	ΔDRI	1.228E-03	.000	.334	7.366	.000
3	(Constant)	1.113	.031		35.394	.000
	ΔDA	8.866E-03	.002	.323	5.820	.000
	ΔDRI	1.094E-03	.000	.297	6.607	.000
	TPD	4.779E-04	.000	.239	4.225	.000
4	(Constant)	1.014	.035		28.659	.000
	ΔDA	1.022E-02	.001	.372	6.903	.000
	ΔDRI	1.122E-03	.000	.305	7.068	.000
	TPD	6.358E-04	.000	.318	5.663	.000
	TAD	8.311E-03	.002	.254	5.367	.000
5	(Constant)	.890	.049		18.298	.000
	ΔDA	7.922E-03	.002	.288	4.994	.000
	ΔDRI	1.075E-03	.000	.292	6.878	.000
	TPD	2.023E-03	.000	1.011	5.070	.0002
	TAD	1.065E-02	.002	.325	6.451	.000
	ΔPA	-5.434E-03	.002	-.633	-3.617	.0006
6	(Constant)	.893	.048		18.489	.000
	ΔDA	7.230E-03	.002	.263	4.523	.000
	ΔDRI	1.174E-03	.000	.319	7.329	.000
	TPD	1.981E-03	.000	.990	4.999	.0007
	TAD	1.065E-02	.002	.325	6.502	.000
	ΔPA	-5.380E-03	.001	-.627	-3.609	.0000
	ΔDWI	4.185E-04	.000	.108	2.457	.015
7	(Constant)	.903	.048		18.958	.000
	ΔDA	5.494E-03	.002	.200	3.309	.001
	ΔDRI	1.043E-03	.000	.283	6.413	.000

	TPD	1.655E-03	.000	.827	4.111	.0008
	TAD	1.040E-02	.002	.317	6.438	.000
	ΔPA	-4.175E-03	.002	-.486	-2.760	.0068
	ΔDWI	7.352E-04	.000	.190	3.800	.000
	TDRI	5.052E-04	.000	.181	3.284	.001
8	(Constant)	.969	.049		19.583	.000
	ΔDA	3.239E-03	.002	.118	1.881	.061
	ΔDRI	9.937E-04	.000	.270	6.234	.000
	TPD	-1.413E-04	.001	-.071	-.234	.8150
	TAD	9.665E-03	.002	.295	6.083	.000
	ΔPA	1.551E-03	.002	.181	.747	.4561
	ΔDWI	6.327E-04	.000	.163	3.314	.001
	TDRI	1.743E-03	.000	.625	4.989	.0006
	TDWI	1.352E-03	.000	.473	3.925	.000
9	(Constant)	.960	.033		28.933	.000
	ΔDA	3.246E-03	.002	.118	1.889	.060
	ΔDRI	9.943E-04	.000	.270	6.248	.000
	TAD	9.804E-03	.001	.299	6.664	.000
	ΔPA	1.075E-03	.000	.125	2.576	.010
	ΔDWI	6.327E-04	.000	.163	3.320	.001
	TDRI	1.681E-03	.000	.603	7.333	.000
	TDWI	1.291E-03	.000	.452	5.757	.000

a. Dependent Variable: ENERGY

Table D.70: Coefficients^a

Model		95% Confidence Interval for $\hat{B}$		Correlations			Colinearity Statistics	
		Lower Bound	Upper Bound	Zero Order	Partial	Part	Tolerance	VIF
1	(Constant)	1.169	1.277					
	ΔDA	.012	.017	.514	.514	.514	1.000	1.000
2	(Constant)	1.144	1.245					
	ΔDA	.010	.015	.514	.502	.459	.978	1.023
	ΔDRI	.001	.002	.403	.385	.330	.978	1.023
3	(Constant)	1.051	1.175					
	ΔDA	.006	.012	.514	.313	.254	.620	1.612
	ΔDRI	.001	.001	.403	.351	.288	.942	1.062
	TPD	.000	.001	.509	.233	.184	.597	1.674
4	(Constant)	.944	1.083					
	ΔDA	.007	.013	.514	.365	.289	.602	1.661
	ΔDRI	.001	.001	.403	.373	.296	.941	1.063
	TPD	.000	.001	.509	.306	.237	.556	1.798
	TAD	.005	.011	-.072	.292	.225	.783	1.276
5	(Constant)	.795	.986					
	ΔDA	.005	.011	.514	.273	.205	.505	1.979
	ΔDRI	.001	.001	.403	.364	.282	.934	1.071
	TPD	.001	.003	.509	.277	.208	.042	23.602
	TAD	.007	.014	-.072	.345	.265	.663	1.508
	ΔPA	-.008	-.002	.439	-.202	-.148	.055	18.186
6	(Constant)	.798	.988					
	ΔDA	.004	.010	.514	.250	.184	.490	2.043
	ΔDRI	.001	.001	.403	.385	.298	.875	1.143
	TPD	.001	.003	.509	.274	.203	.042	23.647
	TAD	.007	.014	-.072	.347	.265	.663	1.508
	ΔPA	-.008	-.002	.439	-.201	-.147	.055	18.190
	ΔDWI	.000	.001	.186	.139	.100	.856	1.168
7	(Constant)	.809	.997					

	ΔDA	.002	.009	.514	.186	.133	.440	2.273
	ΔDRI	.001	.001	.403	.344	.257	.822	1.216
	TPD	.001	.002	.509	.228	.165	.040	25.178
	TAD	.007	.014	-.072	.345	.258	.661	1.512
	ΔPA	-.007	-.001	.439	-.156	-.111	.052	19.328
	ΔDWI	.000	.001	.186	.212	.152	.644	1.554
	TDRI	.000	.001	.385	.184	.132	.528	1.894
8	(Constant)	.871	1.066					
	ΔDA	.000	.007	.514	.107	.074	.391	2.558
	ΔDRI	.001	.001	.403	.336	.244	.817	1.224
	TPD	-.001	.001	.509	-.013	-.009	.017	59.250
	TAD	.007	.013	-.072	.328	.238	.652	1.533
	ΔPA	-.003	.006	.439	.043	.029	.026	38.151
	ΔDWI	.000	.001	.186	.186	.130	.632	1.583
	TDRI	.001	.002	.385	.274	.195	.098	10.226
	TDWI	.001	.002	.144	.219	.154	.105	9.484
9	(Constant)	.895	1.025					
	ΔDA	.000	.007	.514	.107	.074	.391	2.557
	ΔDRI	.001	.001	.403	.336	.244	.818	1.223
	TAD	.007	.013	-.072	.355	.261	.759	1.318
	ΔPA	.000	.002	.439	.145	.101	.647	1.545
	ΔDWI	.000	.001	.186	.186	.130	.632	1.583
	TDRI	.001	.002	.385	.386	.287	.226	4.417
	TDWI	.001	.002	.144	.312	.225	.248	4.030

a. Dependent Variable: ENERGY

Table D.71: Excluded Variables^j

Model		β in	t	Sig.	Partial Correlation	Colinearity Statistics		
						Tolerance	VIF	Minimum Tolerance
1	ΔPA	.244 ^a	4.503	.000	.247	.755	1.325	.755
	TPD	.311 ^a	5.255	.000	.285	.620	1.612	.620
	ΔPD	.209 ^a	4.047	.000	.223	.844	1.185	.844
	TDRI	.213 ^a	4.124	.000	.227	.838	1.193	.838
	ΔDRI	.334 ^a	7.366	.000	.385	.978	1.023	.978
	TDWI	.006 ^a	.122	.903	.007	.928	1.078	.928
	ΔDWI	.042 ^a	.821	.412	.046	.918	1.090	.918
	TAS	.153 ^a	2.954	.003	.165	.858	1.165	.858
	ΔAS	.093 ^a	1.867	.063	.105	.946	1.058	.946
	TAD	.152 ^a	2.927	.004	.163	.847	1.180	.847
2	ΔPA	.189 ^b	3.680	.000	.204	.736	1.359	.736
	TPD	.239 ^b	4.225	.000	.233	.597	1.674	.597
	ΔPD	.168 ^b	3.474	.001	.193	.831	1.203	.831
	TDRI	.098 ^b	1.888	.060	.106	.735	1.361	.735
	TDWI	.079 ^b	1.673	.095	.094	.889	1.125	.889
	ΔDWI	.130 ^b	2.719	.007	.152	.867	1.153	.867
	TAS	.182 ^b	3.837	.000	.213	.853	1.173	.848
	ΔAS	.109 ^b	2.377	.018	.134	.944	1.060	.929
	TAD	.183 ^b	3.837	.000	.213	.841	1.189	.838
3	ΔPA	-.190 ^c	-1.111	.267	-.063	6.498E-02	15.389	5.274E-02
	ΔPD	-.017 ^c	-.181	.856	-.010	.228	4.389	.164
	TDRI	.098 ^c	1.923	.055	.109	.735	1.361	.562
	TDWI	.018 ^c	.378	.706	.021	.798	1.253	.536
	ΔDWI	.109 ^c	2.327	.021	.131	.857	1.167	.590
	TAS	.248 ^c	5.293	.000	.288	.799	1.251	.560
	ΔAS	.140 ^c	3.124	.002	.175	.923	1.083	.584
	TAD	.254 ^c	5.367	.000	.292	.783	1.276	.556
	ΔPA	-.633 ^d	-3.617	.000	-.202	5.499E-02	18.186	4.237E-02
	ΔPD	-.258 ^d	-2.704	.007	-.152	.188	5.317	.126

4	TDRI	.109 ^d	2.245	.025	.127	.733	1.364	.549
	TDWI	.037 ^d	.787	.432	.045	.794	1.260	.508
	ΔDWI	.110 ^d	2.463	.014	.139	.857	1.167	.550
	TAS	.060 ^d	.279	.781	.016	3.793E-02	26.365	3.718E-02
	ΔAS	.018 ^d	.353	.724	.020	.650	1.538	.552
5	ΔPD	.500 ^e	1.870	.062	.106	2.333E-02	42.856	6.822E-03
	TDRI	.077 ^e	1.569	.118	.089	.703	1.423	4.075E-02
	TDWI	.006 ^e	.128	.898	.007	.766	1.306	3.950E-02
	ΔDWI	.108 ^e	2.457	.015	.139	.856	1.168	4.229E-02
	TAS	.026 ^e	.125	.901	.007	3.786E-02	26.417	3.651E-02
	ΔAS	.019 ^e	.371	.711	.021	.650	1.538	4.237E-02
6	ΔPD	.240 ^f	.789	.431	.045	1.789E-02	55.899	5.360E-03
	TDRI	.181 ^f	3.284	.001	.184	.528	1.894	3.972E-02
	TDWI	-.070 ^f	-1.294	.196	-.074	.569	1.756	3.907E-02
	TAS	.026 ^f	.126	.900	.007	3.786E-02	26.417	3.651E-02
	ΔAS	.019 ^f	.374	.709	.021	.650	1.538	4.229E-02
7	ΔPD	.390 ^g	1.290	.198	.074	1.752E-02	57.086	5.341E-03
	TDWI	.473 ^g	3.925	.000	.219	.105	9.484	1.688E-02
	TAS	.030 ^g	.146	.884	.008	3.785E-02	26.417	3.649E-02
	ΔAS	.019 ^g	.378	.706	.022	.650	1.538	3.972E-02
8	ΔPD	.503 ^h	1.696	.091	.097	1.737E-02	57.584	5.073E-03
	TAS	.041 ^h	.201	.841	.012	3.785E-02	26.422	1.685E-02
	ΔAS	.019 ^h	.384	.702	.022	.650	1.538	1.688E-02
9	TPD	-.071 ⁱ	-.234	.815	-.013	1.688E-02	59.250	1.688E-02
	ΔPD	.361 ⁱ	1.510	.132	.086	2.660E-02	37.593	2.202E-02
	TAS	.038 ⁱ	.191	.848	.011	3.792E-02	26.374	3.706E-02
	ΔAS	.019 ⁱ	.385	.701	.022	.650	1.538	.226

a. Predictors in the Model: (Constant), ΔDA

b. Predictors in the Model: (Constant), ΔDA , ΔDRI

c. Predictors in the Model: (Constant), ΔDA , ΔDRI , TPD

d. Predictors in the Model: (Constant), ΔDA , ΔDRI , TPD, TAD

- e. Predictors in the Model: (Constant), ΔDA , ΔDRI , TPD, TAD, ΔPA
- f. Predictors in the Model: (Constant), ΔDA , ΔDRI , TPD, TAD, ΔPA , ΔDWI
- g. Predictors in the Model: (Constant), ΔDA , ΔDRI , TPD, TAD, ΔPA , ΔDWI , TDRI
- h. Predictors in the Model: (Constant), ΔDA , ΔDRI , TPD, TAD, ΔPA , ΔDWI , TDRI, TDWI
- i. Predictors in the Model: (Constant), ΔDA , ΔDRI , TAD, ΔPA , ΔDWI , TDRI, TDWI
- j. Dependent Variable: ENERGY

Table D.72: Coefficient Correlations, part A^a

Model			ΔDA	ΔDRI	TPD	TAD
1	Correlations	ΔDA	1.000			
	Covariances	ΔDA	1.775E-06			
2	Correlations	ΔDA	1.000	-.149		
		ΔDRI	-.149	1.000		
	Covariances	ΔDA	1.551E-06	-3.095E-08		
		ΔDRI	-3.095E-08	2.781E-08		
3	Correlations	ΔDA	1.000	.000	-.605	
		ΔDRI	.000	1.000	-.193	
		TPD	-.605	-.193	1.000	
	Covariances	ΔDA	2.320E-06	-1.452E-12	-1.042E-07	
		ΔDRI	-1.452E-12	2.740E-08	-3.605E-09	
		TPD	-1.042E-07	-3.605E-09	1.279E-08	
4	Correlations	ΔDA	1.000	.006	-.530	.171
		ΔDRI	.006	1.000	-.177	.033
		TPD	-.530	-.177	1.000	.262
		TAD	.171	.033	.262	1.000
	Covariances	ΔDA	2.194E-06	1.308E-09	-8.817E-08	3.918E-07
		ΔDRI	1.308E-09	2.518E-08	-3.157E-09	8.015E-09
		TPD	-8.817E-08	-3.157E-09	1.261E-08	4.558E-08
		TAD	3.918E-07	8.015E-09	4.558E-08	2.398E-06
	Correlations	ΔDA	1.000	.039	-.520	-.013
		ΔDRI	.039	1.000	-.129	-.003
		TPD	-.520	-.129	1.000	.443
		TAD	-.013	-.003	.443	1.000
		ΔPA	.401	.083	-.961	-.392
5	Covariances	ΔDA	2.517E-06	9.543E-09	-3.290E-07	-3.515E-08
		ΔDRI	9.543E-09	2.441E-08	-8.032E-09	-7.152E-10
		TPD	-3.290E-07	-8.032E-09	1.593E-07	2.923E-07
		TAD	-3.515E-08	-7.152E-10	2.923E-07	2.727E-06
		ΔPA	9.563E-07	1.956E-08	-5.764E-07	-9.731E-07
	Correlations	ΔDA	1.000	-.008	-.503	-.013

6		ΔDRI	-.008	1.000	-.135	-.003
		TPD	-.503	-.135	1.000	.443
		TAD	-.013	-.003	.443	1.000
		ΔPA	.392	.084	-.961	-.392
		ΔDWI	-.176	.252	-.043	.000
	Covariances	ΔDA	2.555E-06	-1.957E-09	-3.189E-07	-3.436E-08
		ΔDRI	-1.957E-09	2.564E-08	-8.597E-09	-7.356E-10
		TPD	-3.189E-07	-8.597E-09	1.571E-07	2.877E-07
		TAD	-3.436E-08	-7.356E-10	2.877E-07	2.684E-06
		ΔPA	9.347E-07	2.014E-08	-5.676E-07	-9.575E-07
		ΔDWI	-4.797E-08	6.866E-09	-2.934E-09	-1.347E-10
7	Correlations	ΔDA	1.000	.071	-.384	.003
		ΔDRI	.071	1.000	-.067	.009
		TPD	-.384	-.067	1.000	.441
		TAD	.003	.009	.441	1.000
		ΔPA	.283	.020	-.963	-.392
		ΔDWI	-.303	.089	-.159	-.025
		TDRI	-.318	-.245	-.247	-.048
	Covariances	ΔDA	2.756E-06	1.915E-08	-2.566E-07	7.988E-09
		ΔDRI	1.915E-08	2.644E-08	-4.379E-09	2.397E-09
		TPD	-2.566E-07	-4.379E-09	1.621E-07	2.866E-07
		TAD	7.988E-09	2.397E-09	2.866E-07	2.607E-06
		ΔPA	7.119E-07	4.906E-09	-5.865E-07	-9.567E-07
		ΔDWI	-9.747E-08	2.815E-09	-1.242E-08	-7.662E-09
		TDRI	-8.134E-08	-6.126E-09	-1.527E-08	-1.202E-08
8	Correlations	ΔDA	1.000	.093	.017	.042
		ΔDRI	.093	1.000	.016	.018
		TPD	.017	.016	1.000	.374
		TAD	.042	.018	.374	1.000
		ΔPA	-.044	-.041	-.980	-.359
		ΔDWI	-.238	.099	.001	-.008
		TDRI	-.431	-.176	-.754	-.126
		TDWI	-.334	-.078	-.758	-.117
	Covariances	ΔDA	2.963E-06	2.548E-08	1.791E-08	1.145E-07

		ΔDRI	2.548E-08	2.541E-08	1.539E-09	4.615E-09
		TPD	1.791E-08	1.539E-09	3.643E-07	3.589E-07
		TAD	1.145E-07	4.615E-09	3.589E-07	2.525E-06
		ΔPA	-1.584E-07	-1.355E-08	-1.228E-06	-1.185E-06
		ΔDWI	-7.810E-08	3.015E-09	9.367E-11	-2.459E-09
		TDRI	-2.589E-07	-9.794E-09	-1.589E-07	-7.016E-08
		TDWI	-1.979E-07	-4.305E-09	-1.576E-07	-6.408E-08
9	Correlations	ΔDA	1.000	.093		.038
		ΔDRI	.093	1.000		.013
		TAD	.038	.013		1.000
		ΔPA	-.136	-.126		.040
		ΔDWI	-.238	.099		-.009
		TDRI	-.635	-.249		.255
		TDWI	-.492	-.102		.276
	Covariances	ΔDA	2.953E-06	2.532E-08		9.660E-08
		ΔDRI	2.532E-08	2.532E-08		3.090E-09
		TAD	9.660E-08	3.090E-09		2.165E-06
		ΔPA	-9.770E-08	-8.340E-09		2.433E-08
		ΔDWI	-7.786E-08	3.006E-09		-2.544E-09
		TDRI	-2.503E-07	-9.094E-09		8.615E-08
		TDWI	-1.896E-07	-3.628E-09		9.092E-08

a. Dependent Variable: ENERGY

Table D.73: Coefficient Correlations, part B^a

Model			ΔPA	ΔDWI	TDRI	TDWI
1	Correlations	ΔDA				
	Covariances	ΔDA				
2	Correlations	ΔDA ΔDRI				
	Covariances	ΔDA ΔDRI				
3	Correlations	ΔDA ΔDRI TPD				
	Covariances	ΔDA ΔDRI TPD				
4	Correlations	ΔDA ΔDRI TPD TAD				
	Covariances	ΔDA ΔDRI TPD TAD				
5	Correlations	ΔDA ΔDRI TPD TAD ΔPA	.401 .083 -.961 -.392 1.000			
	Covariances	ΔDA ΔDRI TPD TAD ΔPA	9.563E-07 1.956E-08 -5.764E-07 -9.731E-07 2.257E-06			
	Correlations	ΔDA	.392	-.176		

6		ΔDRI	.084	.252		
		TPD	-.961	-.043		
		TAD	-.392	.000		
		ΔPA	1.000	.015		
		ΔDWI	.015	1.000		
	Covariances	ΔDA	9.347E-07	-4.797E-08		
		ΔDRI	2.014E-08	6.866E-09		
TPD		-5.676E-07	-2.934E-09			
TAD		-9.575E-07	-1.347E-10			
ΔPA		2.222E-06	3.786E-09			
ΔDWI		3.786E-09	2.903E-08			
7	Correlations	ΔDA	.283	-.303	-.318	
		ΔDRI	.020	.089	-.245	
		TPD	-.963	-.159	-.247	
		TAD	-.392	-.025	-.048	
		ΔPA	1.000	.133	.243	
		ΔDWI	.133	1.000	.498	
		$TDRI$	.243	.498	1.000	
	Covariances	ΔDA	7.119E-07	-9.747E-08	-8.134E-08	
		ΔDRI	4.906E-09	2.815E-09	-6.126E-09	
		TPD	-5.865E-07	-1.242E-08	-1.527E-08	
		TAD	-9.567E-07	-7.662E-09	-1.202E-08	
		ΔPA	2.288E-06	3.906E-08	5.646E-08	
		ΔDWI	3.906E-08	3.743E-08	1.483E-08	
		$TDRI$	5.646E-08	1.483E-08	2.367E-08	
8	Correlations	ΔDA	-.044	-.238	-.431	-.334
		ΔDRI	-.041	.099	-.176	-.078
		TPD	-.980	.001	-.754	-.758
		TAD	-.359	-.008	-.126	-.117
		ΔPA	1.000	-.002	.708	.702
		ΔDWI	-.002	1.000	.089	-.137
		$TDRI$	.708	.089	1.000	.903
		$TDWI$	.702	-.137	.903	1.000
	Covariances	ΔDA	-1.584E-07	-7.810E-08	-2.589E-07	-1.979E-07

		ΔDRI	-1.355E-08	3.015E-09	-9.794E-09	-4.305E-09
		TPD	-1.228E-06	9.367E-11	-1.589E-07	-1.576E-07
		TAD	-1.185E-06	-2.459E-09	-7.016E-08	-6.408E-08
		ΔPA	4.314E-06	-7.974E-10	5.141E-07	5.024E-07
		ΔDWI	-7.974E-10	3.643E-08	5.933E-09	-8.994E-09
		$TDRI$	5.141E-07	5.933E-09	1.221E-07	1.086E-07
		$TDWI$	5.024E-07	-8.994E-09	1.086E-07	1.186E-07
9	Correlations	ΔDA	-.136	-.238	-.635	-.492
		ΔDRI	-.126	.099	-.249	-.102
		TAD	.040	-.009	.255	.276
		ΔPA	1.000	-.006	-.226	-.308
		ΔDWI	-.006	1.000	.136	-.209
		$TDRI$	-.226	.136	1.000	.773
		$TDWI$	-.308	-.209	.773	1.000
	Covariances	ΔDA	-9.770E-08	-7.786E-08	-2.503E-07	-1.896E-07
		ΔDRI	-8.340E-09	3.006E-09	-9.094E-09	-3.628E-09
		TAD	2.433E-08	-2.544E-09	8.615E-08	9.092E-08
		ΔPA	1.742E-07	-4.802E-10	-2.164E-08	-2.880E-08
		ΔDWI	-4.802E-10	3.632E-08	5.955E-09	-8.926E-09
		$TDRI$	-2.164E-08	5.955E-09	5.257E-08	3.972E-08
		$TDWI$	-2.880E-08	-8.926E-09	3.972E-08	5.024E-08

a. Dependent Variable: ENERGY

Table D.74: Colinearity Diagnostics^a

Model	Dim	Eigenvalue	Condition Index	Constant	Variance Properties								
					Δ	Δ	T	T	Δ	Δ	T	T	
					D	D	P	A	P	D	D	D	
					A	I	D	D	A	I	I	I	
1	1	1.789	1.000	.11	.11								
	2	.211	2.908	.89	.89								
2	1	2.086	1.000	.07	.07	.08							
	2	.703	1.723	.06	.06	.92							
	3	.211	3.141	.87	.87	.00							
3	1	2.963	1.000	.02	.02	.03	.01						
	2	.731	2.014	.02	.02	.95	.01						
	3	.211	3.743	.54	.56	.00	.00						
	4	9.482E-02	5.590	.42	.40	.02	.98						
4	1	2.981	1.000	.02	.02	.03	.01	.00					
	2	1.055	1.681	.01	.00	.04	.00	.60					
	3	.716	2.040	.01	.02	.91	.01	.04					
	4	.165	4.256	.34	.79	.01	.04	.20					
	5	8.400E-02	5.957	.63	.16	.01	.94	.15					
5	1	3.825	1.000	.00	.01	.02	.00	.00	.00				
	2	1.061	1.898	.00	.00	.02	.00	.51	.00				
	3	.735	2.282	.00	.01	.94	.00	.02	.00				
	4	.215	4.221	.03	.39	.00	.00	.01	.04				
	5	.158	4.926	.32	.35	.00	.00	.26	.01				
	6	6.469E-03	24.317	.63	.24	.01	1.00	.20	.95				
6	1	4.038	1.000	.00	.01	.01	.00	.00	.00	.01			
	2	1.065	1.947	.00	.00	.00	.00	.51	.00	.02			
	3	1.003	2.007	.00	.00	.37	.00	.01	.00	.35			
	4	.520	2.787	.00	.01	.61	.00	.02	.00	.58			
	5	.212	4.360	.04	.37	.00	.00	.01	.04	.02			

	6	.156	5.091	.31	.39	.00	.00	.25	.01	.02		
	7	6.463E-03	24.995	.63	.22	.01	1.00	.20	.95	.00		
7	1	4.555	1.000	.00	.01	.01	.00	.00	.00	.01	.01	
	2	1.175	1.969	.00	.00	.16	.00	.00	.00	.25	.05	
	3	1.065	2.068	.00	.00	.00	.00	.51	.00	.01	.00	
	4	.566	2.837	.00	.01	.79	.00	.01	.00	.18	.06	
	5	.328	3.725	.00	.04	.00	.00	.01	.02	.25	.35	
	6	.164	5.272	.35	.00	.01	.00	.23	.02	.04	.13	
	7	.141	5.684	.02	.82	.02	.00	.04	.00	.24	.36	
	8	6.102E-03	27.323	.62	.13	.00	1.00	.20	.96	.02	.06	
8	1	4.951	1.000	.00	.01	.01	.00	.00	.00	.01	.00	.00
	2	1.535	1.796	.00	.00	.08	.00	.00	.00	.11	.01	.01
	3	1.068	2.153	.00	.00	.01	.00	.51	.00	.00	.00	.00
	4	.592	2.893	.00	.01	.85	.00	.01	.00	.02	.01	.01
	5	.428	3.402	.00	.01	.02	.00	.01	.00	.71	.02	.03
	6	.214	4.813	.04	.12	.01	.00	.01	.02	.09	.00	.04
	7	.151	5.718	.24	.40	.00	.00	.23	.00	.06	.00	.00
	8	5.834E-02	9.212	.17	.46	.03	.00	.09	.00	.00	.37	.32
	9	2.665E-03	43.101	.54	.00	.00	1.00	.14	.97	.00	.58	.59
9	1	4.004	1.000	.01	.01	.01		.00	.01	.01	.00	.00
	2	1.535	1.615	.00	.00	.08		.00	.00	.11	.02	.03
	3	1.067	1.937	.01	.00	.01		.59	.00	.00	.00	.00
	4	.586	2.614	.00	.01	.82		.02	.00	.01	.03	.01
	5	.410	3.125	.01	.00	.03		.01	.04	.77	.03	.09
	6	.191	4.582	.09	.07	.01		.02	.92	.03	.00	.06
	7	.150	5.158	.52	.43	.00		.26	.00	.07	.00	.01
	8	5.784E-02	8.320	.35	.48	.03		.10	.02	.00	.91	.79

a. Dependent Variable: ENERGY

D.4.3 C6211 Data Reduction

D.4.3.1 C6211 Full

Table D.75: C6211 Data Reduction SPSS Output Notes

Output Created	19-FEB-2002 22:19:42	
Comments		
Input	Data	M:\school\thesis\code\c6x\c6xScaled.sav
	Filter	<none>
	Weight	<none>
	Split File	<none>
	Rows in Working Data File	174
Missing Value Handling	Definition of Missing	User-defined missing values are treated as missing.
	Cases Used	No missing values for any variable used.
Syntax		REGRESSION /DESCRIPTIVES MEAN STDDEV CORR SIG N /MISSING LISTWISE /STATISTICS COEFF OUTS CI BCOV R ANOVA COLLIN TOL CHANGE ZPP /CRITERIA=PIN(.05) POUT(.10) /NOORIGIN /DEPENDENT energy /METHOD=STEPWISE ΔPA TPD ΔPD ΔDA $TDRI$ ΔDRI $TDWI$ ΔDWI TAS ΔAS TAD .

Table D.76: Descriptive Statistics

	Mean	Std. Deviation	N
<i>ENERGY</i>	1.3273	.6832	174
ΔPA	46.0690	34.1143	174
<i>TPD</i>	926.8966	488.1242	174
ΔPD	116.7586	96.7914	174
ΔDA	12.5862	11.1987	174
<i>TDRI</i>	132.9655	224.5517	174
ΔDRI	62.3448	167.1208	174
<i>TDWI</i>	204.1379	250.3535	174
ΔDWI	73.9310	167.9459	174
<i>TAS</i>	22.0690	42.2264	174
<i>TAD</i>	11.0345	21.1132	174

Table D.77: Pearson Correlation, part A

	ENERGY	ΔPA	TPD	ΔPD	ΔDA	TDRI	ΔDRI	TDWI
<i>ENERGY</i>	1.000	.594	.697	.858	.569	-.144	-.016	.812
ΔPA	.594	1.000	.983	.808	.806	.587	.365	.423
<i>TPD</i>	.697	.983	1.000	.878	.818	.426	.264	.584
ΔPD	.858	.808	.878	1.000	.730	.111	.085	.748
ΔDA	.569	.806	.818	.730	1.000	.362	.271	.465
<i>TDRI</i>	-.144	.587	.426	.111	.362	1.000	.626	-.486
ΔDRI	-.016	.365	.264	.085	.271	.626	1.000	-.306
<i>TDWI</i>	.812	.423	.584	.748	.465	-.486	-.306	1.000
ΔDWI	.657	.233	.320	.685	.316	-.262	-.165	.544
<i>TAS</i>	-.616	-.710	-.722	-.634	-.591	-.311	-.196	-.429
<i>TAD</i>	-.616	-.710	-.722	-.634	-.591	-.311	-.196	-.429

Table D.78: Pearson Correlation, part B

	ΔDWI	TAS	TAD
<i>ENERGY</i>	.657	-.616	-.616
ΔPA	.233	-.710	-.710
<i>TPD</i>	.320	-.722	-.722
ΔPD	.685	-.634	-.634
ΔDA	.316	-.591	-.591
<i>TDRI</i>	-.262	-.311	-.311
ΔDRI	-.165	-.196	-.196
<i>TDWI</i>	.544	-.429	-.429
ΔDWI	1.000	-.231	-.231
<i>TAS</i>	-.231	1.000	1.000
<i>TAD</i>	-.231	1.000	1.000

Table D.79: Significance: 1-tailed, part A

	<i>ENERGY</i>	ΔPA	<i>TPD</i>	ΔPD	ΔDA	<i>TDRI</i>	ΔDRI
<i>ENERGY</i>	.	.000	.000	.000	.000	.029	.417
ΔPA	.000	.	.000	.000	.000	.000	.000
<i>TPD</i>	.000	.000	.	.000	.000	.000	.000
ΔPD	.000	.000	.000	.	.000	.073	.132
ΔDA	.000	.000	.000	.000	.	.000	.000
<i>TDRI</i>	.029	.000	.000	.073	.000	.	.000
ΔDRI	.417	.000	.000	.132	.000	.000	.
<i>TDWI</i>	.000	.000	.000	.000	.000	.000	.000
ΔDWI	.000	.001	.000	.000	.000	.000	.015
<i>TAS</i>	.000	.000	.000	.000	.000	.000	.005
<i>TAD</i>	.000	.000	.000	.000	.000	.000	.005

Table D.80: Significance: 1-tailed, part B

	<i>TDWI</i>	ΔDWI	<i>TAS</i>	<i>TAD</i>
<i>ENERGY</i>	.000	.000	.000	.000
ΔPA	.000	.001	.000	.000
<i>TPD</i>	.000	.000	.000	.000
ΔPD	.000	.000	.000	.000
ΔDA	.000	.000	.000	.000
<i>TDRI</i>	.000	.000	.000	.000
ΔDRI	.000	.015	.005	.005
<i>TDWI</i>	.	.000	.000	.000
ΔDWI	.000	.	.001	.001
<i>TAS</i>	.000	.001	.	.000
<i>TAD</i>	.000	.001	.000	.

Table D.81: Number of data points, part A

	<i>ENERGY</i>	ΔPA	<i>TPD</i>	ΔPD	ΔDA	<i>TDRI</i>	ΔDRI
<i>ENERGY</i>	174	174	174	174	174	174	174
ΔPA	174	174	174	174	174	174	174
<i>TPD</i>	174	174	174	174	174	174	174
ΔPD	174	174	174	174	174	174	174
ΔDA	174	174	174	174	174	174	174
<i>TDRI</i>	174	174	174	174	174	174	174
ΔDRI	174	174	174	174	174	174	174
<i>TDWI</i>	174	174	174	174	174	174	174
ΔDWI	174	174	174	174	174	174	174
<i>TAS</i>	174	174	174	174	174	174	174
<i>TAD</i>	174	174	174	174	174	174	174

Table D.82: Number of data points, part B

	<i>TDWI</i>	ΔDWI	<i>TAS</i>	<i>TAD</i>
<i>ENERGY</i>	174	174	174	174
ΔPA	174	174	174	174
<i>TPD</i>	174	174	174	174
ΔPD	174	174	174	174
ΔDA	174	174	174	174
<i>TDRI</i>	174	174	174	174
ΔDRI	174	174	174	174
<i>TDWI</i>	174	174	174	174
ΔDWI	174	174	174	174
<i>TAS</i>	174	174	174	174
<i>TAD</i>	174	174	174	174

Table D.83: Variables Entered or Removed^a

Model	Variables		Method
	Entered	Removed	
1	ΔPD	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
2	TDWI	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
3	TAS	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
4	ΔDWI	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).
5	ΔDRI	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).

a. Dependent Variable: ENERGY

Table D.84: Model Summary

Model	R	R^2	Adjusted R^2	Std. Error of the Estimate	Change Statistics				
					R^2 Change	F Change	d of f		Sig. F Change
1	.858 ^a	.736	.734	.3520	.736	479.583	1	172	.000
2	.896 ^b	.802	.800	.3058	.066	56.920	1	171	.000
3	.903 ^c	.816	.812	.2959	.014	12.643	1	170	.000
4	.912 ^d	.832	.828	.2832	.016	16.531	1	169	.000
5	.916 ^e	.840	.835	.2775	.008	8.111	1	168	.005

a. Predictors: (Constant), ΔPD

b. Predictors: (Constant), ΔPD , TDWI

c. Predictors: (Constant), ΔPD , TDWI, TAS

d. Predictors: (Constant), ΔPD , TDWI, TAS, ΔDWI

e. Predictors: (Constant), ΔPD , TDWI, TAS, ΔDWI , ΔDRI

Table D.85: ANOVA^f

	Model	Σ of Squares	df	Mean Square	F	Sig.
1	Regression	59.430	1	59.430	479.583	.000 ^a
	Residual	21.314	172	.124		
	Total	80.744	173			
2	Regression	64.753	2	32.376	346.213	.000 ^b
	Residual	15.991	171	9.352E-02		
	Total	80.744	173			
3	Regression	65.860	3	21.953	250.738	.000 ^c
	Residual	14.884	170	8.755E-02		
	Total	80.744	173			
4	Regression	67.186	4	16.797	209.367	.000 ^d
	Residual	13.558	169	8.023E-02		
	Total	80.744	173			
5	Regression	67.810	5	13.562	176.163	.000 ^e
	Residual	12.934	168	7.699E-02		
	Total	80.744	173			

a. Predictors: (Constant), ΔPD

b. Predictors: (Constant), ΔPD , TDWI

c. Predictors: (Constant), ΔPD , TDWI, TAS

d. Predictors: (Constant), ΔPD , TDWI, TAS, ΔDWI

e. Predictors: (Constant), ΔPD , TDWI, TAS, ΔDWI , ΔDRI

f. Dependent Variable: ENERGY

Table D.86: Coefficients, part A^a

Model		Unstandardized Coefficients		Standardized Coefficients β	t	Sig.
		$\hat{B}$	Std. Error			
1	(Constant)	.620	.042		14.807	.000
	ΔPD	6.055E-03	.000	.858	21.899	.000
2	(Constant)	.643	.037		17.616	.000
	ΔPD	4.012E-03	.000	.568	11.082	.000
	TDWI	1.056E-03	.000	.387	7.545	.000
3	(Constant)	.778	.052		15.019	.000
	ΔPD	3.248E-03	.000	.460	7.904	.000
	TDWI	1.099E-03	.000	.403	8.083	.000
	TAS	-2.460E-03	.001	-.152	-3.556	.000
4	(Constant)	.883	.056		15.793	.000
	ΔPD	2.091E-03	.000	.296	4.306	.000
	TDWI	1.080E-03	.000	.396	8.288	.000
	TAS	-3.486E-03	.001	-.215	-4.919	.000
	ΔDWI	7.672E-04	.000	.189	4.066	.000
5	(Constant)	.872	.055		15.891	.000
	ΔPD	1.380E-03	.001	.195	2.568	.011
	TDWI	1.328E-03	.000	.487	8.592	.000
	TAS	-3.382E-03	.001	-.209	-4.865	.000
	ΔDWI	9.279E-04	.000	.228	4.801	.000
	ΔDRI	4.610E-04	.000	.113	2.848	.005

a. Dependent Variable: ENERGY

Table D.87: Coefficients, part B^a

Model		95% Confidence Interval for $\hat{B}$		Correlations			Colinearity Statistics	
		Lower Bound	Upper Bound	Zero Order	Partial	Part	Tolerance	VIF
1	(Constant)	.538	.703					
	ΔPD	.006	.007	.858	.858	.858	1.000	1.000
2	(Constant)	.571	.715					
	ΔPD	.003	.005	.858	.647	.377	.440	2.271
	TDWI	.001	.001	.812	.500	.257	.440	2.271
3	(Constant)	.676	.880					
	ΔPD	.002	.004	.858	.518	.260	.320	3.126
	TDWI	.001	.001	.812	.527	.266	.437	2.290
	TAS	-.004	-.001	-.616	-.263	-.117	.593	1.686
4	(Constant)	.773	.993					
	ΔPD	.001	.003	.858	.314	.136	.210	4.763
	TDWI	.001	.001	.812	.538	.261	.436	2.293
	TAS	-.005	-.002	-.616	-.354	-.155	.518	1.931
	ΔDWI	.000	.001	.657	.298	.128	.462	2.166
5	(Constant)	.764	.981					
	ΔPD	.000	.002	.858	.194	.079	.165	6.075
	TDWI	.001	.002	.812	.553	.265	.297	3.365
	TAS	-.005	-.002	-.616	-.351	-.150	.517	1.936
	ΔDWI	.001	.001	.657	.347	.148	.422	2.367
	ΔDRI	.000	.001	-.016	.215	.088	.608	1.644

a. Dependent Variable: ENERGY

Table D.88: Excluded Variables^f

Model		β in	t	Sig.	Partial Correlation	Colinearity Statistics		
						Tolerance	VIF	Minimum Tolerance
1	ΔPA	-.287 ^a	-4.551	.000	-.329	.347	2.885	.347
	TPD	-.243 ^a	-3.047	.003	-.227	.230	4.353	.230
	ΔDA	-.123 ^a	-2.169	.031	-.164	.467	2.143	.467
	TDRI	-.242 ^a	-6.930	.000	-.468	.988	1.012	.988
	ΔDRI	-.090 ^a	-2.311	.022	-.174	.993	1.007	.993
	TDWI	.387 ^a	7.545	.000	.500	.440	2.271	.440
	ΔDWI	.130 ^a	2.460	.015	.185	.531	1.883	.531
	TAS	-.121 ^a	-2.426	.016	-.182	.598	1.672	.598
	TAD	-.121 ^a	-2.426	.016	-.182	.598	1.672	.598
2	ΔPA	-.107 ^b	-1.645	.102	-.125	.271	3.684	.146
	TPD	-.127 ^b	-1.758	.081	-.134	.218	4.593	.145
	ΔDA	-.058 ^b	-1.139	.256	-.087	.452	2.214	.254
	TDRI	-.075 ^b	-1.114	.267	-.085	.253	3.945	.113
	ΔDRI	.079 ^b	1.929	.055	.146	.683	1.464	.303
	ΔDWI	.107 ^b	2.323	.021	.175	.529	1.891	.331
	TAS	-.152 ^b	-3.556	.000	-.263	.593	1.686	.320
	TAD	-.152 ^b	-3.556	.000	-.263	.593	1.686	.320
3	ΔPA	-.258 ^c	-3.794	.000	-.280	.218	4.592	.145
	TPD	-.295 ^c	-3.913	.000	-.288	.175	5.701	.145
	ΔDA	-.103 ^c	-2.070	.040	-.157	.428	2.338	.230
	TDRI	-.236 ^c	-3.281	.001	-.245	.198	5.057	9.696E-02
	ΔDRI	.057 ^c	1.422	.157	.109	.665	1.504	.274
	ΔDWI	.189 ^c	4.066	.000	.298	.462	2.166	.210
	TAD	. ^c	.	.	.	.000	.	.000
4	ΔPA	-.111 ^d	-1.029	.305	-.079	8.517E-02	11.741	3.734E-02
	TPD	-.145 ^d	-1.206	.229	-.093	6.858E-02	14.582	3.732E-02
	ΔDA	-.047 ^d	-.925	.356	-.071	.386	2.588	.137
	TDRI	-.030 ^d	-.265	.792	-.020	7.774E-02	12.863	3.764E-02
	ΔDRI	.113 ^d	2.848	.005	.215	.608	1.644	.165
	TAD	. ^d	.	.	.	.000	.	.000
	ΔPA	-.147 ^e	-1.388	.167	-.107	8.407E-02	11.895	3.682E-02

5	TPD	$-.185^e$	-1.569	.118	-.121	6.770E-02	14.772	3.679E-02
	ΔDA	$-.069^e$	-1.383	.169	-.106	.378	2.645	.123
	TDRI	$-.067^e$	-.603	.547	-.047	7.670E-02	13.037	3.712E-02
	TAD	$.^e$	.	.	.	.000	.	.000

- Predictors in the Model: (Constant), ΔPD
- Predictors in the Model: (Constant), ΔPD , TDWI
- Predictors in the Model: (Constant), ΔPD , TDWI, TAS
- Predictors in the Model: (Constant), ΔPD , TDWI, TAS, ΔDWI
- Predictors in the Model: (Constant), ΔPD , TDWI, TAS, ΔDWI , ΔDRI
- Dependent Variable: ENERGY

Table D.89: Coefficient Correlations^a

Model			ΔPD	TDWI	TAS	ΔDWI	ΔDRI
1	Correlations	ΔPD	1.000				
	Covariances	ΔPD	7.646E-08				
2	Correlations	ΔPD	1.000	-.748			
		TDWI	-.748	1.000			
	Covariances	ΔPD	1.311E-07	-3.791E-08			
		TDWI	-3.791E-08	1.959E-08			
3	Correlations	ΔPD	1.000	-.682	.523		
		TDWI	-.682	1.000	-.089		
		TAS	.523	-.089	1.000		
	Covariances	ΔPD	1.689E-07	-3.810E-08	1.486E-07		
		TDWI	-3.810E-08	1.849E-08	-8.395E-09		
		TAS	1.486E-07	-8.395E-09	4.785E-07		
4	Correlations	ΔPD	1.000	-.530	.605	-.586	
		TDWI	-.530	1.000	-.070	-.037	
		TAS	.605	-.070	1.000	-.356	
		ΔDWI	-.586	-.037	-.356	1.000	
	Covariances	ΔPD	2.358E-07	-3.354E-08	2.080E-07	-5.371E-08	
		TDWI	-3.354E-08	1.696E-08	-6.475E-09	-9.099E-10	
		TAS	2.080E-07	-6.475E-09	5.021E-07	-4.761E-08	
		ΔDWI	-5.371E-08	-9.099E-10	-4.761E-08	3.561E-08	
5	Correlations	ΔPD	1.000	-.650	.510	-.632	-.465
		TDWI	-.650	1.000	-.028	.136	.564
		TAS	.510	-.028	1.000	-.325	.052
		ΔDWI	-.632	.136	-.325	1.000	.292
		ΔDRI	-.465	.564	.052	.292	1.000
	Covariances	ΔPD	2.886E-07	-5.396E-08	1.905E-07	-6.563E-08	-4.041E-08
		TDWI	-5.396E-08	2.389E-08	-3.035E-09	4.049E-09	1.412E-08
		TAS	1.905E-07	-3.035E-09	4.832E-07	-4.364E-08	5.898E-09
		ΔDWI	-6.563E-08	4.049E-09	-4.364E-08	3.735E-08	9.133E-09
		ΔDRI	-4.041E-08	1.412E-08	5.898E-09	9.133E-09	2.620E-08

a. Dependent Variable: ENERGY

Table D.90: Colinearity Diagnostics^a

M	D	Eigenvalue	Condition Index	Variance Properties					
				C	Δ	T	T	Δ	Δ
o	i			n	P	D	A	D	D
d	m			s	D	I	S	I	P
e				t					
l									
1	1	1.771	1.000	.11	.11				
	2	.229	2.779	.89	.89				
2	1	2.511	1.000	.05	.03	.03			
	2	.376	2.585	.66	.01	.26			
	3	.114	4.702	.29	.96	.71			
3	1	2.555	1.000	.02	.02	.03	.01		
	2	1.143	1.495	.01	.01	.02	.32		
	3	.228	3.348	.27	.03	.58	.24		
	4	7.492E-02	5.839	.69	.95	.37	.43		
4	1	3.052	1.000	.01	.01	.02	.00	.02	
	2	1.209	1.589	.02	.00	.01	.25	.02	
	3	.473	2.539	.05	.01	.02	.10	.54	
	4	.215	3.771	.14	.03	.76	.13	.06	
	5	5.139E-02	7.706	.79	.96	.18	.52	.36	
5	1	3.108	1.000	.01	.01	.01	.00	.02	.00
	2	1.249	1.577	.02	.00	.01	.17	.03	.07
	3	1.037	1.732	.00	.00	.00	.11	.01	.34
	4	.425	2.705	.03	.00	.07	.05	.52	.07
	5	.137	4.760	.40	.01	.52	.32	.02	.34
	6	4.380E-02	8.425	.54	.98	.38	.35	.40	.18

a. Dependent Variable: ENERGY

D.4.3.2 C6211 ALU Only

Table D.91: C62 ALU only Data Reduction SPSS Output Notes

Output Created	08-AUG-2004 13:05:46	
Comments		
Input	Filter	<none>
	Weight	<none>
	Split File	<none>
	Rows in Working Data File	51
Missing Value Handling	Definition of Missing	User-defined missing values are treated as missing.
	Cases Used	No missing values for any variable used.
Syntax		REGRESSION /DESCRIPTIVES MEAN STDDEV CORR SIG N /MISSING LISTWISE /STATISTICS COEFF OUTS CI BCOV R ANOVA COLLIN TOL CHANGE ZPP /CRITERIA=PIN(.05) POUT(.10) /NOORIGIN /DEPENDENT energy /METHOD=STEPWISE TAS

Table D.92: Descriptive Statistics

	Mean	Std. Deviation	N
ENERGY	.4937216	.12409064	51
TAS	75.2941	45.62600	51

Table D.93: Pearson Correlation

	ENERGY	TAS
ENERGY	1.000	.439
TAS	.439	1.000

Table D.94: Significance: 1-tailed

	ENERGY	TAS
ENERGY	.	.001
TAS	.001	.

Table D.95: Number of data points

	ENERGY	TAS
ENERGY	51	51
TAS	51	51

Table D.96: Variables Entered or Removed^a

Model	Variables		Method
	Entered	Removed	
1	TAS	.	Stepwise (F prob to enter $\leq .050$, F prob to remove $\geq .100$).

a. Dependent Variable: ENERGY

Table D.97: Model Summary

Model	R	R^2	Adjusted R^2	Std. Error of the Estimate	Change Statistics				
					R^2 Change	F Change	d of f		Sig. F Change
							1	2	
1	.439 ^a	.193	.176	.112641	.193	11.681	1	49	.001

a. Predictors: (Constant), TAS

Table D.98: ANOVA^b

Model		Σ of Squares	df	Mean Square	F	Sig.
1	Model	.148	1	.148	11.681	.001 ^a
	Residual	.622	49	.013		
	Total	.770	50			

a. Predictors: (Constant), TAS

b. Dependent Variable: ENERGY

Table D.99: Coefficients^a

Model		Unstandardized Coefficients		Standardized Coefficients β	t	Sig.
		$\hat{B}$	Std. Error			
1	(Constant)	.404	.031		13.174	.000
	TAS	.001	.000	.439	3.418	.001

a. Dependent Variable: ENERGY

Table D.100: Coefficients^a

Model		95% Confidence Interval for $\hat{B}$		Correlations			Colinearity Statistics	
		Lower Bound	Upper Bound	Zero Order	Partial	Part	Tolerance	VIF
1	(Constant)	.342	.465					
	TAS	.000	.002	.439	.439	.439	1.000	1.000

a. Dependent Variable: ENERGY

Table D.101: Coefficient Correlations^a

Model			TAS
1	Correlations	TAS	1.000
	Covariances	TAS	1.219E-07

a. Dependent Variable: ENERGY

Table D.102: Colinearity Diagnostics^a

Model	Dim.	Eigenvalue	Condition Index	Variance Properties	
				Const.	TAS
1	1	1.789	1.000	.11	.11
	2	.211	2.908	.89	.89

a. Dependent Variable: ENERGY