LEABHARLANN CHOLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH | TRINITY COLLEGE LIBRARY DUBLIN
Ollscoil Átha Cliath | The University of Dublin

Terms and Conditions of Use of Digitised Theses from Trinity College Library Dublin

Copyright statement

All material supplied by Trinity College Library is protected by copyright (under the Copyright and Related Rights Act, 2000 as amended) and other relevant Intellectual Property Rights. By accessing and using a Digitised Thesis from Trinity College Library you acknowledge that all Intellectual Property Rights in any Works supplied are the sole and exclusive property of the copyright and/or other IPR holder. Specific copyright holders may not be explicitly identified. Use of materials from other sources within a thesis should not be construed as a claim over them.

A non-exclusive, non-transferable licence is hereby granted to those using or reproducing, in whole or in part, the material for valid purposes, providing the copyright owners are acknowledged using the normal conventions. Where specific permission to use material is required, this is identified and such permission must be sought from the copyright holder or agency cited.

Liability statement

By using a Digitised Thesis, I accept that Trinity College Dublin bears no legal responsibility for the accuracy, legality or comprehensiveness of materials contained within the thesis, and that Trinity College Dublin accepts no liability for indirect, consequential, or incidental, damages or losses arising from use of the thesis for whatever reason. Information located in a thesis may be subject to specific use constraints, details of which may not be explicitly described. It is the responsibility of potential and actual users to be aware of such constraints and to abide by them. By making use of material from a digitised thesis, you accept these copyright and disclaimer provisions. Where it is brought to the attention of Trinity College Library that there may be a breach of copyright or other restraint, it is the policy to withdraw or take down access to a thesis while the issue is being resolved.

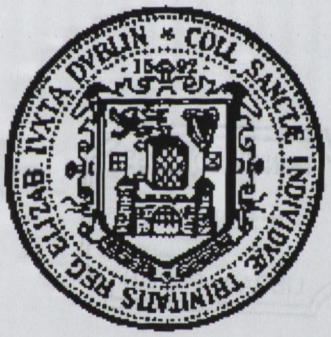
Access Agreement

By using a Digitised Thesis from Trinity College Library you are bound by the following Terms & Conditions. Please read them carefully.

I have read and I understand the following statement: All material supplied via a Digitised Thesis from Trinity College Library is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of a thesis is not permitted, except that material may be duplicated by you for your research use or for educational purposes in electronic or print form providing the copyright owners are acknowledged using the normal conventions. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

CMOS Power Estimation

Paul Comiskey



Thesis submitted for the degree of Ph.D.

May 2003

Department of Electronic and Electrical Engineering

University of Dublin

Ireland

TRINITY COLLEGE
0 3 JUL 2003
LIBRARY DUBLIN
THESIS
7441
7351

I, Paul Comiskey, declare that this thesis is entirely my own work, except where otherwise accredited, and that it has not been submitted for a Degree to any other University or Institution.

I further agree that the Library may lend or copy this thesis upon request:

Signed: Paul Comiskey

Date: 26 May 2003

Acknowledgements

I would like to thank my supervisor Dr. J.B. Foley for all his help and encouragement during this thesis.

I would also like to thank Robbie Dempsey for his invaluable assistance on many occasions.

My good friend and colleague Dr. Andreas Schwarzbacher has helped me with very many aspects of this work, always with good humour, insight and tea. It is fair to say that without his friendship and support there would be no thesis.

Lastly I want to acknowledge the very great contribution made by my wife Annette who has shown great patience and understanding during the course of this work.

Abstract

The last decade has seen the inclusion of power consumption criteria in the list of design goals targeted by silicon designers and architects. Together with speed and area, low power is now a common part of a system specification. In tandem with this new technical challenge, market forces have driven electronic product development increasingly in the direction of portability. This consideration, in conjunction with a growing awareness of the environmental impact of increasing global energy consumption, has led to a substantial body of research devoted to minimising the energy consumption of electronic systems. This is a relatively new field of research within which there are many unexplored challenges, none more so than in the area of prediction of the likely operational power dissipation of the system. Such techniques must be capable of producing efficient, accurate estimates in the shortest possible time.

It is the objective of this thesis to explore the problem of power estimation, with particular emphasis on processor systems. The focus of this work is on the input data sequence applied to the simulation model of the system. The power dissipation is a function of this sequence and it is demonstrated that by extracting only those parameters in the input sequence that relate to the power performance of the processor, the input sequence may be compacted when compared to the original data sequence. Additionally, development times are reduced by providing a statistical model of the input code, which, while containing valid instructions represents the power of the system rather than its functionality. The concepts of a Markov model of the instruction stream and a probabilistic measure of signal activity are combined to successfully represent the processor input data with a reduced data set, allowing speedier power estimates.

The techniques developed in this thesis are incorporated into a set of power estimation tools and utilities that are extensively verified with a complete processor design, in addition to a large set of typical arithmetic and logical submodules. A novel data generation technique for random inputs is also presented. This enables reference levels of power dissipation to be accurately obtained. By employing a genetic algorithm to optimise the generator properties, high quality repeatable sources of data are produced for the first time.

Table of Contents

ACKNOWLEDGEMENTS ..... **II**

ABSTRACT ..... **III**

TABLE OF CONTENTS ..... **IV**

LIST OF FIGURES ..... **VI**

LIST OF TABLES ..... **VIII**

1 INTRODUCTION **1**

1.1 ADAPTABILITY: A TOP-DOWN APPROACH 3
1.2 EFFICIENCY: FASTER POWER ESTIMATION 5
1.3 RELEVANCE: COMBINING CONCEPTS 6
1.4 SUMMARY 6

2 POWER CONSUMPTION, DISSIPATION AND ESTIMATION **7**

2.1 POWER DISSIPATION IN CMOS DEVICES 8
2.1.1 Consumption and Dissipation in a CMOS Inverter *9*
2.1.2 Estimation of the Correct Power Quantity *14*
2.2 POWER ESTIMATION IN CMOS CIRCUITS 14
2.3 SOFTWARE POWER ESTIMATION 15
2.4 BEHAVIOURAL POWER ESTIMATION 19
2.5 REGISTER TRANSFER LEVEL POWER ESTIMATION 23
2.6 GATE LEVEL POWER ANALYSIS 26
2.7 CIRCUIT LEVEL POWER ESTIMATION 34
2.8 SUMMARY AND CONCLUSION 36

3 RANDOM BINARY VECTORS **38**

3.1 CHARACTERISTICS AND VISUALISATION OF RANDOM BINARY DATA 40
3.2 THE LINEAR CONGRUENTIAL EQUATION 44
3.3 THE SPECTRAL TEST 46
3.3.1 Visualisation and Accuracy *46*
3.3.2 Implementing The Spectral Test *50*
3.3.3 Sample Results of the Spectral Test *52*
3.3.4 Implementation Issues with the Spectral Test. *53*
3.4 RANDOM BINARY VECTOR GENERATION USING GENETIC OPTIMISATION 59
3.4.1 Genetic Algorithms *59*
3.4.2 Implementation of the Genetic Algorithm *62*
3.4.3 Results from GA Operation *64*
3.5 TEST AND ANALYSIS OF GENERATOR OUTPUT 67
3.5.1 Test 1: Analysis of Histogram *68*
3.5.2 Test 2: Power Spectral Density *72*
3.5.3 Test 3: CDF Test of Power Spectral Density *79*
3.5.4 Test 4: A Novel Test *89*
3.6 SUMMARY AND CONCLUSIONS 93

4 DEVELOPMENT OF A POWER ESTIMATION TOOL **95**

4.1 LLAMA, A POWER ESTIMATION TOOL 95
4.2 OPERATION OF LLAMA 101
4.3 USE OF GENETICALLY-OPTIMISED GENERATORS 106
4.4 OPEN-LOOP MONTE CARLO ESTIMATION 108
4.5 TWO OBSERVATIONS ON POWER ESTIMATION 108
4.6 SUMMARY AND CONCLUSIONS 111

5 MODELLING REAL DATA WITH MARKOV CHAINS AND TRANSITION PROBABILITY 113

5.1 INPUT DATA MODELLING FOR COMBINATIONAL MODULES 114
5.2 MODELLING TEMPORAL AND SPATIAL CORRELATION 118
5.3 SEQUENTIAL CIRCUIT DATA MODELLING USING MARKOV CHAINS 122
5.4 POWER ESTIMATION IN COMBINATIONAL MODULES 125
5.5 POWER ESTIMATION IN SEQUENTIAL MODULES 128
5.6 SUMMARY AND CONCLUSIONS 134

6 POWER ANALYSIS IN A RISC PROCESSOR: A CASE STUDY 136

6.1 RISC PROCESSOR DESIGN 136
6.1.1 Single Cycle Processor Design 137
6.1.2 Writing an Application 140
6.2 SYNTHESISING THE PROCESSOR 148
6.2.1 Control Unit 149
6.2.2 Register File 150
6.2.3 ALU 150
6.2.4 Multiplier 150
6.2.5 Datapath 151
6.3 POWER ESTIMATION IN COMPLEX SYSTEMS 152
6.3.1 Submodule Power Analysis 154
6.3.2 Modelling Control Signals 158
6.4 SYSTEM-LEVEL POWER ESTIMATION WITH A COMBINATION MODEL 159
6.5 SUMMARY AND CONCLUSIONS 166

7 CONCLUSIONS 167

7.1 SPECIFIC CONCLUSIONS 167
7.2 GENERAL CONCLUSIONS 171
7.3 FUTURE WORK 172

8 REFERENCES 174

9 AUTHORS PUBLICATIONS 193

List of Figures

FIGURE 2.1: THE CMOS INVERTER 10
FIGURE 2.2: INVERTER CURRENT AND VOLTAGE WAVEFORMS 13
FIGURE 2.3: TRANSITION ACTIVITY IN A SAMPLE OF 16-BIT DATA 24
FIGURE 2.4: ALTERNATIVES FOR POWER ESTIMATION 28
FIGURE 2.5: EXAMPLE OF A BINARY DECISION DIAGRAM 31
FIGURE 2.6: MONTE CARLO POWER ESTIMATION 33
FIGURE 2.7: AVERAGE AND INSTANTANEOUS CURRENT WAVEFORMS 35
FIGURE 3.1: THE STRUCTURE OF THIS CHAPTER 40
FIGURE 3.2: GENERATORS *X* AND Y OUTPUT 42
FIGURE 3.3: GENERATORS *X* AND *Y* OUTPUT PLOT IN TWO DIMENSIONS 42
FIGURE 3.4: GENERATORS *X* AND *Y* OUTPUT PLOT IN THREE DIMENSIONS 43
FIGURE 3.5: TRUNCATED OUTPUT OF GENERATOR *Y* IN THREE DIMENSIONS 43
FIGURE 3.6: VARIATION IN POWER ESTIMATES WITH GENERATOR MULTIPLIER 45
FIGURE 3.7: EXAMPLE OF TWO DIMENSIONAL GENERATOR ACCURACY 48
FIGURE 3.8: FLOWCHART FOR ARRAY MULTIPLICATION 56
FIGURE 3.9: FLOWCHART FOR THE GENETIC ALGORITHM 60
FIGURE 3.10: GA OPERATIONS 61
FIGURE 3.11: APPLICATION OF SPECTRAL TEST TO A VARIETY OF GENERATORS 62
FIGURE 3.12: IMPROVEMENT IN SPECTRAL SCORE 64
FIGURE 3.13: GENETIC OPTIMISATION OF SPECTRAL TEST SCORE 65
FIGURE 3.14: SELECTION OF GENERATOR PARAMETER BY GENETIC OPTIMISATION 65
FIGURE 3.15: HISTOGRAM OF GENETICALLY IMPROVED SPECTRAL SCORES 66
FIGURE 3.16: HISTOGRAM OF SCORES WITHOUT OPTIMISATION 66
FIGURE 3.17: VARIATION IN PSD ESTIMATE WITH OVERLAP 77
FIGURE 3.18: PSD OF SAMPLES WITH THE LOWEST AND HIGHEST VARIANCE 78
FIGURE 3.19: RELATIONSHIP BETWEEN VARIANCE AND SPECTRAL SCORE 79
FIGURE 3.20: THE KOLMOGAROV-SMIRNOV TEST 82
FIGURE 3.21: KS TEST RESULTS 84
FIGURE 3.22: PSD FROM OPTIMISED GENERATOR 85
FIGURE 3.23: EMPIRICAL AND THEORETICAL CDF OF OPTIMISED GENERATOR 86
FIGURE 3.24: PERIODOGRAM OF POOR QUALITY GENERATOR 87
FIGURE 3.25: CDF OF PERIODOGRAM OF POOR QUALITY GENERATOR 87
FIGURE 3.26: RELATIONSHIP BETWEEN SPECTRAL SCORE AND KS TEST 89
FIGURE 3.27: KS TEST RESULTS FOR FILTERED DATA 93
FIGURE 4.1: MONTE CARLO POWER ESTIMATION 96
FIGURE 4.2: DISTRIBUTION OF NODE TRANSITIONS FOR NODE *N*1 97
FIGURE 4.3: NORMAL SCORES PLOT FOR NODE *N*1 98
FIGURE 4.4: HISTOGRAM OF NODE TRANSITIONS FOR NODE *N*2 98
FIGURE 4.5: NORMAL SCORES PLOT FOR NODE *N*2 98
FIGURE 4.6: SIMULATION TIME AS A FUNCTION OF INPUT SAMPLE SIZE 100
FIGURE 4.7: NUMBER OF ITERATIONS OF MC LOOP FOR CONVERGENCE 101
FIGURE 4.8: OPERATION OF LLAMA 102
FIGURE 4.9: TYPICAL MONTE CARLO SIMULATION RUN 103
FIGURE 4.10: VARIATION OF AVERAGE POWER AND ERROR DURING SIMULATION 104
FIGURE 4.11: EFFECT OF GENERATOR QUALITY ON POWER ESTIMATES 106

FIGURE 4.12: IMPROVED PERFORMANCE OF OPTIMISED GENERATORS 107
FIGURE 4.13: COMPARISON OF TIME PERFORMANCE OF GENERATORS 107
FIGURE 4.14: ENERGY DISTRIBUTION OF AN 8-BIT ADDER 109
FIGURE 4.15: TRAJECTORY OF CONFIDENCE INTERVAL AND ERROR FOR A 4-BIT ADDER 110
FIGURE 4.16: TRAJECTORY OF CI AND ERROR FOR A 16-BIT ADDER 110
FIGURE 5.1: ANALYSIS AND SYNTHESIS OF SIMULATION INPUT DATA 115
FIGURE 5.2: SIGNAL AND TRANSITION PROBABILITIES OF AN AUDIO SAMPLE 115
FIGURE 5.3: SIGNAL AND TRANSITION PROBABILITIES OF ORIGINAL AND SYNTHETIC DATA ... 117
FIGURE 5.4: PRESENCE OF SPATIAL CORRELATION IN SYNTHESISED SAMPLE 121
FIGURE 5.5: TEMPORAL CORRELATION IN SYNTHESISED DATA 122
FIGURE 5.6: STATE TRANSITION DIAGRAM EXAMPLE 123
FIGURE 5.7: POWER DISSIPATION AS A FUNCTION OF TRANSITION PROBABILITY 126
FIGURE 5.8: EFFECT OF SPATIAL CORRELATION ON POWER ESTIMATES 127
FIGURE 5.9: VARIATION IN POWER DISSIPATION WITH TEMPORAL CORRELATION 127
FIGURE 5.10: STATE TRANSITION DIAGRAM FOR MARKOV MODEL EVALUATION 129
FIGURE 5.11: COMPARISON OF VHDL AND MARKOV MODELS 130
FIGURE 5.12: RELATIVE MAGNITUDES OF MARKOV, VHDL AND UWN POWER ESTIMATES .. 133
FIGURE 5.13: EFFECT OF CORRELATION AND SAMPLE SIZE ON ESTIMATES 133
FIGURE 6.1: PROCESSOR ENTITY 137
FIGURE 6.2: STRUCTURAL VIEW OF THE DATAPATH 138
FIGURE 6.3: CONTROLLER ENTITY 140
FIGURE 6.4: KARPLUS-STRONG PLUCKED STRING ALGORITHM 141
FIGURE 6.5: IMPULSE RESPONSE OF FILTER 142
FIGURE 6.6: RESPONSE TO INPUT NOISE BURST 142
FIGURE 6.7: ENVELOPE OF OUTPUT WAVEFORM 143
FIGURE 6.8: SPECTROGRAM OF SYNTHESISED PLUCKED STRING 143
FIGURE 6.9: AUDIO INPUT SIGNAL 144
FIGURE 6.10: AUDIO OUTPUT SIGNAL WITH REVERBERATION 144
FIGURE 6.11: IMPLEMENTATION OF KARPLUS-STRONG ALGORITHM IN VHDL ROM 146
FIGURE 6.12: PROCESSOR IMPULSE RESPONSE DATA 147
FIGURE 6.13: PROCESSOR OUTPUT DATA 148
FIGURE 6.14: CONTROL UNIT SCHEMATIC 149
FIGURE 6.15: PROCESSOR AND DATAPATH AREA 152
FIGURE 6.16: APPROPRIATE POWER MODELS FOR HIERARCHICAL APPROACH 153
FIGURE 6.17: ALU ACTIVE CAPACITANCE FOR UWN INPUT DATA 155
FIGURE 6.18: ALU ACTIVE CAPACITANCE FOR REAL INPUT DATA 155
FIGURE 6.19: ALU ACTIVE CAPACITANCE FOR SYNTHETIC INPUT DATA 156
FIGURE 6.20: MULTIPLIER ACTIVE CAPACITANCE 157
FIGURE 6.21: REGISTER FILE ACTIVE CAPACITANCE 157
FIGURE 6.22: POWER ESTIMATES OF SUBMODULES 158
FIGURE 6.23: COMBINATION POWER ESTIMATION MODEL 160
FIGURE 6.24: INSTRUCTION LEVEL MODEL 162
FIGURE 6.25: PROCESSOR POWER ESTIMATES 163
FIGURE 6.26: LONG-TERM PROCESSOR POWER ESTIMATES 165

List of Tables

TABLE 2.1: DESIGN CYCLE STAGES 15
TABLE 2.2: ENERGY ASSOCIATED WITH AN INSTRUCTION SET 17
TABLE 2.3: ENERGY ASSOCIATED WITH A SAMPLE OF CODE 18
TABLE 2.4: SIGNAL PROBABILITIES 30
TABLE 3.1: MAXIMUM TWO AND THREE DIMENSIONAL ACCURACY 49
TABLE 3.2: SPECTRAL TEST RESULTS FOR VARIOUS GENERATORS 52
TABLE 3.3: SPECTRAL TEST EXECUTION TIME 53
TABLE 3.4: SPECTRAL TEST EXECUTION TIME FOR VARIOUS MULTIPLIERS 53
TABLE 3.5: MAXIMUM INTEGER SIZES FOR SPARC AND PC SYSTEMS 54
TABLE 3.6: SPECTRAL TEST ERRORS 55
TABLE 3.7: IMPROVED RESULTS FOR SPECTRAL TEST 57
TABLE 3.8: IMPROVED SPECTRAL TEST RESULTS 58
TABLE 3.9: MAXIMUM SIZE OF CONSTANTS 58
TABLE 3.10: TYPICAL GENERATOR OUTPUT TESTS 67
TABLE 3.11: SAMPLE MEAN AND VARIANCE 70
TABLE 3.12: RESULTS FROM THE APPLICATION OF THE χ^2 TEST 71
TABLE 3.13: BINOMIAL TEST OF DATA SAMPLES 72
TABLE 3.14: RELATIONSHIP BETWEEN VARIANCE AND SPECTRAL SCORE 78
TABLE 3.15: CRITICAL VALUES FOR KS TEST 82
TABLE 3.16: KS TEST APPLIED TO GENERATOR SAMPLES 83
TABLE 3.17: KS TEST STATISTICS FOR SAMPLED SPECTRUM 86
TABLE 3.18: KS TEST RESULTS FOR SAMPLED SPECTRUM OF POOR QUALITY GENERATOR 88
TABLE 3.19: SPECTRAL TEST AND KS TEST RESULTS FOR A NUMBER OF GENERATORS 88
TABLE 3.20: SPECTRAL AND KS TEST RESULTS FOR OPTIMISED GENERATORS 90
TABLE 3.21: SENSITIVITY OF BINOMIAL TEST WITH FILTER PARAMETER -0.1 91
TABLE 3.22: SENSITIVITY OF BINOMIAL TEST WITH FILTER PARAMETER 0.1 92
TABLE 4.1: SUMMARY OF TEST CIRCUITS 104
TABLE 4.2: SIMULATION RESULTS WITH TEST CIRCUITS 105
TABLE 5.1: ERROR IN SYNTHESISED INPUT DATA 117
TABLE 5.2: UNCORRELATED DATA, *X* AND CORRLEATED DATA, *Y* 120
TABLE 5.3: RELATIVE FREQUENCY OF MARKOV MODEL OUTPUTS 124
TABLE 5.4: PROPORTION OF INSTRUCTION TYPES FOR MARKOV MODEL 128
TABLE 5.5: POWER ESTIMATES OBTAINED WITH MARKOV AND VHDL DATA MODELS 132
TABLE 6.1: INSTRUCTION TYPES 139
TABLE 6.2: INSTRUCTION SET 139
TABLE 6.3: REGISTER FILE SYNTHESIS RESULTS 150
TABLE 6.4: ALU SYNTHESIS RESULTS 150
TABLE 6.5: MULTIPLIER SYNTHESIS RESULTS 151
TABLE 6.6: DATAPATH SYNTHESIS 151
TABLE 6.7: MODULE ACTIVE CAPACITANCE 158

1 Introduction

The present state of electronic systems design may be characterised by a drive towards the allied goals of systems integration and convergence of functionality. The cellular phone, for example, becomes smaller and lighter and in the process acquires web browser capability [Levi02] [Spec01]. The combined effects of shrinking device geometries and increasing sophistication of design tools have also enabled designers to realise complete systems on a chip with relative ease [Jone01] [Gold01] [Eyre01]. At the same time, increasing amounts of functionality is finding its way out of slow memories onto silicon, in the form of firmware [Sher01]. Such acceleration has resulted in an increasingly complex paradigm for the specification and design of digital systems. This has resulted in challenges to the designer that could not possibly have been foreseen when pioneers such as Kilby [Gold97], Weimer [West93] and Wanlass [Riez91] were taking their first tentative steps along the path of integration. These challenges include increased clock speeds, shorter product development cycles and most recently, lower power consumption. Driven by an expanding market for portable electronic and communication systems, much research effort continues to be expended in the development of techniques aimed at reducing power consumption at all levels of circuit design [Giva01] [Simm01] [Wehm01] [Krus01] [Bhar01] [Kozh01].

In any endeavour, enhanced quality implies rigorous measurement and the field of low-power electronic systems design is no exception. Without the ability to assess the effect of architectural changes, the system designer cannot hope to influence the power performance of the design. The usual constraints imposed on project timelines also apply, in that the designer is also looking for speed and efficiency when assessing the likely power dissipation [Conte00] [Chen00] [Xie99].

Integrated circuits, while containing large amounts of functionality, do not function in isolation. They must interact in an environment where they reside in a hierarchy, communicating with other components at different levels in the system. This stratified approach is also reflected in the model used by the IC designer. Basic modules are combined into larger aggregates, which in turn are treated as components to be combined with other units of similar complexity, and so on. While system specification and functional decomposition are undoubtedly performed top-down [Espe02], much of the implementation

and coding work in systems design is, paradoxically, designed bottom up, or at best in parallel [ibid.]. The power-conscious designer therefore has a requirement to estimate power at several levels in the system, often at the same stage of the product cycle. The nature of the power analysis can also take several forms. Broad comparisons between competing solutions may need a universal approach with measurements that are portable across designs and between successive generations of a product [Chen00] [Gupt00a]. More specific and refined information concerning the power behaviour is generally required at higher levels of the design, where real world inputs may be incorporated into the scenario [Beni00] [Nema99]. Finally, a power rating may need to be determined at the highest system level, which should reflect the typical power rating likely to be encountered under typical working conditions [Maci01] [Gupt00b] [Laks99]. All of these disparate approaches fall under the umbrella of power estimation, yet each requires a different technique and is obtained under distinct operating conditions. The principal aim of this work is to present a unified, coherent approach to the problem of power estimation in integrated systems implemented in silicon.

Low power systems design for digital systems became a serious research topic following Anatha Chandrakasan's seminal paper of 1992, produced while still a postgraduate student at Berkeley [Chan92]. In it, he focuses on the challenges of predicting and minimising the power dissipation for CMOS circuits, in particular fixed-function digital signal processing applications. In so doing, he identifies the pivotal idea that underpins all subsequent work in this field, the trade-off between area and power when minimising power consumption. Following this breakthrough there followed a period of intense activity, with many significant contributions to the burgeoning field of Low Power Design [Nema96] [Chou96] [Land95] [Najm95] [Raba95] [Land94] [Lin94] [Najm94] [Najm93] [Land93]. In tandem with the emerging design techniques, there appeared a body of work devoted to the prediction of the likely power performance of systems, generally termed CMOS Power Estimation. This need for such a discipline was evident, providing as it does a mechanism for the evaluation of novel architectures and concepts. The recent history of power estimation is generally considered to have begun at the University of Illinois, where Farid Najm published his ideas on applying a technique first suggested by the 18th Century French mathematician Bouffon, to the efficient appraisal of circuit power consumption, the so-called Monte Carlo method [Burc93]. Early contributions looked at circuits in isolation without reference to a context for those circuits. In particular, a number of static, probability-based methods for power

prediction evolved which were mathematically very intensive while never managing to handle more than a few dozen transistors, always represented as isolated, out of context circuits [Uchi96] [Mont94] [Chou94].

The importance of the circuit context soon became apparent, as researchers realised for example that processor power could depend on the program executed and that for CMOS circuits, power was heavily influenced by the input signals into the circuit and therefore on the circuit's environment [Niko00] [Beni99] [Gupt97] [Gonz96] [Chan95b]. In recent years, as trends in portability and mobile communication have continued, issues such as low energy software design, adiabatic computing and synthesis for low power have received much attention from researchers [Levi02] [Beni99] [Ye97]. This emphasis on system level considerations has not been greatly reflected in the work being currently presented by those active in power estimation. While some research is being carried out at subsystem level, an overall systematic approach has yet to emerge. In addition, many of the high-level techniques that have been developed offer very little optimisation of the estimation algorithms presented. If such work is to be of relevance to the designer in industry, it must offer efficient, speedy estimates, with an approach that recognises the role of the module in its environment. System level estimates must be based on real data so that the designer can be confident that the results will reflect real working conditions. These attributes, efficiency, adaptability to hierarchical structures and the ability to reflect real world conditions when estimating power in digital systems, are the three objectives adopted at the outset of this present work.

Under the title of CMOS Power Estimation, this thesis aims to investigate the application of power estimation techniques in a comprehensive and systematic manner, using the three stated goals of adaptability, efficiency and relevance. The current state of the art and the intended focus for this work is now discussed for each of these criteria, in relation to power estimation in digital systems.

1.1 Adaptability: A Top-down Approach

The model most often adopted for systems design is that of a tree structure, with communication between successively higher modules in the system [Sjoh01]. As a commercial project is often a team activity, many of the levels are implemented in parallel. This has a number of consequences. Firstly, product development cycles are shortened and so an integrated circuit may be completed before the host module for which it is intended. In

such cases the absence of real world stimuli for test and for power estimation may be a problem [Marc00]. However, to determine a power rating for a component requires exactly such data or a synthetic equivalent that will yield an equivalent behaviour from a power perspective. The qualities required from such data vary, as the point of application moves deeper into the structure. A processor, for example, demands real code and data, if it is to function correctly while the power is assessed. The necessary operating system or application software may not yet exist, even the instruction set may not be finalised. The requirement at this stage is therefore for a source of data, which has the power sensitive attributes of real code, is capable of meaningful interaction with the processor but may be generated in a fraction of the time it takes to provide a software environment for a new system. The efficient synthesis of such data is one aim of this work. Moving deeper into the hierarchy, down to the level of functional units such as multipliers, finite state machines and the like, input stimuli do not appear as code and data, but rather as sequences of binary vectors [Burd02]. Although ultimately derived from program execution, such data streams have characteristics from a power perspective which must be accurately represented if meaningful estimates of power dissipation are to be produced [Ragh99]. Again obtaining a source of such data can be problematic, as the module may be designed well in advance of the system, for example as part of a library for synthesis.

Data sequences for submodule power estimation also form part of the aims of the thesis. At the lowest level of any design reside basic combinational and sequential circuits, often forming part of a design database of reusable intellectual property, seeing service on many fronts, from fixed function signal processors to large scale computing components. For such elements, there is a requirement for a universal system of power classification, independent of the data and also for a means of determining the application-dependent power consumption [Hsia00]. Both of these types of power estimation, the universal and the specific are necessary for basic circuits if real adaptability is required of the power estimation process. They too are added to the list of aims for this work. In the next section the question of how well previous research has approached these issues is addressed and intended enhancements to the current body of knowledge in this area are outlined.

1.2 Efficiency: Faster Power Estimation

The notion of speed and therefore efficiency in the context of the evaluation of a process such as power estimation requires careful definition. To the designer, a meaningful and welcome improvement in the speed with which a process such as power estimation may be concluded is determined by reference to the best alternative. By analogy, processor A might be considered to be faster than processor B only when two useful applications are observed side by side and not by a simple comparison of clock speeds [Haye98]. Similarly, an objective of this thesis is to provide a meaningful improvement in power estimation efficiency resulting in speedier estimates when the overall context in which the designer works is taken into consideration. In particular, the techniques developed in this work will be applied to a meaningful case study, the design and test of a reduced instruction set 32-bit processor [Kain96] [Feld94]. Using this structure as an architectural vehicle, existing and proposed techniques will be evaluated at a systems level, where any speed improvement would be observed by the designer. The concept of measurement efficiency, while most useful to the systems designer, is not limited to the analysis of high level modules. Basic circuit modules also require characterisation of a form discussed in the previous section. In this instance however, efficiency is manifested in repeatability and accuracy of the particular type of Monte Carlo measurements, which is a direct function of the quality of the input data, a fact often noted in the literature [Ding00] [Burc94]. This work proposes a solution to the problem of ensuring consistent data quality for submodule power analysis and thus efficient power analyses, which has so far not been successfully resolved by researchers. The verification of system level estimation techniques has been carried out in conjunction with an actual system design to fully validate the work undertaken. Similarly, to demonstrate the general application of the gains in efficiency from the proposed techniques, a power estimation tool will be demonstrated, incorporating the ideas developed in the thesis. This allows a comprehensive overview to be taken of the estimation problem at the circuit level and efficiencies in simulation methodology, in addition to data generation, are proposed and evaluated.

1.3 Relevance: Combining Concepts

While the initial part of this thesis concentrates on analysis and estimation for circuits and submodules, the concepts are then combined to provide a real solution to the problem of systems level power estimation. Today's designs are predominantly systems on silicon and a relevant power estimation technique must reflect this. A high level approach to system power is proposed by combining the concepts introduced in the early part of the work. This methodology has a high level of adaptability to different types of systems with varying data, control and communication signals equally represented. A working system is again used as the platform from which to evaluate this technique. The relevance of this model to power estimation for programmable systems is demonstrated by the implementation of an audio processing algorithm and the simultaneous estimation of power with a high level model.

1.4 Summary

Three approaches to enhance power estimation for digital systems are proposed. Firstly, increased adaptability to allow the different types of signals encountered at various levels in a system to be adequately modelled such that the power sensitive qualities of the data are retained. Secondly, the introduction of an efficient technique for data generation along with a rigorous approach to the measurement of the data quality and thirdly, the development of a high level methodology that is relevant to the wide variety of systems encountered today. To achieve these aims, a power estimation tool is developed to validate the approach to data generation, while a processor design is completed, along with a signal processing application to test the high level concepts introduced. The aim of this thesis may therefore be stated as being the integration of diverse power estimation techniques into a coherent framework, together with several new approaches designed to improve the efficiency and relevance of power estimation to the systems designer.

2 Power Consumption, Dissipation and Estimation

When the power performance of a system forms part of the design specification, the design cycle must include a means of estimating the probable power requirements of the system. This takes place before the actual implementation of the system as, for example, an integrated circuit. During this design cycle, the ability to effect improvements in the power performance of a system implies a requirement for power estimation algorithms to provide information during the design phase. Additionally, after implementation power measurement techniques may be employed to evaluate the system and assign a nominal power rating. In the context of digital CMOS design, such power measurement is usually performed on a functional die, chip or system by the application of input test vectors [West93]. The simultaneous monitoring of supply current is then performed and the waveform is integrated over time to yield a measurement of the average power consumption. The accuracy of such a procedure, and the measurement error, is solely a function of the test apparatus and the test vectors applied. Power estimation, on the other hand, can take place at several stages in the development of the design [Theo00] [Saty00] [Turg98] [Hsie98] [Ding98a] [Liu94]. Estimates of power consumption can be made early in the design cycle, at the algorithmic level, while later estimates may be obtained from circuit-level information in a design database after place and route has been performed [Huiz90]. There is a trade-off to be made between speed and accuracy of the power estimate during this process [Najm94]. Later estimates use a design representation that more closely resembles the physical implementation of the chip, while initial behavioural estimates are based on abstract software-like constructs. In addition, the amount of data required to capture a design grows with the progress of the design [Wolf94]. Simulation of the performance of the system therefore takes increasing amounts of processor time, or equivalently processing power, as the system description becomes more refined. Exhaustive simulation at all phases of the project is therefore not feasible and the designer must select the levels of simulation at which to operate. While designers of analogue circuitry, for example, may choose to simulate exclusively at the level of transistor operation, those involved in the synthesis of digital systems may opt for simulation at gate or register transfer (RT) level [Blai94]. The RT level of operation is of most application to the design of digital systems as the system is usually specified at this level, or higher [Pirs98]. This is a

fundamental and important decision, given the relatively undeveloped state of this area, because of the shrinking time frame within which electronic systems must be produced [Chan92]. Often, bespoke tools must be built to meet particular needs [Chen00] [Nema99] [Kuma95] [Mehr94].

Therefore, the aim of this Chapter is firstly to explore the physical processes which underlie the need for measurement and estimation of power in electronic systems. A number of alternatives for power estimation are then explored. In particular, the choices currently available when simulating digital ASICs are set out. Subsequently, one particular approach to power estimation used in this work and based on a Monte Carlo strategy is described, along with an approach employing probability concepts. The usefulness of each approach at different stages in a design is also suggested.

2.1 Power Dissipation in CMOS Devices

The physical origin of power dissipation in all CMOS circuits is now explored from a historical perspective, using the inverter as an example. The terminology employed in the literature as regards power dissipation and power consumption is also clarified. Power is separated into three components and the average and instantaneous values of these quantities are identified. Typical simulation waveforms from a circuit simulator are also provided, by way of illustration of the relative magnitudes of the components of power identified in the discussion.

Early developments in IC technology were based on PMOS fabrication technology due to the ease with which a metal gate enhancement mode device could be reliably produced with only four masks [Mill72], a simple but robust process, which today still finds occasional application in academic laboratories. This technology was eventually replaced by an NMOS-based silicon gate process because of the superior mobility inherent in the majority carriers. Typically, in such NMOS circuits, the majority carrier electron mobility is 2.5 times the minority hole mobility [Brac87]. Since the switching time for an inverter is inversely proportional to the carrier mobility, NMOS logic devices were correspondingly faster. However, for both logic families considerable power was consumed even under quiescent or static conditions, resulting in power consumption of the order of milliwatts per gate [Kang96]. In 1963, Fairchild Corporation patented three circuits, an inverter, a NAND gate

and a NOR gate, that exhibited nanowatt power consumption levels, two orders of magnitude lower than the then current NMOS technology [Scie97]. The new circuits initially used discrete transistors of both polarities and the concept was termed CMOS. The more complex production techniques required did not hinder this technology from becoming dominant, due to its superior power performance, particularly under static conditions [Kang96]. An understanding of the mechanisms by which this energy is consumed in a CMOS circuit is central to any attempt to measure and minimise the effect of such mechanisms.

2.1.1 Consumption and Dissipation in a CMOS Inverter

The term power consumption will be used to refer to the total power delivered by the dc supply to the circuit. Power dissipation refers to that portion of the power consumption that is transformed into heat by resistive layers on the die. However, the terms are used interchangeably in the literature [Nico01] [Leme98] [Maci96] [Najm95] [Chan95a] since ultimately all the energy delivered is dissipated, primarily in heat but with some photon emission. The average and the instantaneous values of these quantities are both of interest to circuit designers [Bella95]. The instantaneous power consumption is a function of the current waveform associated with the power supply. The maximum value of the instantaneous power coincides with the maximum value of the current waveform. Two unwanted phenomena in CMOS circuits which are both a function of maximum power are the power supply noise, which is due to the finite output impedance of the supply and its distribution system, and a temperature rise within the device [Hsia00] [Wu01] [Jian00]. Both of these have an adverse effect on the performance of the circuit. For portable applications where battery life is of concern, the average power consumption is usually considered [Chan95b] [Xant00] [Namg00] [Simo00] [Wang00]. It is also instructive to consider the flow of energy that takes place when logic transitions occur. The CMOS inverter consists of an NMOS device in series with a PMOS device, as shown in Figure 2.1.

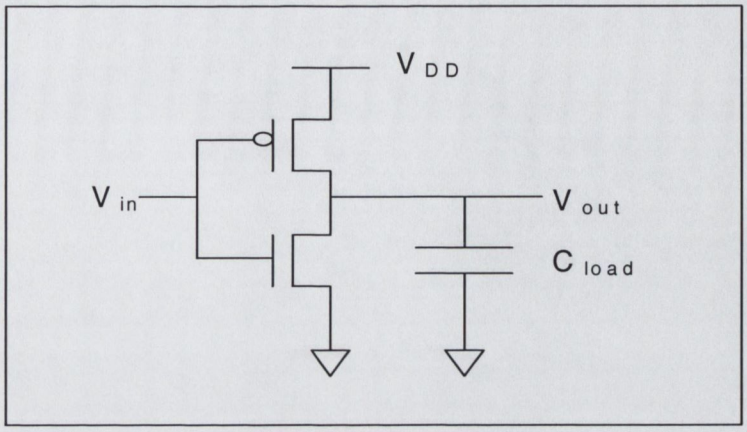


Figure 2.1: The CMOS Inverter

The total power dissipation P_{total} can be separated into three components as indicated in (1)

$$P_{total} = P_{static} + P_{dynamic} + P_{short-cct} \tag{1}$$

The static power dissipation, represented by P_{static}, is caused by a combination of two factors, the leakage current due to the parasitic reverse-biased diodes in the CMOS structure and a current that flows due to the input voltage [Blai95]. The contribution to static power due to leakage is approximately 1fA per device junction and so may be neglected [Jaeg97]. The other factor in determining the static component of power, the current due to the input voltage, can be of significance under certain conditions, as is explained.

The input voltage to the inverter is V_{in} and when it is in the range $0<V_{in}<V_T$, where V_T is the threshold voltage of the NMOS device, a weak channel forms between drain and source of this transistor. The device is ON, with a small temperature dependent current in the subthreshold region of operation [Blai95]. The drain current may therefore be approximated by the expression in (2) where W_{eff} is the effective channel width, W_0 is gate width at $V_{GS}=V_T$, I_0 is the drain current at the same point, and S is the subthreshold swing parameter [ibid.].

$$I_D = I_0 \frac{W_{eff}}{W_0} 10^{\left(\frac{V_{in}-V_T}{S}\right)} \tag{2}$$

The parameter S is the amount of gate-source voltage required to reduce the drain current by one decade and is typically in the range 70-90mV/decade [ibid.]. For a 1μm process with a typical supply voltage of V_{DD} = 3.3V and a subthreshold swing parameter S = 70mV/decade, reducing the threshold voltage to V_T = 0.14V for example, results in a drain current of

approximately $I_D = 25\mu A$ and a static power dissipation of approximately 820nW when V_{in}=0. This is not a sustainable level of power dissipation and is one of the reasons why the threshold voltage is not reduced to very low levels. For typical devices, the threshold voltage is designed to be at a sufficiently high level so that the static component of power dissipation is not of significance.

The second component of power dissipation, the dynamic power $P_{dynamic}$, may be obtained by examining the current that flows when the gate is driven with an ideal step input. In this case current is drawn from the supply as the total capacitative load C_L is charged. The instantaneous current drawn from the supply is given by (3) and the instantaneous power consumption may be obtained by multiplying this expression by the fixed supply voltage V_{DD} [Jaeg97].

$$I_{DD}(t) = C_L \frac{dV_{out}}{dt} \tag{3}$$

For an input waveform with period T, the average power consumption over one period may be obtained by the integration of this expression, yielding (4).

$$P_{DD} = \frac{1}{T} \int_0^{Vdd} C_L V_{dd} dv_{out} = \frac{C_L Vdd^2}{T} \tag{4}$$

Under the same conditions, the instantaneous dynamic power dissipation in the circuit may be expressed as the sum of the instantaneous power dissipation for both devices, as given in (5).

$$P_D(t) = V_n(t) I_n(t) + V_p(t) I_p(t) \tag{5}$$

This may be rewritten as shown in (6).

$$P_D(t) = V_{out}(-C_L \frac{dV_{out}(t)}{dt}) + (V_{DD} - V_{out})(C_L \frac{dV_{out}(t)}{dt}) \tag{6}$$

The integration of this expression over the period T in (7) yields the average dynamic power dissipation P_D.

$$P_D = \frac{1}{T} C_L V_{DD}^2 \tag{7}$$

Energy is only drawn from the supply during an output transition from 0 to V_{DD}. This energy is $C_L V_{DD}^2$. Half of this energy is dissipated in the channel resistance of the PMOS device as the load capacitance charges up [West93]. The remainder of the energy, $0.5 C_L V_{DD}^2$, is dissipated in the NMOS channel when the capacitance discharges as a result of the V_{DD} to 0 output transition. Equation (7) shows the importance of distinguishing between energy and power in characterising circuit performance. While an increase in the period would lower the dynamic power dissipation, the energy required for the process remains unchanged. For this reason, a measure termed the energy-delay product is often quoted as a figure of merit for a circuit [Raba96]. It is the product of total energy transferred from the supply to the circuit during a HIGH to LOW input transition times the propagation delay.

The third component of the total power dissipation is the short-circuit power dissipation, $P_{short\text{-}cct}$. This dissipation occurs because of the finite input signal slope, which allows short current pulses to flow between V_{DD} and Ground while both devices are momentarily on. It is also a function of the output load. An analysis [Veen84] of an unloaded inverter yields an expression for approximate short circuit power as indicated in (8).

$$P_{short-cct} = \frac{\beta}{12}(V_{DD}^2 - V_T^2)^3 \frac{t_{r/f}}{T} \tag{8}$$

In this expression, $t_{r/f}$ is the rise/fall time of the input waveform and β is the transistor conductance parameter. For a loaded gate, with equal rise and fall times, the short-circuit power dissipation is typically less than 10% of the total power [ibid.].

The distinction between power consumption and power dissipation and the relative importance of short-circuit and dynamic components of power dissipation may be easily seen by means of a circuit simulation. In Figure 2.2, a CMOS inverter is driven by a pulse waveform. The input signal has a frequency of 50MHz, with equal rise and fall times of 2ns. The voltage levels of the pulse are V_{DD}=5V and 0V. The inverter was designed with a gate length of 1μm and transistor widths of Wn=10μm and Wp=20μm. Two lumped capacitances representing input and output capacitance were included in the simulation. The values used were C_{in}=20fF and C_{load}=1pF. The large output capacitance serves to accentuate the device

currents, for clarity in the simulation. The input waveform, the instantaneous supply current, the load current and the two transistor currents are recorded. The circuit simulator SPICE [Nage75] was used. The CMOS process was a commercial 1μm technology.

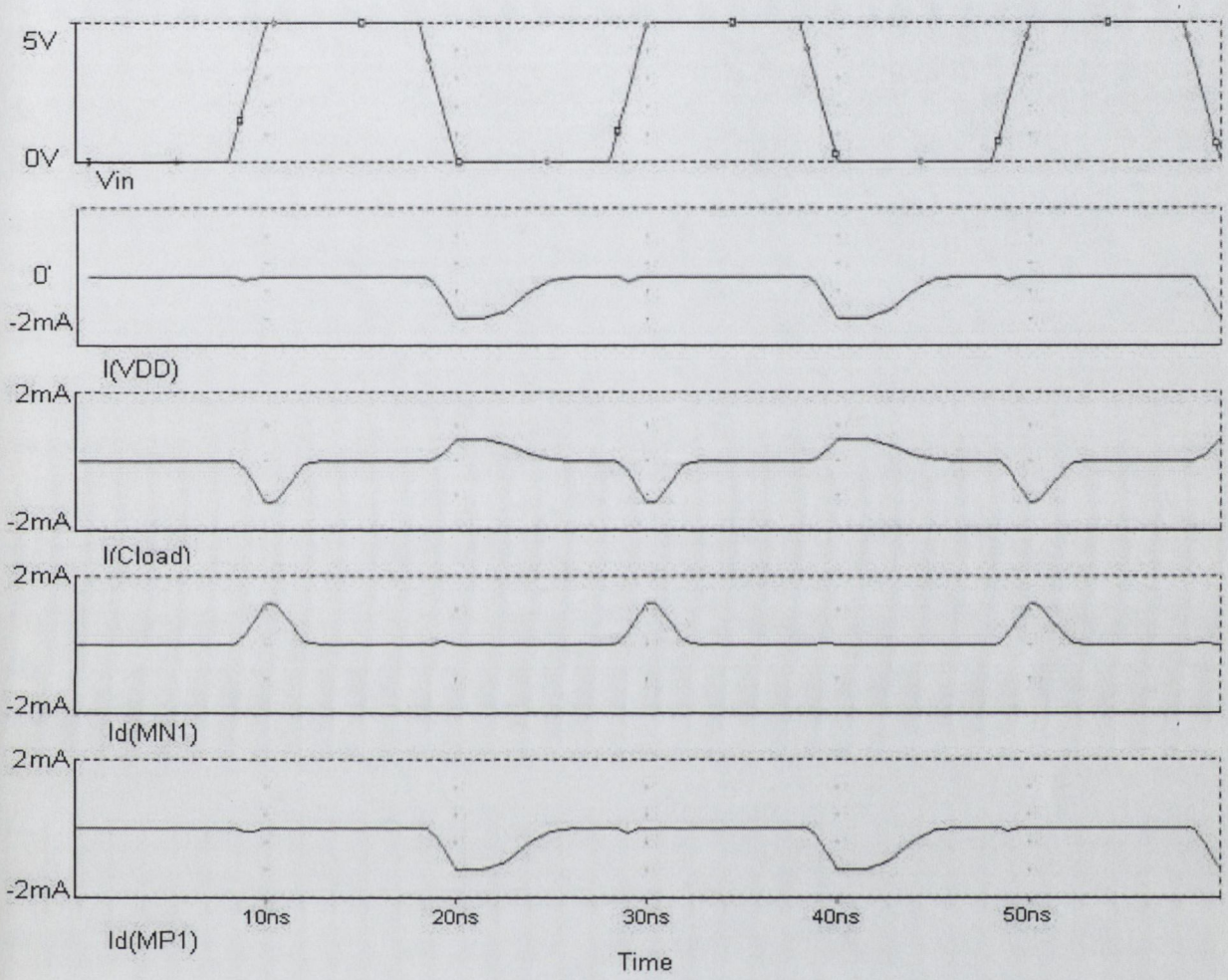


Figure 2.2: Inverter Current and Voltage Waveforms

A number of observations may be made. Firstly, the instantaneous supply current $I(V_{DD})$, is not the same as the device load current, which consists of the sum of the two components $I_D(MN1)$ and $I_D(MP1)$. Consequently, the instantaneous power consumption is very different to the instantaneous power dissipation. The average values of these quantities are the same. The short-circuit current is indicated by the smaller current peaks for each device. Power-conscious layout can minimise this component of power consumption, which may be easily quantified for typical transistor configurations [Nose00].

2.1.2 Estimation of the Correct Power Quantity

The average dynamic power dissipation, as indicated by (7), is independent of transistor characteristics such as device geometry, sheet resistance or delay. This allows some general conclusions concerning power dissipation in larger static CMOS circuits to be made. Firstly, designers wishing to design a 'low power' system must first specify if longer battery life and lower operational die temperature is a design goal. Other parameters that may also be of interest are electromigration, voltage sag on the power bus and component reliability [Deng88]. When battery life forms part of the system specifications, then average power dissipation is the key parameter to be addressed. When parameters such as circuit reliability or the proper design of power and ground lines are of more concern, then instantaneous, and therefore the maximum, current and power should be measured. While instantaneous and average power may each be used as an approximate estimator of the other, researchers building specific power estimation tools, have usually focussed on one or the other [Simu01] [Kouk01] [Gonz96]. For this work, power estimation for design of portable systems is the primary aim. Portable systems require a minimisation of battery current, and therefore the average power dissipation of the system is the focus. This quantity is exactly equivalent to the average power consumption. In addition, as the short-circuit and leakage elements of power dissipation are considerably smaller than the dynamic component [Veen84], minimisation strategies employed should concentrate primarily, but not exclusively, on reducing the average dynamic power dissipation.

2.2 Power Estimation in CMOS Circuits

The design of a complex system such as a processor often follows a sequence of tasks, known as a design flow, that takes the project from an abstract system-level specification, through simulation and verification, down to a transistor netlist or a set of layout masks, ready for fabrication. This process can be divided into a number of well-defined steps, the output from which represents a further refinement in the transition from concept to silicon. One such set of sub-tasks is indicated in Table 2.1 [Maci98].

Level	Design Abstraction	Power Estimation Technique
1	Software Optimisation	Software Power Analysis
2	Behavioural Transformation	Behavioural Power Analysis
3	High Level Synthesis	Register-level Power Analysis
4	Logic Synthesis	Gate-level Power Analysis
5	Technology Mapping	Circuit and Switch Level Power Analysis

Table 2.1: Design Cycle Stages

At each level of abstraction, a power estimator is used to loop through the design alternatives to predict the power at that level. The accuracy of the power estimate increases as the design detail increases, but this is at the expense of increased computing resources and simulation time. Power optimisations applied at high levels in the design flow yield shorter development times and quicker power estimates. However, more accurate and reliable results are achieved by the use of tools lower down the table, due to a more mature state of development [Roy00]. As long as the design process takes place at a number of different levels, from the behavioural down to the transistor level, power estimation must be available at that level to evaluate the effect of circuit modifications. Software and hardware development increasingly takes place concurrently and so the relatively new area of software-hardware power co-optimisation is of importance [Heal00]. For each of the five levels of power estimation indicated, current techniques will now be described.

2.3 Software Power Estimation

The aim of software power estimation is to provide a figure for the power dissipation of a section of code. There are currently four techniques available for achieving the Level 1 power estimation referred to in Table 2.1. Some of these techniques, while introduced here in the context of software power estimation, are applicable to more than one level. The four techniques available for estimating the power implications of software are now discussed.

Gate level power estimation is one of the most accurate methods currently in use for software power estimation [Chou96] [Najm94]. This uses a logic level description of the processor, post synthesis, with a behavioural level model of external memory for code and data. The method is slow, but can provide useful data when characterising modules for use in more abstract software estimation tools.

Architectural level power estimation of software models the processor as a set of interconnected functional modules such as the ALU, register file, controller, etc. The components are characterised for power *a priori* and the number of accesses associated with each instruction is also obtained. The energy cost of a program operation is then determined by adding the dissipation of each module, over all cycles of the program run [Sata95].

Bus switching activity is another approach to software estimation that assumes that signal transitions on the processor busses are a good indictor of overall processor activity. A sample program is run and statistics are gathered for the amount of activity on each bus. A simplified processor model may be used, that takes into account those elements of software that generate bus transitions, such as op-codes, typical input data and address space usage. These sequences of code can be translated into binary sequences, which directly provide the required activity. This information is used to choose between software alternatives [Chan95a] [Land95].

Instruction level Power Analysis is the fourth technique available for software power estimation based on the average power dissipation of each instruction [Tiwa96] [Roy00]. Processor operation is divided into two categories: a normal mode when the bulk of the processing occurs such as arithmetic functions, register and memory accesses, etc. and a mode where the prior state of the processor is grossly altered. Pipeline stall, buffer stalls and cache misses fall into this latter category. Instruction energy is obtained for each category type. For instructions such as add, multiply and shift, the power dissipation is divided into two components, the Base Cost B_i and the Circuit State Effect O_{ij}. Base cost is that fraction of the power dissipation of instruction i that does not depend on the previous state of the processor. It can be determined by running a fragment of code consisting of a number of instances of the target instruction. Power and energy estimates for the base cost of the instruction are then made using the circuit activity, the number of execution cycles and the supply voltage. An average value may be obtained by simulation, using different examples of the instruction, for example by varying the registers accessed during an add instruction. The circuit state effect is the additional energy O_{ij} that is dissipated when the processor changes state during the execution of instruction i followed by instruction j. When instruction i and instruction j are of the same type, e.g. add, the circuit state effect is an average value, due to the change of registers, constants and addresses within the instruction. Published work in this

area [Tiwa96] does not distinguish between instruction *i* followed by instruction *j* and vice versa, due to the measurement technique employed. In this work, measurements were taken from a processor while a section of code consisting of the two instructions was repeated many times. This meant that the transitions *i* to *j* and *j* to *i* were indistinguishable. This difficulty could be overcome in a simulation environment.

For example, a memory access is followed by an arithmetic operation in the following two lines of code: Reg1 Mem[10], Reg1 Reg1+Reg2. In this case, circuit state effects would include changes in the instruction bus, new controller outputs and rerouting of ALU inputs. The sequence is repeated several times and the energy is then obtained. The energy associated with a stall or a miss is also termed an effect with associated energy E_i and is determined with a special simulator which can generate such events. [Tiwa96]. The cache or miss is simulated in isolation and the energy dissipated is thereby obtained. Equation (9) allows the energy dissipation of a sample program to be estimated from these instruction level power measurements.

$$E_P = \sum_i (B_i \cdot N_i) + \sum_{i,j} (O_{i,j} \cdot N_{i,j}) + \sum_k E_k \tag{9}$$

In this expression, N_i is the number of occurrences of instruction *i*, while $N_{i,j}$ counts the number of times instruction *j* is followed by instruction *i*. The energy associated with effects such as stalls and misses is represented by E_k. For example, the base costs and circuit state effects associated with part of the instruction set of a hypothetical processor are given in Table 2.2.

Instruction j	Base Cost (pJ)	Circuit State Effects (pJ)			
		LoadWord	Add	StoreWord	Subtract
LoadWord (lw)	2.37	0.81	1.05	1.32	0.68
Add	2.25		0.98	2.41	1.65
StoreWord (sw)	3.51			0.62	2.85
Subtract (sub)	0.98				0.44

Table 2.2: Energy Associated With An Instruction Set

The following fragment of code evaluates the expression (*x*-*y*)+*z*, where *x*, *y* and *z* are memory variables. The processor executes the following sequence of instructions:

register 1 <- *x*

register2 <- *y*

register3 <- *z*

register4 <- *x*-*y*

register4 <- register 4 + register 3.

Assuming that an add was the last instruction executed before this section of code, the energy costs associated with the code may be evaluated as shown in Table 2.3. Each line of the program contributes two components to the overall energy cost of the code. The first component is the base cost unique to each instruction. To this is added the incremental energy cost incurred when one instruction is followed by another. This is the circuit state effect.

Line Number	Instruction	Base Cost (pJ)	Circuit State (pJ)
1	lw	2.37	1.05
2	lw	2.37	0.81
3	lw	2.37	0.81
4	sub	0.98	0.68
5	add	2.25	1.65
	Totals	10.34	5.00

Table 2.3: Energy Associated with a Sample of Code

The total energy associated with this code is therefore (10.34+5.00)pJ=15.34pJ. With a clock period of 50ns, for example, the average power dissipation is 15.34pJ/50ns=308μW. In [Tiwa96], a cache simulator is used to estimate the frequency of cache misses, to add the energy cost of these events to the overall energy consumption of the code. This technique may be extended to characterise blocks of assembly code within a large program. The frequency of blocks may then be used to estimate the power dissipation of the program.

This approach to power estimation is only applicable to programmable systems where there is some element of stored program control, either at the instruction level or at the level of microcode. Evaluation of other forms of architectures such as fixed-function DSP systems, array processors and individual datapath elements would prove more problematic. Many systems are characterised by the relative sizes of the controller and datapath, so-called data-dominated and control-dominated systems. This technique makes no attempt to capture the

influence of data or operands on the power dissipation. It would therefore be unsuitable for such data-dominated systems as audio and video processors. The utility of this methodology would be most easily observed in quantifying the power dissipation of an embedded system with a relatively small instruction set, where the code sequence is fixed, with little programmability. Otherwise the amount of simulation required to capture all the possible base and circuit effects would become excessive. Another possible effective use would be in an incremental design or a re-design, where for example only the effect of additional instructions required analysis.

2.4 Behavioural Power Estimation

At the behavioural level of the design cycle, the system is represented by high level modules such as memory, ALU and register file. These are described only in terms of their function and no circuit information is yet available. As there is no gate or netlist information available, power estimates must depend on abstract models of both capacitance and switching activity. A number of approaches have been suggested for estimating the likely power dissipation of the final system. These are now outlined.

Information Theory Models use the output entropy of a function to predict the average area and so estimate the switched capacitance and average power dissipation of the final CMOS implementation of the function [Nema96] [Pipp77]. For example, a Boolean function F has an n-bit input vector X, and an m-bit output vector Y. The input entropy of the function, $H(X)$, is given in (10), where p_i is the probability that the input vector X takes the i^{th} value X_i.

$$H(X) = \sum_{i=1}^{2^n} p_i \log_2 \frac{1}{p_i} \tag{10}$$

The output entropy $H(Y)$ is defined in (11). There are 2^m possible output values and the probability that the output takes the particular value Y_i is given by p_i. The summation is taken over the 2^m output values.

$$H(Y) = \sum_{i=1}^{2^m} p_i \log_2 \frac{1}{p_i} \tag{11}$$

When input signals are random, with signal probability of 0.5, the function *H(Y)* can be used to estimate the area of the average minimised implementation using (12), where *A* is the circuit area and *K* is a constant of proportionality.

$$A = K\frac{2^n}{n}H(Y) \tag{12}$$

Empirical results have shown the effectiveness of this approach for small and medium sized circuits in predicting circuit area and power dissipation. In [Marc96], it has been shown that the average switching activity of a bit is upper-bounded by half its average entropy. Therefore, the average power dissipation may be approximated by (13), where the average switching activity is estimated by half the average entropy.

$$P_{avg} = 0.5V^2 fC_{total}\frac{H_{avg}}{2} \tag{13}$$

The average circuit entropy H_{avg} is estimated in turn using an expression proposed in [Nema96], as indicated in (14).

$$H_{avg} = \frac{2}{3(n+m)}(H(X)+H(Y)) \tag{14}$$

In addition, the entropy is used to predict the total switched capacitance of the circuit. Ferrandi *et al.* [Ferra98] proposed a capacitance estimator based on the number *N* of nodes in the binary decision diagram representation [Brya86] of the circuit function, as expressed in (15).

$$C_{total} = \alpha\frac{m}{n}NH(Y)+\beta \tag{15}$$

The coefficients of the model α and β, are obtained through a linear regression on a set of previously synthesised circuits. Tyagi, in [Tyag97], explicitly addresses controller circuits, where a lower bound is obtained for the entropy of the average Hamming distance [Beni99] between *T* states drawn from a state set *S*. In this expression, given in (16), $p_{i,j}$ is the steady-state transition probability between two states, $H(s_i,s_j)$ is the Hamming distance between the two states and $h(p_{i,j})$ is the entropy of the probability distribution $p_{i,j}$.

$$\sum_{s_i, s_j \in S} p_{i,j} H(S_i, S_j) \geq h(p_{i,j}) - 1.52 \log T - 2.16 + 0.5 \log(\log T) \quad (16)$$

Complexity-based models attempt to express the power dissipation as a function of circuit complexity, an abstract measure which reflects a variety of parameters such as the number of literals in a Boolean expression or the number of states in a controller description [Mull91]. The notion of a gate equivalent count is central to the complexity-based paradigm. This is an estimate of the number of basic gates, such as NAND gates, which will result when the function is eventually synthesised. In addition, an average energy is associated with the equivalent gate. Thus, obtaining an approximation for the number of equivalent gates in a function allows an estimate of the average power associated with the function. One such model is the Chip Estimation System [ibid.], which is based on the average power estimate provided by (17).

$$P_{avg} = fN(Energy_{gate} + 0.5V^2 C_{load}) E_{gate} \quad (17)$$

In this expression f is the clock frequency, N and $Energy_{gate}$ are respectively, the number of equivalent gates and the average energy consumption of an equivalent gate. The average capacitive load for an equivalent gate, including fanout and interconnect is given by C_{load} and is estimated statistically, based on fanout and wire load models. The average output activity per cycle is given by E_{gate}. The value used depends on the functionality of the module. Other work in this area focuses on fitting a variety of regression curves with data from fabricated circuits. Landman and Rabaey [Land95] use this technique to predict the power dissipation of a finite state machine using a precharged psuedo-NMOS technology, amongst others. Nemani and Najm [Nema97a] [Nema97b] predict the area of a single output Boolean function f using a complexity measure based on the on-set and off-set [Nema97a] of the function. The complexity of the on-set of f is given by (18), while the equivalent function for the off-set is given in (19).

$$C_1(f) = \sum_{i=1}^{N} c_{i(on)} p_{i(on)} \quad (18)$$

$$C_0(f) = \sum_{i=1}^{N} c_{i(off)} p_{i(off)} \quad (19)$$

The individual sizes of the N essential prime implicants [DeMi94] of the on-set are given by the set of integers $\{c_1, c_2, \ldots c_N\}$ and the weight $p_{i(on)}$ is the probability of all minterms in the on-set of f, which are covered by essential primes of size less than or equal to i. The area complexity of the off set of f is also calculated in a similar manner. The overall area complexity of function f is then given by (20).

$$C(f) = \frac{C_1(f) + C_0(f)}{2} \tag{20}$$

The final step in the process involves a set of regression curves for random logic functions obtained using a logic optimisation program. These curves relate the area complexity to actual circuit area. This approach has also been applied to multiple output functions [Nema97b].

Synthesis Based Models rely on a register transfer level template for the circuit, thereby reducing the final choices in the system architecture. The designer chooses from a limited set of options, such as pipeling structure, controller design or bus architecture. The power estimate is then obtained from a register-transfer level simulation of the system. Typical system data is provided by static or statistical profiling of input data [Chan95] [Chan96] and by dynamic profiling, where a behavioural simulation with input data is performed [Kuma95] [San96].

Given the diminishing time allocated to product design, an algorithm that delivers a quick estimate based on the Boolean function, is a useful analytical tool. For this reason the entropy-based information theory model may be regarded as a good compromise between time and accuracy. While some prior knowledge of the relationship between area and power is assumed, a simple linear relation could be easily developed from one or two designs. While not providing the accuracy of some of the more time-intensive approaches, this technique represents a good engineering compromise that has a reasonable chance of being used in commercial designs, as a first estimate of power dissipation.

2.5 Register Transfer Level Power Estimation

RT-level power estimation is predominantly based on the use of switched capacitance models of medium-complexity circuit modules such as adders, multipliers and memories [Raba95] [Theo00]. Signals at this level are often modelled in terms of their statistical properties. While there are several ways of describing the activity at the RTL level, two measures of the behaviour of a digital signal have, in particular, have proved useful. They are the signal probability, P_s,[Park75] [Bala89] and the signal transition probability, P_t [Xake94] [Najm92]. Conceptually, the signal probability is defined as the probability that a signal will be at the HIGH logic level. The transition probability is the probability that the signal will switch states during the clock cycle. Thus for a clock signal, P_s is 0.5 and P_t is 1. Uniform digital white noise (UWN) is defined as a digital signal for which both P_s and P_t are 0.5. The switching activity α_i of a circuit node i is another useful concept. It determines how many 0 to V_{DD} transitions occur at the node per unit time. The observed switching activity is therefore an estimator of the probability that the node will undergo a LOW to HIGH transition during any given clock cycle. [Bella95].

The power model used to characterise an RTL subsystem is referred to as a power macro model. The power macro model for a multiplier, for example, is a multivariable expression for the power dissipation of that multiplier. It is usually parameterised in terms of quantities such as input bit width, module structure and supply voltage. It is developed by simulating a gate- or switch-level representation of a set of representative modules with differing input sizes, signal statistics and internal structures with random input data and thereby fitting a regression curve to the results. The regression curve is then used as the power macro equation and is evaluated for components instantiated in individual designs. The additional power dissipation from random logic blocks or interface circuitry is obtained by gate level simulation, or by probabilistic techniques. A number of forms have been proposed for the power macro equation, each of which contains trade-offs between accuracy and computational requirements. One of the earliest and simplest power macro models is found in [Powe90], where it is termed the power factor approximation technique. This work uses a constant type model where an experimentally determined constant C, is used as weighing factor. For example, the power dissipation of a multiplier with two n-bit inputs is given by (21).

$$P_{diss} = 0.5V_{DD}{}^2 n^2 C f_{activ} \quad (21)$$

The supply is V_{DD}, and f_{activ} is the frequency with which the module is accessed. This model does not take the input data dependency of the power dissipation into account. When input data other than UWN is applied, i.e. data containing temporal correlation (between samples) or spatial correlation (between bit positions), then this expression over-estimates the power dissipation because the effect of such correlation is to reduce the activity at the individual nodes [Najm95]. Thus the UWN-based estimate of power is useful as an upper bound on the power.

A more sophisticated model, which takes into account the amount of activity at the inputs of the model, has been developed by Landman and Rabaey [Land93]. This model, called the dual bit type model, incorporates a relationship between successive samples of the input data, into the power estimate. The transition activity for each bit position is also factored into the model. For example, Figure 2.3 shows the transition activity P_t for each bit position of a 16-bit correlated data stream, for correlation coefficients in the range -0.99 to 0.99.

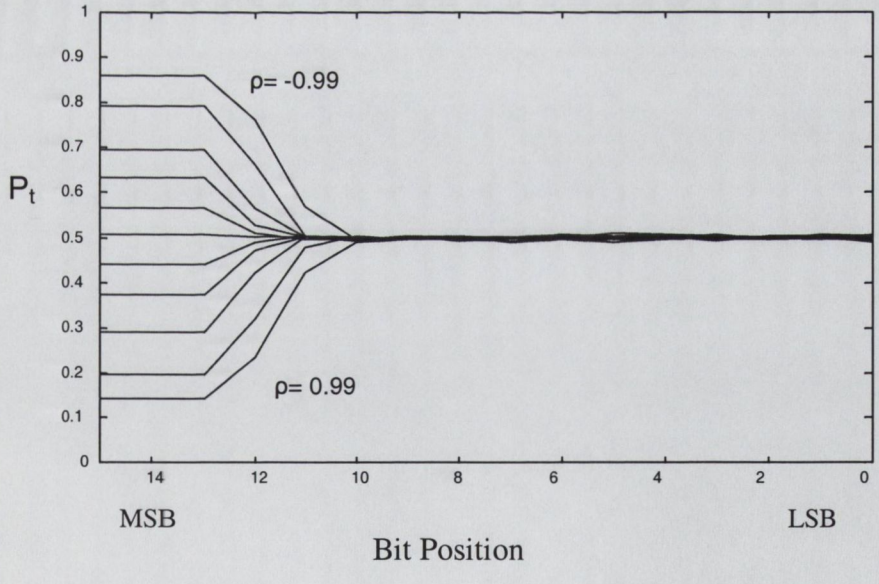


Figure 2.3: Transition Activity In A Sample Of 16-bit Data

The data is drawn from a gaussian process for which the temporal correlation is given by (22), where $cov(x_{t-1},x_t)$ is the covariance between two successive samples.

$$\rho = \mathrm{cov}(x_{t-1}, x_t) / \sigma^2 \quad (22)$$

This corresponds to the correlation coefficient at lag one. Transition probabilities are given for a number of values of correlation. Note that when the correlation between samples is zero, the data becomes white noise. To produce this data, a sample of gaussian white noise was sampled and filtered with an autoregressive filter, such that correlated data is obtained. By varying the filter parameters, correlation coefficients in the range 0.9 to -0.9 were generated. The transition probability for the 16-bit data was then plotted against bit position, for each value of correlation. Given that power dissipation is a function of the activity of the input data, it can be seen that use of white noise as input data gives at best an upper bound to the power dissipation. While this information can be of use, precise power estimates are obtained only with real data, or by generating synthetic bit streams which have the same statistical properties as the original data. In order to generate such data, input streams are divided into three regions, from the LSB to the MSB. For example, the data stream analysed in Figure 2.3 contains a region from bit 15 to bit 12, termed the sign region. The region from bit 8 to bit 0 is a white noise region. There is also a small intermediate region between the other two regions, where the signal probability varies from 0.5 to a value determined by the correlation coefficient. The location of the boundaries of each region, called breakpoints, may be expressed in terms of the signal properties. Breakpoint *BP1* defines the transition from the sign region to the intermediate region and is given by (23), where the signal mean and standard deviation are given by μ and σ respectively.

$$BP1 = \log_2(|\mu| + 3\sigma) \tag{23}$$

Breakpoint *BP0*, between the intermediate region and the white noise region, is given by (24).

$$BP0 = \log_2 \sigma \tag{24}$$

This model also replaces fixed capacitive coefficients with a parameterised capacitance, the value of which depends on the exact statistical behaviour of the input data. The overall effect is a power modelling technique for modules such as register files, multipliers and adders which give results comparable to switch level simulations. As long as the underlying technology does not change, or new synthesis algorithms implemented, the results are generally valid. This work has been extended [Gupt97] to a three dimensional technique that

utilises the output switching of the circuit in addition to the switching activity discussed above.

These RTL procedures are based on a regression model of the subsystem. The parameters of the model are extracted from a large number of implemented systems. They are therefore valid only for a given technology, place and route tool and standard cell library, as altering these parts of the design flow would result in a different physical capacitance for the system. Where the design process is relatively static, as in an academic environment and when a large amount of prior work has been completed on modules and subsystems, then this approach should yield good results. This has been the format adopted for power estimation results reported using these techniques. Systems are analysed based on the results of previous research into the modules incorporated into the system. However, for an *ab initio* power estimation exercise, the effort required to characterise all cell, modules and memories to build the regression models, would become very time consuming.

2.6 Gate Level Power Analysis

At the gate level stage of the design, a synthesis tool is invoked to automatically translate the register-level VHDL description of the system into a netlist of basic gates. Good estimates of the capacitive loading at each node are also available, created with a wire load model of the interconnect and the fanout. As a result, there are a large variety of options for estimating the power dissipation at this point. The dynamic component of both average and instantaneous power dissipation is due to the charging and discharging of the drain- and source-to-substrate capacitances, the gate-to-substrate and the interconnect capacitance of the transistors used to implement the gate function. Transitions only occur as a consequence of a change in logic inputs to the gate and may be categorised as either useful or spurious. Useful transitions are those that represent a change of logic level from one steady state level to another. A finite time is required to charge and discharge the device and interconnect capacitances associated with the gate. There is therefore a delay between the change in logic inputs and the corresponding output transition. One of the effects of this propagation delay is the occurrence of two or more additional transitions, which may take place in addition to the transition determined by the logic function of the gate. The exact nature of these so-called hazards or glitches will be discussed next, together with some methods for accurately modelling the

delay that causes them. An important result of this classification of transitions into wanted and unwanted events is that the dynamic power dissipation incurred may be similarly divided into two components. The hazard power component may be typically 20% of the total dynamic power on average, and up to 70% in circuits such as combinational adders [Naim94]. The ability to accurately estimate both components of the dynamic power is therefore a useful discriminator between power estimation methods at the gate level.

How successful a tool is in estimating the hazard component of the dynamic power dissipation depends on the model used for propagation delay. There are three basic approaches to modelling a gate delay. The simplest method, known as the zero-delay model, ignores all gate delays. This model therefore neglects the dynamic power component due to hazards and so provides the lowest estimate of dynamic power dissipation. A slightly more sophisticated approach is to incorporate a unit-delay model, which assigns a nominal, fixed delay to each gate. This allows an estimate of the power component due to hazards. The most accurate method of modelling delays at the gate level involves the use of variable delays. There are several approaches to this [Roy00], one very common approach being the division of the delay into two components, one fixed and one variable. The fixed component is called the inertial delay and is constant for a given gate. The second component, termed the transport delay, is a function of the gate fanout. The overall gate delay is then given by (25).

$$delay = inertial + transport \times fanout \tag{25}$$

Refinements to this model may include temperature and process dependencies and statistical variation due to manufacturing tolerances. This model provides the most accurate delay estimate and also the optimum method for determining gate level power dissipation.

Regardless of the delay model used, gate level power estimation may be divided into two categories, as shown in Figure 2.4, termed probabilistic and statistical.

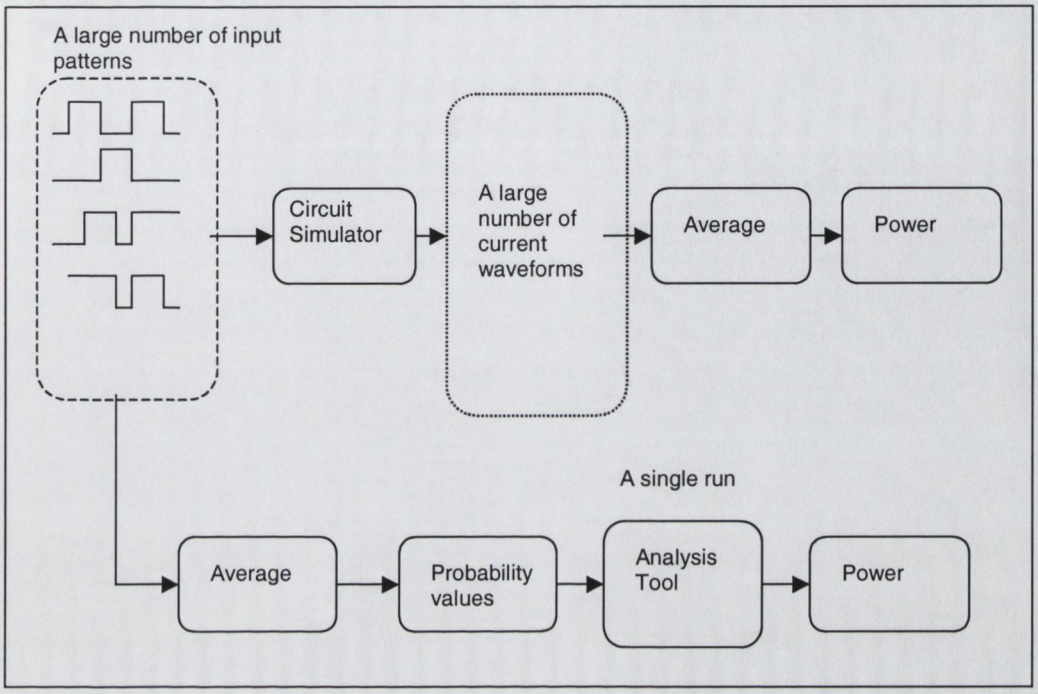


Figure 2.4: Alternatives For Power Estimation

The input in either case is a large number of input patterns, which are applied synchronously. The probabilistic method uses these waveforms to estimate a measure of how often each input bit is likely to switch state. Measures used include the signal probability, transition probability and switching activity. Various algorithms have been proposed using these quantities [Najm94]. Typical probabilistic estimation uses the switching activity $A(x_i)$ which is the probability that the logic signal x_i will switch during any given clock cycle. The switching activity $A(x_i)$ at each circuit node i is calculated and the dynamic power is given by (26).

$$P_{avg} = \frac{1}{2} V_{DD}^2 \sum_{i=1}^{n} C_i A(x_i) \tag{26}$$

The capacitance at each node is estimated using the gate output capacitance, the fanout and an estimate of interconnect capacitance using a wire load model. It has been shown [Najm93] that if the inputs x_i, $i = 1, \ldots, n$, to logic function M, are known, the signal activity $A(f_j)$ at the output f_j of the logic gate may be calculated using (27), where the partial derivative of output f_j is given by (28), and P is the signal probability of the derivative.

$$A(f_j) = \sum_{i=1}^{n} P\left(\frac{\partial f_j}{\partial x_i}\right) A(x_i) \tag{27}$$

$$\frac{\partial f_j}{\partial x_i} = f_j\,|_{x_i=1} \oplus f_j\,|_{x_i=0} \tag{28}$$

The behaviour of the input must be specified by supplying values for the activity of each input bit. As indicated in (28), the partial derivative, also known as the Boolean difference, is calculated using the EXCLUSIVE-OR function. The term $f_j|_{x_i=1}$ is called the cofactor of f_j with respect to x_i, while the term term $f_j|_{x_i=0}$ is the cofactor of f_j with respect to $\overline{x_i}$. The Boolean difference indicates whether f_j is sensitive to changes in input x_i. When the Boolean difference is zero, the function f_j does not depend on x_i and when it is one, the function is said to be sensitised to input x_i. The probability P corresponds to the likelihood that the output is sensitive to input x_i, while the product of P and A is the fraction of the switching that takes place at the output, contributed by input x_i. The overall amount of switching at the output is then given by the summation of each contribution from the n input bits.

The probability P is calculated using an algorithm first proposed by Parker and McCluskey [Park75]. A useful implementation technique for this algorithm employs the Binary Decision Diagram (BDD) [Aker78]. Both algorithm and implementation are outlined next.

As described above, signal probability P is used to determine signal activity, which in turn provides an estimate of dynamic power. Parker and McCluskey's general algorithm for the calculation of P at the internal nodes of a gate-level logic circuit contains three steps.

Algorithm: Compute Signal Probabilities

Inputs: Signal probabilities of all inputs to the circuit

Output: Signal probabilities of all nodes in the circuit

Step 1: For each input signal and gate output in the circuit, assign a unique variable.

Step 2: Starting at the inputs and proceeding to the outputs, write the expression for the signal probability of each gate as a function of its input probability expressions. Use

standard expressions for the output signal probability of each gate in terms of its input probabilities.

Step 3: Suppress all exponents (P^2, P^3 etc.) in a given expression to obtain the correct probability for that signal. The inputs are assumed to be independent of each other and there is no reconvergent fanout, which means that gate outputs do not recombine at further levels in the circuit.

The standard expressions for the output signal probabilities P, as derived by Parker and McCluskey, are given in Table 2.4. Each gate has inputs A and B, with corresponding signal probabilities P_A and P_B. For example, a 2-input NAND gate may be logically viewed as an AND function followed by an inverter. The output probability P_{out}, is therefore given by P_{out}= $1-P_A.P_B$.

Logic Function	Signal Probability
And	$P_A.P_B$
Or	P_A+P_B
Invert	$1-P_A$
Nand	$1-P_A.P_B$
Nor	$(1-P_A)(1-P_A)$

Table 2.4: Signal Probabilities

For anything other than the simplest circuits, the signal probabilities must be calculated using a technique capable of automation. The most commonly used technique uses a circuit description based on the BDD. A brief outline of the background and application of the BDD in the context of this work is now given.

There are three ways of representing Boolean functions: tabular form, logic equations and binary decision diagrams. The truth table is the simplest tabular form, while logic equations are expressions that use combinations of literals, linked by operators such as '+' . A two-level sum-of-products or product-of-sums equation may be used to represent any arbitrary Boolean function. A binary decision diagram is a set of binary-valued decisions, which provides an overall decision, which may be true or false. The decision is made as a result of evaluating a variable. An example of a BDD, implemented as a rooted and directed acyclic graph (DAG) [Demi94], more commonly known as a tree structure, is shown in Figure 2.5. The ordering of the variables in this example is *a,b,c*.

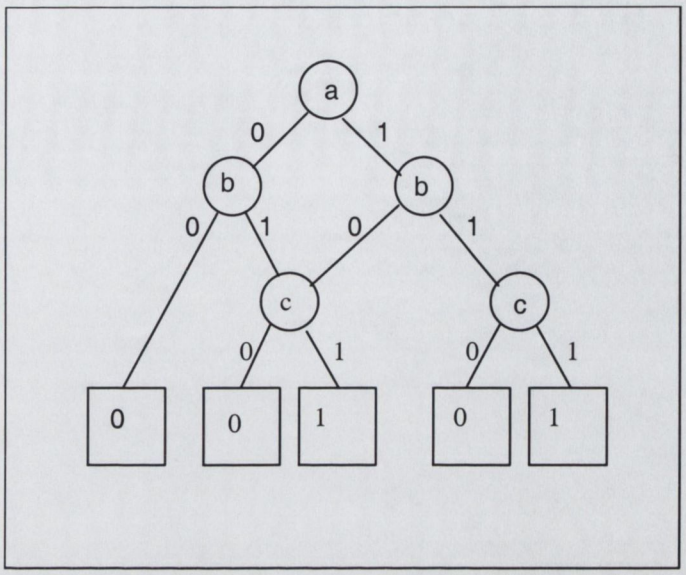


Figure 2.5: Example of a Binary Decision Diagram

The figure shows one form of the expression $f = (a+b).c$. The graph consists of a set of vertices, linked by directed edges. The size of the graph is usually given in terms of the cardinality of the set of vertices, that is, the number of elements in the set. BDDs were originally proposed by Lee [Lee75] and later by Akers [Aker78]. Bryant demonstrated in [Brya86] that an ordering of the variables within the BDD allows efficient algorithms to be developed for manipulating the data. Such diagrams are called Ordered Binary Decision Diagrams (OBDD).

There are two reasons why BDDs are useful in the context of evaluation of Boolean functions. Firstly, for any given Boolean function, a canonical form OBDD can be found that uniquely characterises the function. Secondly, BDD operations can be done in polynomial time of their vertex set cardinality, i.e. their size. This makes them preferable for computer-based solutions when compared to exponential time approaches, as BDD operations require less time to process the graph. Signal probability for any arbitrary Boolean function may be calculated using a BDD.

For example, a function $f(x_1, x_2,..., x_n)$, has n inputs. The function may be represented by Shannon's expansion [Shan48] as indicated in (29).

$$f = x_i \cdot f(x_1,...x_{i-1},1,x_{i+1},...,x_n) + \overline{x_i} \cdot f(x_1,...x_{i-1},0,x_{i+1},...,x_n) \tag{29}$$

The cofactors of f with respect to x_i and $\overline{x_i}$ are defined in (30) and (31).

$$f_{xi} = f(x_1,...x_{i-1},1,x_{i+1},...,x_n) \tag{30}$$

$$f_{\overline{xi}} = f(x_1,...x_{i-1},0,x_{i+1},...,x_n) \tag{31}$$

Using these three equations an expression is obtained for the signal probability of a function f. By successively applying Shannon's Theorem to each term, a simple algebraic expression in $P(x_i)$ and $P(\overline{x}_i)$ is produced. This yields the desired signal probability. Using the BDD, this calculation may be automated, as the underlying form of the BDD follows the Shannon decomposition of the expression, where the remaining tree to the left of a node x_i represents $f(x_i)$ and the tree to the right represents $f(\overline{x}_i)$. When circuit behaviour such as glitching, simultaneous switching of inputs and reconvergent fanout are present, probabilistic techniques can become unwieldy and mathematically intensive. They often represent a lower bound on the actual circuit activity due to their difficulty in dealing with awkward aspects of circuit operation such as gate delays and glitching, as mentioned above. An alternative approach uses a statistical technique that easily incorporates such phenomena. One such method has been adopted in this thesis.

Statistical techniques are based on the repeated use of a functional or circuit simulation of the system. A standard event-driven logic simulator is used to simulate a gate level netlist description of the circuit. The number of transitions occurring at each node during the simulation run is recorded and is used as an estimator for the true circuit activity. These techniques fall into the class of one dimensional Monte Carlo simulation techniques [Sobo94]; therefore, well-known statistical techniques may be applied to the process in order to control the accuracy and confidence of the estimate. This class of statistical simulations also has the ability to terminate the simulation run when a sufficiently accurate result has been calculated, thereby minimising the simulation time required. An important issue in the successful application of Monte Carlo methods is the quality of the input vectors applied to the circuit. When random input data is required, any trace of non-randomness can distort the result [Najm93]. A novel solution to this problem was proposed and implemented as part of this thesis and will be discussed in Chapter3. The typical steps employed in a Monte Carlo simulation are shown in Figure 2.6.

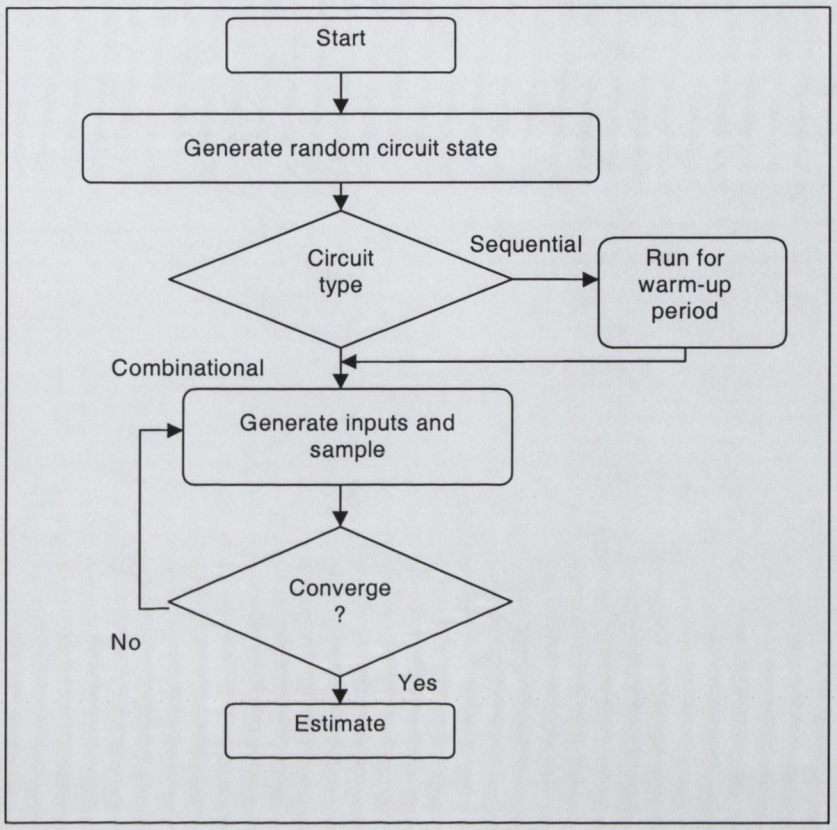


Figure 2.6: Monte Carlo Power Estimation

Simpler versions are possible, depending on requirements. The simulation is initialised in a random state. For many circuits, this step may be omitted, as a power-on reset signal, which initialises all flipflop outputs, forms part of most practical systems. For a combinational circuit and for sequential circuits represented by gate level netlists, the next step is to generate a set of random inputs and initiate a simulation run. For alternative circuit representations, such as state transition diagrams, this step is preceded by a short warmup period, where inputs are applied but no transitions are counted. This has been found to be necessary to obtain meaningful estimates [ibid.]. At the end of the simulation run, a statistical method is used to calculate an average value for the circuit activity and to decide if it is sufficiently accurate, as specified by the user prior to simulation. When this target accuracy has been obtained, the simulation terminates, reporting a value for the average circuit activity. The exact nature of the statistical operations performed will be discussed in detail in Chapter 4, in the context of the Monte Carlo simulator developed for this thesis. Sophisticated delay models are routinely incorporated into gate level circuit simulators, allowing both useful and

glitching transitions to be captured. This provides a good estimate of dynamic power dissipation.

In assessing the two gate level techniques of probabalistic analysis and Monte Carlo estimation, it is apparent that the former is not yet at a stage of development where meaningful circuits may be submitted for analysis. The use of binary decision diagrams requires large processing power, while the approach has yet to be successfully applied to sequential circuits, as well as to problematic combinational circuits with variable delays and reconvergent fan-out. In contrast, the MC technique encompasses all these types of circuits, includes glitching power and may be applied to gate and circuit level descriptions with equal ease. As the MC approach works in tandem with a logic or circuit simulator, the development time is short. Additionally, the same input data may be applied as gate level and then circuit level if required, with little extra effort. The extra time required at the gate level to simulate the circuit is compensated by the accuracy obtained in the results. For an academic project, which may not have the same time scales as its commercial counterpart, this approach represents the best balance between effort and accuracy, which is required for such research activity.

2.7 Circuit Level Power Estimation

Circuit simulation, in the context of CMOS technology, is based on a physical model of the enhancement-mode MOSFET transistor. The circuit is represented as a set of n- and p-channel devices, interconnected by capacitive wires. The behaviour of each device is governed by a set of device equations, each of which has a number of input parameters. The set of input parameters for any given device is referred to as its device model. Currently, the program SPICE [Nage75] and commercial variants, is the most widely encountered circuit simulator. The behaviour of the circuit is captured as a series of time-domain waveforms, representing selected voltages and currents within the circuit. These waveforms, which are the graphical version of sets of time-voltage and time-current pairs, are obtained by the numerical solution of sets of simultaneous equations for each node in the circuit. When the current waveform of the power supply is specified for simulation, the resulting data may be numerically integrated, for example using trapezoidal integration, to estimate the average power consumption for that simulation run. All three power dissipation mechanisms, namely

the static, dynamic and short-circuit components, are included in the device model and so this technique produces an accurate estimate of the total power dissipation. However, the circuit simulation is slow and only a limited number of devices can be simulated, typically less than one thousand [Schw01]. Also, because the input patterns employed have a large influence on the current that flows, this method may take more iterations of the Monte Carlo loop to reach a power estimate of specified accuracy. Figure 2.7 shows a typical power supply current waveform, together with the average value of that waveform.

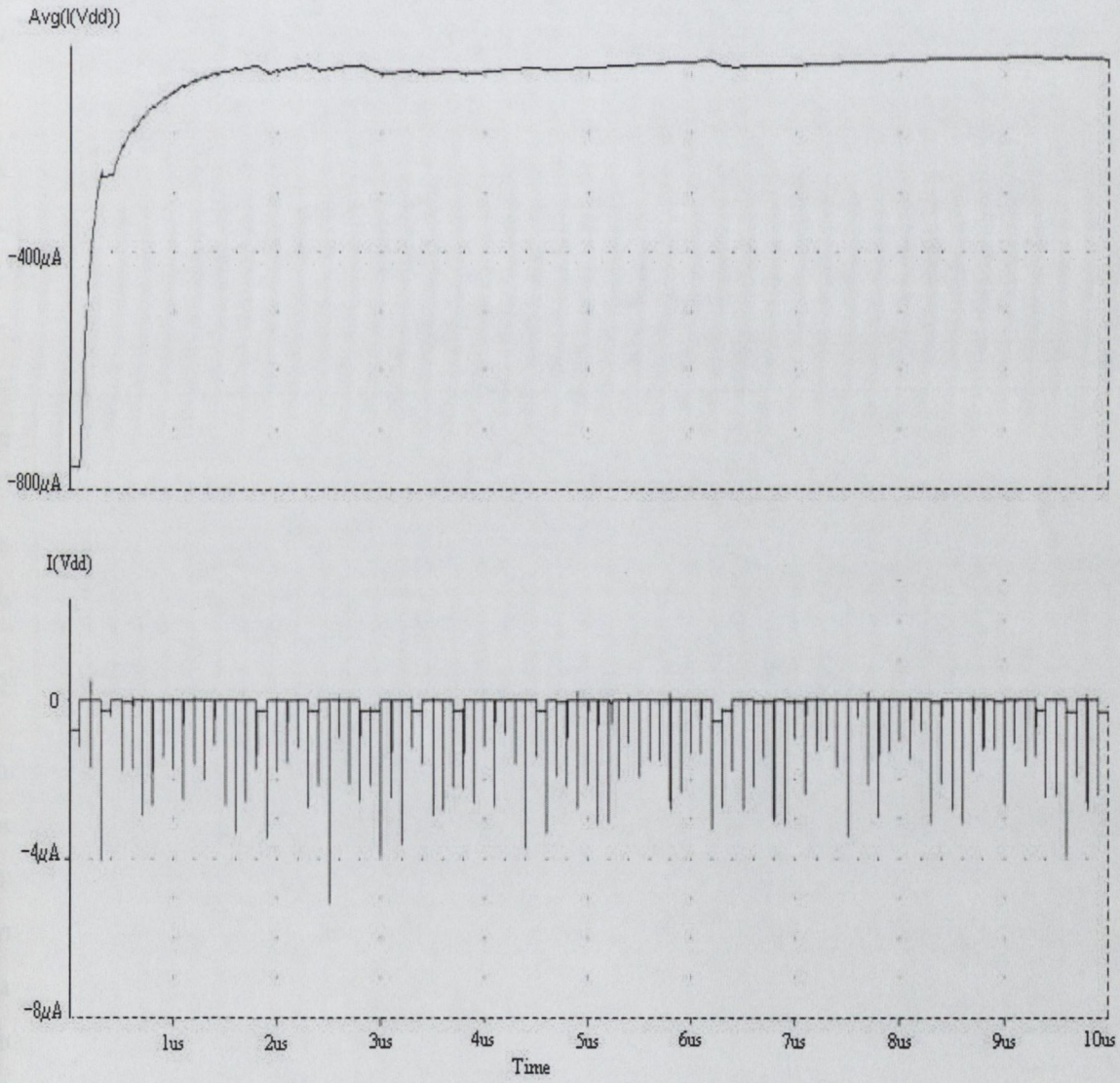


Figure 2.7: Average And Instantaneous Current Waveforms

It can be seen that the average current converges towards a value, which may be used to calculate the average power consumption for the simulation run. This is obtained by forming

the product of the average current obtained at the end of the simulation and the magnitude of the power supply voltage V_{DD}. The strong dependence of the instantaneous current on the input patterns can also be seen from the lower plot. Specifically, the magnitude of the current will be a function of the input bit pattern and its successor. Each input bit that toggles causes energy-consuming logic transitions within the circuit. If two successive input patterns are similar, then relatively few transistors will change state and the magnitude of the current will be correspondingly small for that input pair. This is also the reason why a UWN input, with maximum changes between input pattern pairs, causes the highest level of average dynamic power consumption.

Circuit level analysis was the first type of analysis available to ASIC designers and is still an important part of the design process. However, its relevance today is more at the level of standard cell characterisation, mixed-signal interfaces and analogue design. When used in conjunction with a gate level simulator in a Monte Carlo environment, the circuit simulator provides unrivalled accuracy at the cost of large simulation times. For an academic project, such as this work, the exactitude of the results must be rated more highly than reduced simulation time. Therefore, the combined approach of gate level plus circuit level simulation, with a Monte Carlo program to specify the error and confidence of the power estimation results, represents the optimum trade-off between time and accuracy. This is the approach adopted during this work.

2.8 Summary and Conclusion

The origins of power dissipation in CMOS circuits were shown to be caused by the flow of energy to and from the capacitive structures inherent in all circuits. The three components of power dissipation, namely static, leakage and dynamic were identified and quantified. The dynamic component was identified as being of most relevance in this thesis. A distinction was drawn between power dissipation and power consumption, although these terms are widely used interchangeably. Power estimation was introduced as an indispensable aid to the integrated cicuit designer. Estimates are obtained at various times in the life cycle of a design, with consequent trade-offs in speed and accuracy. Power estimation is necessary at all stages of the design cycle, from high level software estimation which analyses individual instructions, down to circuit level techniques based on the magnitude of the power supply

current. The gate and circuit levels of design were explored in detail, as they form the bulk of the design descriptions used in this thesis. At this level, two techniques for power estimation predominate. Probabilistic techniques estimate the activity at circuit nodes by applying a static analysis of the signal probabilities throughout the circuit. These techniques have difficulty in dealing with circuit delays and glitching, which contribute to the overall power dissipation in a real circuit. They also become very slow when analysing large circuits. In contrast, statistical techniques use existing circuit and logic simulators to run several simulations while monitoring circuit current or node activity. They provide an accuracy that is limited only by the simulator. Gate level logic simulations in particular are very useful in the context of statistical power estimation, allowing thousands of gates to be simulated in a reasonable time. Circuit level simulation, as provided by SPICE, gives very accurate power estimates in conjunction with statistical simulation. The next chapter will present the power estimation tools developed and used as part of this thesis, together with a method for generating input data for such simulations.

3 Random Binary Vectors

As described in the previous Chapter, the Monte Carlo class of parameter estimation techniques is based on the application of random digital signals to the circuit inputs. The parameter to be statistically evaluated, which in this work is the average power dissipation of a circuit, is estimated at the end of each simulation run. This power estimation process terminates when an error measure reduces to a pre-determined value. The use of such simulations, carried out with random or uniform white noise inputs, is a well-established technique with many other applications, such as [Bind88], [Bori83] and [Oliv81]. The testing of circuits with UWN is also an important comparative evaluation technique [Burd94]. It is of particular relevance when selecting between alternative architectures and when, early in the design process, real data may not be available. In order to assign a power rating to a circuit, however, typical data for the application must be applied. When such statistical methods are used to estimate the average power, the quality of the input data can have a profound influence on the answer obtained [Najm94]. Data which is not sufficiently random can produce biased results, while appearing to behave correctly by converging to the wrong answer. Such inconsistencies can be difficult to detect and suspect simulation results must be re-run with an alternative source of UWN input data in order to obtain consistent results with minimal error. This process is very time consuming. In order to minimise the likelihood of inaccurate power estimates, it is vital to ensure the quality, i.e. the randomness of the input data source. The solution is to have available a set of well characterised generators that consistently produce acceptable results. The method of production should also be straightforward, to minimise the impact of this peripheral process on the power estimation phase and consequently on the development time of the design.

While there are many individual examples in the literature of methods of producing good numbers, generally acknowledged to have been started by Lehmer, [Lehm51], no general method for producing good generators has been encountered. To build a power estimation tool, it would have been necessary to code into the application a specific generator, such as those proposed by Marsaglia [Mars68] or Park [Park88]. This work has developed a method for the creation of any number of random generators, each with pre-determined and consistent quality. This represents a significant increase in flexibility when using random

data, as results may now be cross-checked several times with a fresh source of high quality data. The implementation of a novel method for obtaining generators using a genetic algorithm is the initial focus for the work described in this Chapter.

The attributes of the data source must then be rigorously evaluated before use, as there are many instances in the literature where poor quality, or insufficiently tested, data has caused difficulties in the context of simulation [Vatt95], [Park88], [Dagp88]. The data should therefore be subjected to a series of demanding tests for randomness, using a variety of techniques [Nied92], [Lecu91], [Savi83], [Cham67]. There are two approaches available for such analysis of random binary data. The first comprises a collection of standard statistical tests such as Kolmogarov-Smirnov goodness-of-fit [Mass51], quantile-quantile plots [Kenn87] and tests of linearity on the cumulative distribution function that may be formed from the sampled power spectral density [Widr85]. The second type of randomness test is structural, and analyses the properties of the generator parameters. This type of test examines the lattice structure that is inherent in all pseudo-random sequences. The lattice test and the spectral test are among this class of tests [Atki79], [Cove67]. All tests are discussed in detail in later sections of this Chapter.

This Chapter contains five linked topics corresponding to the major sections of the Chapter. Firstly, a limited definition of randomness is presented with a visualisation technique for examining data in Section 3.1. Then, in Section 3.2, the equation used to generate random integers is described. Next, a test known as the Spectral Test is given in Section 3.3 for evaluating the parameters of the equation. A general method for the selection of the equation parameters using a genetic algorithm is then developed in Section 3.4, using the Spectral Test to evaluate parameters within the genetic algorithm. Finally in Section 3.5, comprehensive test results from the generators produced by the genetic algorithm are presented. The structure of the material presented in this Chapter is summarised in the diagram of Figure 3.1.

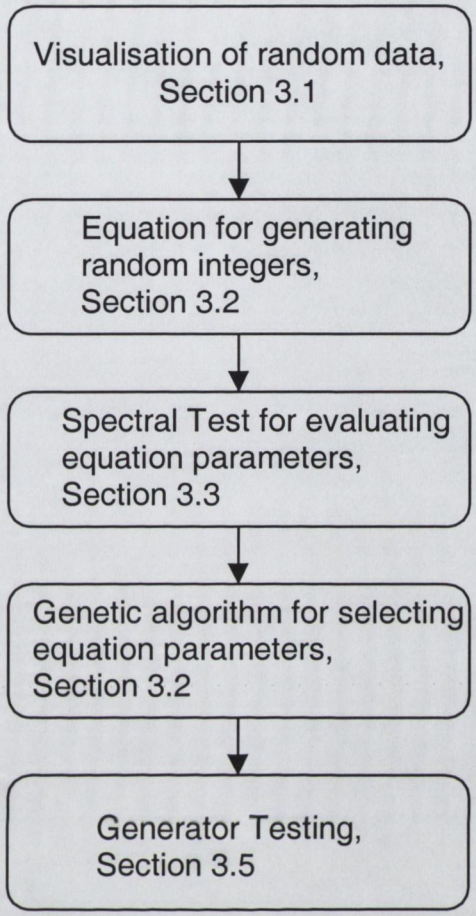


Figure 3.1: The Structure of This Chapter

Section 3.1 now deals with a visualisation technique that provides a means of comparing generators and their output.

3.1 Characteristics and Visualisation of Random Binary Data

While Nature contains many random processes such as Brownian motion, transistor noise and solar flares, Science has yet to successfully mimic those processes by supplying a method for producing truly random numbers. The best that can be produced is a pseudo-random sequence, which can pass an arbitrary number of tests for randomness and which has sufficient statistical properties to make it appear adequately random for a given application [Hamm67]. These sequences are finite and periodic, even though generators with large periods have been developed [Mars94], [Komo91], [Robe82]. The period alone however, is no guarantee of randomness in a data sample from that sequence [Hull62]. Thus the problem becomes one of generating a pseudo-random sequence, followed by the rigorous and

comprehensive testing of the data produced. In addition, because VLSI systems have a finite number of inputs, any random digit so produced must be truncated to form a binary input vector with the appropriate number of input bits. As the most significant bits of a pseudo-random pattern provide better UWN patterns, the least significant portion of the data is usually removed. The effect of this truncation on the sequence is to reduce the quality of the data, by effectively throwing away the randomness contained in the least significant bits [Knuth81]. Using untruncated data is not a solution either and may also cause error, as the period becomes small. For example, a circuit with an 8-bit input may be driven by a generator with exactly eight bits, but the data repeats after only 256 samples, which is the maximum period for such a generator. This may be a significant source of error in the power estimate when larger sequences are required [Smit71].

The generation of circuit input data consists of a two-step process. Firstly an integer sequence with a large period of the form 2^m, is produced. As m is greater than the number of circuit inputs for large periods, the data generated must be truncated to form an input with the correct number of bits. The second step in the production of input data involves the sampling of smaller truncated subsequences from within the larger sequence. This smaller sequence is then applied to the circuit and a simulation is performed.

A useful aid for visualising randomness in a data sequence is the two-dimensional plot obtained by plotting the data in a number of ways [Rich89], [Voel88], [Fol84]. The data may be plotted against its sequence number, or a set of 2-tuples can be obtained by pairing the n^{th} element of the sequence, R_n with its neighbour, R_{n+1}. These plots are also extended to three dimensions, where each point consists of a sample, its predecessor and its successor in the sequence. Qualitative judgements may be made concerning the randomness of the sequence, even with this simple technique. When the data is truncated to a fixed number of bits, a larger measure of visual regularity appears in the plot, indicative of the decrease in randomness. Again, at this point a visual check of the data is possible. The following series of diagrams illustrates the concept. Two generators, X and Y, are compared untruncated. Generator X has been chosen so that it is of poorer quality than generator Y. Figure 3.2 shows the output from each generator, plotted against sequence number. The differences at this stage are not immediately obvious, as both sets of data appear to be evenly distributed over the plot, with no apparent order.

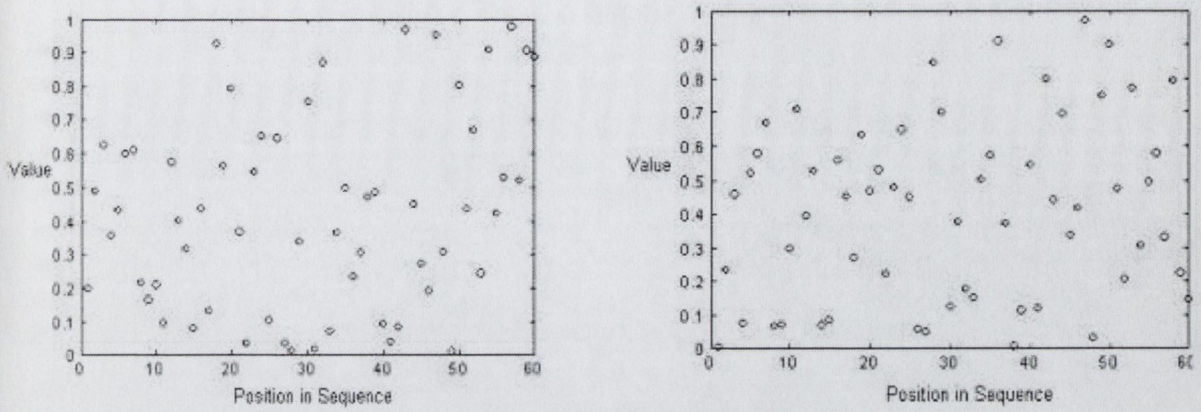


Figure 3.2: Generators *X* andY Output

The next two diagrams in Figure 3.3, show two-dimensional plots of the 2-tuples formed by neighbouring elements of each sequence. Some differences between the generators are visible at this stage. The plotted vectors for generator *X* lie in a series of lines. Figure 3.4 then plots the sequences in three dimensions. Note the regular structure of generator *X*. In this sample the data forms a series of planes, separated by a fixed distance. Neither sequence has been truncated, which means that any non-randomness is inherent in the generators themselves.

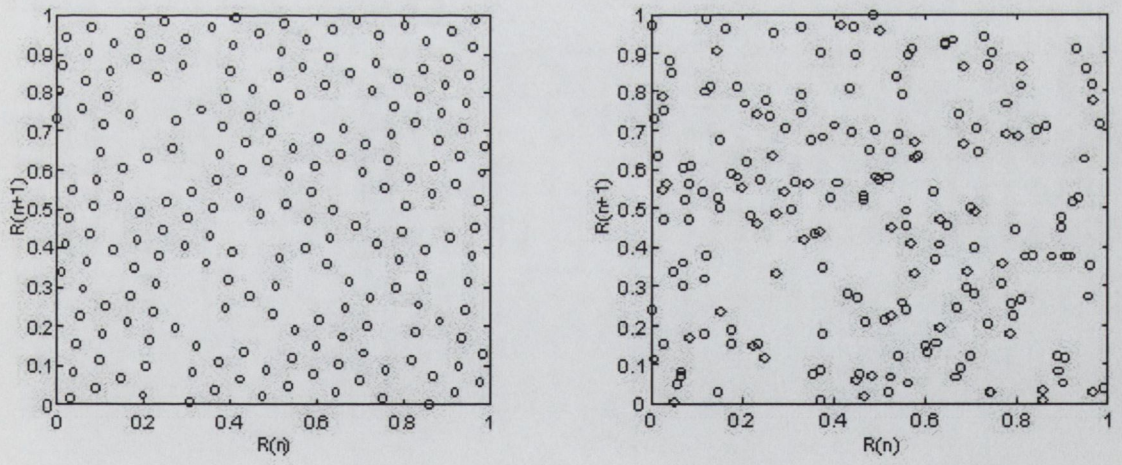


Figure 3.3: Generators *X* and *Y* Output Plot in Two Dimensions

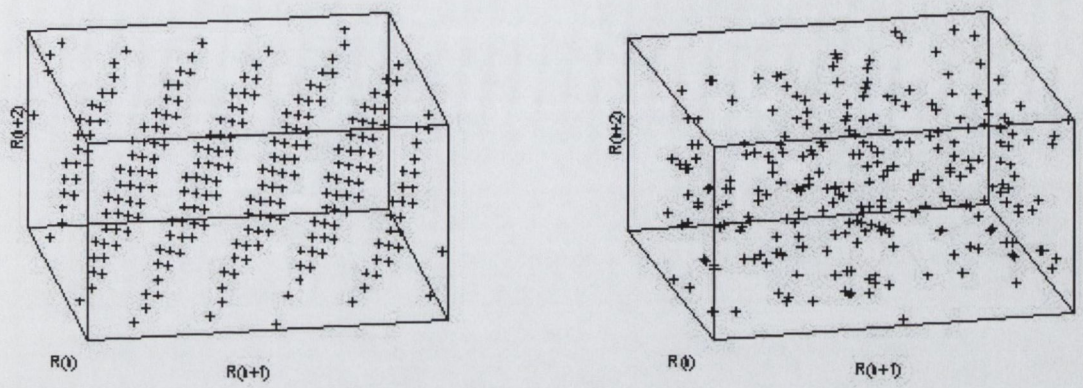


Figure 3.4: Generators *X* and *Y* Output Plot in Three Dimensions

The final diagram, Figure 3.5, shows generator *Y*, having been truncated to eight bits. Again the planar structure appears, demonstrating that this artefact is an inevitable consequence of the use of truncation. Generator *Y* is the 'rand' function provided by the Matlab algorithmic prototyping environment [Math95], while *X* is a linear congruential generator of low period.

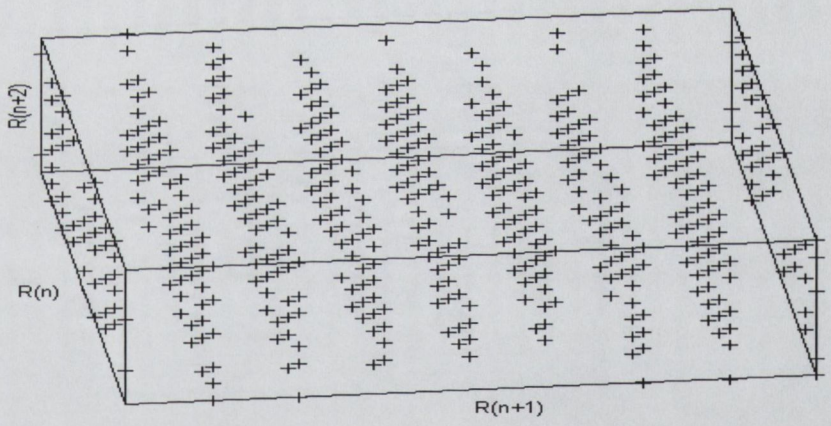


Figure 3.5: Truncated Output of Generator *Y* in Three Dimensions

The amount of regularity inherent in a sequence may be used as a measure of quality for the data. The points for generator *X* in Figure 3.3 for example, may be covered with several different sets of parallel lines. The maximum distance between lines taken over all sets of lines, represents a worst-case measurement of the resolution of the data. In particular, this maximum interplanar distance is of significance when selecting a generator, with smaller inter-planar distances indicative of better data [Cove67]. This concept may be extended into three or more dimensions, for a more rigorous test of the generator. The closer the worst-case

set of planes are to each other, the better the properties of the sequence [Knuth81]. The maximum inter-planar distance is a useful figure of merit when searching for a suitable pseudo-random generator. In general, the maximum inter-planar distance is measured in six or more dimensions, where the data forms a set of parallel hyper-planes. The equation used to generate this data is now discussed.

3.2 The Linear Congruential Equation

Typically, when running Monte Carlo experiments it can be important to verify results with alternative sources of random data. This provides a means of verifying the properties of the input data while reducing the likelihood of a spurious result from the simulation, which may occur for a variety of reasons [Ding98b]. The technique for generating simulation input data, developed for this thesis, consists of a method that provides as many sources of random data as are required, all with equally good statistical properties. Therefore, experimental results may be cross-checked as often as required, with a different generator of consistent properties each time.

The method used to produce the simulation inputs is based on the generation and truncation of pseudo-random integers and is an application of Lehmer's Linear Congruential Generator (LCG) scheme [Lehm51]. A sequence $\{R_n\}$ is obtained as shown in (32).

$$R_{n+1} = (aR_n + c) \bmod m \tag{32}$$

The sequence is initiated by a start-up value R_0 called the seed. The modulus m determines the range of values generated. Correct choice of multiplier a and increment c will ensure that all integers 0 to m-1 occur before repetition, which is the maximum period for this type of generator. In practice, this may be achieved by setting $m = 2^r$, $c = 1$ and $(a \bmod 8)=5$, where r is an integer [Dagp88]. The value of a has a strong influence on the quality of the numbers generated for a given value of the period m [Cove67]. For example, Figure 3.6 indicates the variation in power estimates produced by using different LCG generators to drive a particular circuit. Each generator used a different multiplier and would therefore produce a different set of inputs. The circuit is a 10-bit adder implemented in a 0.8μm CMOS technology.

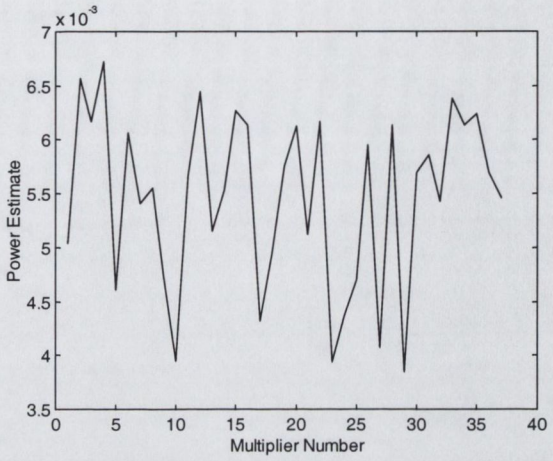


Figure 3.6: Variation in Power Estimates with Generator Multiplier

The power estimates were produced by driving the circuit with 38 different 32-bit generators (m=2^{32}), truncated to 10 MSBs. Each result is the output from a Monte Carlo simulation run with a pre-defined error limit of 10% and a 95% confidence interval. The multiplier a, for each generator, was a different 32-bit number. The estimations of average power ranged from approximately 3.8mW to 6.7mW. This represents a variation between the highest and lowest power estimates obtained of more than 75%. The error bound of 10% should have ensured that the maximum variation observed was at most 10%. This is clearly not the case, which leads to the conclusion that the input data was compromised due to the different multipliers, leading to termination of some simulations with the wrong answer. While the choice of multiplier is important, the increment term c does not have a strong influence on the quality of the numbers generated [Knuth81].

In general, numbers with a higher resolution than that required must first be obtained and then truncated to give the appropriate number of binary inputs for the circuit under test, as discussed previously. For $m = 2^r$, where r is an integer, n-bit truncated binary data is obtained from the original integer sequence U_1, U_2, U_3... as indicated in (33), where the *floor()* function implies a rounding down to the nearest integer.

$$R_i = floor\left(U_i \times 2^{n-r}\right) \quad for\ n < r \tag{33}$$

If a multiplier a of sufficient quality is used in the original integer sequence, then the binary data so produced will also be suitable for Monte Carlo estimation [Najm95]. Furthermore, a sequence with the maximum period m-1 is produced when the mathematical conditions on the multiplier, stated previously, are observed. The task of generating suitable

input data then reduces to that of finding a good multiplier a for the intended application. The quantitative Spectral Test used to evaluate multipliers is now discussed in detail, in Section 3.3.

3.3 The Spectral Test

The aim of this Section is to describe a useful test of the properties required of a pseudorandom, periodic sequence for use in a Monte Carlo simulation. There are many varied tests and definitions for the random qualities of a set of data, but the approach chosen for this thesis uses what is generally considered to be the most rigorous approach. This test, known as the Spectral Test, is based on the spectral quality of the data, which in itself is a familiar starting point for many engineering applications. In the Spectral Test, data is examined for independence [Sher93] between successive samples, as well as between samples separated by an arbitrary number of intermediate samples. This approach has the added advantage that it closely relates to the visualisation of data described in Section 3.2 while not limiting the analysis to three dimensions. Using the concept of accuracy, the link between the planar plots of pseudorandom data and the Spectral Test is presented. The mechanism of the test is then given together with some implementation issues encountered. First, the discussion of planar visualisation of data is now extended to define the term accuracy of data.

3.3.1 Visualisation and Accuracy

The qualitative visualisation presented earlier is now developed into a quantitative measure of data quality, using the concept of accuracy as a link. As a starting point, the linear congruential sequence with maximum period length, introduced earlier, may be defined by a set of four parameters (R_0, a, c, m), where

R_0 = the initial value in the sequence, known as the seed.

a = multiplier

c = a constant term in the equation

m = the period of the sequence.

Using the linear congruential expression of (33), the sequence $\{(R_n, R_{n+1},\ldots, R_{n+t})\}$ may be written as indicated in (34), where $s(x)$ is the successor of x in the sequence, as given in (35).

$$\left\{\left(x, s(x), s(s(x),\ldots s^{t-1}(x)\right) \mid 0 \le x < m\right\} \tag{34}$$

$$s(x) = (ax + c) \bmod m \tag{35}$$

Hence a 2-dimensional plot of the generator output consists of the set of all points $\{x, s(x)\}$, while a volumetric plot appends a height component by plotting the set $\{x, s(x), s(s(x))\}$. These 2- and 3-dimensional were plotted in Figures 3.2 to 3.8 in Section 3.1.

It is important to reiterate at this point the significance of this planar structure. Given a hypothetical source of truly random numbers between 0 and 1, this planar structure would not be present. However, when these numbers are truncated to a finite accuracy, so that each one is an integer multiple of $1/v$ for some number v, then the data plotted according to (34) above will be planar [Atki79]. Because a pseudorandom generator has a finite sequence length, the same planar structure is observed. For example, a pseudorandom generator with period m can be viewed as producing data in the range 0 to 1, where each number is an integer multiple of $1/m$. This data will also produce a planar structure. Therefore, the planar structure, and in particular the interplanar distance, is a good measure of how closely the pseudorandom approximates the true, truncated random data. The closer the planes are to each other, the finer the granularity of the data and therefore the more random it is. In two dimensions, for example, the structure consists of a set of points in 2-dimensional space, i.e. on a plane. The points may be covered by a number of different sets of parallel lines. In practice, the two-dimensional accuracy of the generator is given by v_2, where $1/v_2$ is the maximum distance between these lines, having examined all families of parallel lines that cover the points $\{(x/m, s(x)/m)\}$ in two dimensions. The diagram of Figure 3.7 shows the two-dimensional accuracy for a simple generator.

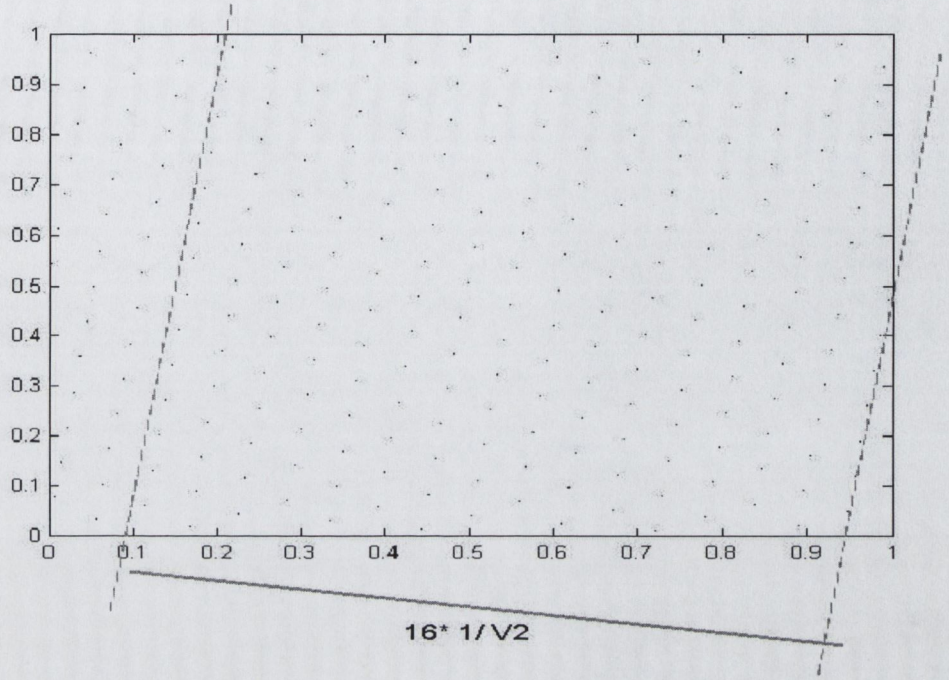


Figure 3.7: Example of Two Dimensional Generator Accuracy

The distance between 16 parallel lines was measured, and the distance between lines, $1/v_2$, was measured to be approximately 0.056. This yields an accuracy of $v_2 = 17.7$. This graph was generated using the Matlab RNG function *rand*, with truncation to 8 bits before plotting. It is an indication of the quality of the *rand* generator that the accuracy, at 17.7, is better than the maximum accuracy obtainable from an untruncated 8-bit generator with $m = 2^8$. Such a process can provide a maximum value of $v_2=m^{1/2}=16$. This result indicates that the data in Figure 3.7 originated in a generator with a much larger period. In a similar manner, the accuracy in three dimensions is given by v_3, where $1/v_3$ is the maximum distance between planes, taken over all families of parallel planes that cover all points $\{(x/m, s(x)/m), s(s(x))/m)\}$. In general, the t-dimensional accuracy v_t is the reciprocal of the maximum distance between hyperplanes, taken over all families of parallel t-1 dimensional hyperplanes that cover all points $\{(x/m, s(x)/m), \ldots, s^{t-1}(x)/m)\}$.

The primary difference between a periodic sequence and truly random truncated sequence is that while the accuracy of the latter is the same in all dimensions, the accuracy of the periodic sequence decreases with increasing dimension t [Knuth81]. With a period of m, there are m points to be distributed in a t-dimensional cube. Therefore, the best distribution of points that can be achieved is a uniform placement of points, separated by a distance of $m^{1/t}$.

This value is used in (36) to determine $n_{t(max)}$, the maximum number of random bits obtainable from an LCG generator, when t-dimensional accuracy is considered.

$$n_{t(\max)} = \log_2 m^{1/t} \tag{36}$$

For example, a generator with period $m = 256$ has a maximum 2-dimensional ($t = 2$) accuracy of $256^{1/2} = 16$, with 4 random bits.

The example in Table 3.1 demonstrates this concept. Three cases are considered. A hypothetical perfectly random number generator, truncated to n bits, has n-bit accuracy in all dimensions and is used as a reference. In this example $n = 8$ so the hypothetical sample has 8-bit accuracy in each dimension. LCG generators have a maximum accuracy of only $\log_2 m^{1/t}$ bits, where m is the period and t is the dimension number. These maximum values are calculated for two values of the period, 256 and 4096.

Generator	*Max.* v_2	$\log_2 v_2$	*Max.* v_3	$\log_2 v_3$
Hypothetical generator	256	8	256	8
LCG with period = 256	16	4	8	3
LCG with period = 4096	64	6	16	4

Table 3.1: Maximum Two and Three Dimensional Accuracy

The table indicates that in two dimensions, the maximum LCG accuracy for a period of 256 is 4 bits in two dimensions and 3 bits in three dimensions. This type of generator is therefore equivalent to a perfectly random source, when comparing dimensions 2 and 3, only when the three most significant bits of each are compared. In other words, such a source would provide at best only a 3-bit generator. When the period is increased to 4096 the characteristics are better, in that 6 random bits are potentially available in dimension two and 4 random bits in dimension three. It is apparent that large periods are required if typical circuit input vectors of 16 or more bits are to be accommodated with the LCG method. Using (36), the minimum period required for a 16-bit generator when three dimensions are considered is 2^{48}. Suitable LCG parameters must then be found for this period if this level of accuracy is to be attained.

As stated earlier, pseudorandom data exhibits decreasing accuracy with increasing dimension. In Table 3.1, only three dimensions were considered for the sake of clarity. In general, the question must be asked, how many dimensions are significant and how many should be considered when searching for a suitable generator. The spectral test can measure the accuracy in any dimension and is to be used as the objective function of a genetic algorithm; therefore, this is an important issue. The question, reformulated empirically, is how much accuracy do we need from a generator. The answer, based on such practical considerations as ease of implementation, computing power and experiences documented within the literature [Dagp88], [Atki80] would suggest that upper dimensions are less important and diminish in contribution to the accuracy of the simulation as the dimension increases. For dimensions higher than 3, it appears to be satisfactory to have some of the most significant bits random. For any given n-bit input, there exists a generator with period $m>>2^n$, such that the above conditions may be satisfied. In practice, if a large enough period is used, then the n most significant bits of that generator may be used to meet the required randomness criteria. The maximum accuracy in dimension t is $m^{1/t}$. It may therefore be deduced that squaring the period will square the potential accuracy in higher dimension, i.e. the number of useful bits is doubled. This gives a useful value for the period when searching for a generator. When a candidate generator has been identified, it is evaluated using a test of dimensional accuracy based on a spectral analysis. The test is now described.

3.3.2 Implementing The Spectral Test

Knuth states, in [Knuth81], concerning the spectral test, that, "not only do all good (linear congruential) generators pass this test, all generators now known to be bad actually fail it. Thus it is by far the most powerful test known...". As previously stated, the quality of the data used to drive a Monte Carlo simulation can affect the answer obtained. Therefore, any generator proposed must be rigorously tested. There are many theoretical tests of generator properties, in addition to empirical tests of samples of the generator output. The spectral test however, is the only test that examines the full period of the output, while also examining the theoretical properties of the generator. In this way, it is by the far the most demanding test of a linear congruential generator. By addressing the properties of the full period of the sequence, it is a rigorous, structural test of the generator properties [ibid.]. In addition, the use of a computer program to calculate the most important parameters of the generator makes it a

strong empirical test. Knuth refers to the implementation of this procedure as Algorithm *S* (The Spectral Test) [ibid.] and it is now described.

The objective of the algorithm is to determine the value of v_t^2, such that (37) is satisfied for values of t in the range $2 <= t <= T$ given a, m and T, where $0 < a < m$ and a is relatively prime to m, as discussed in Section 3.1.

$$v_t^{\,2} = \min\left\{\sqrt{x_1^{\,2} + \ldots + x_t^{\,2}} \,\middle|\, x_1 + ax_2 + \ldots + a^{t-1}x_t^{\,2} = 0 (\mathrm{mod}\, m)\right\} \tag{37}$$

The number v_t then measures the t-dimensional accuracy of the generator. The algorithm comprises eleven steps and functionally has two sections. The eleven-step algorithm is now outlined. An example of the program operation is also given.

Step 1 [Initialisation] Set $h <= a$, $h' <= m$, $p <= 1$, $p' <= 0$, $r <= a$, $s <= 1 + a^2$, $t = 2$.

Step 2 [Euclidean step] Set $q <=$ round_down(h'/h), $u <= h' - qh$, $v <= p' - qp$. If $u^2 + v^2 < s$, set $s <= u^2 + v^2$, $h' <= h$, $h <= u$, $p' <= p$, $p <= v$ and repeat Step 2.

Step 3 [Compute v2] Set $u <= u - h$, $v <= v - p$, and if $u^2 + v^2 < s$, set $s <= u^2 + v^2$, $h' <= u$, $p' <= v$. Then output $v^2 = s^{1/2}$.

Step 4 [Advance t] If $t = T$, the program is finished. Set $t <= t+1$ and $r <= (ar) \mathrm{mod}\, m$. Set U_t to the new row $(-r,0,0,\ldots,0,1)$ of t elements and set $u_{it} <= 0$ for $1 <= i <= t$. Set V_t to the new row $(0,0,\ldots,0,m)$. Set $q <=$ round($v_{i1}r/m$), $v_{it} <= v_{i1}r - qm$ and $U_t <= U_t + qU_i$, where the function *round*() rounds to the nearest integer. Set $s <= \min(s, U_t \cdot U_t)$, $k <= t$ and $j <= 1$.

Step 5 [Transform] For $1 <= i <= t$, do: if $i <> j$ and $2|V_i \cdot V_j| > V_j \cdot V_j$, set $q <=$ round$(V_i \cdot V_j)/V_j \cdot V_j)$, $V_i <= V_i - qV_j$, $U_j <= U_j + qU_i$ and $k <= j$.

Step 6 [Examine the new bound] If $k = j$, set $s <= \min(s, U_j \cdot U_j)$.

Step 7 [Advance j] If $j = t$, set $j <= 1$; otherwise set $j <= j + 1$. If $j <> k$, return to Step 5.

Step 8 [Prepare for search] Set X <= Y <= $(0,\ldots,0)$, set $k <= t$ and for $1 <= j <= t$, set

$$z_j = floor\left(\sqrt{(V_j \cdot V_j)s/m^2}\right) \tag{38}$$

Step 9 [Advance x_k] If $x_k = z_k$, got to Step 11. Otherwise increase x_k by 1 and set $Y <= Y + U_k$.

Step 10 [Advance k] Set $k <= k + 1$. If $k <= t$, set $x_k <= -z_k$, $Y <= Y - 2z_kU_k$ and repeat Step 10. But if $k > t$, set $s <= \min(s, Y \cdot Y)$.

Step 11 [Decrease k] Set $k <= k - 1$. If $k >= 1$, return to Step 9. Otherwise output $v_t = s^{1/2}$ and return to Step 4.

3.3.3 Sample Results of the Spectral Test

The algorithm of the spectral test was implemented in a number of programming languages (ANSI C and Matlab) and on different platforms (Windows and Unix), in order to characterise its behaviour for different applications. As a result of this investigation, an analysis of the operational limits of this algorithm is given, together with some suggested modifications that are intended to improve the performance of the various implementations. Table 3.2 shows the output from the spectral test for various generators.

Gen.	a	m	$\log_2 v_2$	$\log_2 v_3$	$\log_2 v_4$	$\log_2 v_5$	$\log_2 v_6$
1	229	2^8	3.99	2.20	1.29	1.29	1.28
2	1445	2^{16}	7.34	5.17	3.78	2.98	2.40
3	10058589	2^{24}	12.09	7.09	5.88	4.12	3.84
4	2265756725	2^{32}	16.10	8.81	7.36	6.04	4.39
5	813558501373	2^{40}	20.07	13.04	9.29	7.06	5.46

Table 3.2: Spectral Test Results for Various Generators

The results are interpreted as follows. For generator 4, for example, $\log_2 v^2 = 16.1$ bits. Therefore, only the two most significant bytes are random, when considering two dimensions. The other results for this generator show decreasing accuracy with increasing dimension. Generator 4 is therefore suitable for a circuit with 16 input bits or less.

Execution times for the spectral test vary with the magnitude of the period m and the number of dimensions specified. The slowest version was implemented on a modestly specified personal computer based on a Pentium II with 64MB of RAM using Matlab. As the code is interpreted rather than compiled, the measured execution time should be considered as an

approximate upper bound for the computation time. The execution times for the generators in Table 3.2, using Matlab, are provided in Table 3.3 below.

Generator	1	2	3	4	5
Execution Time	0.11s	0.06s	0.11s	0.39s	0.6s

Table 3.3: Spectral Test Execution Time

The execution times for five different multipliers using the same period, and hence for five different searches of the same solution space, are given in Table 3.4. The period is 2^{48}.

Multiplier	1	2	3	4	5
Execution Time	0.72s	0.77s	0.93s	0.82s	0.66s

Table 3.4: Spectral Test Execution Time for Various Multipliers

The algorithm is fast. However, the computational accuracy of the algorithm, as presented by Knuth [ibid], does cause some difficulties when large numbers are involved. The principal difficulty arises because of the magnitude of the numbers generated internally by the various multiplication steps. There is also a difficulty when Matlab is used, as the accuracy of the functions is dependent on the magnitude of the parameters passed to the various functions. This problem is explored and a number of solutions proposed which allow numbers as large as 2^{100} to be accommodated by the spectral test.

3.3.4 Implementation Issues with the Spectral Test.

The various implementation issues referred to at the end of the previous section are now discussed, starting with the calculation of the modulo arithmetic needed to calculate the next random integer with the LCG equation. This equation provides a series of pseudorandom integers using the relation in (39).

$$Next_integer = (a \times Current_integer + c) \text{ modulo } m \tag{39}$$

In Matlab, for example, this is implemented with the following code:

$$Next_integer = \text{rem}(a \times Current_integer + c, m) \tag{40}$$

where the built-in function $rem(x,y)$ is $x\text{-}y.\times\text{fix}(x./y)$ if $y \neq 0$. The *fix* function rounds towards zero. In addition, Step 4 of the spectral test algorithm includes the multiplication and modulus operations $r \leftarrow (a \times r) \bmod m$. A difficulty arises when the operands in either function are

large. For example, the code *rem*(2447152529^2, 2^{40}) returns a value 548323073024. This is less than the correct result 548323073057. As a further example, *rem*(680011341^2, 2^{32}) returns 3347772224, which is larger than the correct result 3347772201. The error is caused because the product $a \times r$ cannot grow larger than the maximum integer within Matlab. When it does, *rem*() returns an incorrect value. As this function is a built-in precompiled operation, the source code was not available for modification or examination to determine the exact cause. However, an alternative approach was developed, as outlined below.

The maximum value taken by both the multiplier a and r is approximately m and this product must not exceed the biggest integer representable in the programming environment in use. This means that the modulus m must not exceed the square root of the maximum integer within the system. For example, at the time this work was carried out, using Matlab IV, that value was 2^{53}. This limits the maximum value of the modulus to a value lower or equal to 2^{26} such that $a \times r < m^2 < 2^{53}$. An accurate computation of the remainder is closely related to the least significant digits because these least significant digits will constitute the remainder after division. When the integer capacity is exceeded, the results become unpredictable and the error increases with the size of the multiplier.

The variables a, m and c are by definition integer, but in general the capacity of integer data types is not large enough. For example, the header file *limits.h* shows the maximum integer sizes for some typical programming environments. Depending on the magnitude of the modulus, integer arithmetic in most systems will probably impose a limit on the range of values acceptable. Some typical values are indicated in Table 3.5.

Constant	Interpretation	Value on Sparc (cc compiler)	Value on a PC (Turbo C)
ULONG_MAX	Largest value of an unsigned long integer	2^{32}-1	2^{32}-1
LONG_MAX	Largest value of a long integer	2^{31}-1	2^{31}-1

Table 3.5: Maximum Integer Sizes for Sparc and PC Systems

This function, rem(), is called during the spectral test and so there is an error introduced into the spectral test. Typical errors are shown in the following table, where the Matlab answer and the correct answer are both given. The generator is specified by the parameters a=12420108765, m=2^{40}. The spectral test is an iterative programme and the small error introduced by an inaccurate computation of the remainder is amplified to give a significant

error. Moreover, the spectral test calculates the resolution in higher dimensions using the resolution already computed from a lower level. In this way the error is propagated through the dimensions even though the magnitude of the product $a \times r$ decreases when the dimension increases. Table 3.6 gives some numerical examples of this effect.

Method	v_3 / bits	v_4 / bits	v_5 / bits	v_6 / bits
Answer using rem()	9912 /13	1059 / 10	178 / 7	58 / 5
Correct answer	6520 / 12	661 / 9	161 / 7	83 / 6

Table 3.6: Spectral Test Errors

There are errors in most dimensions of ± 1 bit. This may have an effect on the accuracy of the power estimates obtained. Two solutions to this problem were implemented. The first solution uses an array to store the product. This technique has the advantage that it is portable when programmed in ANSI C. It allows a maximum modulus of 2^{50}. The second method uses a symbolic data type. This, however, is a much slower method.

The array solution computes the modulo arithmetic of for example, $r_{i+1} = a \times r_i \bmod m$ by utilising the transformation of (41).

$$r_{i+1} = a \times r_i - m \times \text{floor}\left(a \times \frac{r}{m} \right) \tag{41}$$

The steps of the algorithm are outlined in the flowchart of Figure 3.8. The inputs are the values of a, r and m and the output is the remainder as calculated in (41). There are six steps in the process. The values of a and r are stored and the larger of the two values becomes the multiplicand. The two variables are read into an array of integers, with one digit per location. The array size is $n \times 2$, where n is the number of decimal digits in the multiplicand. The data is stored with the least significant digit of both a and r at location $(i,1)$ two dimensional array,where $i = 1$ for a and $i=2$ for r. The normal shift and add algorithm for multiplication is performed on the two rows of the array to give the product $a \times r$. The product $m \times \text{floor}(a \times r/m)$ is then formed from another array multiplication and the remainder is output.

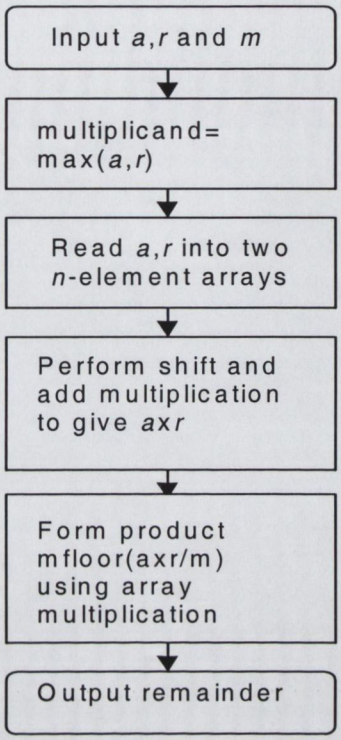


Figure 3.8: Flowchart for Array Multiplication

The array operations are unusual in that the data and the operations are both reversed in the array while the array holds individual digits which must be assembled to form the answer. However, the effect of this is to allow a greater range of data to be accurately processed by the spectral test because each individual location in the array can store the maximum system integer. This improved procedure was incorporated into the spectral test. In order to quantify the improvement to the spectral test, the outputs from the original and the modified spectral test are compared in Table 3.7

Accuracy:	$\log_2 v_2$	$\log_2 v_3$	$\log_2 v_4$	$\log_2 v_5$	$\log_2 v_6$
Test1: a=3141592653 m=2^{35}					
Original Spectral Test	15.74	10.52	8.66	6.67	5.67
Improved Spectral Test	15.74	9.98	7.38	5.06	5.06
Published Data	15.7	10.0	7.4	5.0	5.0
Test2: a=4219755981 m=10^{10}					
Original Spectral Test	16.66	10.10	7.73	6.18	5.11
Improved Spectral Test	16.66	10.65	7.80	6.26	4.84
Published Data	16.7	10.7	7.8	6.3	4.8
Test3: a=31167285 m=2^{48}					
Original Spectral Test	24.10	15.97	11.71	8.84	7.95
Improved Spectral Test	24.10	15.97	12.02	9.11	7.93
Published Data	24.1	21.3	16.0	12.7	10.4

Table 3.7: Improved Results for Spectral Test

The results have been compared against published data [ibid] which is given to one place of decimals. The improved spectral test therefore agrees exactly with the reference data in all cases examined. Some sample data are given in the above tables. The array technique has been validated for moduli of up to 2^{50}.

The above array technique will improve the algorithm for all programming languages. A second method of improvement was also implemented, specifically for Matlab V. This employs the symbolic toolbox. The symbolic data type is a complex structure, which manipulates numbers as character chains, and it is possible to access the symbolic objects by using an interface, which converts a symbolic number into a chain of characters. A version of the spectral test was written using the symbolic data type. This gives a spectral test with no limit on the size of the input value. The symbolic toolbox has limited functionality and integration with the Matlab V environment, in particular comparison operations '>' and '<' are not defined for symbolic type. This resulted in the occasional reuse of double precision. In addition, the functions *floor*() and *round*() are not overloaded for the symbolic type and were rewritten. In this spectral test there are virtually no limits on the magnitude of the inputs, but the time penalty for the increased performance is considerable. For example, a generator with a = 6364136223846793005 and m = 2^{64} takes 331s to compute the accuracy for dimensions 2 to 6, as opposed to under 1s for the original implementation. The results obtained for this generator are given in Table 3.8. The published data appears to have been rounded but there is still excellent agreement between the data and the results obtained with the symbolic method.

Accuracy	Published Data (v_i^2)	Symbolic Spectral Test (v_i)t
v_2	8.8×10^{18}	8810664174654508192
v_3	6.4×10^{12}	6398304806574
v_4	4.1×10^{9}	4112636266
v_5	45662836	45662836
v_6	1846368	1846368

Table 3.8: Improved Spectral Test Results

This spectral test makes it possible to evaluate multipliers with a large modulus, when required. For the versions coded in ANSI C a different approach was taken. The language offers two data types, double and long double. The sizes of theses two types are given in Table 3.9, for two programming environments used in this work.

Constant	Interpretation	Value on Sparc (cc compiler)	Value on a PC (Turbo C)
FLT_MANT_DIG	Bits of the mantissa (with sign) for float.	24	24
DBL_MANT_DIG	Bits of the mantissa (with sign) for double.	53	53
LDBL_MANT_DIG	Bits of the mantissa (with sign) for long double.	113	64

Table 3.9: Maximum Size of Constants

If the modulus and the multiplier are limited to double format, the product $a \times r$ will fit exactly into a long double. Since no remainder function exists in ANSI C for floating point numbers, it is necessary compute the remainder as given by (42).

$$a\times r-(long\ double)floor((double)(a\times r/m))\times m \tag{42}$$

In this expression, a, r and m are long double and *floor()* takes and returns double. The maximum modulus allowed is 2^{53}. When programming in C or C++ on a Sun Sparc system, it is possible to bring the maximum modulus allowed to over 2^{100}. This is achieved by using the non-ANSI C library (sunmath.h) or the ANSI C++ library where the functions *floor()* and *pow()* are overloaded for the type long double, in conjunction with a function using an array with the type long double to compute this remainder [Wolf98]. The spectral test is now incorporated into a general method for finding good generators using a genetic algorithm which is now described.

3.4 Random Binary Vector Generation Using Genetic Optimisation

The technique used in this work to obtain the multiplier *a,* for a given modulus *m*, employs an integer-based Genetic Algorithm (GA) [Tang96], [Gold89]. The algorithm operates by minimising an objective function based on estimating the distance between planes for the binary data in each of several dimensions. The resulting generator is optimised for randomness for up to 6-tuple random vectors. This means that all possible sets of 2, 3, 4, 5 and 6 consecutive binary vectors are sufficiently random over the complete set of data. Analysis in higher dimensions is also possible, if required. Firstly, the structure of a typical genetic algorithm is discussed. The objective function, which the algorithm uses to grade the solution it produces, is then described. The operation of the algorithm is presented, with some typical results.

3.4.1 Genetic Algorithms

The concept of the genetic algorithm was first proposed by John Holland [Holl75] and has been applied to a variety of optimisation problems with some success [Gold94], [Karr91, [Star90]. It is a probabilistic search mechanism, inspired by the Darwinian laws of Natural Selection and by the process of genetic evolution. It belongs to a broad grouping of algorithms that mimic natural processes, such as simulated annealing [Kirk83] and differential evolution [Pric97]. It is generally considered to be very effective when searching a large solution space and when handling optimisation problems with a large number of variables [Gibb96]. The first step in applying a GA to a problem is to represent each point in the solution space as a string, referred to as a chromosome. This is known as encoding and can be as simple as assigning an integer value to every solution, as in this work. An initial population of chromosomes, each of which represents a solution to the problem, is generated randomly. This is the first generation of the solution. Each chromosome is evaluated by an objective function, which produces a numerical fitness value for it. The GA then mimics a process of natural selection by selecting the best chromosomes, allowing them to become parents of the next generation. Offspring are produced from these winners, which in turn are evaluated and the simulation terminates when a chromosome with the desired objective value has been produced, or a predetermined number of generations has elapsed. Genetic optimisation is based on three operators which are applied to each chromosome: (1)

selection/reproduction (2) crossover and (3) mutation. These operators are outlined in Figure 3.9.

```
Initialise the population
For count = 1 to last_generation
      Calculate the fitness of each individual
      Construct the roulette wheel for reproduction
      While ( no_of_individuals )<= popsize do
          Select two parents using roulette wheel
          Perform crossover based on the probability of crossover
          Mutate offspring based on probability of mutation
          Increase the no_of_individuals counter
       End while loop
       Evaluate the fitness of the new population
       Test for exit condition
End for loop
Output chromosome with highest fitness
```

Figure 3.9: Flowchart for the Genetic Algorithm

Selection is the mechanism used to determine the parents of the next generation. Parents are selected with a biased random process, where strings with a higher fitness have a greater chance of contributing offspring to the next generation. In this way a natural selection is made every generation from among the fittest. There are a number of selection procedures available. The one implemented in this thesis is called roulette wheel selection. Individuals in the population are assigned a slice of a virtual roulette wheel, based on their fitness. A random slot on the wheel is chosen and the owner of that slot is allowed reproduce, using the crossover mechanism described next.

Crossover is the operator used to combine two parents to produce an offspring. In the GA implemented in this work, crossover is achieved by randomly selecting two bit positions on each parent, dividing each parent into two sections. The offspring is created by concatenating the upper section of one parent with the lower section of the other parent. This retains the characteristics of both parents in the offspring and hopefully improving upon them.

Mutation, used sparingly in this implementation, chooses a bit position within the chromosome at random and inverts it, changing its value and its fitness. The probability of

this occurring is made very low, typically less than 1% [Tang96]. Nevertheless, its function within the GA is significant, as it reduces the chance of the solution getting trapped at a local minimum. As mutation removes information from the population it can slow the speed of convergence of the algorithm and so its probability is kept low. It also allows the creation of a bit value in the population that may not arise through crossover.

The GA is also described in terms of the operations performed on the population. The relationship between each of these operations is represented graphically in Figure 3.10.

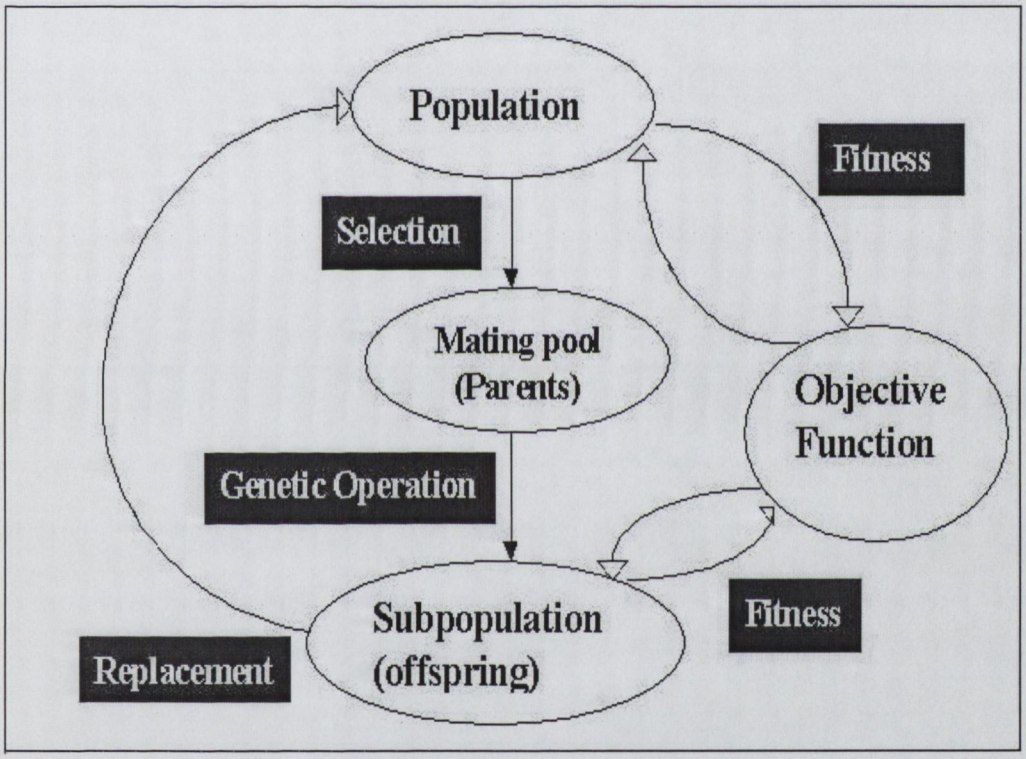


Figure 3.10: GA Operations

Figure 3.10 demonstrates how the GA continually refines the population through the mechanisms of selection, mating and replacement. The extent to which the optimisation is successfully addressed depends on how the objective function is formulated, as this is the only criterion applied to the individuals within the population. The objective function in this application, namely the spectral test, has proved to be a useful measure for the GA implementation and it is now discussed in the context of the overall optimisation problem.

3.4.2 Implementation of the Genetic Algorithm

The specific problem to be addressed by the GA is the search for an adequate multiplier for use in the LCG scheme. By using the spectral test as an objective function in a GA implementation, a population of multipliers may be optimised to provide a suitable multiplier for a given LCG generator period. In addition, the nature of the solution space is such that the GA is an effective technique for selecting the generator multiplier parameter. The solution space is now explored.

Application of the spectral test to an arbitrary selection of multipliers from the solution space for a given period will quickly reveal the multi-modal, discontinuous and extremely non-regular nature of the relationship between the magnitude of the multiplier and the randomness of the sequence produced. This is demonstrated in Figure 3.11 for a sample of different 32-bit generators. This figure gives three traces, one each for v_2, v_3 and v_4.

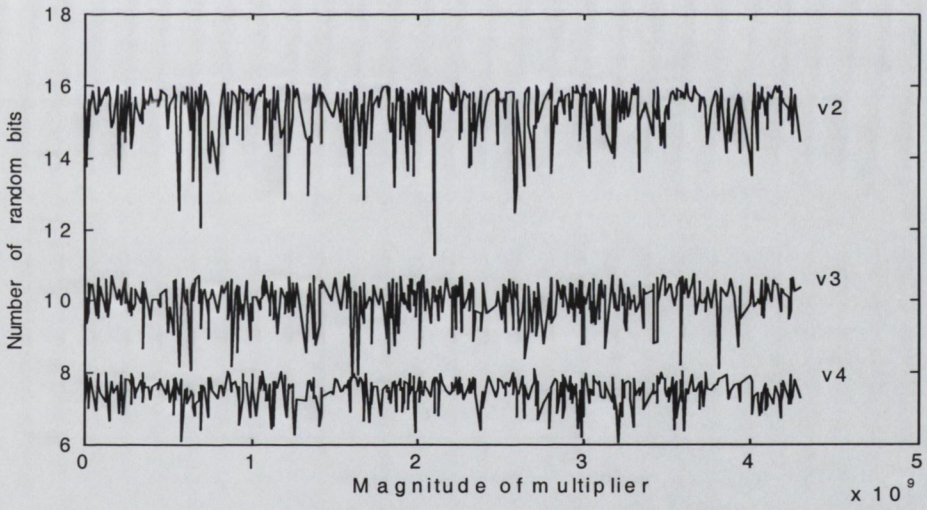


Figure 3.11: Application of Spectral Test to a Variety of Generators

There is a large variation in the quality of the generators, as determined by this application of the spectral test. Genetic algorithms are used to solve difficult problems of this type, where the objective function is not continuous or differentiable as in the present case and so analytical solutions are not applicable [Houc97]. Several features of this algorithm suggest its suitability for the generation of random vectors. Firstly, the fundamental approach is based on the selection and reproduction of a population of binary strings. The generator parameters are integers and so there is a natural correspondence between the binary and integer domains. Secondly, the mathematical conditions imposed on the generator parameters

for maximum period may be neatly implemented with the use of schema [ibid] or fixed bit positions within the chromosome string. There is also an explicit parallelism in the operation of the algorithm as several solutions are pursued within the population from one generation to the next. The production of random data does not require the optimum solution, merely an adequate one for the given specifications. Hence, the pursuit of several simultaneous good solutions to the problem means that meaningful results can be obtained in relatively few generations. For these reasons, the genetic algorithm [ibid] was chosen as an appropriate mechanism for the optimisation of the generator multiplier.

The algorithm has been adapted in several ways. This implementation uses an alphabet of integer variables, with each one taking values within user-specified bounds. It also utilises a binary representation and performs only two genetic operations: simple crossover and binary mutation. Operating on one variable only, the multiplier a, the program includes a mechanism for ensuring that the conditions for maximum period are maintained throughout the population. This requires that the condition $a \bmod 8 = 5$ is implemented. It imposes a condition on the binary representation of the multiplier that the three least significant bits are 101_2. These characteristics must be propagated through each generation of the GA operation. These fixed bit positions are known as schema and are essential for maximum period.

The most critical part of any GA is the formulation of the objective function, used to evaluate the fitness of each individual. The more demanding the test applied to potential solutions, the better the results obtained. For this reason, the objective function was based on what is generally considered to be a demanding test of randomness, the spectral test. In particular, the five-dimensional volume formed by the product of the application of the test in dimensions two to six, forms the figure of merit for each multiplier in the population. This is evaluated within the GA using the expression in (43).

$$score = \prod_{t=2}^{6} v_t \tag{43}$$

Some sample results for typical runs are given in the next section. They provide a qualitative view of the operation of the program. The results are also compared to those obtained by a simple random selection procedure to demonstrate its effectiveness. A number

of facets of the program operation are explored in the following section, including tracking improvements over generations, processing time and the effect of population size on the quality of results. The effect of seeding the initial population with a known good generator is also explored.

3.4.3 Results from GA Operation

The aim of the algorithm is to provide continuous improvements to the individuals in the population, the multipliers, over successive generations. As an example of this process, the diagrams of Figures 3.12-3.14 show typical improvements experienced when populations of individuals were optimised for different numbers of generations. Three different runs are shown. The search space is 2^{40} for the first example in Figure 3.12, with a population size of 30.

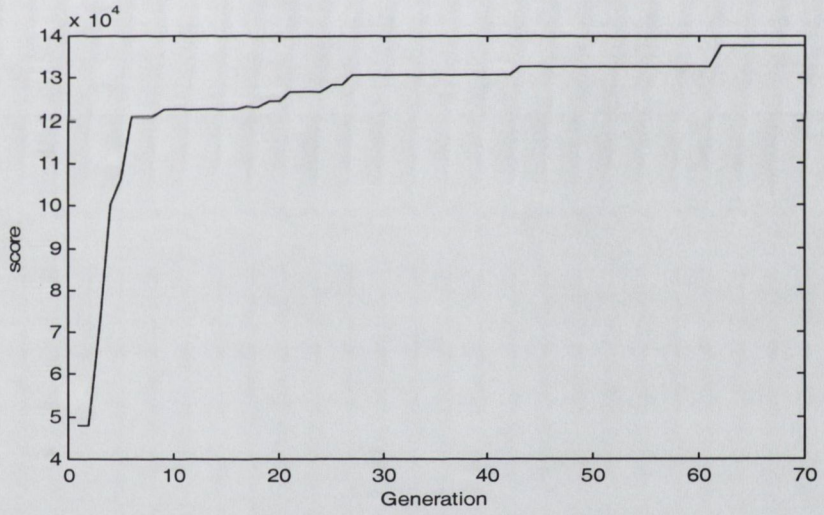


Figure 3.12: Improvement in Spectral Score

The second run is for $m = 2^{24}$, a population size of 40 and the algorithm was run for 20 generations. The results are given in Figure 3.13.

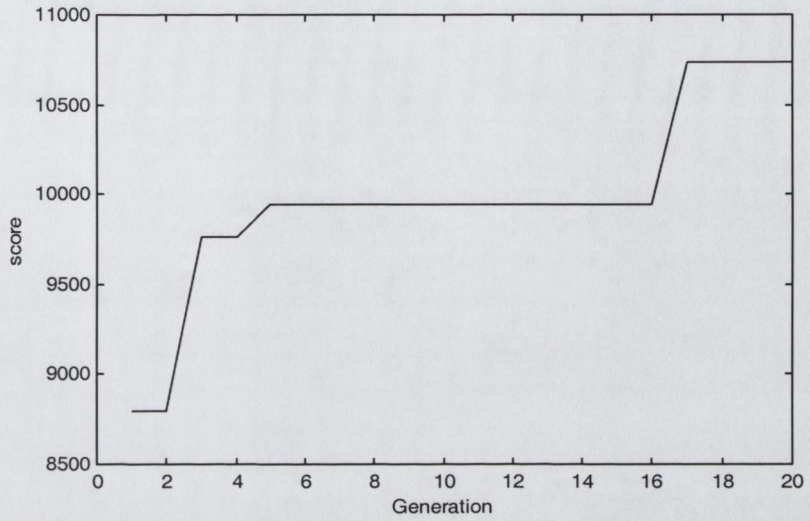


Figure 3.13: Genetic Optimisation of Spectral Test Score

Figure 3.14 contains the data from the third sample run with a period of 2^{32}, a population size of 20 and optimisation for 20 generations.

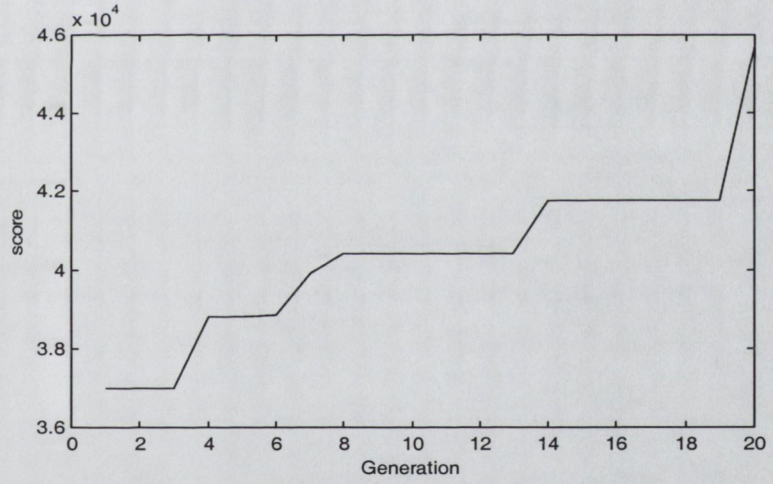


Figure 3.14: Selection of Generator Parameter by Genetic Optimisation

In each case a useful improvement in the score of the generator is produced. The potential scores, indicated by the y-axis values, increase with the period. Most significantly, the algorithm produces results of a consistent value. The effectiveness of the GA in reliably producing an acceptable result is demonstrated in the histogram of Figure 3.15. It shows the spectral test results of 100 GA runs, in this case with a period of 2^{32}. The consistency of the results is the most significant point here, as the GA can be used to produce a suitable generator with a minimum level of accuracy, as determined by the spectral test.

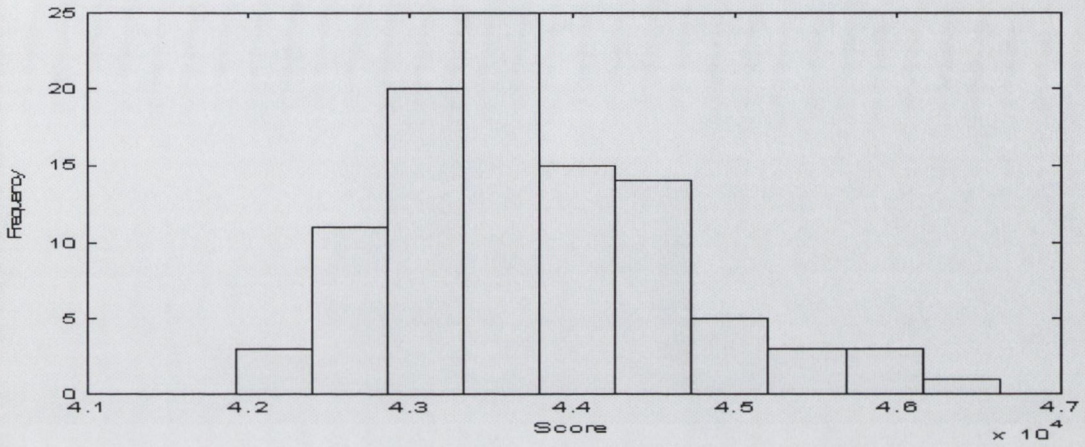


Figure 3.15: Histogram of Genetically Improved Spectral Scores

In contrast, the histogram of Figure 3.16 shows the generators produced when random multipliers are chosen. It can be seen that the results are not as consistent. There is a larger spread of results and a lower average result, with a peak at approximately $3x10^4$. Thus the principal benefit of applying the GA is in ensuring a consistent generator quality.

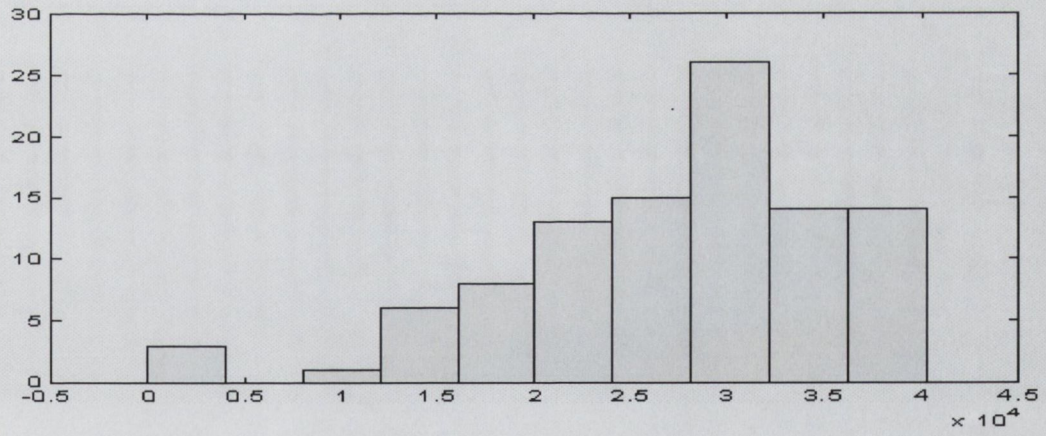


Figure 3.16: Histogram of Scores Without Optimisation

The outputs of such generators are required to be random enough for the purpose of power estimation. Therefore, several standard tests were applied to the output of the generators to test whether the data was drawn from a uniformly distributed independent population. If so, then the generators may be acceptable for the intended application. The tests applied, together with the results obtained, are described in the following section.

3.5 Test and Analysis of Generator Output

The spectral test is an a-priori, analytical, structural test, which investigates the full period of a generator. The application of a genetic algorithm has proved to be a convenient mechanism for the selection of generator parameters based on the test. It is also necessary to look at the output of such generators, sampling them to ensure that the data produced is of sufficient quality. Such tests are statistical in nature, and they require the generation of samples of output vectors and the subsequent application of rigorous tests, to test whether or not the sequence under test is drawn from a population with the required properties. The two fundamental properties that any sequence of random data should possess are uniformity and independence. The uniformity of a sequence refers to the distribution of individual vectors. The data should appear as though drawn from a population with a rectangular distribution, i.e., all elements of the population are equally likely to occur. Independence of data occurs when there is no correlation between successive members of a sequence. Independence between samples separated by more than one sample period may also be considered. In addition, any source of random data should also be subjected to application-specific tests representing typical problems for which the generator is intended. The sequences generated have been subjected to a variety of tests of both types, the most common of which are indicated in Table 3.10.

Test	Property
1. Analysis of Histogram	Uniformity
2. Binomial Test	Independence
3. Kolmogarov-Smirnov (KS) test of Power Spectral Density	Independence
4. KS Sample Test	Uniformity
5. Estimation of average power for sample circuits	Suitability for application (power estimation)
6. χ^2 Goodness of Fit Test	Uniformity

Table 3.10: Typical Generator Output Tests

The precise property tested is also indicated. Three representative tests have been selected for this work from this table. Each of the three tests is now discussed in a separate subsection. A novel test strategy is also proposed.

3.5.1 Test 1: Analysis of Histogram

A finite set of uniformly distributed numbers on the interval [0,1) is one where each number has an equal probability of occurring in the set [Spie88]. The uniform random variable X, with parameters a, b has a sample space $S_x = [a,b]$, where a, b are reals with $a<b$. The probabiltiy distribution function (PDF) is given by (44).

$$f_X(x) = \frac{1}{b-a}, x \in [a,b] \tag{44}$$

The first moment, or mean, of X is a useful and easily estimated quantity. It is given by (45), where it is evaluated for the PDF of (44).

$$E(X) \stackrel{\Delta}{=} \int_{-\infty}^{\infty} x f_X(x) dx = \frac{b+a}{2} \tag{45}$$

The second moment, or variance, of X is equally useful and is defined by the expression in (46).

$$\text{var}(X) \stackrel{\Delta}{=} \int_{-\infty}^{\infty} (x - EX)^2 f_X(x) dx = \frac{(b-a)^2}{12} \tag{46}$$

The pseudorandom sequences implemented utilise a finite number of bits and may be considered to represent a finite subset on the interval [0,1), where the bracketing indicates that the value one is usually excluded. Thus with values of $b = 1$ and $a = 0$, the theoretical mean and variance are 1/2 and 1/12 respectively. Using these values, the output sample from the generator may be tested. Several tests are available. Estimates of the mean and variance may be tested under the hypothesis that the sample has been drawn from a uniform distribution, giving a statistical decision within a predetermined level of significance. In addition, a χ^2 goodness-of-fit test may be applied to the data. This determines how well the empirical distribution fits the theoretical distribution obtained from the sample data.

Uniformity is tested by sorting the generator output into a number of sub-intervals and then counting the number of occurrences in each one. The flatness of the resulting histogram may then be tested under the null hypothesis that the data is drawn from the theoretical

distribution. However, even a bad generator produces a uniformly flat histogram during a complete period. The data, therefore, is possibly not independent. Thus a test of uniformity is a necessary but insufficient test which is often performed in conjunction with estimates of the mean and standard deviation of the sample, as outlined above. It is also very useful when applied to transforms of the data, particularly those involving the frequency domain. This will be discussed in a later section. It is also important to note that the global properties of a generator with a large period can be different to the local properties observed in individual samples of the data. This results in individual samples from good generators failing such statistical tests. The likelihood of such a failure depends on the initial conditions for the test and is of particular relevance in this context, given the large amount of data that is usually generated, i.e. the large sample space. This additional factor must be taken into account when implementing tests, so that local results do not skew the more significant global behaviour under investigation. The background to the approach taken to this problem is now given.

When a hypothesis such as "is the generator sufficiently random?" is tested, two types of error are possible. If the hypothesis is rejected when it should be accepted, a Type I error has been made [ibid]. In this case a good generator will provide a sample that will fail the test, when it should really have passed. Conversely, if the hypothesis is accepted when it should be rejected, a Type II error has been made. This occurs when a bad generator passes the test. In order for any test to be useful, such errors must be minimised. Since the null hypothesis adopted for this work states that the generator is good, Type I errors are of more interest. In any test, the maximum probability associated with the occurrence of a Type I error is termed the level of significance of the test. This parameter, denoted α, is specified as part of the test. In general, an examination of the literature indicates that a level of significance of 0.05 or 0.01 is often adopted. In the context of this work, a value of $\alpha = 0.05$ means that there is a 5% probability of rejecting the hypothesis when it should be accepted. If 100 samples are tested, we expect no more than 5 of them to fail. This provides a very powerful method for investigating the global behaviour of the generator. When a large number of samples are considered, the number of passes and fails may be tested for significance using a binomial distribution. This technique allows the overall global behaviour of the generator to be assessed, rather than the less reliable results obtained from a single local sample. In

particular, in addition to the histogram, the power spectral density of the data has been efficiently tested with this procedure.

The binomial distribution used is a discrete distribution, which states that given the probability p of an event occurring, the probability of X such events in N trials is given by (47), where the probability of the non-occurrence of the event is given by $q = 1 - p$.

$$p(X) = {}^{N}C_{X}\, p^{X} q^{N-X} = \frac{N!}{X!(N-X)!} p^{X} q^{N-X} \tag{47}$$

In the context of the analysis of generator output, the number of failed samples may be tested to see if it is significantly high. In particular, the significance level of the goodness-of-fit test α, which states the probability of an error, should determine the number of failures. By setting $p = \alpha$, N = the number of tests and X = the number of failures, the number of failures is tested using the binomial distribution. If a significantly higher number of failures is encountered than that predicted by the distribution, the generator has failed to demonstrate sufficient randomness, for this test. This technique has been used to investigate both the histogram and the power spectral density [Sher95].

For each test, three generators are examined: a good generator selected by the genetic mechanism, a poor-quality generator and the generator provided by Matlab. The good generator has parameters $m = 2^{32}$, $a = 1494270789$, $c = 1$, with a spectral score of 45330.0, as defined by equation (43). The poor quality generator has parameters $m = 2^{32}$, $a=15$, $c = 1$, with a spectral score of 913.9. First and second moments are calculated and 100 χ^2 goodness-of-fit tests are performed on samples of each generator output. A binomial test for global randomness is performed on the results of the χ^2 tests.

The data in Table 3.11 contains the mean and variance for the three generators, using one sample from each generator. Again while these figures give a general indication of the data quality, it is difficult to draw meaningful conclusions from them.

Generator	Mean	Variance
Genetically optimised	0.51	0.09
Non-optimised	0.48	0.08
Matlab	0.50	0.09

Table 3.11: Sample Mean and Variance

A χ^2 goodness-of-fit test was also performed on one sample from each generator, at the 1%, 5% and 10% levels of significance. The data was divided into ten subintervals and o_j, the frequency of occurrence within each interval was counted. A test statistic χ^2 was then calculated using the observed and expected frequencies, o_j and e_j respectively. The test statistic is given by (48).

$$\chi^2 = \sum_{j=1}^{k} \frac{(o_j - e_j)^2}{e_j} \tag{48}$$

If χ^2 is zero then the observed and expected frequencies agree exactly, while if $\chi^2 > 0$, there is a discrepancy between the expected and observed frequencies. The sampling distribution of χ^2 is given by the Chi-squared distribution, with v degrees of freedom. For this test, v is given by N_{sub} -1, where N_{sub} = the number of subintervals, 10 in this instance. The results, along with the relevant percentile values from the distribution, are given in Table 3.12. The bracket values refer to the test results from each generator. The critical value is the maximum value of test result for passing the test. The subscript for the percentile value refers to the value 100(1-α)%, where α is the level of significance for the test. As the test is applied only to a single sample, this is not a very demanding or useful test.

Generator	$\chi^2_{0.90, v=9}$	$\chi^2_{0.95, v=9}$	$\chi^2_{0.99, v=9}$
Genetically optimised (11.2)	Pass	pass	Pass
Non-optimised (24.6)	Fail	fail	Fail
Matlab (10.1)	Pass	pass	Pass
Critical Value	14.7	16.9	21.7

Table 3.12: Results From the Application of the χ^2 Test

A more powerful test of this data is realised by repeating the above tests a number of times and then examining the number of passes and fails with a binomial distribution [ibid]. For the three generators used to illustrate these ideas, the χ^2 test was run 100 times at three levels of significance. The number of fails was recorded and tested against the expected binomial distribution. The results of this test are presented in Table 3.13. Both the genetically optimised generator and the Matlab generator pass the binomial test at all levels. The non-optimised generator fails at all levels of significance.

Generator	Number of Passes (/100) at 10% level	Number of Passes (/100) at 5% level	Number of Passes (/100) at 1% level
Critical Value	90	95	99
Optimised	91	95	99
Non-optimised	89	93	95
Matlab	91	96	99

Table 3.13: Binomial Test of Data Samples

3.5.2 Test 2: Power Spectral Density

The pseudorandom data examined so far may be viewed as a series of discrete time samples from an ensemble of stochastic member functions. As the probability density function for the signal is not a function of time or equivalently, is not a function of sample number, the signal is said to be stationary. As such, the properties of the signal may be described using a number of time-domain and frequency domain measures. These include autocorrelation and autocovariance in the time domain and the discrete Fourier transform (DFT) and power spectral density in the frequency domain [McCl99], [Krau95]. These quantities are very useful in characterising the properties of such signals. They may be estimated in order to deduce the behaviour of the signal. In the time domain, the relation between two samples at different time indices m and n is also of interest. One measure of this relation is given by the autocorrelation, which for a random discrete time signal is given by (49) where E is the expectation operator or mean.

$$\phi_{xx}[k] = E(X[n]X[n+k]) \tag{49}$$

When dealing with real signals, there are two estimators for autocorrelation. The unbiased estimate at lag k, for $|k| < N$, is given by (50).

$$\hat{\phi}\, xx(k) = \frac{1}{N} \sum_{i=0}^{N-1-|k|} X(i+|k|)X(i) \tag{50}$$

Another estimate is called the biased autocorrelation estimate, and is given by (51). It is defined for $|k| < N$.

$$\hat{\phi}\, xx(k) = \frac{1}{N-|k|} \sum_{i=0}^{N-1-|k|} X(i+|k|)X(i) \tag{51}$$

As the mean square error of the biased estimate is less than that of the unbiased estimate, [Chil97], the biased estimate is usually the one employed. Ideally, the autocorrelation sequence should show no relations between samples, for lags 1 to *k*. The autocovariance is the mean-removed autocorrelation and is given by (52).

$$\gamma_{xx}[n] = E\left((X[n] - m_{x[n]})(X[n+k] - m_{x[n+k]})\right) \tag{52}$$

The autocovariance [Mitr98] may be estimated using a sufficiently large sample of data using the estimator of (53).

$$\hat{C}xx(k) = \frac{1}{N}\sum_{n=0}^{N-1-|k|}\left(X(n) - m_x\right)\left(X(n+|k|) - m_x\right) \tag{53}$$

This measure is used to examine the linear independence between the terms of a time series, such as are of interest in this work. An alternative to these time domain representations is provided by the power spectral density function. The Wiener-Khintchine theorem [Chil97] states that the Power Spectral Density of a stationary random process X(t) is defined as the Fourier transform of the autocorrelation function of the random process. Since the Fourier transform is invertible, the spectrum and the autocorrelation are equivalent measures of the process. The relevant transforms are given in (54) and (55).

$$S_{xx}(\omega) = \sum_{-\infty}^{\infty} R_{xx}(k)e^{-j\omega k} \tag{54}$$

$$R_{xx}(k) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S_{xx}(\omega)e^{jwk}\,d\omega \tag{55}$$

The power spectral density (PSD) is an even, real and positive function of radian frequency, ω, for a real random process. The units for the Fourier transform of a voltage signal are volts/Hz or volts-sec while the units of the autocorrelation function of a voltage signal are $(\text{volts})^2$. As the PSD is determined by the Fourier transform of the autocorrelation function, the units of the PSD are therefore $(\text{volts})^2$-sec or $(\text{volts})^2$/Hz. It is often assumed that the power of a random process is the power dissipated in a nominal 1Ω load resistance. This yields units of $(\text{volts})^2$-sec across 1Ω or $(\text{volts})^2$/Hz across 1Ω, resulting in units of watts-sec

or watts/Hz. As this is dimensionally equivalent to energy, the PSD may also be quoted in Joules.

For this work, the particular case of a constant PSD $S_{xx}(\omega)$ is of interest. This random process, known as a uniform white noise process, is one in which all frequencies are present with equal power density. In this case the spectrum becomes a constant, as given in (56).

$$S_{xx}(f) = N_0 \tag{56}$$

Approaches to spectral estimation of the signal $X(t)$ fall into two categories, classical or non-parametric techniques and parametric methods [Krau95]. Classical methods require the estimation of the autocorrelation sequence from the data sample. The power spectrum is then estimated using the Fourier transform $F_T[X(t)]$, as shown in (57), where the Fourier transform is given by (58).

$$S_{xx}(\omega) = \lim_{T\to\infty} \frac{1}{2T} E\left[\left|F_T[X(t)]\right|^2\right] \tag{57}$$

$$F[X(t)] = \frac{1}{2\pi} \int_{-\infty}^{\infty} x(t) e^{-j\omega t} d\omega \tag{58}$$

The second class of techniques is termed nonclassical or parametric. This method uses an underlying model of the process, for example an autoregressive or moving average model, and estimates the power spectrum using this assumption.

A non-parametric method was used in this work because it makes no assumptions about the underlying data model and it is easily implemented. The principal techniques of this type include the periodogram, the modified periodogram, Bartlett's method, Welch's method and the Blackman-Tukey algorithm [Hayes96]. With these methods, the technique is often expressed directly in terms of the original signal $x(n)$, where it may be known as a direct method, although the underlying approach is based on the autocorrelation sequence. Each of these methods utilises an estimated spectrum, known as a periodogram. For discrete signals, when a window signal $w(n)$ is used on the data, a modified periodogram is obtained as given in (59).

$$\hat{P}_M(e^{j\omega}) = \frac{1}{N \cdot U}\left(\left|\sum_{n=-\infty}^{\infty} x(n)w(n)e^{-j\omega}\right|^2\right) \tag{59}$$

The parameter N is the length of the window and the constant U is given by (60).

$$U = \frac{1}{N}\sum_{n=0}^{N-1}|w(n)|^2 \tag{60}$$

When $w(n)$ is a rectangular window, then $U = 1$, and the periodogram reduces to the unmodified periodogram. The main function of the window is to damp out the effects of the Gibbs phenomenon resulting from truncation of an infinite series [Math96]. The spectrum of a window typically has unwanted spectral sidelobes or peaks, which can be misinterpreted as peaks in the spectrum of the data [Haye96]. The effects of windowing on a signal are well known and include a trade-off between spectral resolution or main lobe width, and spectral masking, or the amplitude of the sidelobes. For example, replacing a rectangular window with a Hamming window gives a reduction in the sidelobe amplitude of approximately 30dB. The spectral resolution is however reduced by about 50%. Bartlett proposed that the signal $x(n)$ be partitioned into K nonoverlapping sequences of length L, where the overall sequence length $N=K.L$. The periodogram is obtained with data $x_i(n)$ partitioned as indicated in (61).

$$x_i(n) = x(n+iL) \qquad for \quad n = 0,1,...,L-1 \quad and \quad i = 0,1,...,K-1 \tag{61}$$

Bartlett's estimate is then given by (62).

$$\hat{P}_B(e^{j\omega}) = \frac{1}{N}\sum_{i=0}^{K-1}\left|\sum_{n=0}^{L-1} x(n+iL)(e^{-jn\omega})\right|^2 \tag{62}$$

Various tradeoffs are possible between the spectral resolution and the variance of the estimate by altering the values of K and L, with an increase in K and a decrease in L, corresponding to a decrease in the variance.

Decreasing K and increasing L increases the resolution obtained. The spectral estimation technique used in this work uses two modifications to the above method proposed by Welch. The first is to allow the $x_i(n)$ sequences to overlap, and the second is to allow a data window

$w(n)$ to be applied to each sequence producing a set of modified periodograms which are then averaged. For a sequence of length L and an offset of D points between the start of each subsequence, the i^{th} sequence is given by (63).

$$x_i(n) = x(n+iD) \qquad n = 0,1,\ldots,L-1 \tag{63}$$

Thus the overlap between subsequence $x_i(n)$ and its successor $x_{i+1}(n)$ is L-D points, and if there are K subsequences over the entire sample of N points then (64) gives the relation between the subsequence length and the sample length.

$$N = L + D(K-1) \tag{64}$$

With no overlap ($D = L$), there are $K = N/L$ sections of length L as in Bartlett's method. Alternatively, if a 50% overlap is used then $K = 2N/L - 1$ sections are formed, each of length L. This doubles the number of sections, K, and thereby reduces the variance, while obtaining the same resolution by maintaining subsequence length L, when compared to Bartlett's method. Similarly, with a 50% overlap, $K = N/L - 1$ subsequences of length $2L$ are obtained. This has the effect of maintaining the variance while increasing the resolution. This method, therefore, allows reductions in variance and resolution to be traded by increasing the number and/or the length of the subsequences by varying the amount of overlapping. For this reason, the average modified periodogram technique, with a Hamming window, was selected for power spectral analysis of the data.

In practice, the computational requirements and computation time increase with increasing K for a given sample of N data points. In addition, the correlation between subsamples also increases. For this reason, the amount of overlap is typically no more than 50%-75%. Depending on the values chosen for the various parameters, other methods may yield similar resolution and variance. However, the reduction in spectral leakage that occurs because of the attenuated sidelobes of the window makes this a very useful method. In particular, when examining data for correlations, the ability to closely discriminate between nearby frequency components is useful. An implementation of this algorithm was coded using a modified version of the algorithm of Widrow and Stearns [Widr85].

Typical results obtained from the implementation are given in Figure 3.17. A sample of 10,000 data points was generated and the power spectral density was obtained using a subsequence length of 1000 with an overlap varying from 0% to 90%. A Hamming window was used to reduce spectral leakage. For each value of overlap, the variance between the N estimated spectral components $\{P_m\}$ was calculated using the usual statistical expression, as indicated in (65).

$$\text{var}(PSD) = \frac{1}{N}\sum_{m=1}^{N}\left(P_m - \frac{1}{N}\sum_{m=1}^{m=N}P_m\right)^2 \tag{65}$$

Figure 3.17 shows the effect of overlap on the variance estimate obtained. It is also observed that the variance in the estimate is dependent on the amount of overlap used, for a given subsequence length. This is a useful observation, and it will be utilised in a subsequent section. It is also evident that there are diminishing returns to the amount of overlap used, due to the increase in correlation added to the data.

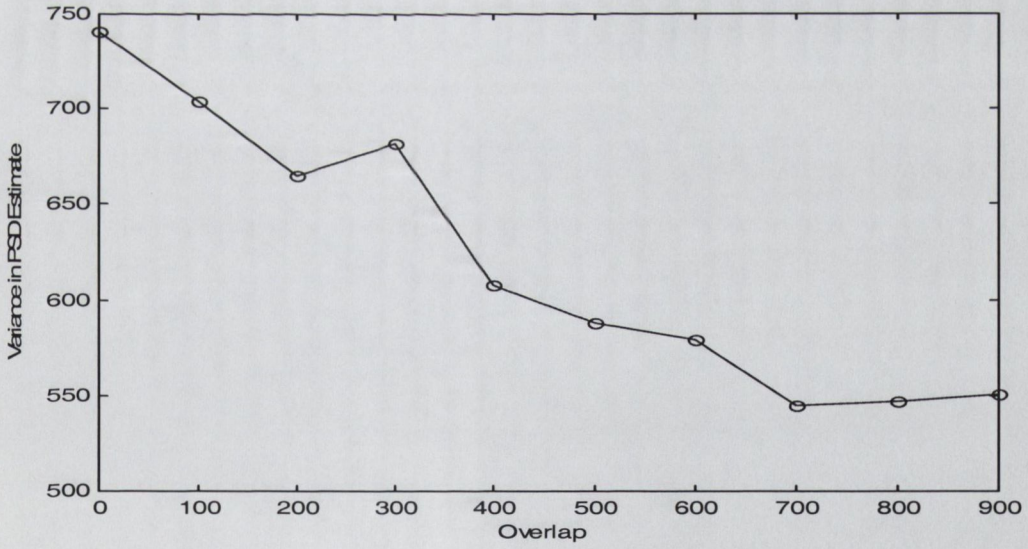


Figure 3.17: Variation in PSD Estimate with Overlap

The PSD for the samples with the two extremes of variance obtained are also given in Figure 3.18 for a visual comparison. The vertical range of the PSD data determines the variance. Therefore, the left hand diagram of Figure 3.18 represents the sample with the lowest variance observed, while the right hand diagram had the largest variance obtained.

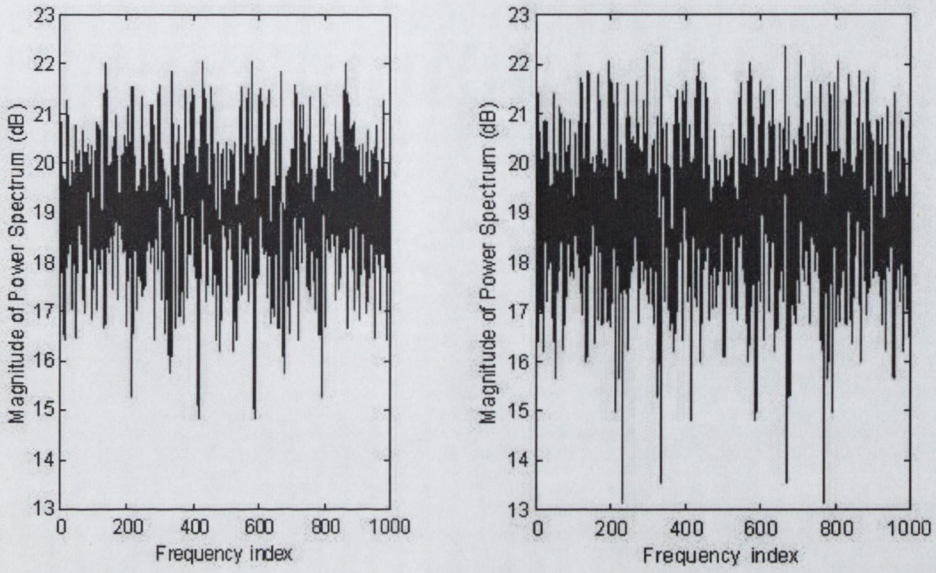


Figure 3.18: PSD of Samples with the Lowest and Highest Variance

The above data was obtained using a good generator and qualitatively demonstrates the general usefulness of the PSD and associated measures when determining the quality of input data for circuit simulation. The variance of the estimated PSD may also be used as a comparative measure, as Table 3.14 shows. The variance of the PSD was obtained for generators of varying quality, as determined by the spectral test. The results in this table indicate that this test may be used to compare the output from two good generators, for example when comparing a known good generator with a candidate generator.

Generator	(1)	(2)	(3)	(4)	(5)	(6)
Spectral Score	7226	15140	20773	25140	29299	31617
Variance	2.07	1.68	1.46	1.35	1.36	1.03

Table 3.14: Relationship Between Variance and Spectral Score

In each case a sample of 1000 data points was taken, with 10 subsequences of length 100. Five percent overlap was used and no windowing was applied. The variance of the resulting PSD estimates are given in the table and the data is plotted in Figure 3.19.

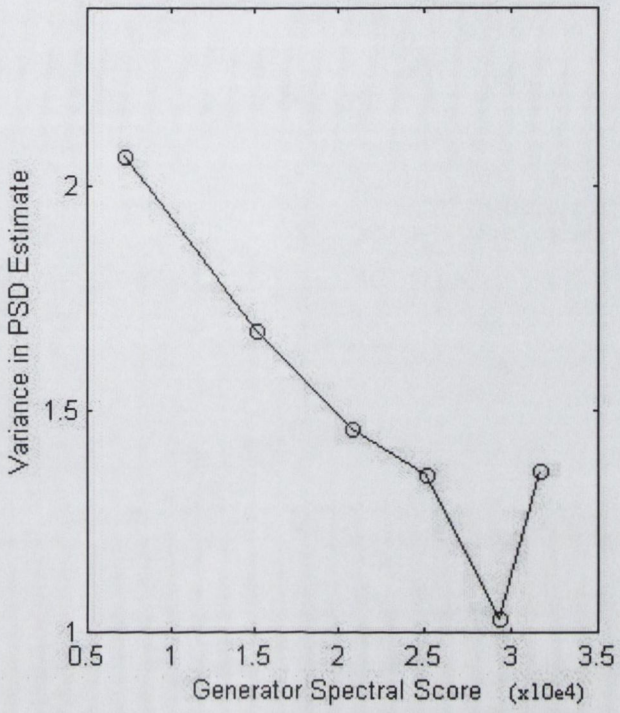


Figure 3.19: Relationship Between Variance and Spectral Score

While there appears to be a general correspondence between the spectral test score and the variance, the relation will depend on the sample size, overlap and the subsample drawn from the generator. However, the general trend is present and may be used to rate a given generator. The statistical nature of the PSD estimate can, however, be used to construct a very powerful test of the generator output, based on the mean of the distribution of spectral lines, rather than the variance. This test is now described.

3.5.3 Test 3: CDF Test of Power Spectral Density

The data generated and tested in the previous section consists of samples drawn from a uniform distribution, also referred to as uniform white noise (UWN). The PSD of such a distribution is also uniform, where the spectral component at each frequency should be the same magnitude. Inevitably, as the PSD is estimated with a periodogram-based technique, there will be variation in the magnitude of the component at any given frequency. These components are therefore used to compute an estimated cumulative distribution function (CDF) which should also be linear. The linearity of the resulting CDF may be subjected to number of tests, notably the Kolmogarov-Smirnov (KS) Goodness-of-Fit test [Gonz77]. This test determines if the underlying hypothesis, that of a flat PSD, is adequately reflected by the

CDF obtained. The linear independence of the generator is therefore rigorously tested. This test has been proposed by Sherif and Dear [Sher95], employing a Walsh Transform [Kart76] to compute the CDF. In this work, a test based on the more widely known Fourier Transform is developed and explored. As the results presented indicate, this approach is one of the most useful tests encountered so far. The construction of the test may be outlined as follows.

Data such as frequency component magnitudes may be distributed into a number of classes or categories, based on numerical value. A tabular arrangement of such data, along with the corresponding class frequencies, is called a frequency distribution. The class interval is a data range defining such a class, for example the range 10dB-12dB. The total frequency of all values less than the upper class boundary of a given class interval, is called the cumulative frequency. The cumulative frequency distribution is a table representing such cumulative frequencies and a graphical representation of the data is termed a cumulative frequency polygon or ogive [Spie88]. A relative cumulative frequency is often quoted, where the data is divided by the total frequency. This allows the data to be expressed within the range 0 to 1, or as a percentage. The application of the CDF to the testing of the generator consists of five steps, which are outlined below.

1. A sequence of samples is obtained from the generator under test.

2. The periodogram is calculated.

3. An empirical cumulative distribution function is calculated by cumulatively summing the values and then scaling the result so that the maximum is one. This is equivalent to a relative frequency distribution.

4. An appropriate goodness-of-fit test is performed at the desired level of confidence, α, under the null hypothesis that the data has been drawn from a uniform CDF. Kolmogarov-Smirnov is used to make this test, which is detailed below.

5. Steps (1)-(4) are repeated N times and the number of acceptances and rejections is tested to determine if it is consistent with a binomial distribution with parameters $n = N$ and $p = \alpha$. This aspect of the test is explored in more detail below.

In advance of some typical results, two aspects of the test require elaboration: the KS test for goodness of fit in Step (4) and the use of repetition in Step (5). These topics are now addressed in turn.

The Kolmogarov-Smirnov (KS) [Birn52] test is used to test hypotheses concerning the agreement between the distribution of a set of sample values and a theoretical population. Such tests fall into two categories, parametic and non-parametric. Test statistics whose sampling distribution depends on the explicit form or the value of parameters in the distribution of the population are termed parametric. For example, the test statistic $t=N^{1/2}(X_{\text{mean}}\mu)/s$ has Student's distribution only if the population is normal [Vini98]. Such tests are limited in their application. Non-parametic tests have no such dependency and therefore have a wider range of applications. The most common non-parametric test is the χ^2 test, but it has the disadvantage that the data is only considered after it has been grouped. This action filters the data and so information concerning the sample is lost. The KS test is also non-parametric but has no such grouping of data and so is generally considered a more discerning, although more complex, test. The KS test can be applied to the PSD distribution and also to the original data, where it replaces the χ^2 test for example. The basic operation of the test is now presented.

A population is assumed to have a specific CDF, $F_0(x)$. For any given value of x, x_i, $F_0(x_i)$ is the proportion of the population having a value less than or equal to x_i. The cumulative step function obtained from N data samples $S_N(x)$ is expected to closely follow the proposed distribution, if the hypothesis is a correct one. The sample distribution is easily obtained by setting $S_N(x) = k/N$, where k is the number of observations less than or equal to x. The difference between the expected and actual CDF is then given by the two statistics K_{n+} and K_n, as given in (66).

$$\begin{aligned} K_n^+ &= \sqrt{n} \max_{-\infty<x<+\infty} \left(S_n(x) - F_0(x)\right) \\ K_n^- &= \sqrt{n} \max_{-\infty<x<+\infty} \left(F_0(x) - S_n(x)\right) \end{aligned} \tag{66}$$

They represent the greatest amount of deviation when $S_N(x)$ is respectively greater than and less than $F_0(x)$. The factor $n^{1/2}$ is included because the standard deviation of $S_N(x)$ is proportional to $1/n^{1/2}$. To make the standard deviation independent of the sample size the

result is multiplied by $n^{1/2}$. The distribution of $K_{n+/-}$ was proposed by Kolmogarov [Kolm33], Smirnov [Smir48] and others, and is independent of $F_0(x)$. The test is represented graphically in Figure 3.20, where the two outer lines show the maximum deviation expected from the distribution. The middle plot represents the actual distribution. Should the empirical distribution exceed the outer boundaries, the data may not be drawn from the proposed distribution.

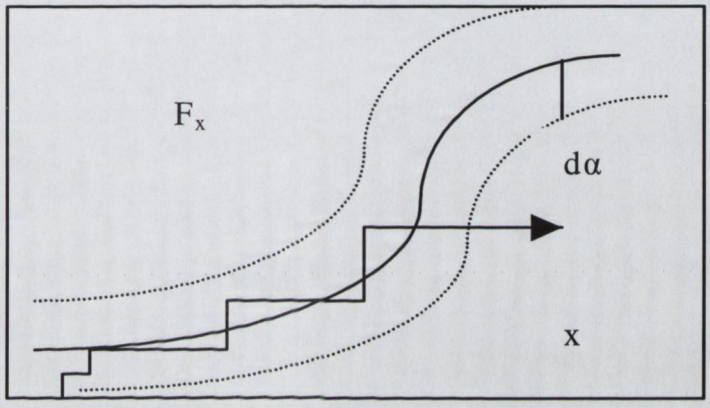


Figure 3.20: The Kolmogarov-Smirnov Test

Table 3.15 gives a short excerpt of the critical values of the maximum absolute difference between sample and population CDF. For example, at a level of significance 1-α = 0.20, the critical value of d for N = 10 is 0.7845. This means that the probability is 80% that either form of K_{10} will be less than or equal to 0.7845. Thus a procedure for testing data requires a number of plots or array equivalents in software. The hypothetical distribution is plotted, along with curves a distance $d_\alpha(N)$ above and below it. If $S_N(x)$ passes outside of this band at any point the data is rejected at the α level of significance, as not belonging to the distribution $F_0(x)$.

Sample Size (N)	α = 0.95	α = 0.50	α = 0.25	α = 0.05	α = 0.01
8	0.1086	0.5392	0.7797	1.1586	1.4327
10	0.1147	0.5426	0.7845	1.1658	1.4440
20	0.1298	0.5547	0.7975	1.1839	1.4698
30	0.1351	0.5605	0.8036	1.1916	1.4801
Over 30	0.1601	0.5887	0.8326	1.2239	1.5174

Table 3.15: Critical Values for KS Test

The algorithm implemented in this work uses this technique by recording in a table the observed and hypothetical distributions and calculating the maximum deviation between them. If this maximum deviation exceeds $d_\alpha(N)$ the distribution is rejected. This technique has been implemented where the data was available as the output of a linear congruential generator routine, with parameters obtained by a genetic optimisation mechanism. It is also important to note that this approach is applicable to data from any generator type, regardless of the algorithm used to generate it. As a simple example of the application of this technique, the KS test was applied to five samples of data drawn from an LCG. The hypothesis under test was that the data was drawn from a uniform population. A sample size of 1000 was used. In addition, the maximum deviations above and below the hypothetical distribution are recorded. If either should exceed the appropriate $d_{5\%}(1000)$ value, that sample fails. The critical percentage point of the distribution is therefore 1.2239. All samples except number 3 pass as indicated in Table 3.16.

Sample	Worst case KS+	Worst Case KS-
1	0.37	1.00
2	0.85	0.21
3	1.92	0.40
4	0.81	0.16
5	0.71	0.42
Reference generator	0.57	0.19

Table 3.16: KS Test Applied to Generator Samples

The data can also be represented visually, as shown in Figure 3.21. The horizontal line indicates the allowable maximum value of the test statistic $K_{n+/-}$. The lower line indicates the $d_\alpha(N)$ value, for $\alpha = 0.95$ and $N = 1000$. This value indicated a level where the data might be considered too good, in the sense that it might not have been generated randomly. None of the test data calculated in this example approach either value, indicating that the generation process is correct and that the samples appear random.

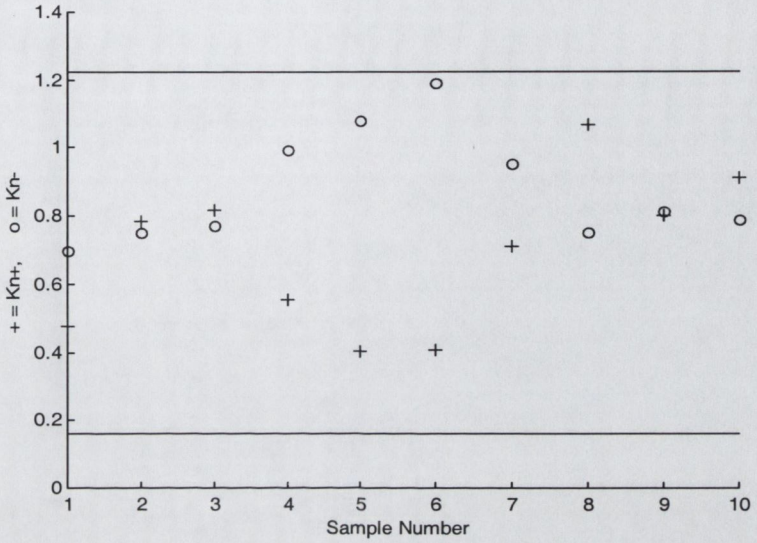


Figure 3.21: KS Test Results

In addition, individual test values derived from the KS test are distributed according to (67) [Birn52].

$$S_{\infty}(x) = 1 - e^{-2x^2}, \quad x \geq 0 \tag{67}$$

This allows the overall, or global behaviour of the data to be investigated by obtaining a number of individual $K_{n+/-}$ values and checking their distribution. The KS test was employed as a goodness-of-fit test for the examination of the power spectral density sample, as described above. While individual PSD samples may pass or fail a KS test, the global performance of the data may only be ascertained by examining the behaviour of a number of such pass/fail results. This is the objective of step (5) in the procedure outlined earlier. Before results of the entire test are presented, the significance of a given number of passes and fails of any test must be clarified. This requires the application of the binomial distribution, which is now briefly described again for the purpose of completeness.

Given p, the probability of success in any single event, and q = 1-p, the probability of failure, the probability of X successes in N trials is given by the expression in (68).

$$p(X) = {}_N C_X \, p^X q^{N-X} = \frac{N!}{X!(N-X)!} p^X q^{N-X}, \quad X \leq N \tag{68}$$

This expression is referred to as the Bernoulli or Binomial distribution. It may be applied to the analysis of power spectral density as follows. When the goodness-of-fit test is implemented on the empirical CDF of the power spectral density, it is performed at a specified level of confidence, alpha. This parameter determines the likelihood of the test returning an error, or equivalently, the number of failed PSD samples that should occur when running a batch of n such tests. By assigning alpha to p and n to N, the number of PSD goodness-of-fit acceptances and rejections may be investigated to determine whether they are appropriate to the corresponding binomial distribution. In this way, the global behaviour in addition to the local i.e. small sample, behaviour may be investigated.

Some typical results of the CDF test of power spectral density are now presented. The relationship between the results obtained and the theoretical spectral test results is explored. Finally, some generators whose parameters were obtained by the genetic selection mechanism are tested and the results are analysed. Firstly, the test is demonstrated on a good quality generator with a spectral test score of 27720. A sample size of 1024 was used, and the periodogram was estimated using the method of Widrow and Stearns, with no overlap. Figure 3.22 shows the PSD obtained, in addition to the theoretical value expected which is indicated as a horizontal line on the graph.

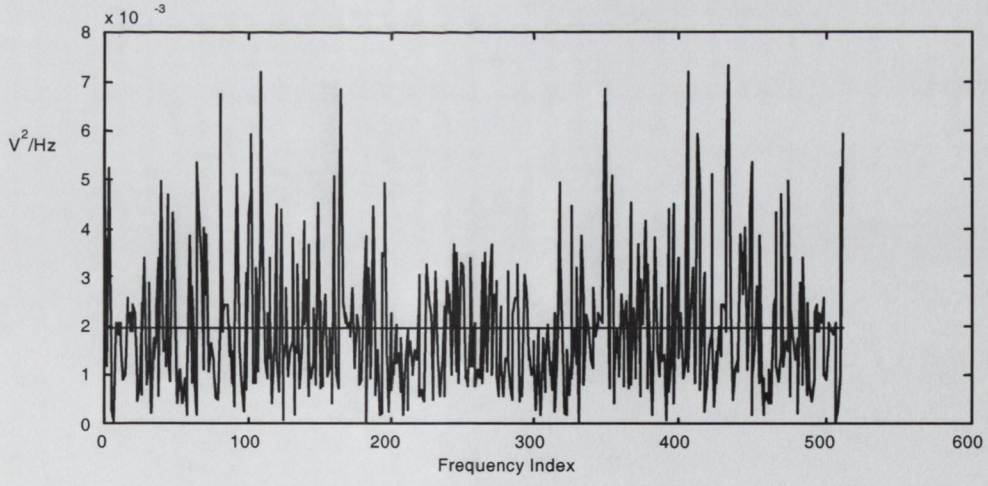


Figure 3.22: PSD from Optimised Generator

Both the empirical and the theoretical CDF distributions for the generator PSD were calculated and plotted in Figure 3.23. The theoretical CDF is given by the straight line, while the curved plot indicates the empirical CDF derived from the data. The maximum vertical

excursion from the theoretical to the experimental plot is the subject of the KS test and occurs at a frequency index of approximately 350.

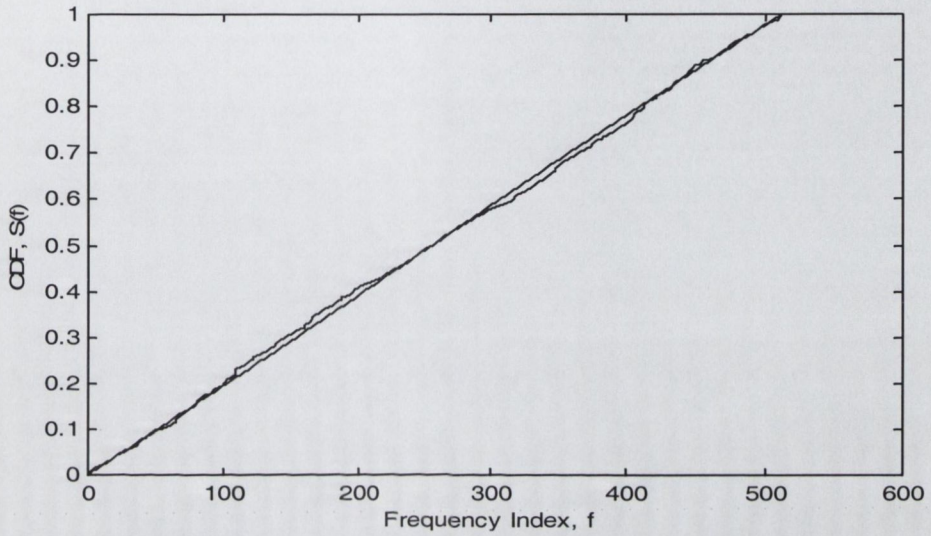


Figure 3.23: Empirical and Theoretical CDF of Optimised Generator

Finally, the KS statistics for the sample spectrum were computed, in addition to the χ^2 value for the sampled spectrum, the latter for comparison purposes only. The application of the KS test to the data yields a set of statistics, which are given in Table 3.17. The sample passes the CDF test of PSD at the 95% level of significance ($\alpha = 0.05$), as the values of K^+_{1024} and K^-_{1024} obtained are lower than the critical value $d_{0.05}(1024)$.

K^+_{1024}	K^-_{1024}	χ^2	$d_{0.05}(1024)$
0.47	0.52	0.49	1.22

Table 3.17: KS Test Statistics for Sampled Spectrum

The above set of tests was rerun, but this time using data from a poor quality generator with a spectral score of only 12.6. The periodogram, shown in Figure 3.24, displays a very uneven response. There are large components at the lower and upper ends of the sampled spectrum, while the portion of the data represented by indices 150 to 350 are lower in magnitude than expected. While the low frequency components indicate a data sample that changes value only very slowly, the exaggerated upper frequency response would seem to represent a sample that also has in addition an occasional excessive transient response.

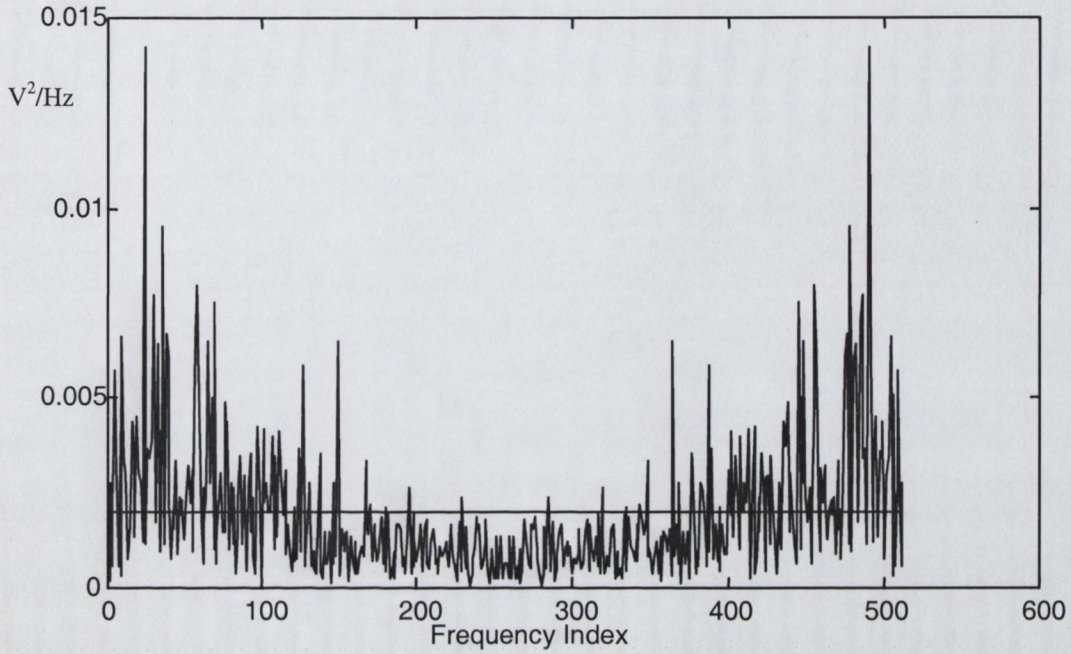


Figure 3.24: Periodogram of Poor Quality Generator

The CDF of the above PSD is given in Figure 3.25 below. As is evident from the diagram, the deviations from the expected function are quite large and in this particular case have an approximately symmetric distribution around the theoretical, linear CDF.

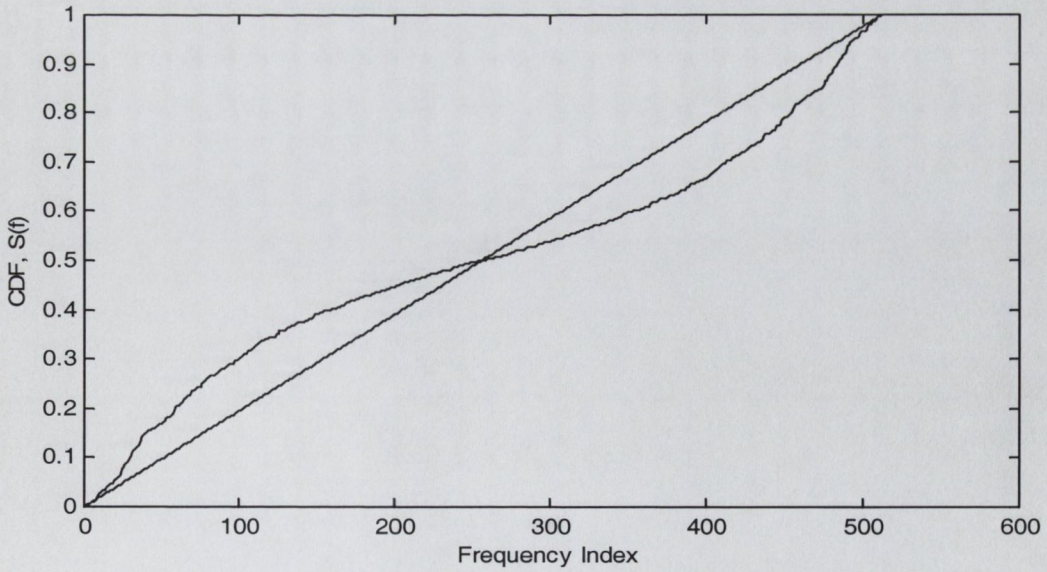


Figure 3.25: CDF of Periodogram of Poor Quality Generator

The relevant statistics for this sample are given in Table 3.18. The values for the two test statistics K^{+}_{1024} and K^{-}_{1024} are equal, which corresponds to equal excursions about the

theoretical CDF near indices 100 and 400. This sample fails the KS test at the $\alpha = 0.05$ level of significance.

K^+_{1024}	K^-_{1024}	χ^2	$d_{0.05}(1024)$
2.61	2.61	0.90	1.22

Table 3.18: KS Test Results for Sampled Spectrum of Poor Quality Generator

The next series of tests explore the relationship between the theoretical spectral test and the empirical KS test. A series of generators are tested, first with the spectral test and then with the KS test. The correlation between both tests is examined. Table 3.19 shows the parameters for a number of generators. The spectral test score is given, in addition to the K^+_{1024} and K^-_{1024} statistics for a sample output. All generators are 32-bit, with a period of 2^{32}.

Generator	Multiplier	Spectral Score	K^+_{1024}	K^-_{1024}
1	21	1637.8	0.66	0.67
2	29	2708.08	1.03	1.06
3	3065293	4042.96	0.33	0.34
4	49364597	8047.55	0.34	0.39
5	21479509	8088.65	0.53	0.54
6	33321709	8127.05	0.74	0.76
7	47375085	10221.75	0.35	0.38
8	134520405	11719.18	0.45	0.49
9	83081725	12591.95	0.46	0.46
10	231390117	12927.79	0.38	0.38
11	1517190677	15859.02	0.35	0.36
12	920531893	20173.08	0.46	0.50
13	660974685	21130.62	0.54	0.57
14	2147106333	23960.48	0.33	0.37
15	2343534885	25193.31	0.42	0.42
16	3413489181	25337.69	0.23	0.25
17	1046426629	25512.32	0.26	0.28
18	3484959045	28970.69	0.27	0.28
19	3565697325	30908.31	0.55	0.57
20	3476492325	32947.25	0.31	0.35

Table 3.19: Spectral Test and KS Test Results for a Number of Generators

The resulting plot of K^+_{1024} test results against the spectral score, given in Figure 3.26, shows that qualitatively, the general trend of higher spectral scores leading to improved PSD performance is observed. The data consists of small samples drawn from a generator with a large period and so there is inevitably considerable noise in the plot, but the improvement is nonetheless apparent.

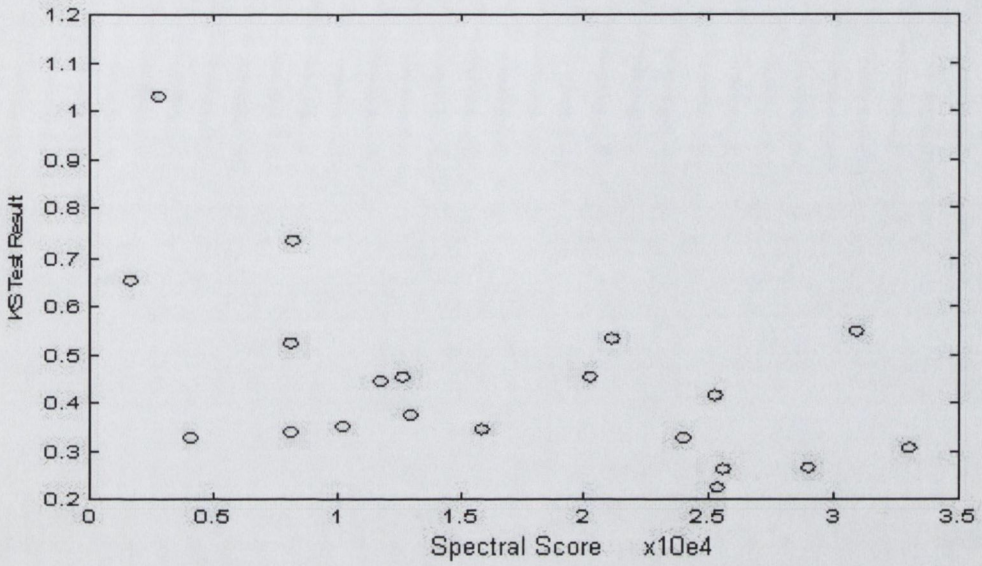


Figure 3.26: Relationship Between Spectral Score and KS Test

3.5.4 Test 4: A Novel Test

The final section of this Chapter applies the techniques discussed and presented in the preceding sections to the development of a novel test. The algorithm was run a number of times and several generators were produced. The spectral score of each of these generators was obtained and a sample of the data output was analysed using the KS test. In each case the Matlab generator was used as a reference source, providing a known good generator as a basis for comparison with the generators produced. Finally, a number of samples from each generator were tested with the binomial test to verify that the number of passes and fails of the KS test observed were in accordance with a uniform PSD.

Each of the generators in Table 3.20 was obtained by running the genetic algorithm for 100 generations with an initial population of 200. The period selected was 2^{42}. Twenty samples of length 10240 were then taken from each generator and the KS test was performed on each sample. The number of passes and fails are recorded, for a 95% confidence level, in addition to the average of the highest and therefore the worst KS value obtained for each sample. The critical KS value for this test is $d_{0.05}(N) = 1.36/N^{1/2}$, where N = sample size, which is 1024 in this test [Mass51]. This yields a test parameter value $d_{0.05}(1024) = 0.0425$. Samples with KS values greater than this are not sufficiently random, according to the standard administration of this test. The results indicate that the GA provides a suitable mechanism for selecting generator parameters, using the tests of output data described earlier.

Generator	Multiplier	Spectral Score	Average KS value	Number of passes
1	96716669	150602	0.0208	20
2	1257893517	147162	0.0213	20
3	4035528829	141922	0.0225	20
4	1105181349	147125	0.0206	20
5	1551086125	148377	0.0189	20
6	278681677	143456	0.0175	20
7	1347387893	128828	0.0202	20
8	2433786117	172278	0.0223	20
9	1851497965	156918	0.0204	20
10	9876765557	158360	0.0192	20
11	1904180213	169245	0.0216	20
12	93151629	149418	0.0207	20
13	11318344693	153155	0.0198	20
14	5055058229	137339	0.0205	20

Table 3.20: Spectral and KS Test Results for Optimised Generators

The average KS values are clustered around 0.02, which probably represents the lower limit for the randomness obtainable from this approach. In the context of Monte Carlo power estimation, this degree of randomness is sufficient. All samples pass the test, which indicates good global randomness.

In the event of a number of samples failing the test, the significance of the amount of fails is examined using a binomial test, as described previously. To demonstrate the application of this test, some less random data than the above samples must be obtained, because all of the above samples have passed. A convenient method of obtaining such data is to first take a known good sample, such as the above generators might produce, and to remove some of the randomness by introducing an element of correlation into the data. In addition, it would be useful to be able to control the degree of randomness removed to ascertain the sensitivity of the test. A wide-sense stationary autoregressive process of order p is a special case of an auto-regressive moving average or ARMA(p,q) process in which $q = 0$ [Haye96]. An AR(p) process may be generated by filtering white noise with an all-pole filter of the form given in (69).

$$H(z) = \frac{b(0)}{1 + \sum_{k=1}^{p} a_p(k) z^{-k}} \tag{69}$$

This process produces correlated, non-random data, where the autocorrelation $r(k)$ is estimated from the N-point data $x(n)$ using (70).

$$\hat{r}(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)x(n-k) \tag{70}$$

An AR(1) process with parameter a_p = 0.1 or –0.1 results in data with non-zero autocorrelation with high frequency and low frequency variations respectively. Samples of such data, generated by filtering the data used for Table 3.17, with parameter a_p = 0.1, were tested for randomness using the same test as described above. The number of samples failing the test was also recorded. The data in Table 3.21 demonstrates the sensitivity of the test, in that the failure of samples indicates the detection of correlation within the sample.

Generator	Average KS value	Number of passes (/20)	Pass/Fail Binomial Test
1	0.0701	5	Fail
2	0.0659	5	Fail
3	0.0678	7	Fail
4	0.0709	5	Fail
5	0.0641	8	Fail
6	0.0703	5	Fail
7	0.0688	7	Fail
8	0.0686	5	Fail
9	0.0645	7	Fail
10	0.0677	4	Fail
11	0.0657	8	Fail
12	0.0751	3	Fail
13	0.0699	3	Fail
14	0.0727	5	Fail

Table 3.21: Sensitivity of Binomial Test with Filter Parameter -0.1

Replacing the filter parameter a_p = -0.1 with the value 0.1 results in the data given in Table 3.22.

Generator	Average KS value	Number of passes (/20)	Pass/Fail Binomial Test
1	0.0505	16	Pass
2	0.0486	19	Pass
3	0.0456	18	Pass
4	0.0472	18	Pass
5	0.0436	19	Pass
6	0.0499	18	Pass
7	0.0514	17	Pass
8	0.0463	19	Pass
9	0.0445	18	Pass
10	0.0443	20	Pass
11	0.0460	18	Pass
12	0.0508	16	Pass
13	0.0469	16	Pass
14	0.0462	18	Pass

Table 3.22: Sensitivity of Binomial Test with Filter Parameter 0.1

Table 3.22 indicates that all the generators pass the test when the filter parameter is 0.1. This suggests that the output from any generator may be classified with respect to its degree of randomness, by associating it with an equivalent filtered process. The graph shown in Figure 3.27 indicates the motivation for such an approach. The response of the test to various degrees of filtering is shown. It can be seen that as the filter parameter increases in magnitude, i.e. as the data becomes less random, the average KS value increases. This may be useful as a means of quantifying the randomness of a given generator, by associating it with an equivalent AR(p) process, particularly when a negative coefficient is used. This would have the advantage of being a more widely known measure than the KS test, and once standard sample sizes were employed, could be used as a comparative figure of merit for a generator.

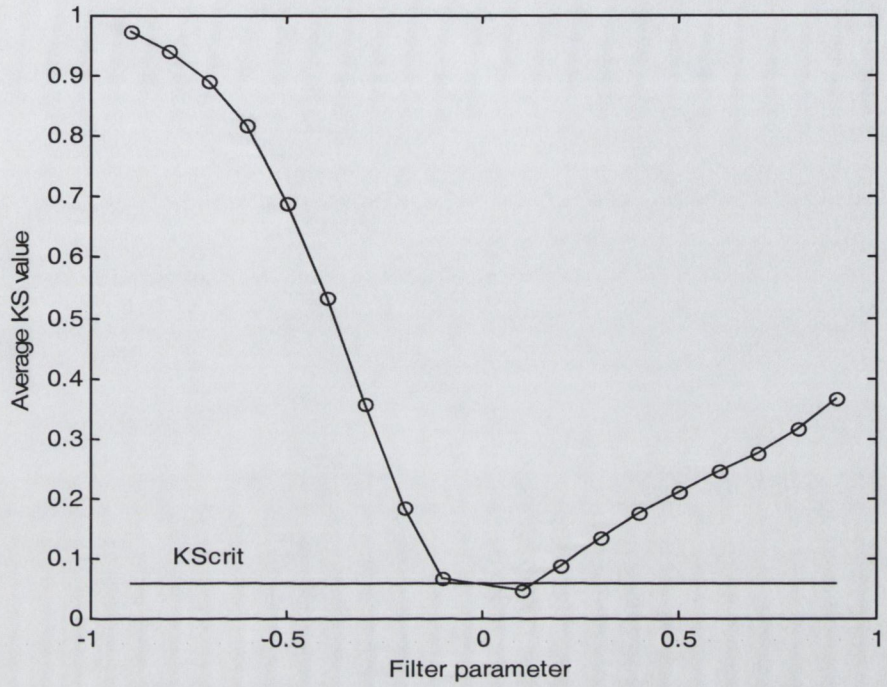


Figure 3.27: KS Test Results for Filtered Data

In addition, Figure 3.27 also demonstrates that the KS test is sensitive to data with a correlation of 0.1 and above, which represents a relatively small amount of non-random behaviour. Therefore, the test exhibits a reasonable amount of sensitivity. The varying response of the test to positive and negative filter coefficients can be clearly observed, with data drawn from positive coefficients having in general lower KS values. This behaviour is not significant in the context of this work as only one type of coefficient should be used to rate generators. This aspect of the test was therefore not investigated further. By choosing to grade generators against a negative coefficient process, a larger range of KS values are possible and therefore this portion of the characteristic is sufficient to successfully implement the test.

3.6 Summary and Conclusions

This Chapter has been concerned with presenting three enhancements to the current body of knowledge concerning the production and test of UWN data for use in Monte Carlo simulations. Firstly, a novel technique for the production of generator parameters has, to the author's knowledge, provided the first automated method for creating a UWN generator based on the LCG equation. Secondly, a useful test technique, the application of the KS test

to the Power Spectral Density has been modified to a form more useful in engineering applications. This was achieved by transformation of a Walsh transform based technique into a Fourier based one. The third novel contribution proposed a new test for UWN data, based on determining an equivalent AR(p) process for the generator under test, with lower p values indicating better data.

The structure of this Chapter consisted of five sections. Firstly, a useful data visualisation technique for pseudorandom data was implemented in two and three dimensions. The LCG equation for generating the data was then presented in Section 3.2. Using the concept of accuracy, a link between the planar structure visualised in Section 3.1 and data quality was established. The Spectral Test was introduced in Section 3.3 as a suitable means of determining the accuracy in any number of dimensions. Various implementation techniques were presented which allowed the maximum precision in the Spectral Test results. In Section 3.4 a genetic algorithm was used to produce generators of any user-defined quality, using the Spectral Test as the objective function. Finally, in Section 3.5, the output from the genetically obtained generators were comprehensively tested in four separate tests including a novel test. In these tests, the uniformity and independence of the data samples were shown to be the most significant characteristics. The first test examined the histogram of the data, while the second test looked at the PSD. The spectral densities of the generators obtained were shown to be adequately uniform. The third test was a test for independence of the data, with a KS goodness of fit test on the uniformity of the PSD. This three-level approach comprises a sampled spectrum, a test of uniformity on the spectrum and a binomial model for the number of test failures. It is therefore a reasonably powerful test of the randomness of the sampled data. Using this technique, genetically derived generators appear to be adequately independent at the 1% level. Based on Test 3, a comparative measure of randomness for individual generators was also proposed, based on the equivalent response of an AR(p) process. This also allows the sensitivity of the test to be gauged, in that it appears to be insensitive only to data with a small amount of correlation, corresponding to the output from a first-order process with parameter magnitude of less than 0.1. The final test, Test 4, associated the generator output with an equivalent AR(p) process. The next Chapter will present a tool for power estimation in CMOS circuits, developed using the techniques explored in this Chapter.

4 Development of a Power Estimation Tool

The previous Chapter described a method for generating and evaluating UWN binary vectors for use as an efficient input data stream when estimating the power dissipation of CMOS circuits. The aim of this Chapter is to describe a new power estimation tool that is based on the use of UWN data for the estimation of average power dissipation in CMOS circuits. Such tools are useful to the designer in that they automate and consequently speed up the circuit analysis and design development cycles. An integral part of this thesis has been the development of LLAMA (Low Level Analysis for Microelectronic Architectures), one such power estimation tool. This tool operates at the circuit, i.e. layout, level. It consists of a Monte Carlo 'wrap-around' program, which uses a SPICE engine for simulation. The output from the genetic algorithm described in the previous Chapter is employed to provide sufficiently random vectors for the program. The incorporation of the genetically modified data into a Monte Carlo power estimation tool is then described. The stopping criterion for the simulator is outlined and simulation results for various circuits are reported. Finally, additional functionality is presented using a novel approach to enable less time-consuming estimates of power to be made in certain circumstances.

4.1 LLAMA, a Power Estimation Tool

Because of its ability to dynamically determine when to end a simulation, the Monte Carlo (MC) technique has been successfully applied in various forms to many problems, including that of power estimation [Sber97], [Sobo94], [Bind92]. It is a statistical technique which is dependent on the input data applied to the circuit. The input patterns are applied and the resulting power is monitored with a simulator. This is continued until the result is obtained with the required accuracy, at a specified confidence level. One major feature of the Monte Carlo method is that it is dimension independent, which means that the number of samples required to achieve an adequate estimate is independent of the circuit size [Burc93]. The block diagram in Figure 4.1 gives an overview of the technique as implemented in this work.

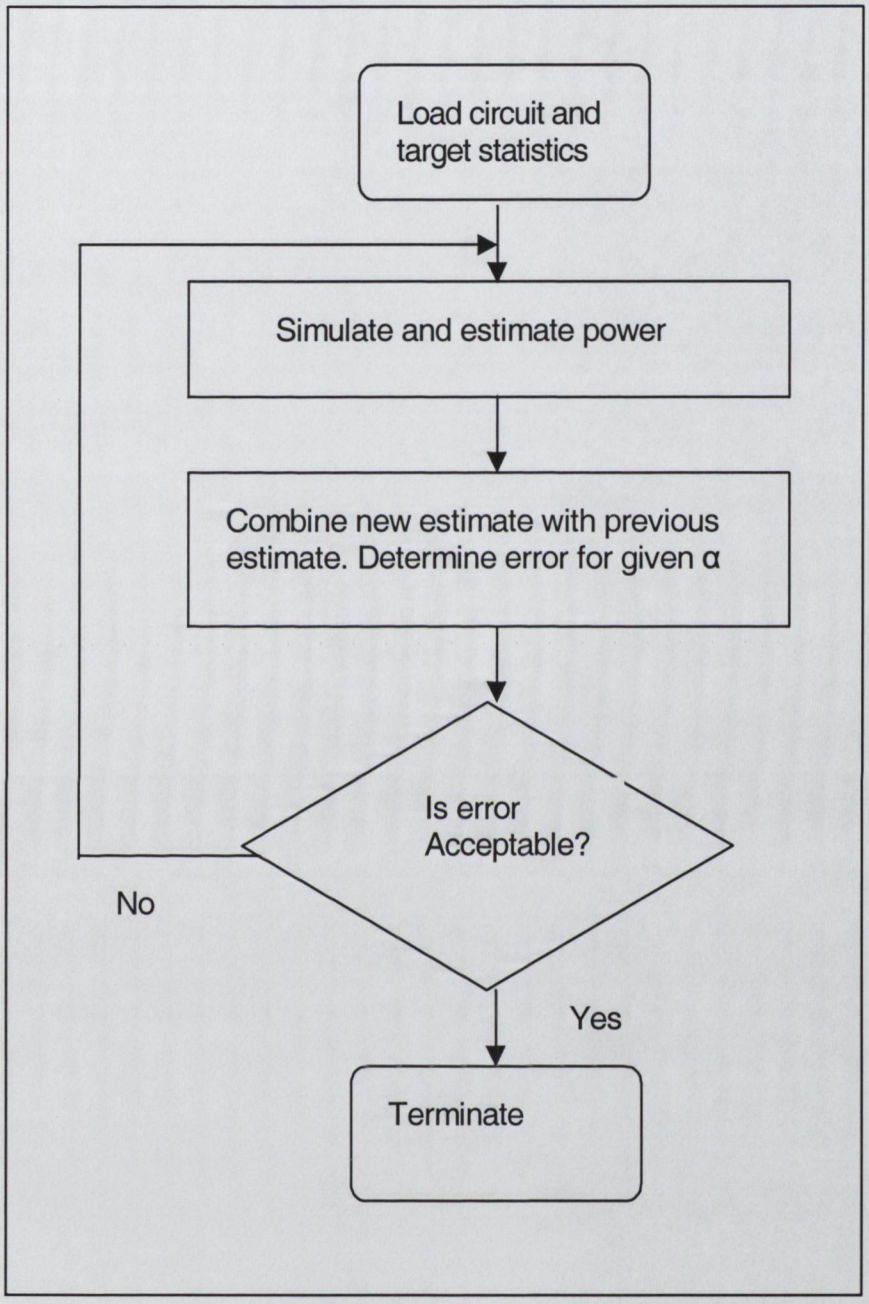


Figure 4.1: Monte Carlo Power Estimation

At the beginning of the simulation, two parameters, the desired error and the confidence level are specified. A simulation run, usually referred to as an iteration, is then executed with a fixed number of input vectors. After a small number of iterations, typically three to five, the error is calculated [Hamm64]. If the estimated error is small enough, the simulation terminates with the average power estimate. If the error is still larger than the value specified by the user at the start of the simulation, the iterations continue until the error reduces to the

required value. This iterative approach gives the MC method two distinct advantages over techniques such as the probabilistic approaches discussed in Chapter 2. It avoids the speed/accuracy trade-off of those techniques, in that the desired accuracy may be achieved in realistic time for even large circuits [Ding98]. The simplicity of the basic algorithm makes it easy to implement in conjunction with existing logic or circuit simulation environments. The operating principle behind the MC method is now briefly described.

Typically, a circuit description consists of a number of active elements, linked by capacitive nodes. Regardless of how the power estimate is obtained, the total power will be determined by the summed contributions from each node, many of which are independent from each other. The central limit theorem [Vini98] indicates that in such circumstances the total power will be approximately normally distributed. This is demonstrated in Figures 4.2 to 4.5. In these figures, the activity statistics from a 1-bit full adder are reported. For each of two internal nodes labelled $N1$ and $N2$, the number of logic transitions was recorded for each one of 1000 simulations. Each simulation consisted of a single iteration with 1000 UWN input vectors. The histogram and normal scores plot [Freu90] for each node were then plotted. The normal distribution of the bit transitions and hence the power dissipation is clearly evident.

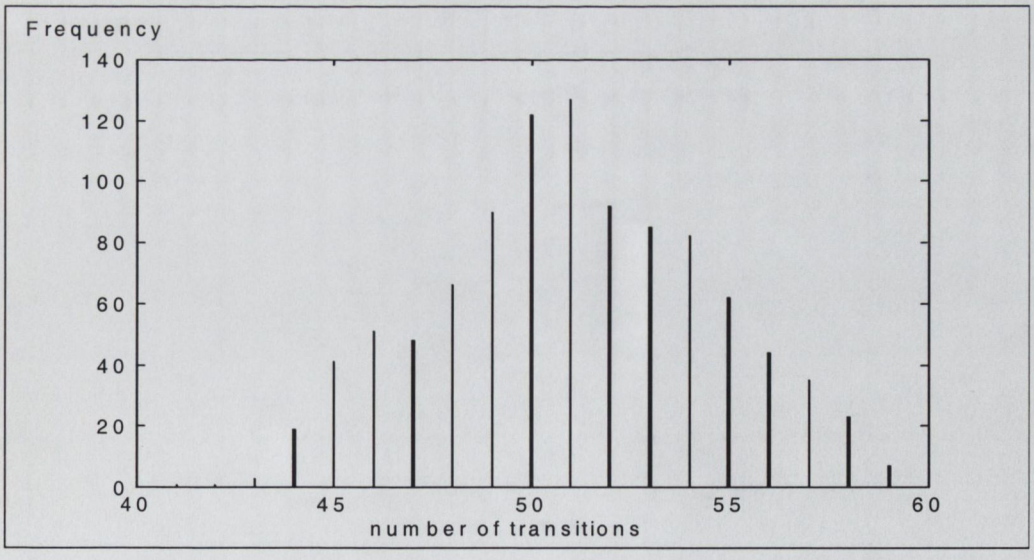


Figure 4.2: Distribution of Node Transitions for Node $N1$

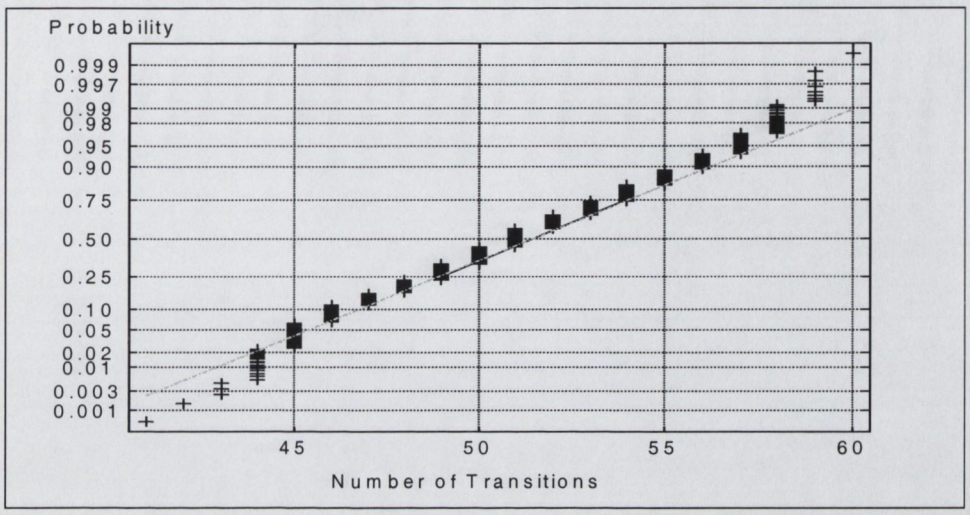


Figure 4.3: Normal Scores Plot for Node *N*1

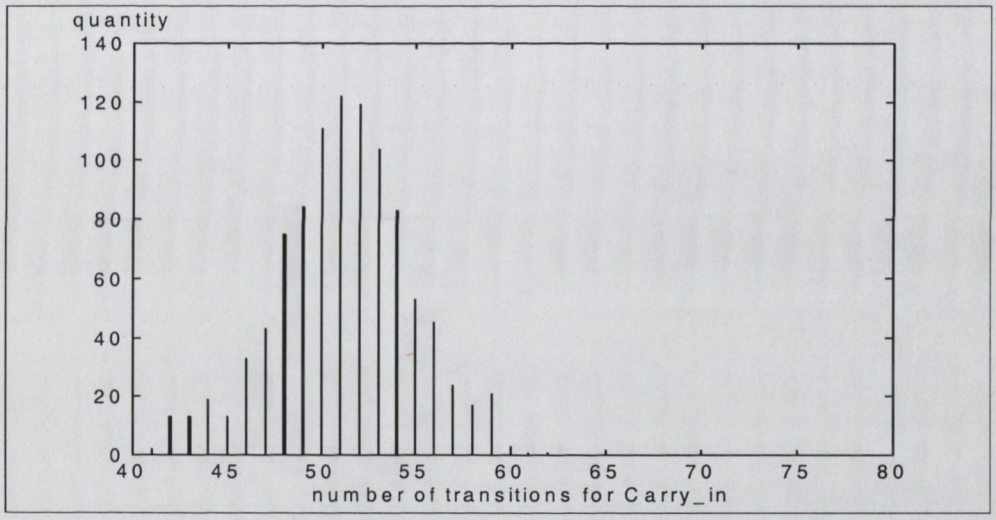


Figure 4.4: Histogram of Node Transitions for Node *N*2

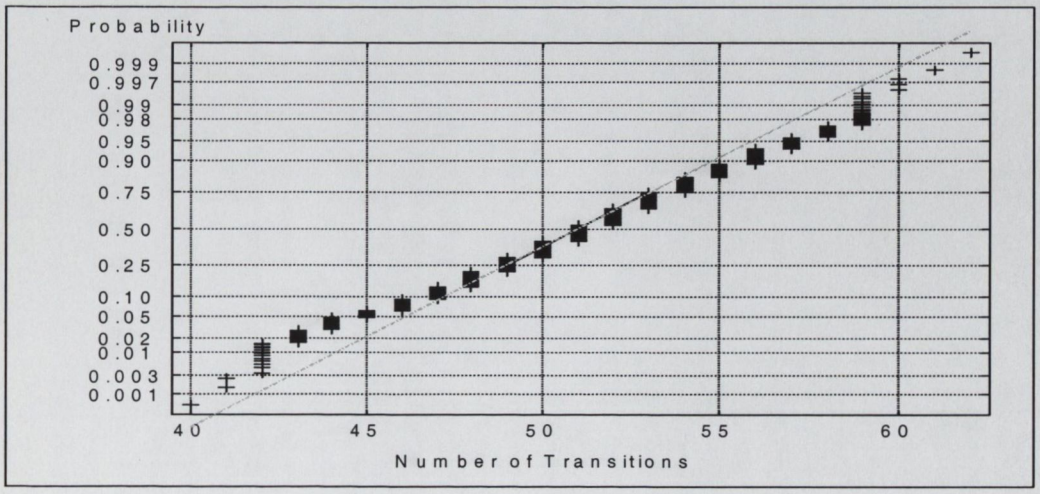


Figure 4.5: Normal Scores Plot for Node *N*2

When the input data is random, then the average of the power estimates converges to the true power dissipation. The more power estimates obtained, the closer the average value converges to the true value [Huiz90]. In the context of this work, the true power is taken to be that power dissipation the value of which may be asymptotically approached by running an arbitrarily long simulation. In practice, estimates of the true value of the power dissipation when UWN data is the stimulus are determined by running very long simulations. This may involve over 100,000 input vectors with simulation times of over a week.

The user may decide how closely the estimate need approach the true value by specifying the required accuracy. The estimate is finally produced within a given confidence interval, typically 95% to 99%. The normally distributed power estimates are used to tell the simulator when to stop by establishing a stopping criterion which is evaluated at the end of each iteration. The stopping criterion is obtained as follows. Given a normally distributed variable of total power P_T, running N iterations of the simulator produces N different estimates of the average power dissipation. A sample average η_T and a sample standard deviation s_T are then formed. There is therefore (1-α)100% confidence [Spie88] that

$$\left|\eta_T - E[P_T]\right| < z_{\alpha/2} s_T / N^{1/2} \tag{71}$$

where $z_{\alpha/2}$ is obtained from the normal distribution. This may be rewritten as in (72), where the left-hand quantity represents the fractional error in the estimated power.

$$\left|\frac{P - \eta_T}{\eta_T}\right| < \frac{z_{\alpha/2} s_T}{\eta_T \sqrt{N}} \tag{72}$$

Therefore, for a specified percentage error ε in the power estimate and for a given confidence level (1-α), the circuit is repeatedly simulated until (73) becomes true. This expression is referred to as the stopping criterion.

$$\frac{z_{\alpha/2} s_T}{\eta_T \sqrt{N}} < \varepsilon \tag{73}$$

At the end of iteration N, both the standard deviation s_T and the mean η_T are estimated. The test statistic above is also calculated. If the value obtained is larger than the specified

error, the result is not accurate enough and a further iteration is performed. This cycle is repeated until the specified accuracy is obtained.

Given the accuracy and the confidence level, the remaining parameter to be selected is the sample size, i.e. the number of input vectors per iteration. The optimum sample size is not readily apparent [Marc99]. The smaller the sample size, the less time is taken by each simulation cycle. However, it is easily observed that using a larger number of input vectors results in a quicker convergence, that is less iterations are required. The overall simulation time is also highly dependent on the size of the circuit. Some typical data are presented in Figure 4.6, where the total simulation time and the average time per iteration is given as a function of the sample size for a 12-bit adder. The data was obtained using the MC simulator LLAMA, the operation of which is detailed in the next section.

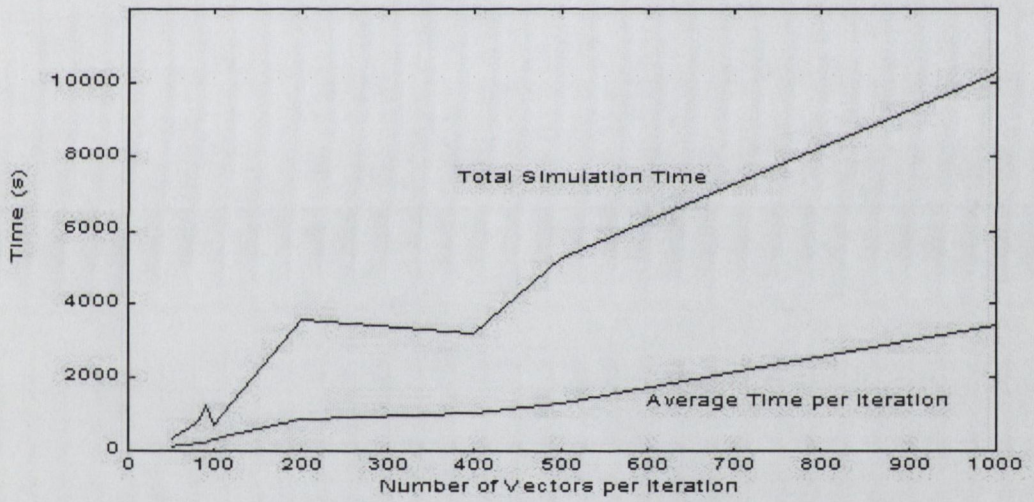


Figure 4.6: Simulation Time as a Function of Input Sample Size

In general, the number of iterations required for convergence declines with the sample size. However, the simulation time for each iteration increases, as there are more vectors per sample. This offsets any benefit in reducing the input sample size unduly. Figure 4.7 gives the number of iterations required for 50 simulations of the 12-bit adder, with 50 input vectors per iteration. There is a spread of values, from 2 to 12, reflecting the uniform distribution from which the data is drawn.

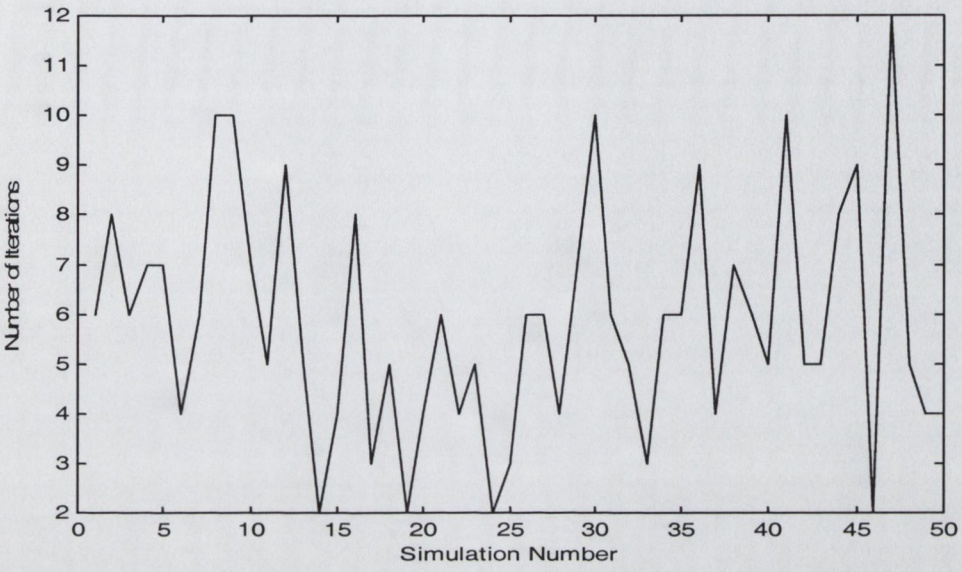


Figure 4.7: Number of Iterations of MC Loop for Convergence

Based on a review of practice as reported in the literature [Huiz90], [Najm94], [Schw01] and the experience of running many such simulations as reported in this section, a figure of 100 inputs per sample appears to be an acceptable choice of sample size. It provides an adequate balance between the three time measures of iteration time, number of iterations and total simulation time. The use of other sample sizes is not greatly significant, affecting one or other of the three measures, but not altering the accuracy or effectiveness of the MC technique in any way.

4.2 Operation of LLAMA

Details are now provided concerning the internal organisation and operation of the power estimation tool LLAMA. Figure 4.8 shows an outline of the implementation. The tool operates in conjunction with a SPICE simulation engine. The input to the estimator consists of a circuit-level netlist and some start-up parameters. Based on the output from the data generator, a stimulus file containing the input vectors is generated for the current iteration. These vectors are applied to the simulator at a rate determined by the user-specified sampling frequency using a file-based input mode within the simulator. The resulting output is a matrix of time current pairs, which represents the total supply current variation over the simulation period. The time resolution of this matrix is dynamically controlled by the simulator and is typically of the order of picoseconds depending on the circuit activity and the simulator parameter RELTOL [Micr96], which sets the lower limit of time increment. This limit exists

to prevent instability in the simulator, which can lead to convergence problems if the increment is too small.

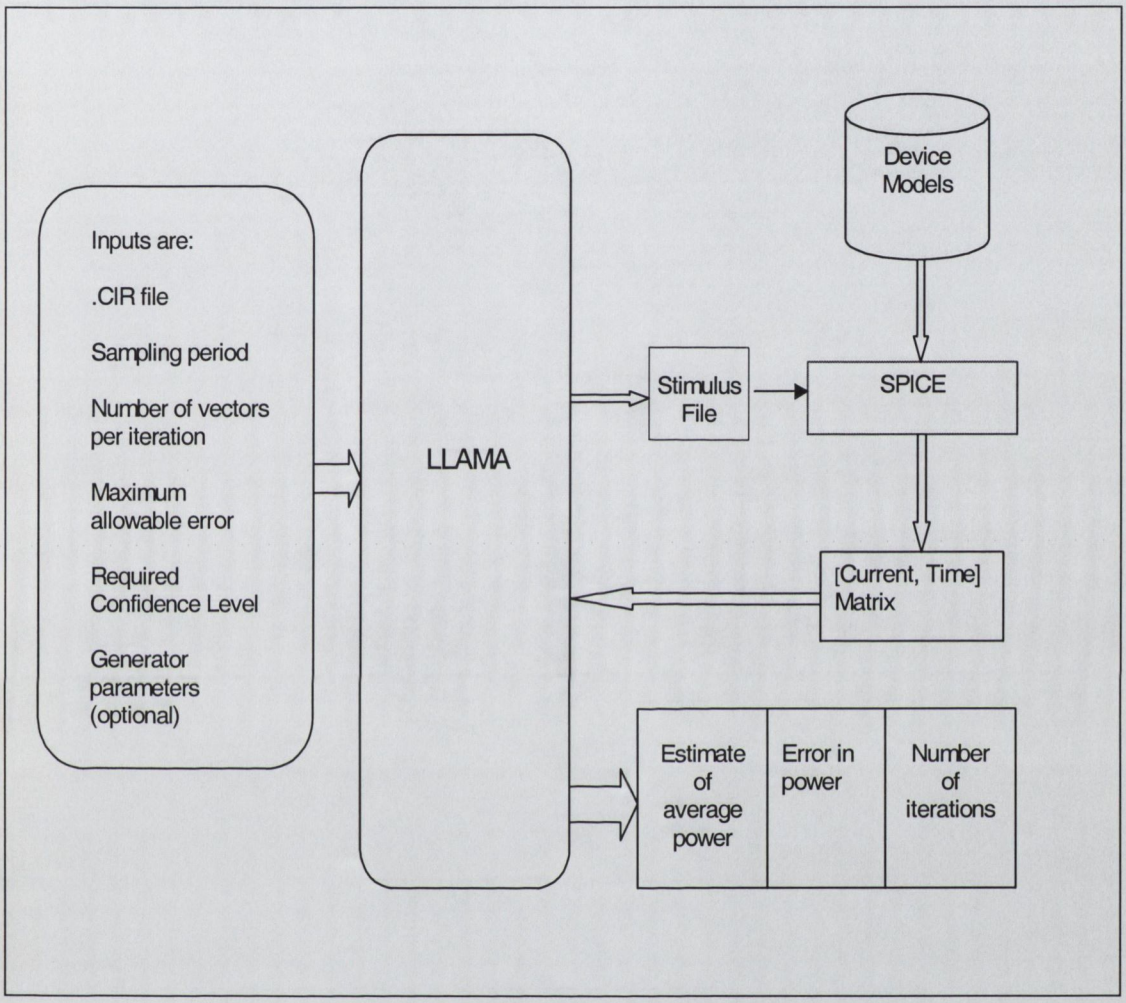


Figure 4.8: Operation of LLAMA

The average value of the current waveform is then obtained using trapezoidal integration, which, when multiplied by the supply voltage, yields the average power dissipation. The stopping criterion is then evaluated and a decision is made on whether to proceed with the next simulation cycle. If the criterion has been met with an error less than the specified value, the program terminates and the error, the number of iterations and the estimated mean power dissipation are reported.

The following figures illustrate the operation of LLAMA. Firstly, a typical simulation run is analysed in Figure 4.9, which illustrates the distribution of power estimates over a simulation run of 30 iterations. The normally distributed values are clustered around a mean

of approximately 0.15mW, which is the final power estimate from the simulation. Two outlying values of 0.26mW occurred during the simulation.

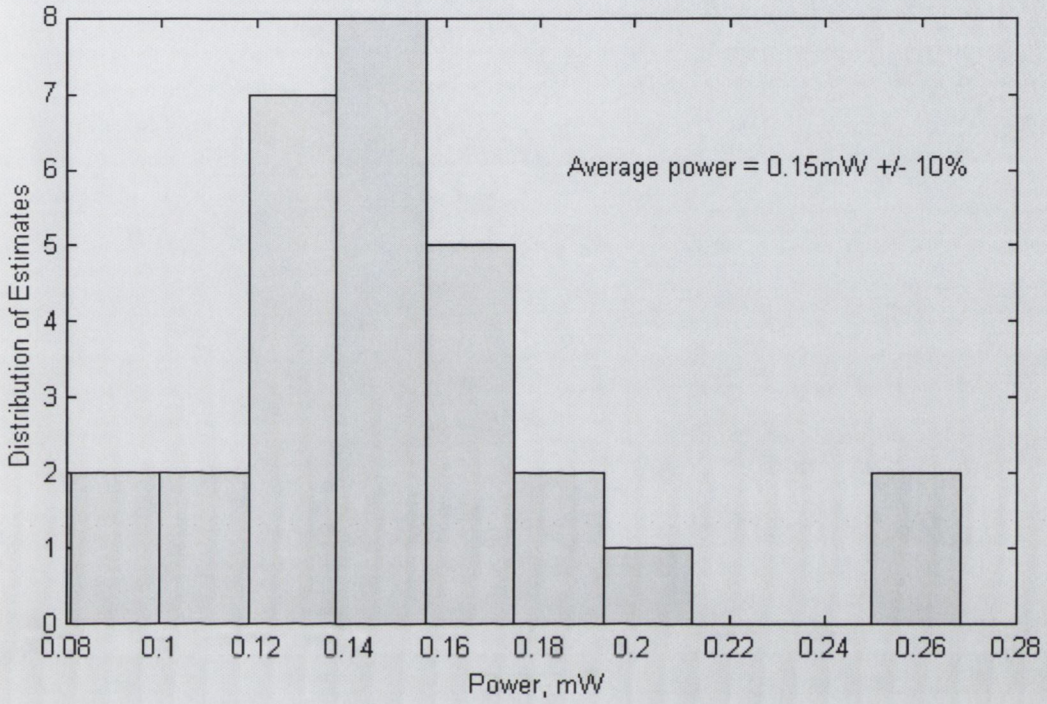


Figure 4.9: Typical Monte Carlo Simulation Run

The effect of this large value on the convergence is seen in Figure 4.10, which plots the error as a function of the iteration number. As a result of the excessive current caused by the particular set of input bit transitions during iteration 3, the power dissipation recorded is well in excess of the average power. As the Monte Carlo method estimates the average power only, such extremes of power dissipation are to be occasionally expected. The effect of such extrema is to prolong the convergence of the error and to extend the simulation time. In this simulation, large power dissipation was recorded during iterations 2 and 11, in each case causing the error to increase. As the effect is cumulative, the impact of the second value is much less. It does mean, however, that the error function is non-monotonic and is subject to statistical variation.

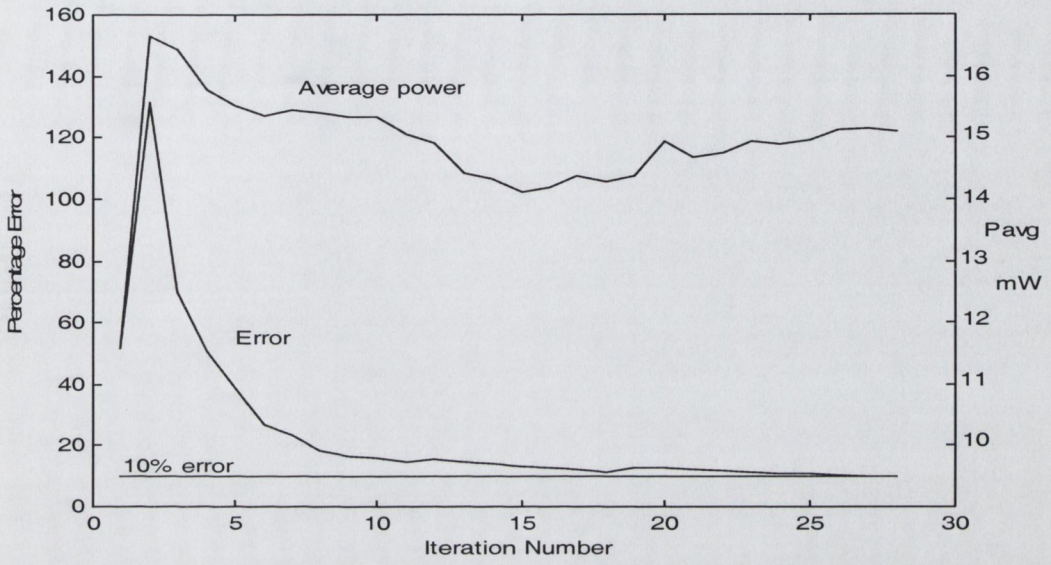


Figure 4.10: Variation of Average Power and Error During Simulation

The figure also shows a running average for the power. The average approaches the final value of 0.15mW as the error decreases to the 10% specified by the user. Although the power calculated during iteration two was 0.26mW, the previous value was much lower, thus giving an average value of approximately 16.5mW.

A number of circuits of increasing size were tested to give an indication of how Monte Carlo power estimation behaves with increasing circuit complexity. The details of each circuit are summarised in Table 4.1. The circuits, n-bit adders, for n = 4, 8, 12, 16 and 20, were automatically created using a synthesis tool which was developed during this work. This allowed the verification and exploration of the power estimation and data generation techniques developed. It also provides a convenient method of producing consistent, repeatable netlists for verification of the power estimation software. The circuits were simulated with 100 vectors per iteration, at a 99% level of confidence. The maximum allowable error was 20%.

Circuit	Total Number of input bits	Number of transistors
adder4	4	72
adder8	8	148
adder12	12	348
adder16	16	548
adder20	20	748

Table 4.1: Summary of Test Circuits

A number of variables were recorded, including the total number of iterations, the average time per iteration and the total simulation time. The results are given in Table 4.2.

Circuit	Number of iterations	Average time/iteration (s)	Total simulation time (s)
adder4	4	56	224
adder8	2	144	288
adder12	3	239	717
adder16	4	314	1257
adder20	5	406	2033

Table 4.2: Simulation Results with Test Circuits

As expected, the simulation time for a circuit is a function of the circuit size. This is most clearly indicated by the relationship between average iteration time and the circuit size. Due to the input pattern dependence of the MC process, there is no deterministic relationship between circuit size and the number of iterations. Large samples of 1000 input vectors and greater do occasionally result in a smaller number of iterations, but this is not guaranteed. They can also lead to a decrease in the total number of iterations per simulation run, as the error between iterations is reduced, due to the larger number of vectors per iteration. One consequence of this practice is that there are insufficient power estimates to use a normal distribution. Student's *t*-distribution [Spie88] provides an approximation to the normal distribution when the sample size is small. It is specified for α degrees of freedom, where, for a sample size N, α is equal to N-1. The stopping criterion is modified by replacing the normal variate with an appropriate *t*-statistic [Najm93]. Thus after N iterations, with a (1-α)100% confidence level, the stopping criterion becomes

$$stopping\ \ criterion = \frac{t_{\alpha/2(N-1)} s_T}{\eta_T \sqrt{N}} \tag{74}$$

In this expression s_T is the estimated standard deviation, n_T is the mean and t is the *t*-statistic with N-1 degrees of freedom. In practice, the majority of simulations require less than thirty iterations and so this stopping criterion is routinely incorporated into Monte Carlo applications, as is the case in this implementation.

4.3 Use of Genetically-Optimised Generators

The motivation for providing a suitable source of data for Monte Carlo estimators can be appreciated by observing the effect of poor quality input data on the power estimates obtained. In the following experiment, the power estimation tool was repeatedly run on a single circuit, with a large number of generators with randomly chosen parameters. The circuit was also simulated using data from a similar number of generators whose parameters were chosen by genetic optimisation. The results from the non-optimised generators are shown in Figure 4.11 and show a wide variation in estimates.

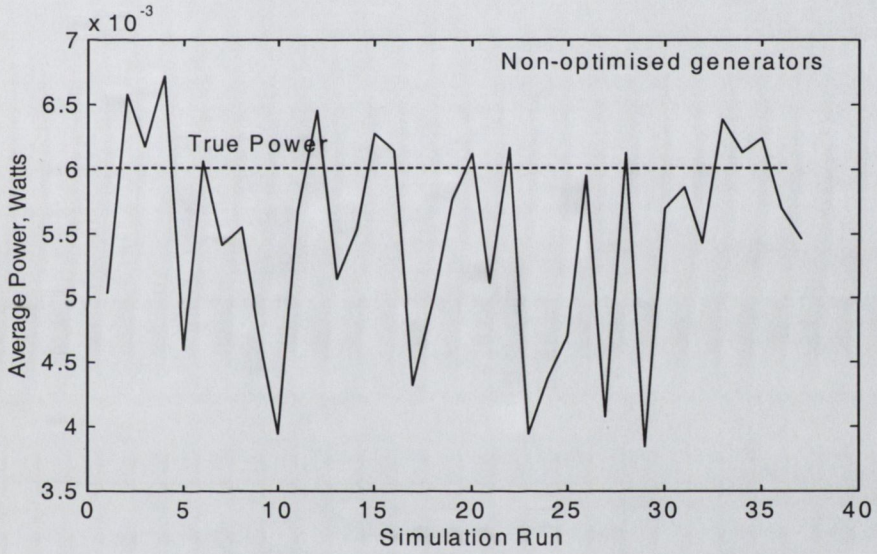


Figure 4.11: Effect of Generator Quality on Power Estimates

The line labelled true power was obtained by running a single long iteration of 10,000 input vectors. In this instance, the results underestimate the actual power, possibly due to reduced circuit activity occurring because of one or more input bits with low transition rates. This is equivalent to an input sample with a poor high frequency spectral response. In contrast, the results obtained using the optimised generators show much less variation and provide better estimates of the true average power. The same circuit was then driven with a set of genetically-optimised generators. The power estimations were obtained using LLAMA under identical conditions. For these results there is a much closer correspondence between the reference power and the values obtained from the genetically optimised generator. Figure 4.12 shows the output produced by these generators. In this case also, the estimate labelled

true power was produced using a known good source of data, namely the Matlab generator, with a 10,000-vector simulation of the circuit.

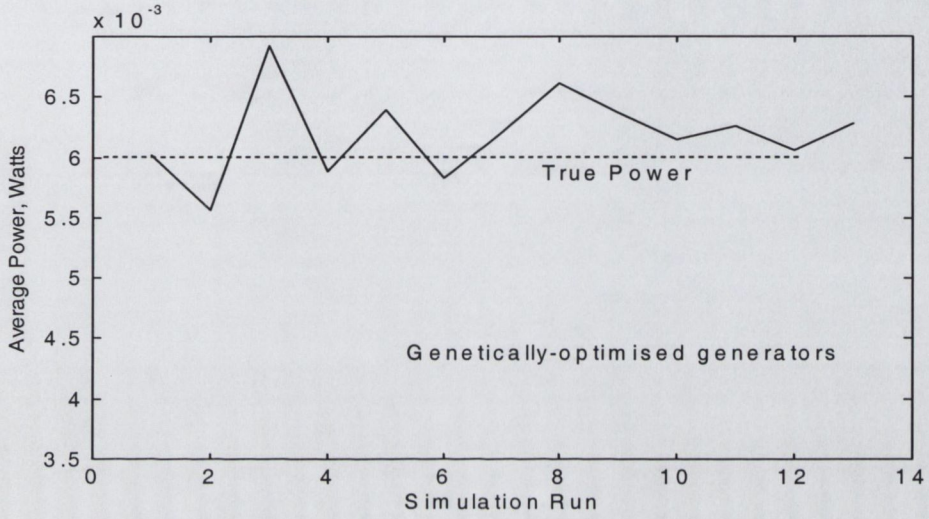


Figure 4.12: Improved Performance of Optimised Generators

These results demonstrate that the input bit transition rates closely resemble those of ideal uniform white noise. The slight over-estimation of power is possibly due to bit transition rates that slightly exceed the ideal value, which is represented by a transition probability of 0.5, as defined in Chapter Two.

The average number of iterations required was also observed over several simulation runs. The non-optimised generators and the reference generator were tested. The results are given in Figure 4.13, which shows the average number of iterations for three types of generators, optimised, non-optimised and the reference generator.

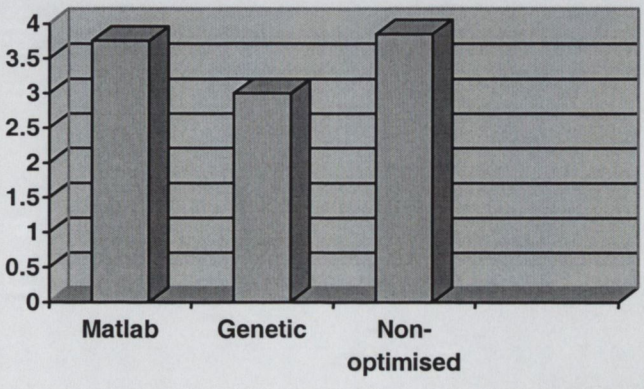


Figure 4.13: Comparison of Time Performance of Generators

The bar graph shows that the genetically-optimised generator is at least as good as the others when the simulation time is an issue. This again is indicative of the overall random quality of the generator. Given that a 500-transistor circuit may take up to 20 minutes per iteration, based on Table 4.2, even one iteration less represents a useful reduction in simulation time.

4.4 Open-loop Monte Carlo Estimation

Much of a circuit designer's time is spent making comparative decisions about functionality, speed and energy. This process is usually represented as a design cycle, through which the designer passes many times, as the design approaches the specified behaviour or power consumption. Only on the first pass through the cycle is the designer without a previous version of the design with which to compare specifications. In addition, design specifications are often mutually exclusive. For example, speed and power or area and power are often traded-off. In this manner a hierarchy of objectives is established, which prioritises the different aspects of the design specification. Achieving the required functionality is obviously the primary aim, often followed by timing considerations such as cycle time and latency. Only after such aims have been met, can the power consumption be optimised. This is, of course, aided by a top-down approach to power consumption where power consumption influences the choice of architecture, data coding, and arithmetic. However, in many cases, minimisation of power consumption in the context of commercial design involves modification to a functionally verified architecture, followed by power estimation. This is a comparative measurement of power and the motivation for this section is the speeding up of such measurements. It will be shown that the opening up of the simulation loop under certain circumstances can provide a reduced time for power estimation, when the designer is making a comparative measurement of power. The basis for this modification to the standard Monte Carlo technique is based on two observations made during this work which are now presented.

4.5 Two Observations on Power Estimation

The average power dissipation is based on the mean of individual power values, each one of which is usually estimated after tens or even hundreds of clock cycles. These individual

power values are distributed with a Gaussian probability distribution function [Vini98] and this fact provides the basis for all standard implementations of Monte Carlo estimators. However, investigations carried out during this work have indicated that the energy consumption associated with a single input vector transition also appears to be normally distributed. This observation means that after a single clock cycle, there is useful information available about the behaviour of the circuit. An example of this behaviour is shown in Figure 4.14, which shows the energy distribution of an 8-bit adder for 1000 input transitions.

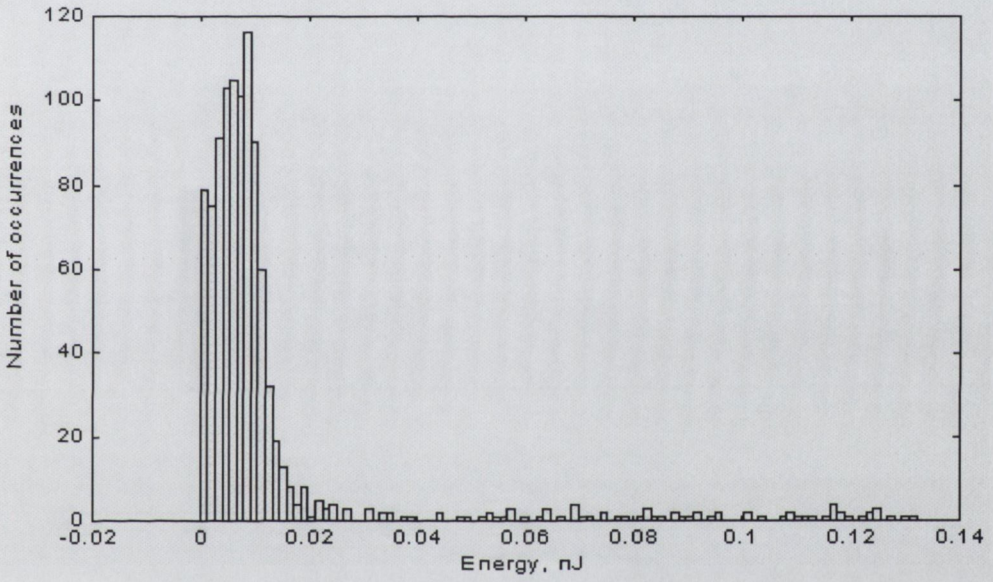


Figure 4.14: Energy Distribution of an 8-Bit Adder

This distribution contains a low-amplitude tail and this form of the distribution has been referred to in [Najm94] as a long-tailed normal distribution. Similarly-shaped distributions of energy estimates may be observed for a variety of circuits of differing sizes. Some larger circuits appear to possess a bimodal energy distribution, but such circuits, while perfectly amenable to statistical analysis, are only rarely encountered [Marc99].

The second observation concerns the nature of the output from an MC simulation. Prior to simulation, the user fixes the level of confidence, typically 95% to 99%. The user also sets a maximum allowable error ε, which is often in the 5% to 20% range. The output is then a figure for power consumption, $P_{diss} \pm \varepsilon$. However, the number P_{diss} has no significance, other than as the algebraic centre of a band of values, within which the true power lies. This band, or confidence interval (CI), is the significant output from the simulation. During the lifetime of a single iteration, the CI may be viewed as exhibiting both a position function and a

magnitude function. That is, as the simulation time progresses, the position of the CI in a two-dimensional energy-time map varies in a definite manner. It displays a trajectory that takes it from time $t = 0$ to the end of the simulation run, while also moving in the vertical energy axis. In addition, the instantaneous magnitude of the CI alters over the length of the run, fluctuating as the simulation proceeds. This behaviour is demonstrated in Figures 4.15 and 4.16, which show the convergence of the confidence interval during 1000 input samples for a 4-bit and a 16-bit adder, respectively. The error shown is the CI magnitude, that is, the difference between the upper and lower CI.

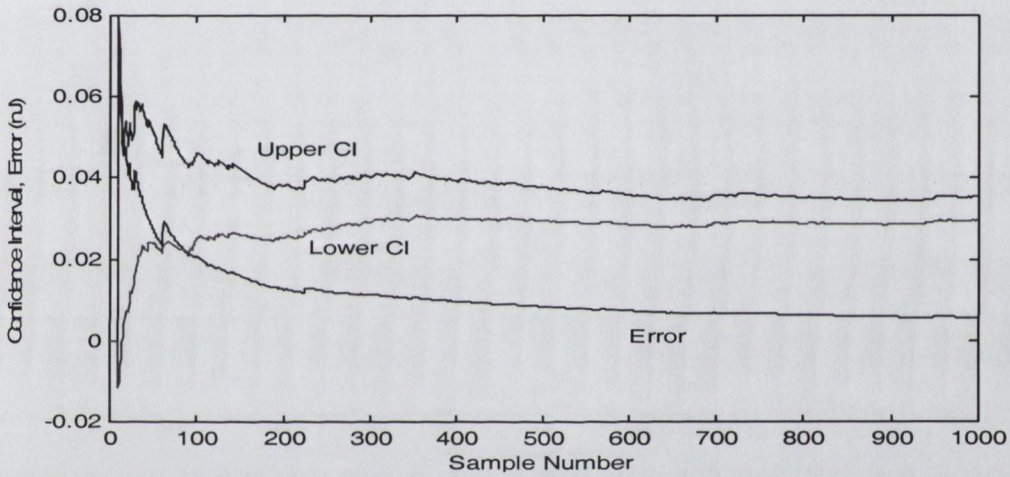


Figure 4.15: Trajectory of Confidence Interval and Error for a 4-Bit Adder

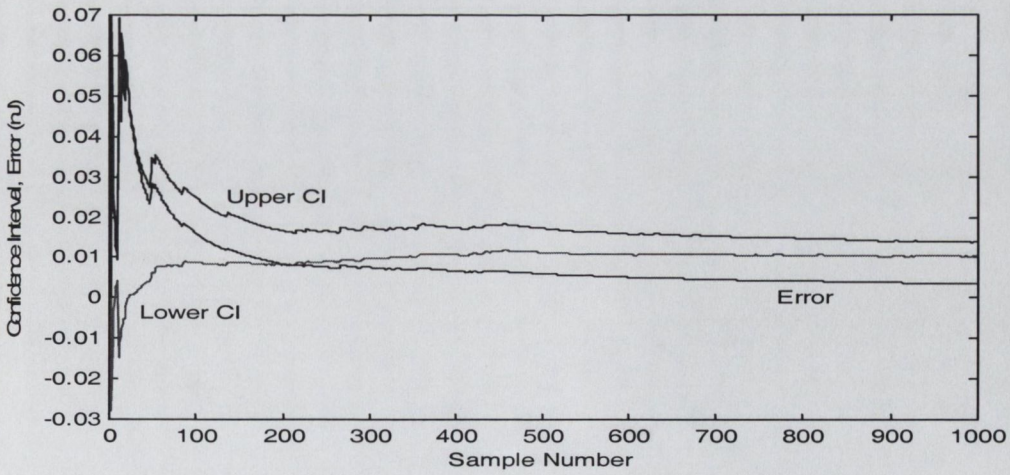


Figure 4.16: Trajectory of CI and Error for a 16-Bit Adder

When, after a circuit modification, a comparative power analysis is performed, the designer may wish only to determine how the power consumption has changed relative to the previous version of the circuit. When a Monte Carlo technique has been used to determine

power, the confidence interval of the previous simulation is available. This information may be used in conjunction with a graphical output of type shown in Figure 4.16 to terminate the simulation. This decision is based on the locus of the CI and also on its magnitude, that is, the error. For example, a circuit simulation returns an estimate of 0.05nJ using the energy technique outlined above. The relative error is 10%. This corresponds to a CI of 0.04nJ to 0.06nJ. If a modified circuit exhibits the performance indicated in Figure 4.16 above for example, then the designer, using the graphical output of both CI and error, should, after a few hundred vectors, realise that the modified circuit represents a more energy efficient circuit. This decision could be made after as few as 200-300 vectors, which would take considerably less time than a full MC simulation. Power or energy consumption to a specified error is only available after a full MC simulation. The approach outlined here should be viewed as an approximation tool, suitable for comparative power and energy estimates, when used in conjunction with a full MC simulator such as LLAMA.

4.6 Summary and Conclusions

In this Chapter the usefulness of UWN-based estimates of average power dissipation for CMOS circuits has been demonstrated. Based on optimised sources of such circuit inputs, a power estimation tool, LLAMA, was developed, incorporating a SPICE simulation engine. The operation and performance of this tool was examined, with typical results presented for a variety of circuits. The circuits employed included automatically generated layouts of adders of various lengths. These were created using a 0.8μm technology by a synthesis tool developed for the purpose of testing the power estimation tool LLAMA. Simulation studies showed that for such a tool, 100 input vectors per iteration was a good compromise between iteration time and number of iterations. The reduced error when using the UWN source developed earlier was demonstrated, in contrast to the variation in estimates obtained when poorer quality data was used. The full functionality of LLAMA is utilised in the next Chapter where it is employed to acquire all power estimation results for a wide variety of circuits. Both the closed loop Monte Carlo operation and the open loop mode of the tool are used to get respectively UWN and correlated estimates of power dissipation. An alternative and novel approach to power estimation was suggested by providing an approximate but speedy estimate of the power dissipation. This new technique is based on the dynamic nature of the confidence interval generated during a simulation. It was noted also that the distribution of

energy estimates follows a Gaussian distribution. This observation allows the designer to terminate a simulation at an early stage, when the error function and confidence intervals reach a steady state. This technique is easily automated, to give the circuit designer an alternative and parallel view of how the power estimation process is progressing. While UWN estimates are invaluable in allowing comparative, unbiased estimates of power dissipation for alternative architectures, they represent an upper bound on the power dissipation observed in normal circuit operation. To model the effect of real data on the circuit power performance, the characteristics of the inputs should represent those of a typical application. The next chapter addresses this issue, using a variety of techniques to represent the temporal and spatial relationships that may be present in a data stream. A Markov model is then used to capture the output characteristics of sequential circuits, giving a complete representation of the outputs of both datapath and controller subsystems.

5 Modelling Real Data with Markov Chains and Transition Probability

Power estimation techniques discussed so far have all relied on uniform white noise input data, which is of importance when evaluating circuits in the early stages of the design cycle, due to the ease of data generation and the universal nature of the data, which facilitates comparative power measurements. As the design progresses, it becomes necessary to determine a power rating for each circuit or subsystem. It is also usual to provide an overall power rating for the system [Namg00], [Simo00], [Baas99], [Murp97]. In this context, the term power rating is taken to represent the typical power consumption of the system or module, given typical input data. Where such data is available, it provides the best method for determining the power rating. However, there are difficulties associated with the production of sufficient quantities of such data. In particular, the availability of system-level input data requires the continuous production of system input, which for example may be audio, video or telecommunications data. Alternatively, if a simulation model of the host system for the ASIC is available, this may be used to generate typical data. In either case, the process is either technically demanding or computationally intensive. There is a further complication when internal subsystems require suitable input. Again the production of such data implies the continuous running of a system simulation to supply such data. This Chapter explores more efficient methods of obtaining data that, when supplied to an MC simulation, will provide a power rating for the module under test. Two approaches will be considered, one relevant to datapath and system-level components and the other of use in the simulation of controller blocks based on the finite state machine. The datapath model will offer three alternatives, depending on assumptions about the nature of the input data. There are two advantages when employing such synthetic data. Given a short sequence of input data to be modelled, once the parameters of that sequence have been determined unlimited quantities of the synthetic data may be produced for the power estimation purposes. Conversely, given a long input sequence then a more compact sequence may be generated which reduces simulation time. Even when no input data is available, a complete exploration of the power

behaviour of the system can be accomplished by varying the nature of the synthetic sequence applied to the system [Wu01], [Gupt00a], [Marc99].

5.1 Input Data Modelling For Combinational Modules

The aim of this part of the work is to identify a suitable technique for modelling complex data and to develop an efficient technique for generating such data. In this regard, two measures of signal activity are particularly useful, the signal probability P_s which is the probability of a signal being HIGH and the signal transition probability P_t [Najm95] which is the probability of a signal transition, as defined in Chapter 2. The maximum transition probability of a data bit in the system is unity while a random bit has a value of $P_t = 0.5$. The approach taken to synthesise typical datapath input involves three steps, represented diagrammatically in Figure 5.1. Firstly, a sample of real input data is obtained. This step is carried out only once. The sample is then analysed and the values for P_s and P_t are obtained for each of n input bits. This is achieved by analysis of a large data sample, or as is the case in this work, by the application of an MC routine to the data. This results in estimates with predetermined accuracy and confidence, as was the case for power and energy estimation in the previous chapter. These n 2-tuples become a template for the third step, which is the efficient generation of the synthetic data. The overall result is a method of quickly and accurately providing as much data as is required for an unlimited number of simulations. This ensures a suitable input from a statistical viewpoint, compared to reusing the same sample for every simulation. The three steps have been incorporated into a synthesis tool, the structure of which is now described.

The structure of the data synthesis tool may be seen by reference to Figure 5.1. The first step requires the extraction of the signal and transition probabilities. Extraction of signal and transition probabilities is a direct application of MC techniques. The MC algorithm used to obtain these measures has been described in Chapter 4 where it was applied to power estimation. In this case, the quantity estimated is the required probability and the technique is identical. The input to the MC program is the original data sample and the output is the n-element set $\{P_s, P_t\}$, for an n-bit data sample.

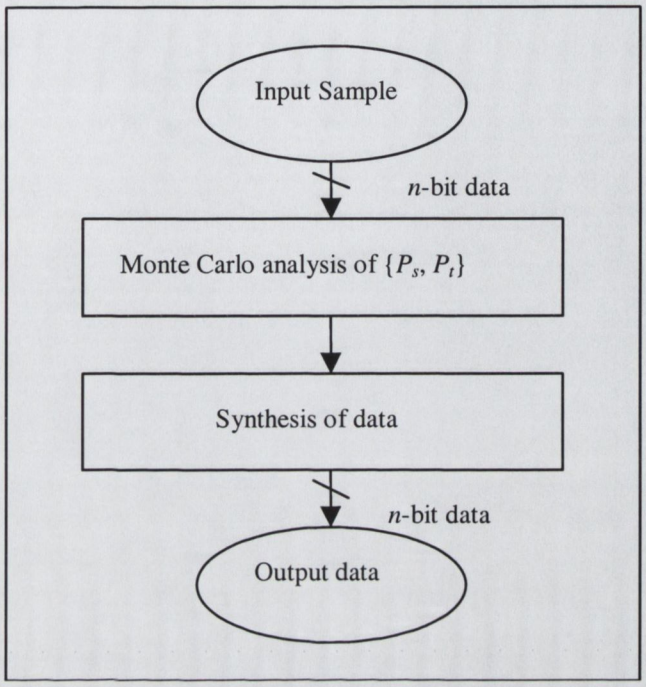


Figure 5.1: Analysis and Synthesis of Simulation Input Data

As an example of the operation of the routine, the output for an 8-bit sound file in the *.wav* format is given in Figure 5.2. This file contains a monophonic elephant trumpet of 5s duration, sampled at 22.050kHz with 8-bit resolution.

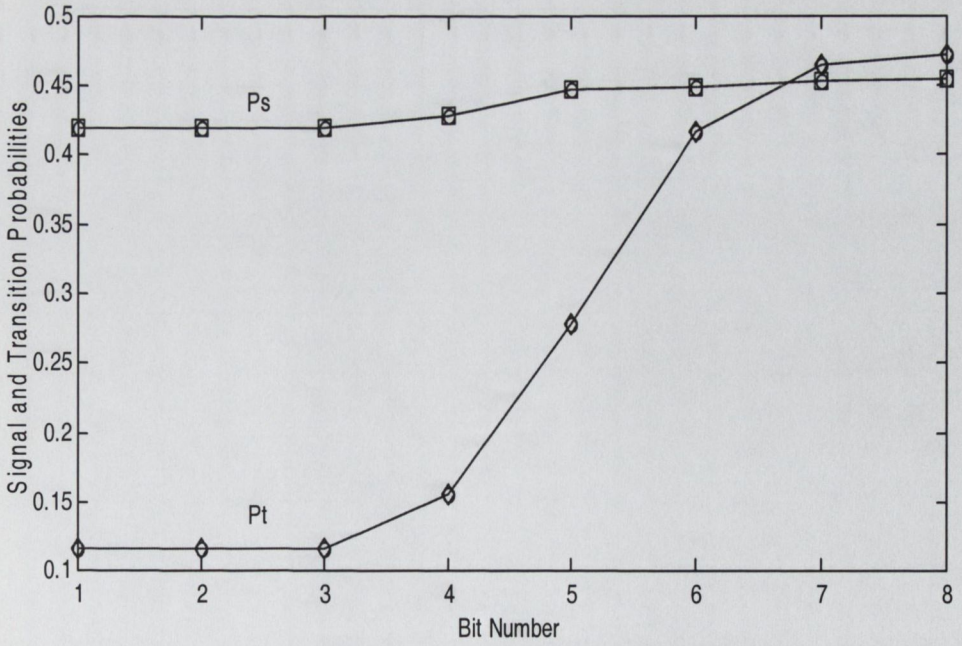


Figure 5.2: Signal and Transition Probabilities of an Audio Sample

The values shown here were obtained with 5% error and a 95% confidence level. On each pass of the MC probability evaluator, the probabilities are evaluated according to (75) and (76), where $x(t)$ is the signal and $n_x(T)$ is the number of transitions in time T [ibid.].

$$P_s(x) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{+T} x(t)dt \tag{75}$$

$$P_t(x) = \lim_{T\to\infty} \frac{n_x}{T} \tag{76}$$

The next step requires the synthesis of an n-bit signal with the same properties as the original signal. The technique used to create the signal is based on the relationship between the signal and transition probabilities [Xake94]. For signal probability P_s and transition probability P_t, the likelihood of a low to high transition P_{01}, and of a high to low transition, P_{10} is given by (77) and (78), respectively.

$$P_{01} = \frac{P_t}{2(1-P_s)} \tag{77}$$

$$P_{10} = \frac{P_t}{2P_s} \tag{78}$$

These relationships have been implemented in the following algorithm, the output from which is a matrix of binary signals with the appropriate probabilistic behaviour.

```
Initialise signal with a random state
While more_data_required
   if previous_state = 0
      if random_number <= P01
         next_state = 1;
       else next_state = 0;
      end
   end
   if previous_state = 1
      if random_number <= P10
          next_state = 0;
      else next_state = 1;
      end
   end
end
```

This algorithm is applied to each bit position, based on the analysed probabilities. The effectiveness of this approach may be judged from Figure 5.3, which shows the signal and transition probabilities of a synthesised file, together with the original data. There is close correspondence between the two sequences. This technique can be easily incorporated into an optimisation loop to provide used-defined error levels.

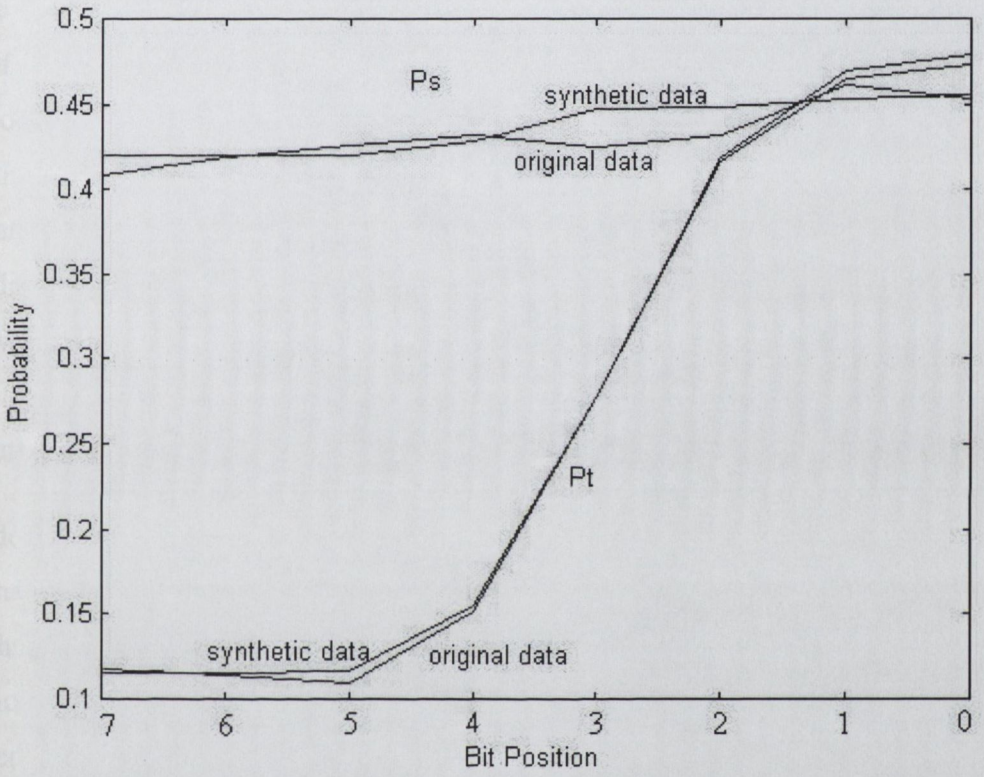


Figure 5.3: Signal and Transition Probabilities of Original and Synthetic Data

Table 5.1 contains the percentage error in both signal and transition probabilities at each of the 8 bit positions of the synthesised sequence.

% Error	7	6	5	4	3	2	1	0
Ps	2.6040	0.1274	1.3960	1.0459	4.9530	3.7503	1.9279	0.5704
Pt	1.5079	-0.9172	-6.3890	2.5795	0.3703	0.5073	1.0926	1.4119

Table 5.1: Error in Synthesised Input Data

The application of this data to power estimation will be considered in Section 5.3, in conjunction with an analysis of the data techniques proposed for sequential circuits, as described in the following section.

5.2 Modelling Temporal and Spatial Correlation

The presence of a relationship between input vectors, separated by one or more clock cycles is of significance when considering power dissipation [Land94]. This relationship is referred to as temporal correlation [Xake94]. It may be measured by the autocovariance function of the input data [Haye96]. Spatial correlation refers to a relationship between signals during the same cycle, for example between two inputs to an adder or between two bits in a word. It is also of consequence when considering system power dissipation. It is measured with the crosscovariance function [Mitr98]. The inclusion of correlation modelling into any synthesis algorithm is a useful addition to the functionality, given the relation between power dissipation and such correlation [Marc94]. The technique used in this work to incorporate such data is based on a technique drawn from the domain of signal processing [Chil97]. However, it has proved to be a quick and efficient method for the production of input data for power estimation. The method is now outlined, starting with the production of spatially correlated data.

A random sequence $X(n)$ of uncorrelated variables may be viewed as a vector, X where each element of the vector is an element of the sequence. It is possible to generate a vector Y from X, with a specified covariance matrix Cov_{YY}, where this matrix is real-valued and symmetric with non-negative eigenvalues. The entries c_{ij} in this matrix indicate the spatial correlation between inputs i and j. When, for example, the random sequence $X(n)$ has two members, i.e. two simultaneously generated, uncorrelated random variables, a $2x2$ symmetric matrix may be derived such that the spatial correlation coefficient is given by the elements $c_{12} = c_{21}$. In addition, many such correlated signals are Gaussian, in which case the input vector is often a zero mean, unit variance random variable [Vini98]. An efficient method for generating the spatially correlated data is suggested in [Chil97]. Given N input variables, Childers' technique is based upon an NxN transformation matrix $[T]$, such that

$$Y = [T] \cdot X \tag{79}$$

The k^{th} element of Y is given by (80).

$$Y_k = \sum_{j=1}^{N} t_{kj} X_j \tag{80}$$

where t_{kj} is the element in the k^{th} row and j^{th} column of $[T]$. When the elements of X are zero mean, the expected value of Y is also zero. The covariance of the elements of Y is therefore given by (81).

$$E[Y_k Y_j] = E\left[\sum_{i=1}^{N} t_{ki} X_i \sum_{m=1}^{N} t_{jm} X_m\right] = \sum_{i=1}^{N}\sum_{m=1}^{N} t_{ki} t_{jm} E[X_i X_m] \tag{81}$$

The elements X_i are zero mean and uncorrelated, therefore $E[X_i, X_m] = 1$ if $i = m$ and it is zero otherwise. Therefore, the covariance may be written as indicated in (82).

$$E[Y_k Y_j] == \sum_{i=1}^{N} t_{ki} t_{ji} \tag{82}$$

This equation may be interpreted as stating that the kj element of the covariance matrix of Y is the dot product of the k^{th} row of $[T]$ with the j^{th} column of $[T]^T$, the transpose of $[T]$.

Thus the covariance of Y is given by (83).

$$Cov_{YY} = [T][T]^T \tag{83}$$

The matrix $[T]$ may be determined using the decomposition expressed in (84).

$$Cov_{YY} = [P][D][P]^T \tag{84}$$

where $[D]$ is a diagonal matrix containing the N eigenvalues of Cov_{YY} and $[P]$ is an NxN matrix with columns consisting of an orthonormal set of eigenvectors of the covariance. Therefore, $[T]$ may be calculated using the expression in (85).

$$Cov_{YY} = [P][D][P]^T \tag{85}$$

In general, Cov_{YY} is specified (hence $[P]$ and $[D]$) and $[T]$ is calculated using $[P]$ and $[D]$, which are obtained from the covariance. For example, using Matlab, the matrices P and D may be obtained with the code $[P,D] = eig(Cov_{yy})$. The spatially correlated data Y is then obtained from the random input vector X using $[T]$. A simple example of the application of this method is now given, where spatially correlated data with a given covariance matrix is generated. The procedure is demonstrated as follows. Using 100 samples of a two-element

random vector X, the following procedure is applied to synthesise 100 samples of a spatially correlated 2-element vector Y. The spatial correlation between the two elements of Y is defined by the covariance matrix Cov_{YY} = [1 0.4; 0.4 1].

Using the Matlab code [P,D] = eig([1 0.4; 0.4 1]) produces two matrices, P and D.

$$P = \begin{bmatrix} 0.7071 & 0.7071 \\ -0.7071 & 0.7071 \end{bmatrix}, \quad D = \begin{bmatrix} 0.6 & 0 \\ 0 & 1.4 \end{bmatrix}$$

The transformation matrix $[T] = [P][D]^{1/2}$ is therefore:

$$[T] = \begin{bmatrix} 0.5477 & 0.8367 \\ -0.5477 & 0.8367 \end{bmatrix}$$

To obtain the output Y, the matrix multiplication $[Y] = [T][X]$ is performed for each pair of input numbers $\{X_1, X_2\}$. Table 5.2 shows a sample of the uncorrelated input data sample from X with the corresponding spatially correlated output data Y.

Sample	1	2	3	4	5
X (X_1)	-0.0715	0.2792	1.3733	0.1798	-0.5420
(X_2)	1.4440	0.6123	-1.3235	-0.6616	-0.1461
Y (Y_1)	-1.2473	0.6652	-0.3551	-0.4550	-0.4191
(Y_2)	-1.1690	0.3594	-1.8595	-0.6520	0.1746

Table 5.2: Uncorrelated Data, X and Corrleated Data, Y

The effect of the transformation may be seen in the plot of Figure 5.4, which shows the ratio of output cross-covariance to input cross-covariance.

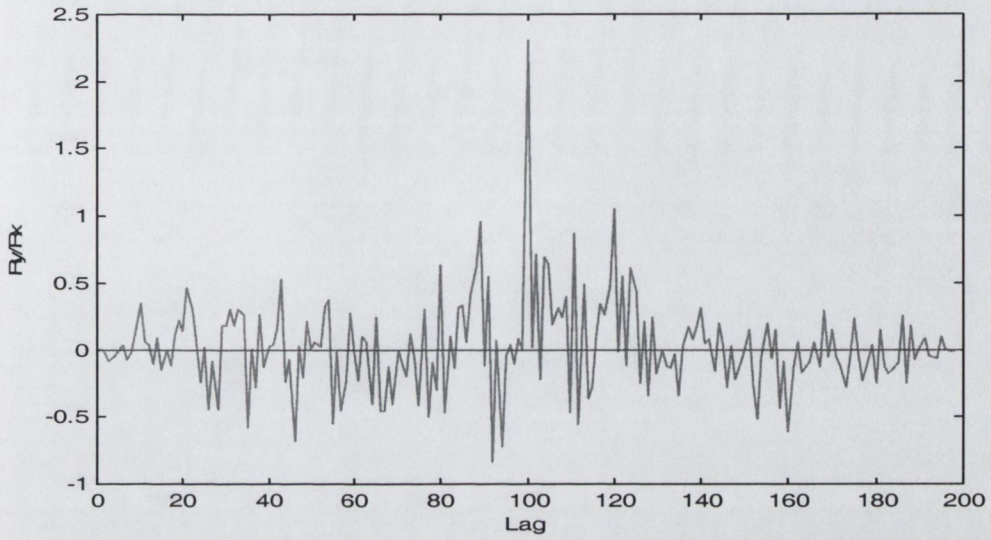


Figure 5.4: Presence of Spatial Correlation in Synthesised Sample

The output is approximately 2.5 times more spatially correlated than the input at correlation sample 100, which in this example corresponds to concurrent samples Y_{1i} and Y_{2i}, for all times t_i. In general, data sequences with the required value of covariance matrix are quickly generated with a minimum of computation. The technique allows samples of arbitrary length to be created with a user-defined number of bits per sample.

The production of temporally correlated data is based on the well-known behaviour of an autoregressive-moving average (ARMA) filter. When such a structure is driven by random data, temporal correlation is introduced into the output data [Haye96]. The latency of the correlated data relative to the input, and the amount of temporal correlation, are functions of the filter order and parameters, respectively. Consider for example, a simple first order ARMA filter with MA coefficients $b_i = 0$, $i \neq 1$ and AR coefficients $a_i = [1\ 0.3]$. This simple structure is sufficient to temporally correlate the data as shown in the output autocorrelation of Figure 5.5, where it may be observed that the temporal correlation at lags -1 and +1 are approximately 0.3, and are determined by the filter coefficient. These results are well known [ibid.]. With this technique temporal correlation may be easily introduced into a data sequence with minimum computation.

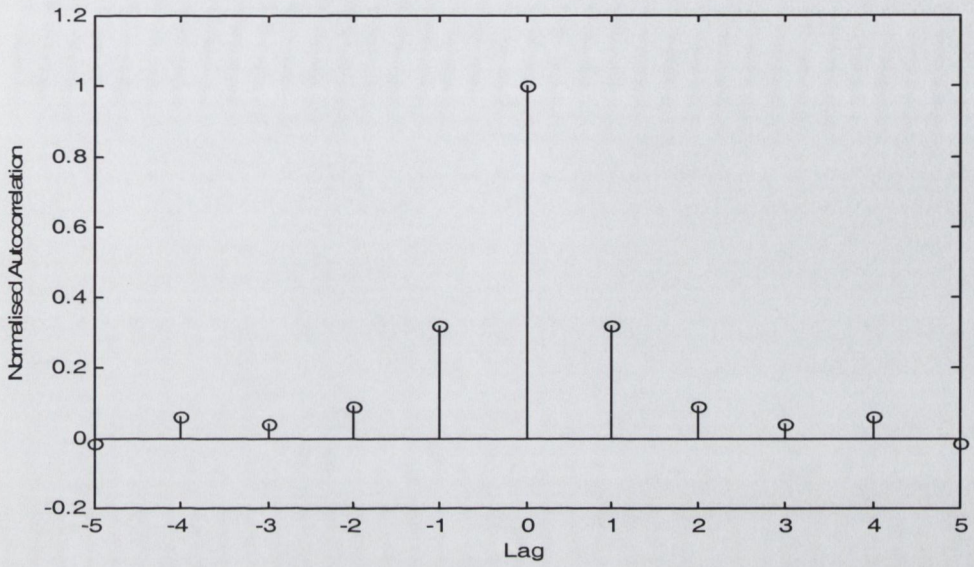


Figure 5.5: Temporal Correlation in Synthesised Data

5.3 Sequential Circuit Data Modelling Using Markov Chains

When the input to a module is the output of a sequential circuit such as a finite state machine, one approach to modelling that data involves the use of Markov Chain theory [Chil97]. The use of a behavioural model of the output data removes the need for a functional model of the circuit that generates the data. This reduces the development time, as the functional model will be a completely specified design described in VHDL or SPICE, while the behavioural model requires a simpler description of the output signals in a language such as C or Matlab. In addition, the behavioural model of the output signals is independent of the internal organisation of the functional block that generates the data. In the context of this work, the modelling of finite state machines is of particular relevance and therefore these circuits will be looked at in detail. Firstly, the application of Markov Chain theory to the modelling of the outputs of sequential circuits is reviewed.

A Markov process is a random process where the occurrence of an event, such as the transition from one logic state to another, is dependent only on the immediately preceding event [Vini98]. The process consists of a set of states S and a corresponding set of probabilities $\{P_{ij}\}$which describe the probability of a transition from state i to state j. A Markov Chain is a process with an initial state, a finite set of possible states and an associated set of transition probabilities, usually given in the form of a square matrix, T. Chains may

reach equilibrium, or steady state, where the probability of reaching a given state becomes a constant and is independent of the step and initial condition. To reach an equilibrium state, a chain must be a regular chain, that is, for some non-zero, non-negative n, T^n contains only positive elements. A fixed point or fixed probability vector F of a regular chain with transition matrix T is a probability vector with all positive elements such that the relationship $F^T = F^T T^n$ holds. If F exists, it may be interpreted as the steady state probabilities for each state. Finally, F may be experimentally obtained as the limiting value of the matrix T^n, as n $\rightarrow \infty$. The relation $F^T(T\text{-I}) = 0^T$ also holds, and may be used in conjunction with the fact that the elements of F sum to unity, to determine F [Chil97].

Some work has been done on modelling the input to sequential circuits using Markov techniques [Qiu01], [Marc99] with a view to obtaining power estimates in the sequential circuit. The focus in this work is on the modelling of the output of such circuits to obtain power estimates of circuits driven by the sequential module. In this regard, the modelling of sequential circuits, and finite state machines in particular, is effectively accomplished with a suitable Markov model. As an example, the state diagram of a simple circuit is given in Figure 5.6. It has a single external input x, and three states, $\{S_0, S_1, S_2\}$. The input signal probability is $P(x = 1) = P(x = 0) = 0.5$. Any number of inputs with arbitrary probabilities are modelled with similar ease.

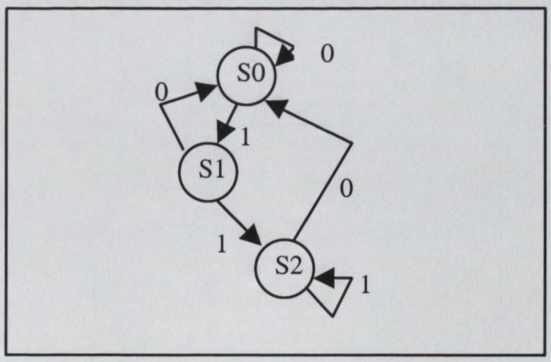


Figure 5.6: State Transition Diagram Example

The transition matrix T may be obtained by inspection as indicated in (86), where for example, the probability of going from state 0 to state 1 is 0.5, which is the probability that the external input x is HIGH. The actual transition that occurs during a simulation run is determined by the value taken by x.

$$T = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0.5 & 0 & 0.5 \end{bmatrix} \tag{86}$$

In general, once the signal probabilities of all external inputs to the state machine are known, the matrix T is uniquely determined by the state diagram.

A program was written that generates a stream of state transitions, which simulates the actual finite machine outputs. The input to the program is the matrix T and the required number of transitions. This is a straightforward operation, requiring less than ten lines of high-level code for this particular example. The description is independent of the implementation of the circuit as either a Mealy or a Moore machine [Katz94], giving the flexibility to implement the structure at a later date. The program operation is verified by determining the steady-state probability vector from T which is then checked against the experimentally calculated value from the simulation output. An example of this procedure is detailed as follows.

The transition matrix T of (125) is now examined. Firstly, T is regular, because T^2 has no zero elements. Also, some matrix manipulation indicates that T^2 and higher powers are identical, giving the steady-state probabilities for each state of the circuit, as given in (87).

$$T^2 = T^4 = T^6 = \ldots = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.5 & 0.25 & 0.25 \\ 0.5 & 0.25 & 0.25 \end{bmatrix} \tag{87}$$

This result indicates that in a simulation run, 50% of the state transitions should be to state 0, 25% to state 1 and 25% to state 2. These probabilities also provide a useful test of the data generation program. The output from the program is a file of binary test vectors, representing the output states simulated. A sample program run of 1000 states was analysed for the relative frequency of each state, and the results are presented in Table 5.3.

State	0	1	2
Relative Frequency	0.4995	0.2520	0.2485

Table 5.3: Relative Frequency of Markov Model Outputs

This table verifies that the correct behaviour was observed and that the technique is therefore useful as a means of generating a compact sequence for power estimation purposes. The application of this technique to system level power estimation enables the output of the controller section of a design to be quickly and easily modelled without designing the structure of the controller. This allows datapath inputs from the controller to be driven with realistic data during a power estimation simulation. The model is a high level behavioural one and any modification to the design, the instruction set or the controller, can be efficiently incorporated with minimum effort.

5.4 Power Estimation in Combinational Modules

To demonstrate the application of the synthesised data sequences to power estimation and to observe the effect of varying the model parameters on the estimate obtained, the following analysis is presented. For each of the models examined, a sample of synthesised data is used to obtain a power estimate for a 12-bit adder, the layout for which was implemented for testing the MC tool in Chapter 4. In addition, the UWN power estimate is included for reference. The variation in power dissipation caused by varying the model parameters in each case is also included. In this way the sensitivity of the power estimates to the parameters chosen for these models is clearly observed.

Figure 5.7 contains power estimates for the 12-bit adder for a number of input sequences, including UWN. For each sequence, the transition probability was varied as indicated on the figure. The signal probability for each sample was fixed at 0.5, with a sample size of 1000. Power dissipation was estimated by averaging the supply current at the end of the run. The power dissipation rises with increasing amount of signal activity up to the UWN case of P_t=0.5. Higher values of transition probability represent a signal that changes almost every cycle, becoming more like a clock signal. This reduced randomness in the signal results in decreasing power dissipation, as the transition probability is increased beyond UWN.

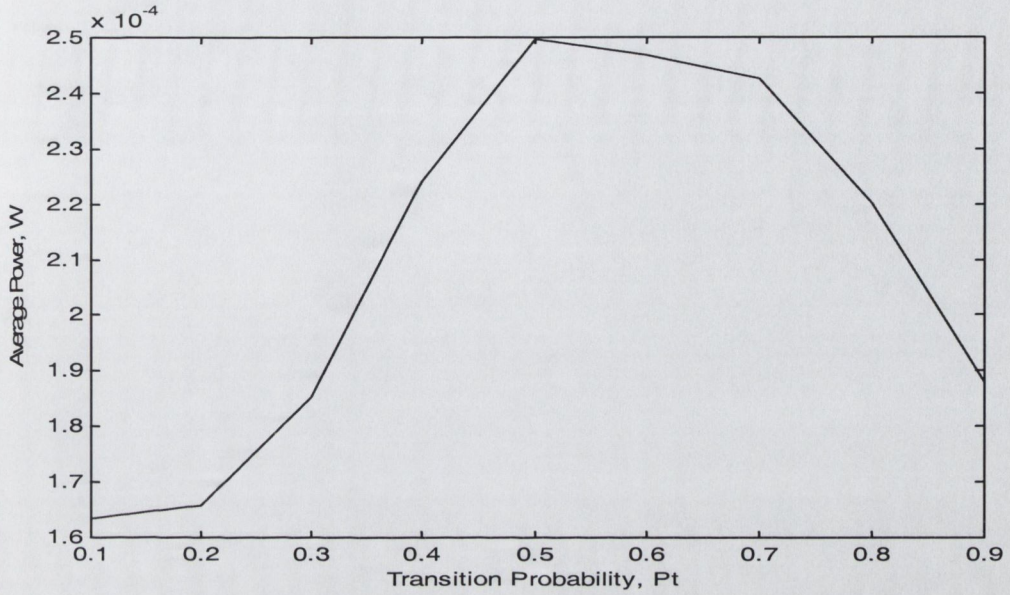


Figure 5.7: Power Dissipation as a Function of Transition Probability

Application of data with spatial correlation in the range 0.1 to 0.9, as measured by the covariance, results in power estimates as given by Figure 5.8. The power dissipation reduces with increasing correlation, as expected. This behaviour depends on the position of the correlated data within the input. When the correlated inputs are divided between signal inputs, the relationship is less pronounced. For the data in Figure 5.8, two 6-bit correlated inputs were applied to a 2-input, 12-bit adder. Any other mapping between input data and circuit inputs, by changing the order of the input bits for example, reduces the spatial correlation and the power observed is more likely to approach the UWN value. Thus the effect of spatial correlation on power dissipation is less pronounced and depends on the ordering of the input data to a large extent.

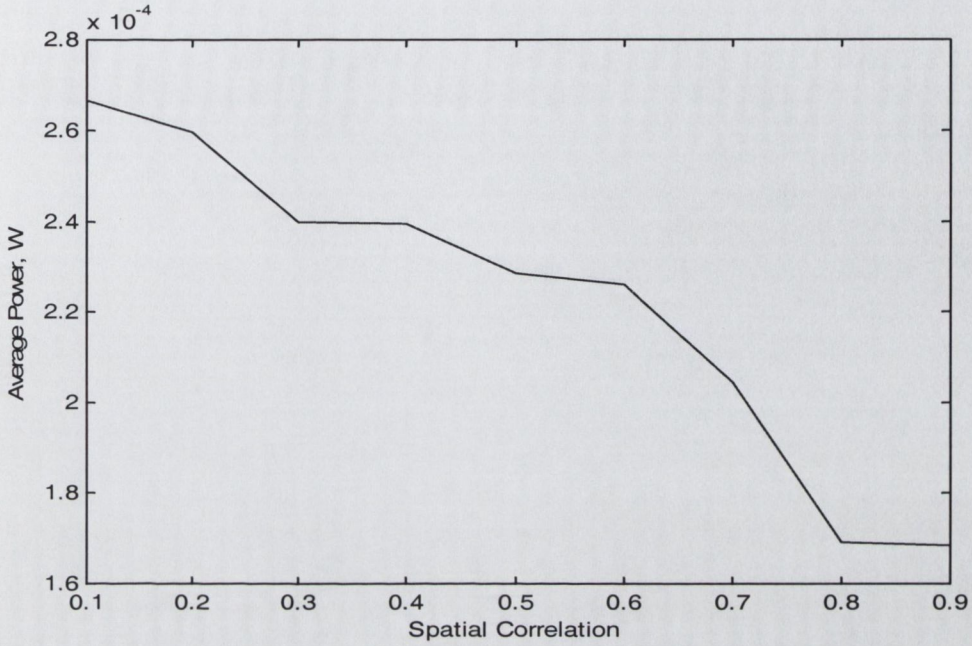


Figure 5.8: Effect of Spatial Correlation on Power Estimates

The variation in power dissipation caused by temporal correlation is reported in Figure 5.9.

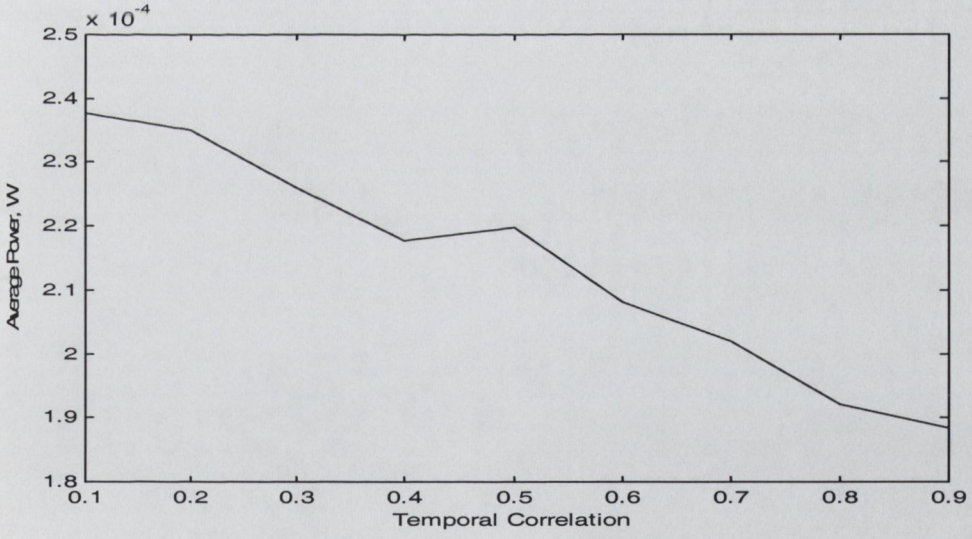


Figure 5.9: Variation in Power Dissipation with Temporal Correlation

The effect of temporal correlation on power estimates appears to vary a certain amount with the circuit under test, but the circuit above, a 12-bit adder, demonstrates the variation in power with input data temporal correlation. For this circuit, the UWN average power was estimated to be 0.2320 mW.

5.5 Power Estimation in Sequential Modules

For the Markov model, a different approach was adopted. A VHDL state machine module was written and a sample of the output data recorded. This sequence was applied to the adder and a power estimate obtained which serves as the reference power estimate. The output from the Markov model was then applied to the same circuit and the power estimates compared. The Markov model is evaluated with the sequential circuit of Figure 5.10, which is the controller module of a multicycle processor used as a test architecture in Chapter 6. This sequential module has ten states, the function of which is annotated on the state diagram. The external input is the signal *OP*, which can take five values, depending on the instruction type. To provide a realistic context for this evaluation, a mix of instruction types according to Table 5.4 is chosen for demonstration purposes. This data represents the ratio of instruction types observed during the execution of the compiler *gcc* [Patt98]. The R-type (register-type) instructions are those that access the ALU.

Instruction Type	Relative Frequency (%)
Load	22
Store	11
Branch	16
Jump	2
R-type	49

Table 5.4: Proportion of Instruction Types for Markov Model

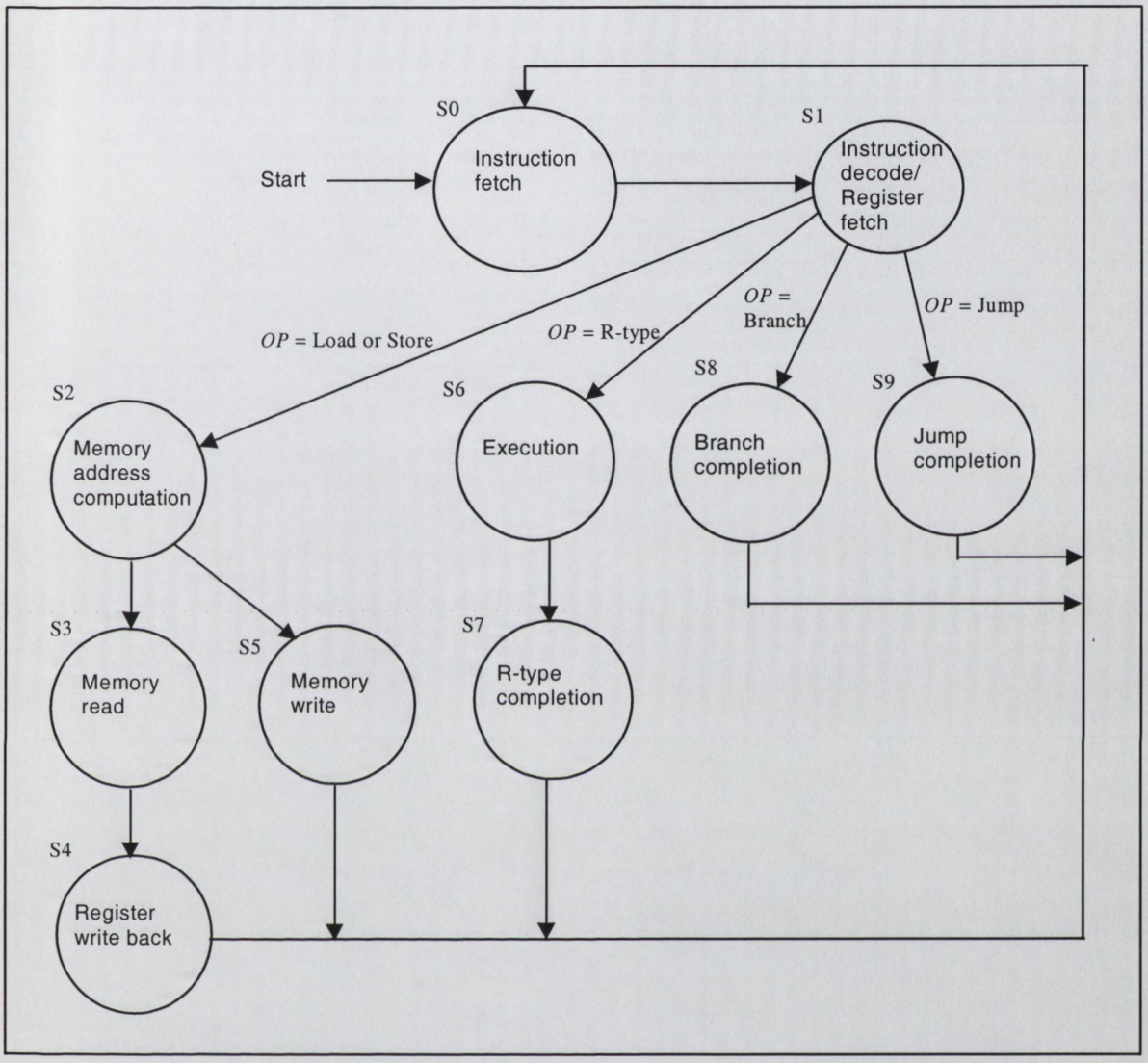


Figure 5.10: State Transition Diagram for Markov Model Evaluation

Two models of this circuit were prepared, a VHDL entity-architecture pair and a Markov model of the state transition diagram. In both cases the models incorporate routines to generate two output files, a list of the sequence of states entered during the simulation and a file of control outputs generated by each state. The relative frequency of each state was also recorded. The output files were converted into a binary format capable of driving a file-based SPICE simulation. Circuit simulations were then performed using the outputs from both models, VHDL and Markov and the results compared. The methodology for evaluation of this Markov model by comparison with a VHDL equivalent, is represented diagrammatically in Figure 5.11.

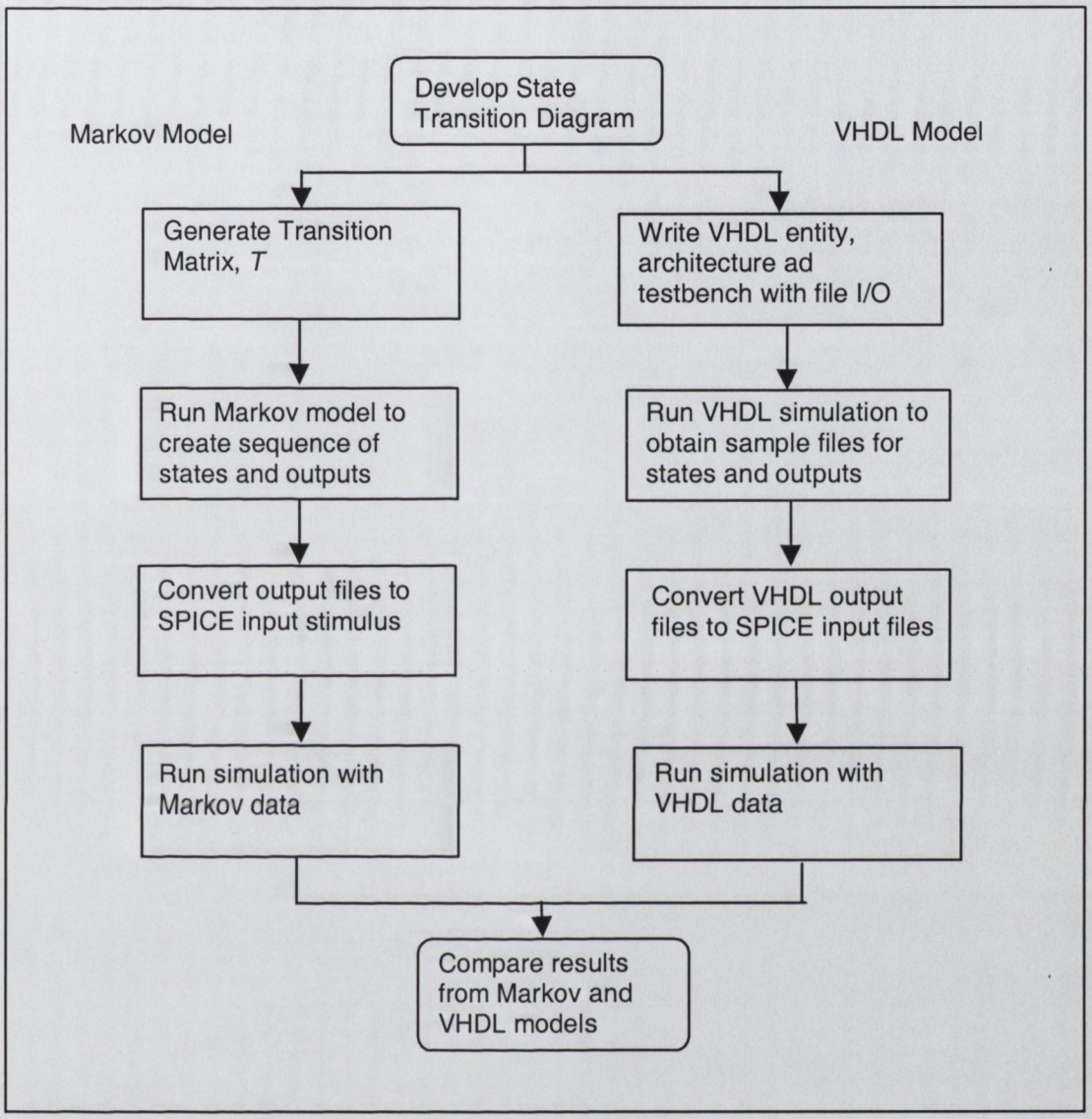


Figure 5.11: Comparison of VHDL and Markov Models

To generate the Markov model a circuit transition matrix T was derived from the state diagram by inspection, as indicated in (88). The data of Table 5.4 provided the necessary transition probabilities between states.

$$T = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.33 & 0 & 0 & 0 & 0.49 & 0 & 0.16 & 0.02 \\ 0 & 0 & 0 & 0.67 & 0 & 0.33 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (88)$$

In this matrix, element t_{ij} represents the probability of a transition from state i to state j, for each of the ten states. According to the state diagram, for example, only two states, 3 and 5, may be reached from state 2, depending on the value of the *OP* input. Table 5.4 informs that there are twice as many load instructions as stores. Therefore, the probability of a transition from state 2 to state 3, which occurs when *OP* = *load*, is 100%x22/(22+11), i.e. 66.66%. In this manner, the elements of T, which is a sparse matrix, may be quickly and easily calculated. A synthetic sequence of states and controller outputs is then obtained from the Markov model using a program based on the technique described in Section 5.2. This data becomes the input data for the comparative power estimation.

The VHDL model was written using a standard approach for the specification and implementation of finite state machines [Sjoh97]. The code contains a number of interconnected processes to update the current state, determine the next state and outputs and to manage file input output. The input to the controller is a sequence of values for the opcode *OP*. These are generated randomly according to their relative frequency, as given by Table 5.4.

Table 5.5 provides a comparison of the power estimates obtained using input data from this model and the estimate resulting from input data from a VHDL model of the circuit. The UWN estimate is also included for comparison. The experiment was run three times, using a 16-bit adder circuit as the load, with a 100ns clock period and a sample size of 1000 vectors. The simulations were repeated for sample lengths of 500, 200 and 100. These results are also included in the table.

Sample Length	Sim. No.	Markov Power (mW)	VHDL Power (mW)	UWN Power (mW)	Markov Error (%)	Markov Ratio	VHDL Ratio
1000	1	0.0382	0.0373	0.361	2.41	0.10	0.10
	2	0.0374	0.0378	0.329	1.05	0.11	0.11
	3	0.0379	0.0371	0.348	2.15	0.11	0.11
500	1	0.0421	0.0408	0.352	3.18	0.12	0.12
	2	0.0413	0.0418	0.346	1.19	0.12	0.12
	3	0.0425	0.0404	0.327	5.19	0.13	0.12
200	1	0.0527	0.0523	0.329	0.76	0.16	0.16
	2	0.0538	0.0527	0.302	2.09	0.18	0.17
	3	0.0542	0.0524	0.359	3.43	0.15	0.15
100	1	0.0736	0.0710	0.331	3.66	0.22	0.21
	2	0.0741	0.0731	0.334	1.37	0.22	0.22
	3	0.0734	0.0724	0.362	1.38	0.20	0.20

Table 5.5: Power Estimates Obtained with Markov and VHDL Data Models

There is a close correspondence between the power estimates obtained using the Markov model and those obtained with the VHDL circuit-based estimate. The column labelled Markov Error tabulates the magnitude of the relative error between these two measures and ranges between approximately 1% and 5%. This indicates that the Markov model has a good level of agreement between the power estimates it provides and those of the standard VHDL-based approach. These results illustrate that the Markov model is an adequate method for the simulation of circuits driven by sequential modules. The figures quoted for the Markov ratio and the VHDL ratio are calculated using the ratio of Markov power to UWN power and VHDL power to UWN power respectively. These measures show a narrow range of values of between 0.10 and 0.22. They give an indication of how both methods behave when the sample size is altered, in comparison with the UWN estimate. As the sample size decreases, both VHDL and Markov estimates increase slightly. This to be expected since decreasing the sample size reduces the amount of correlation, given the small number of states. It is significant that the relative error between the models does not alter appreciably, demonstrating that the variation in power estimates is solely due to varying the sample size.

Figure 5.12 demonstrates the motivation for this approach to power estimation. It shows the relative magnitudes of each of the Markov, VHDL and UWN power estimates, for four sample sizes. The over-estimation of the UWN estimates in this case is apparent, as is the equivalence of the Markov and VHDL techniques.

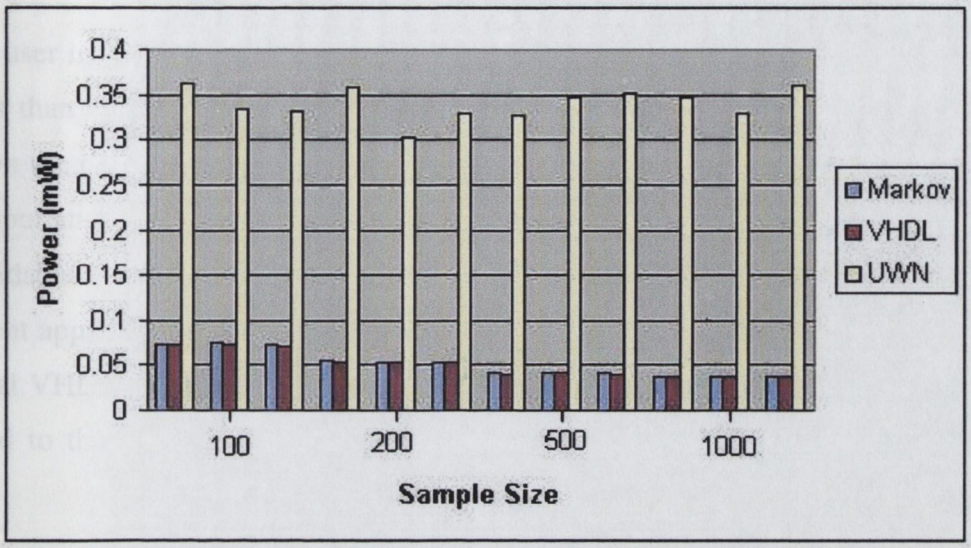


Figure 5.12: Relative Magnitudes of Markov, VHDL and UWN Power Estimates

The data in Figure 5.13 shows the effect of increased correlation on the power estimates, as sample size is increased. The UWN sample, devoid of correlation, is not susceptible to this effect, as seen in the relative uniformity of its estimates. The Markov and VHDL-based power estimates track each other almost exactly.

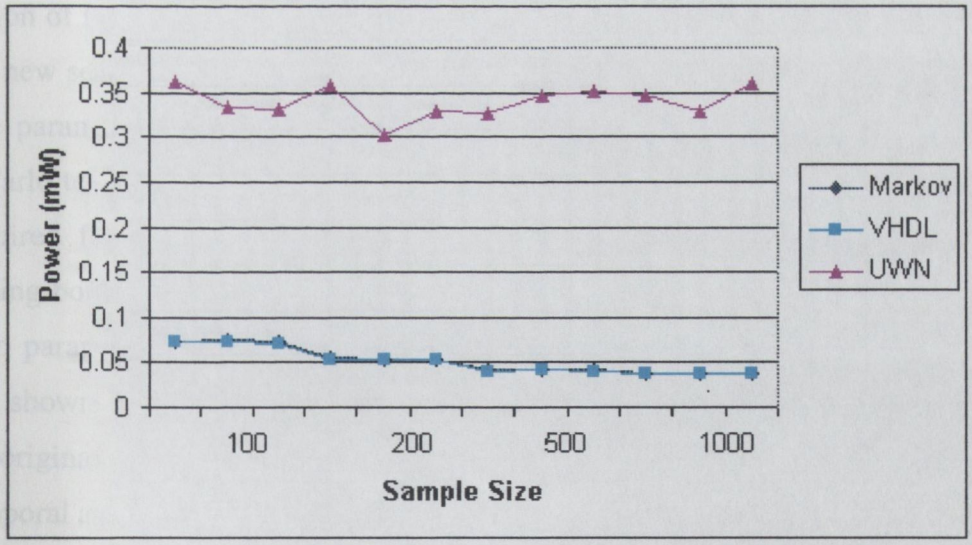


Figure 5.13: Effect of Correlation and Sample Size on Estimates

The final aspect of this approach that provides a motivation for its application is the development time required for each type of model. The UWN approach is the fastest, requiring no preparation time as the data generation routine, outlined in Chapter 3, is automatically included in the power estimation tool. The Markov technique too is very fast,

the only user input being the specification of the transition matrix. In the above instance this took less than two minutes to create and enter into the vector generation tool. The VHDL model, on the other hand, took several hours of text entry, debugging, the inclusion and test of file input and output in addition to the creation of a suitable testbench. While this activity is unavoidable when actually designing the module, the Markov model should provide a more convenient approach to power estimation before any VHDL code has been written. Even if a sequential VHDL circuit is available, the Markov vector generation routine is instantaneous, compared to the time required for a full VHDL simulation to obtain the same amount of vectors.

5.6 Summary and Conclusions

Four models of input data were identified as being of particular relevance to the process of power estimation at the subsystem and system level. They consist of a statistical model based on the use of signal and transition probability, a temporal correlation model, a spatial correlation model and a Markov chain model. The benefits of this approach include the compaction of the data sequence, with resulting savings in simulation time and the ability to generate new sequences whenever required, based on a single original data sequence. In each case, the parameters of the model were extracted from an original sample by estimation. Monte Carlo techniques were shown to be useful in this regard, when a user-specified error was required for the parameters. The statistical model was applied to data signals by determining both the signal probability and the transition probability for the data. Based on these two parameters, the statistical behaviour of the original data is recreated as required. This was shown to be an efficient technique for determining the power dissipation associated with the original sequence. Additionally, the presence of two types of correlation in the input data, temporal and spatial, was also incorporated into the modelling technique. Procedures for generating such correlated sequences were presented. The results obtained were also compared to the power dissipation resulting from UWN inputs. The influence of the model parameters on the power dissipation for this type of data was demonstrated for a number of circuits. The Markov model was used to model the output behaviour of sequential circuits, in particular the finite state machine representation of many controller circuits. Using a minimal amount of matrix manipulation, a data sequence was produced that closely resembled the statistical characteristics of the state machine. Power estimation simulations were undertaken

which indicated the usefulness of this method. The ability of the Markov model to track the VHDL equivalent over several simulations with differing sample size was illustrated. The relative error between the two approaches was less than 5% in most cases. Another feature of the Markov model was the reduced preparation time before the power simulation. The vectors are generated in a few seconds when compared to a VHDL simulation which can take several hours, depending on the length of the simulation. The next chapter will apply some of these techniques to the case study of a complete system, by undertaking the design and power analysis of a processor using a combination method consisting of a Markov model, transition probability sequences and high-quality UWN sources.

6 Power Analysis in a RISC Processor: A Case Study

To verify the techniques developed in the preceding chapters, a complete reduced instruction set processor (RISC) was designed, synthesised and simulated, to act as an architectural vehicle for the application of the data generation routines to a real design. Several software applications were developed using a pseudo-language based on VHDL, to provide meaningful signal activity during program operation. The execution of these programs allows the analysis of real signals, which are then used as reference data for the synthesis routines developed in earlier chapters. Some utilities were written, including an automatic translator for mapping memory contents into a VHDL ROM model and a synthesis tool for synthesising the SPICE file of the ROM. This combination of hardware, software and interface utilities allows the real datapath signals to be captured with relative ease and accuracy, thereby providing a satisfactory method of verifying the work undertaken in previous chapters. Power estimates for the system are obtained using three sets of data. Firstly real signal traces are used to drive the inputs of the system to obtain the actual power dissipation. Secondly, sequences of synthetic input data from a novel data model are applied and a system level power estimate is obtained. Finally, the power dissipation of the system to UWN is acquired as an upper bound to both sets of data. In addition to the system level power estimates obtained with the new model, power analysis of the ALU, the register file, the multiplier subsystem and the combinational control unit are also carried out independently of the system operation using the techniques presented in earlier chapters. The various approaches to synthetic data generation are compared with the results from real program operation, using UWN estimates as a reference.

6.1 RISC Processor Design

A generic RISC processor [Henn96] was designed to allow verification of the synthesis algorithms developed in previous chapters. It operates on a single instruction per cycle, with separate data and instruction memory and a 32-bit wide databus. This approach allows a combinational form of control to be incorporated, which was easily modified to test alternative instruction sets and datapath architectures. The datapath contains a multiplier unit

and a dual-port register file to exercise the datapath modules with realistic data streams. A brief description of the processor is now given, including the datapath-controller structure and the instruction set.

6.1.1 Single Cycle Processor Design

The single cycle approach to processor design requires the controller to issue a new instruction only when the last datapath element has finished processing its inputs. There is therefore a single instruction executed per clock cycle. This method has the advantage of simple controller design, as the controller becomes a combinational module. The disadvantage is that the clock speed is usually much lower than in the equivalent multicycle design, with a minimum clock period of 94ns in the synthesised design in this Chapter for example. Nevertheless, this design could be of use in less demanding applications such as embedded control of consumer electronics, where cost is a higher priority than performance [Burd95]. The block diagram of the version implemented in this work is given in Figure 6.1.

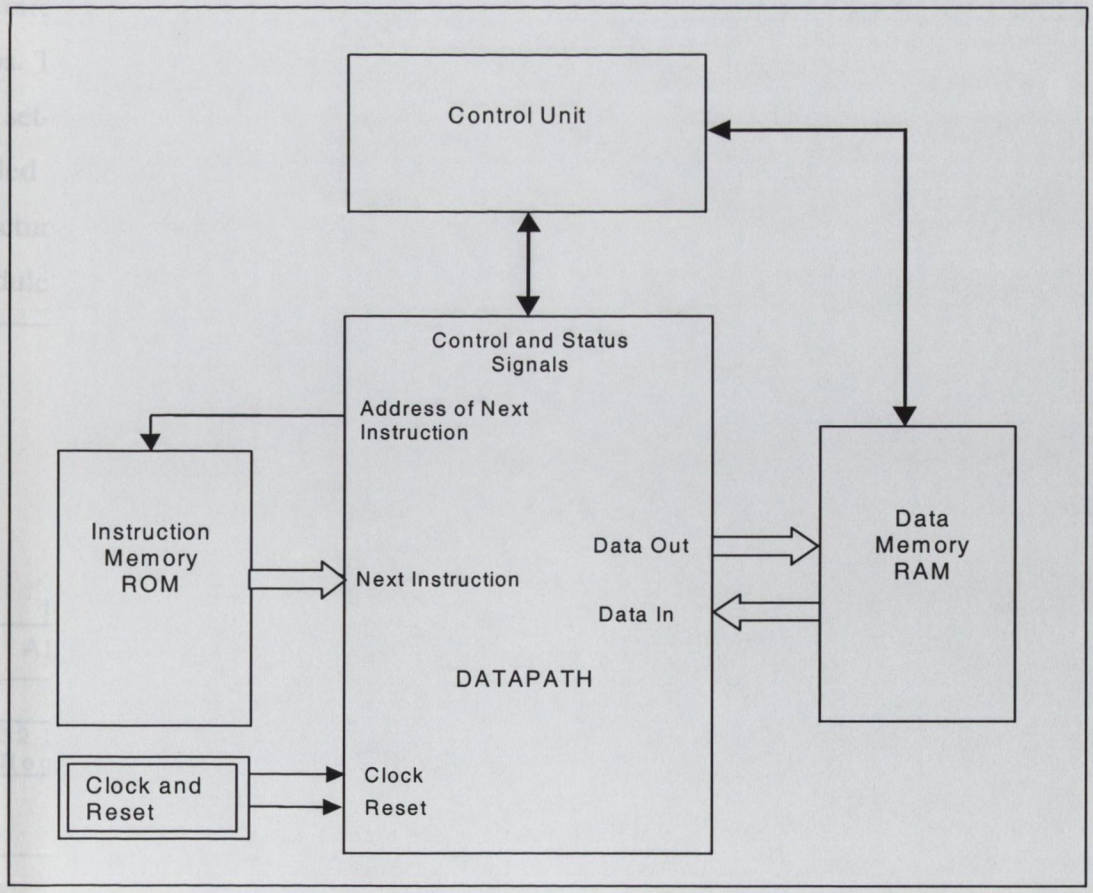


Figure 6.1: Processor Entity

The principal modules of the datapath are the arithmetic-logic unit, the multiplier block, the register file and the controller. The interface with program and data memory is discussed in Section 6.2, where a typical application program is presented. The operation of this processor is briefly summarised as follows. The design is synchronous and positive edge triggered. At initialisation, the processor commences to fetch and execute successive instructions, stored in the instruction memory, which is implemented by a behavioural ROM model. The first clock cycle after reset loads an instruction into the program counter (PC) where it is decoded into a number of fields. This instruction format is described below. Depending on the instruction type, appropriate multiplexors are selected so as to route the data through the associated combinational modules, such as the ALU, to the input of the correct register or data memory location. The data memory is implemented by a behavioural model of a RAM. At the next clock edge the data is latched into a memory location or a register, while the next instruction appears at the PC output. Data may be read from and written to sequential modules simultaneously because of the finite propagation delay through the circuits and the synchronous nature of the design as based on true edge-triggered flip-flops. The clock speed is chosen so that all combinational operations have completed prior to the set-up time preceding the next clock edge. As the clock must wait until signals have settled along the slowest path through the datapath, the speed of the design is limited. A structural view of the datapath tree is provided in Figure 6.2. This shows the hierarchy of modules used to implement the system.

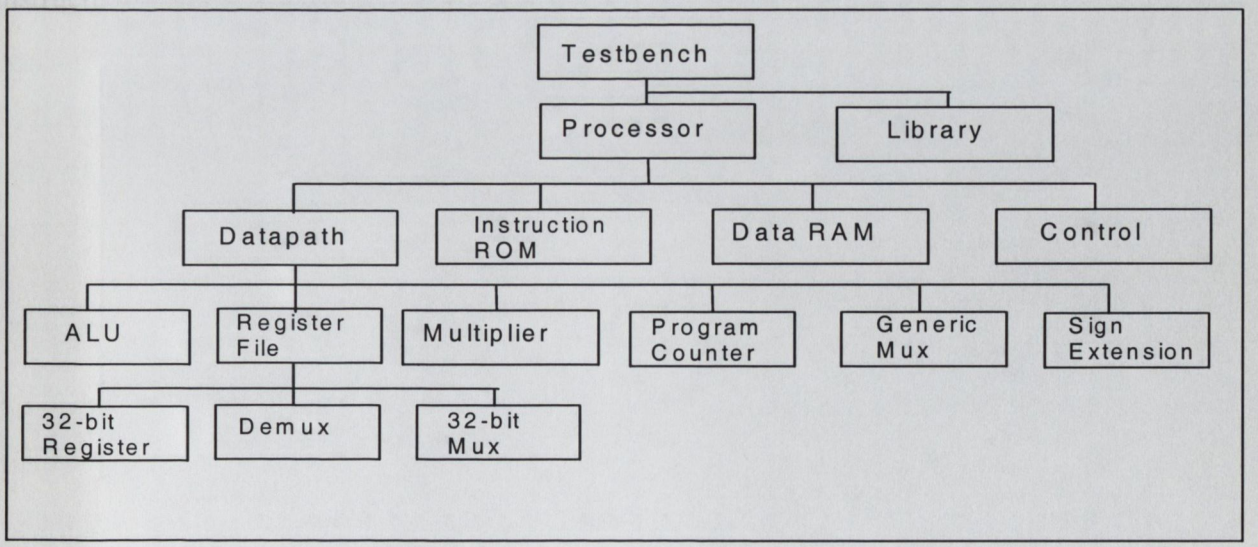


Figure 6.2: Structural View of The Datapath

The instruction set format is described in Table 6.1 and consists of four types of instructions with up to six subgroups of bits or fields in each instruction. R-type instructions are register type instructions such as *add*, *and*, etc. that access the ALU and store the result in a register. For this instruction type, for example, field 1 is zero and field 6 selects the particular ALU operation. In general, fields 1 and 6 are used to determine the instruction type while fields 2,3 and 4 select registers when appropriate. Field 5 represents the number of bits for a shift instruction.

Instruction type		1	2	3	4	5	6
R-type	field	0	rs	rt	rd	shamt	funct
	position	31-26	25-21	20-16	15-11	10-6	5-0
Load/Store	field	35-26	rs	rt	address		
	position	31-26	25-21	20-16	15-0		
Branch	field	4	rs	rt	address		
	position	31-26	25-21	20-16	15-0		
Jump	field	0	address				
	position	31-26	25-0				

Table 6.1: Instruction Types

There have been several such generic designs developed for research and teaching purposes [Hamb02], [Darm02], [AMS02]. This design is an extension of that work, with several modifications and enhancements, such as the multiplier unit design, the inclusion of immediate instructions and the behavioural modelling of data and instructions. The instruction set consists of the instructions given in Table 6.2.

Instruction op-code	Operation	Function field
lw (load/store type)	load word from memory	-----
sw (load/store type)	store word into memory	-----
beq (R-type)	Branch when equal	-----
add (R-type)	add	100000
sub (R-type)	Subtract	100010
and (R-type)	Logical and	100100
or (R-type)	Logical or	100101
slt (R-type)	set on less than	101010
Addi	add immediate	
Mul	Multiply	
Jmp	Unconditional jump	-----

Table 6.2: Instruction Set

The controller for this system functions as a decoding block of combinational logic for the various fields of the instruction. It can be decomposed into two submodules as indicated in Figure 6.3.

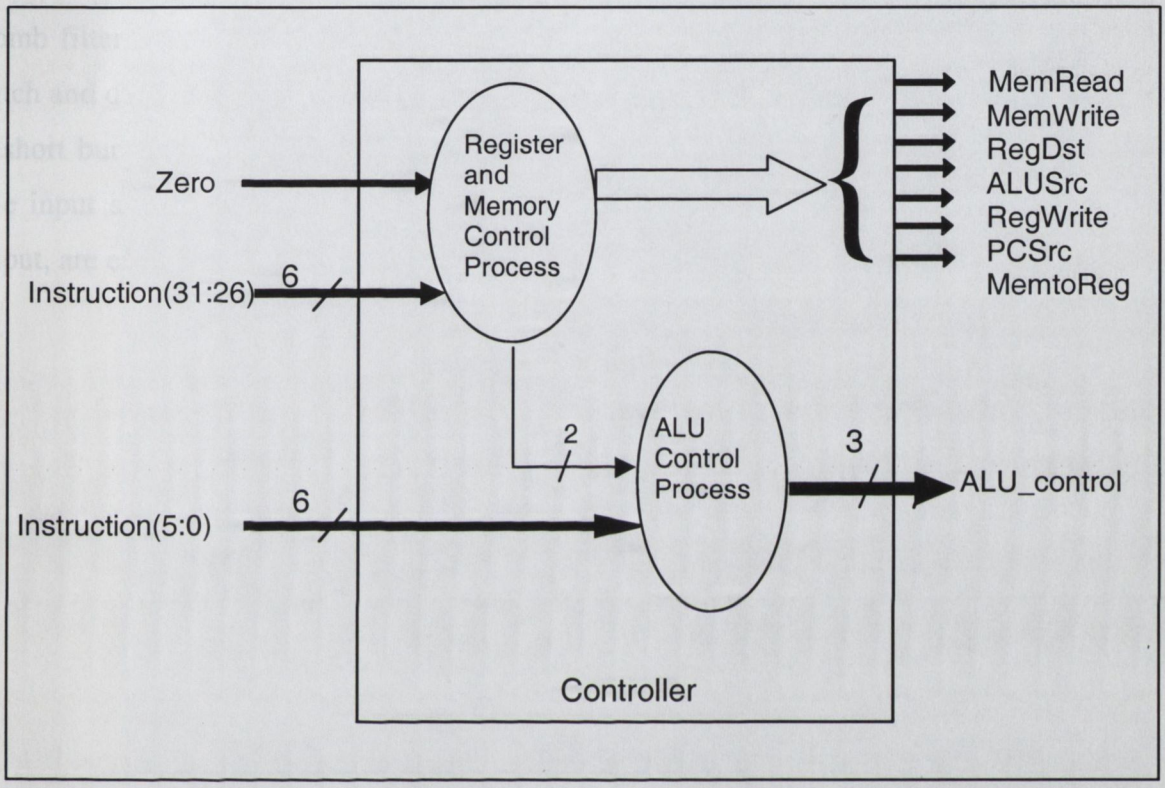


Figure 6.3: Controller Entity

The structure adopted for the controller allows easy modification to the instruction set and its coding. The single cycle design is used to compare the power dissipation for the datapath and its submodules when driven by signals that occur during the operation of an actual program, with the power estimates obtained when the techniques presented in previous chapters are used. Details of the software application used to capture the real input signals are now given.

6.1.2 Writing an Application

A number of applications of varying complexity were written for the processor. As there is no assembler available, the code was written in a pseudo-code developed for the work. This consisted of VHDL constants, one per instruction. These constants were loaded into a behavioural ROM model, which was then included in a top-level testbench with the processor, together with a RAM model. The laborious nature of this process limits the length

of the programs that may reasonably be developed in the absence of an assembler or other software development tools. However, a simple digital signal processing algorithm based on the Karplus-Strong method for synthesising a plucked string was one of the routines implemented [Karp83]. The block diagram of the algorithm is given in Figure 6.4. It is a comb filter structure, with parameters $L > 1$ and $R < 1$, the values of which determine the pitch and duration respectively of the synthesised note [Stei96]. The input sequence is usually a short burst of noise. The algorithm also functions as a reverberation effect, in which case the input signal is an audio waveform. Both aspects of the algorithm, with both types of input, are employed in this section.

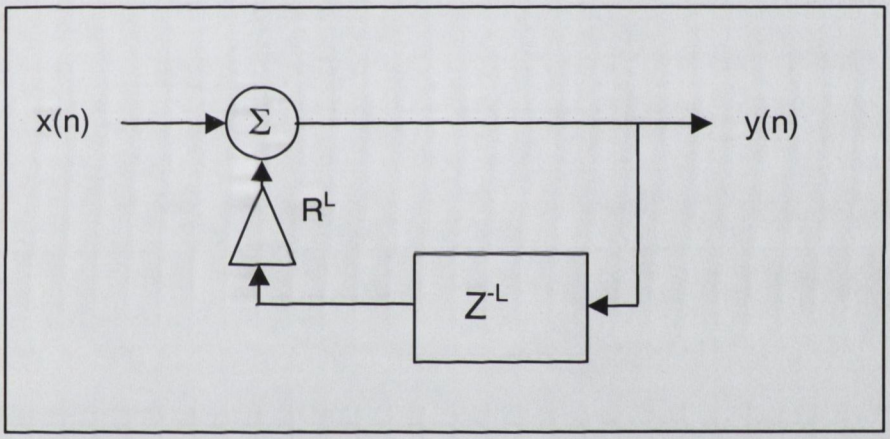


Figure 6.4: Karplus-Strong Plucked String Algorithm

Typically, for plucked string emulation the filter is driven by a short noise burst of approximately 100 samples, followed by a period of silence. The delay L determines the pitch of the sound according to (89), where F_s is the sampling frequency.

$$f_{sound} = F_s / L \tag{89}$$

The algorithm was initially coded in a high level language, with functionality to allow the processing of real audio samples. Figure 6.5 gives the impulse response for parameter values L=3 and R = 0.8598.

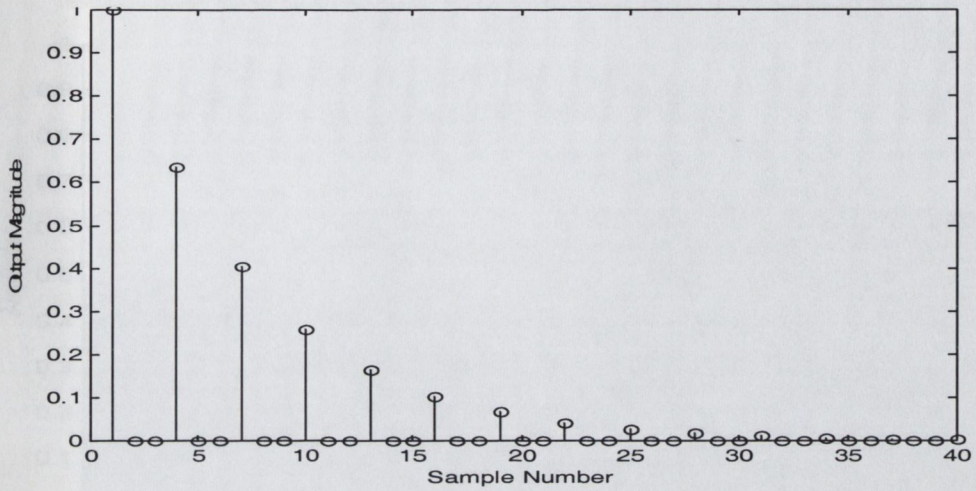


Figure 6.5: Impulse Response of Filter

Figures 6.6 and 6.7 contain a portion of the output waveforms for an input sample consisting of a short burst of 100 random noise samples. The parameter values were R=0.9995, L=75 with a sampling frequency of 22.5kHz. A sound of approximately 1s duration was produced. The periodic nature of the sound with a frequency of 300Hz, is evident.

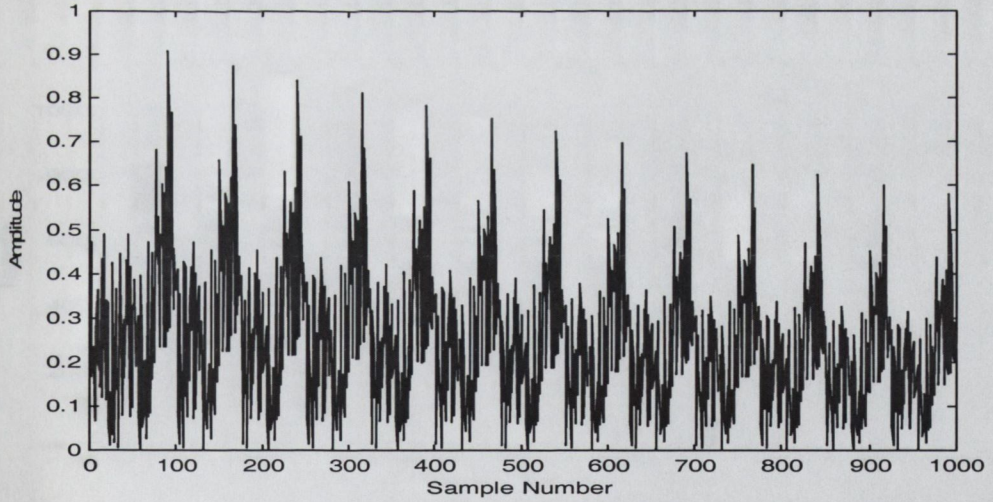


Figure 6.6: Response To Input Noise Burst

The complete synthesised waveform is given in Figure 6.7, where the envelope of the sound may clearly be seen.

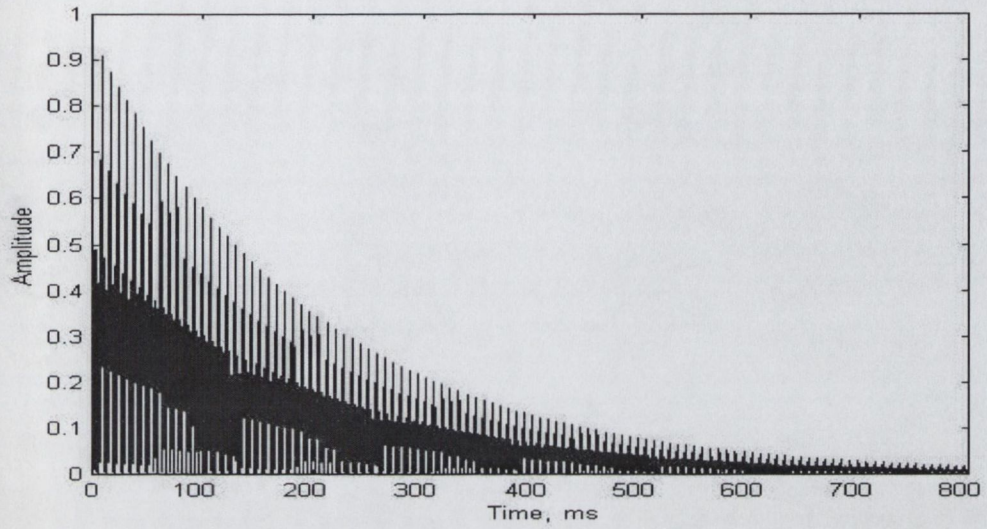


Figure 6.7: Envelope of Output Waveform

The rich harmonic structure of the waveform is indicated by the components of the spectrogram in Figure 6.8. A qualitative test of the aural properties of the sample indicates a very good approximation to the sound of a real plucked string, with a metallic, harpsichord-like quality.

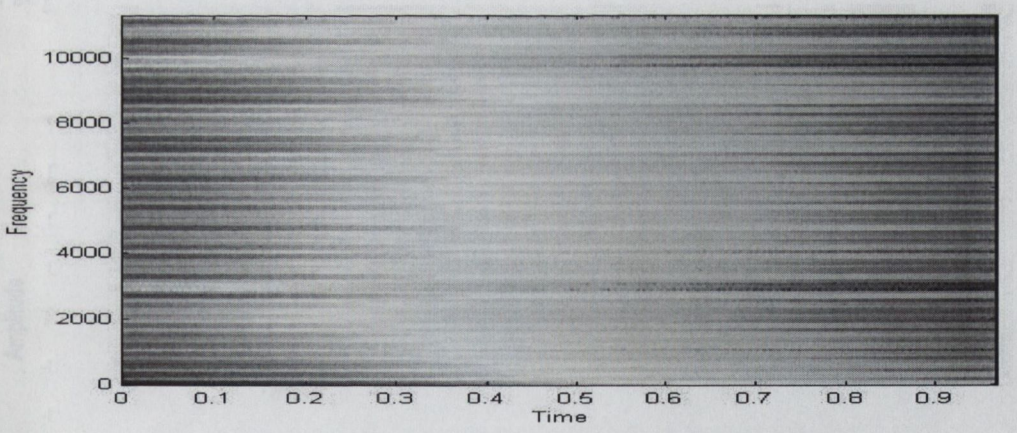


Figure 6.8: Spectrogram of Synthesised Plucked String

As the algorithm also functions as a reverberation algorithm when driven by audio data [Stei96], this too was included in the program test, as the data will provide an alternative source of input statistics for the datapath and its modules during program operation. An input signal consisting of an 8-bit recording of a short segment of speech, monophonically sampled at 44.1kHz was applied to the program. It contains 271040 samples and is approximately of 6s duration. Figures 6.9 and 6.10 show the input and output signals respectively.

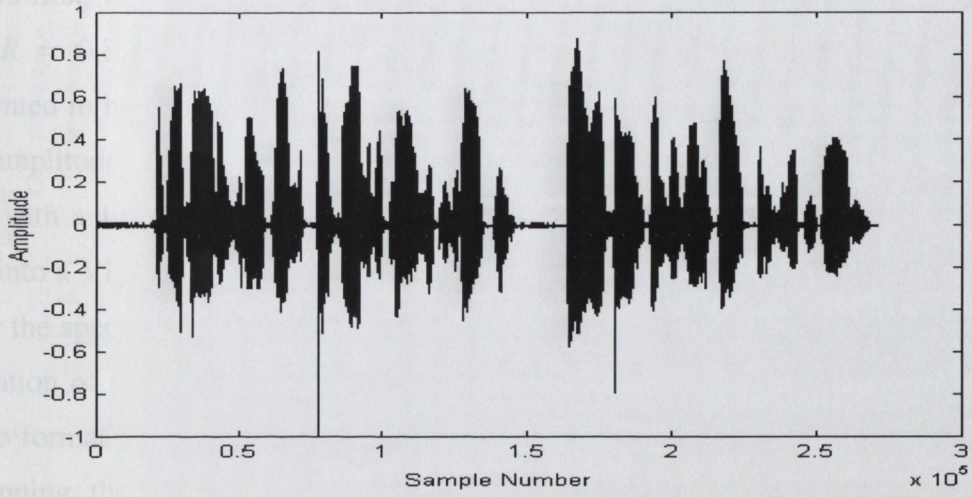


Figure 6.9: Audio Input Signal

The reverberant nature of the processed sample may be clearly seen in Figure 6.10, consisting of bursts of dense echoes and an audio tail, which persists after the input ceases. The output sample is otherwise largely unaltered, as the algorithm does not significantly affect the intelligibility or information content of the audio signal. A listening test reveals that a dense reverberation characteristic, similar to that of a large church, has been added to the original segment of speech.

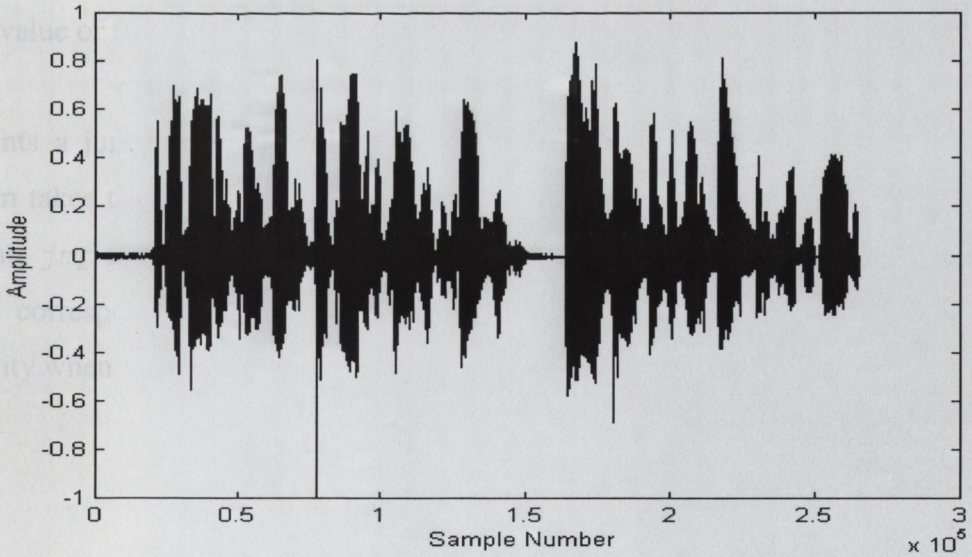


Figure 6.10: Audio Output Signal with Reverberation

The algorithm was coded for the processor with the instruction set specified in Table 6.2 as a means of exercising the instruction set. It was implemented as a machine language program stored in the instruction memory, coded in VHDL. The impulse response was

obtained first, with program parameters $L = 3$ and R = Ox6e0f 0fff, which is equivalent to a value R = 0.85983467055485 in a 32-bit system. The initial impulse amplitude was represented in hexadecimal by 0x7ff ffff, which is equal to a pulse height in a 32-bit system of an amplitude of $1\text{-}2^{-31}$ = 0.99999999953434. The processor was run for 6,000,000 clock cycles with a 100ns clock period. A tool was written to automatically synthesise the audio signal into a VHDL RAM module and to instantiate it in the system. The processor was then run for the specified number of clock cycles and the memory contents were extracted at the termination of the program. Another utility was written to convert the memory data into a suitable format for graphical output and importing into Matlab for analysis. While the code was running, the bit transition statistics for all signals of interest were also recorded. These will be used to verify the transition and signal probability approach to power estimation, as described in Chapter 5.

A small section of the low-level machine code for the Karplus-Strong program is given in Figure 6.11, as an example of how software and hardware may be integrated in a hardware simulation environment. Each line of the ROM contents corresponds to a 32-bit instruction, which is selected from the memory table by the ROM input *address*. The use of a library of constants allows a mnemonic form of each instruction code to be specified, rather than the binary value of the instruction code. Thus, the line of code:

```
jmp&"00"&To_StdLogicVector(X"00")&For_test1
```

represents a jump to the location *For_test1*. Each such instruction in the application program takes the form of a 32-bit word in the instruction ROM. The binary values of the constants *jmp* and *For_test1* are substituted for the codes at simulation time by looking up the corresponding values in the design library. This technique allows a degree of flexibility when coding the contents of the ROM.

```
entity INSTRUC_MEM7 is
    port (
        address: in instruc_mem_address;
        instruction: out risc_word
    );
end INSTRUC_MEM7;

architecture RTL of INSTRUC_MEM7 is

type rom_table is array (0 to instruc_mem_size-1) of
      std_logic_vector(31 downto 0);
constant L: std_logic_vector(15 downto 0):= To_StdLogicVector(X"0003");
          --L is delay, in samples
constant R_upper: std_logic_vector(15 downto 0):=
To_StdLogicVector(X"6e0f");
          --upper 16 bits of multiplier
constant R_lower: std_logic_vector(15 downto 0):=
To_StdLogicVector(X"0fff");
--lower 16 bits of multiplier
constant For_test1: std_logic_vector(15 downto 0):=
To_StdLogicVector(X"0009"); --absolute address of instruction For_test1
constant END_TEST: std_logic_vector(15 downto 0):=
To_StdLogicVector(X"0017");  --absolute address for jmp
.....etc. etc.
constant rom: rom_table:= rom_table'(
          addi&ZERO&a3&NUM_SAMPLES,--0: a3 <- NUM_SAMPLES
          addi&ZERO&a1&L,                    --1: a1 <- L
            addi&ZERO&a2&R_upper,            --2: a2 <- R_upper (16-bits)
-- calculate R^L and store in t3 using a for loop:
    addi&ZERO&s0&To_StdLogicVector(X"0001"), --8: i <= 1
    -- Executes L-1 times
    -- address For_test1 =9 :
         slt&s0&a1&t0&"00000"&slt_func,         --9: t0<-1(continue)
      --PC-relative jump to EXIT1
       mult&t3&a2&ZERO&ZERO&mult_func,         --11:t3*a2 = R^i*R
         mfhi&ZERO&ZERO&t3&ZERO&mfhi_func,     --12: t3 <- R^(i+1)
         sll_op&t3&ZERO&t3&"00001"&sll_func,   --13: remove sign bit
         addi&s0&s0&To_StdLogicVector(X"0001"),--14: i <= i+1
   jmp&"00"&To_StdLogicVector(X"00")&For_test1,     --15: jump to
For_test1
    -- address EXIT1 = decimal 16: PC_relative address for beq =5
        etc. etc.
      To_StdLogicVector(X"00000000")
                                  );

begin
    instruction <= rom(conv_integer(address));
end RTL;
```

Figure 6.11: Implementation of Karplus-Strong Algorithm in VHDL ROM

The reverberation functionality of the algorithm was also exercised with an audio sample replacing the impulse input. Sample output audio waveforms from the execution of the algorithm on the processor are shown in Figures 6.12 and 6.13. These were obtained by running a VHDL simulation within the Synopsys environment [Syno02] using the Karplus-Strong application program stored in the instruction ROM. The testbench for the design created a number of output signal traces and an image of the final contents of the RAM, using a file-based technique. A utility was written that converted the contents of the RAM into an audio format file and into a form importable by Matlab. This allowed the data to be converted from a sequence of 32-bit binary integers in twos-complement form into real data. Two versions of the program were therefore run with two distinct sets of initial input samples in the RAM at the commencement of each simulation, one for the impulse response, the second for audio reverberation. The first run, with an impulse signal as an input, resulted in the impulse response of Figure 6.12. The second simulation operated on the input audio sequence displayed in Figure 6.9, resulting in the output waveform of Figure 6.13. A sequence of 8000 output samples is given in this figure, where the reverberation effect may be observed in the denser portions of the waveform.

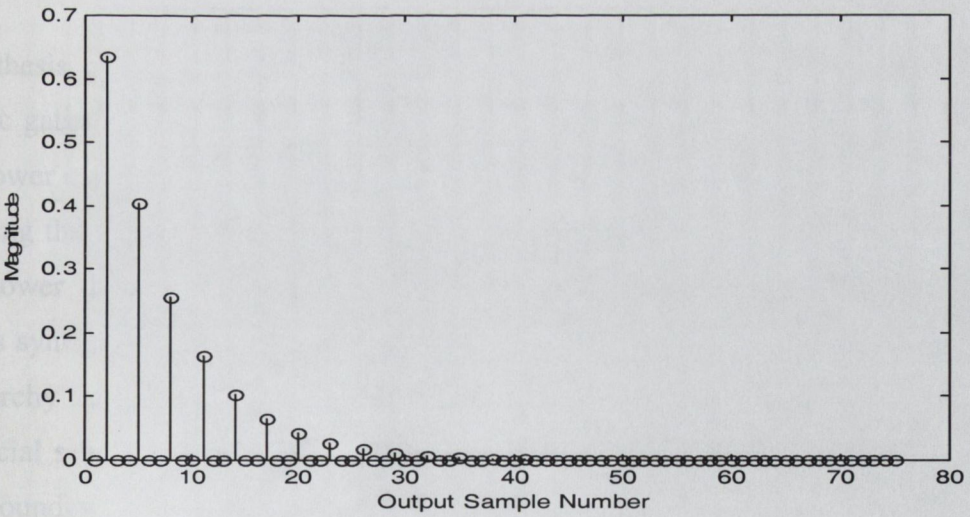


Figure 6.12: Processor Impulse Response Data

In both cases the same algorithm was employed, the only difference being the initial data stored in RAM. This allowed both the plucked string and the reverberation functionality of the algorithm to be observed. The parameters of the algorithm were chosen for this

simulation so as to minimise the VHDL simulation time, while verifying the operation of the processor. A delay of $L = 3$ was again chosen for demonstration purposes.

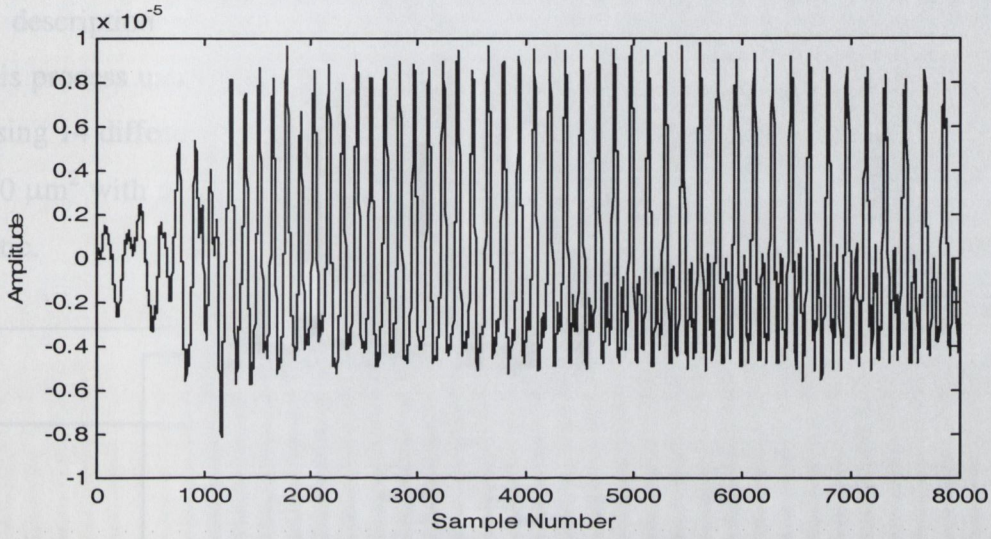


Figure 6.13: Processor Output Data

In the next section, results for the synthesis of the processor are given, whereby the design is converted into a circuit containing real devices.

6.2 Synthesising the Processor

The synthesis procedure, whereby the high-level VHDL code is translated into a network of real logic gates connected by capacitive interconnect, is an important step in the verification of the power estimation model. The inclusion of capacitive nodes in the circuit means that by monitoring the charge and discharge of these nodes during processor operation, an estimate of the power drawn from the supply may be made, as discussed in Chapter 2. When the design is synthesised using a tool such as Synopsys, it is compiled into a netlist from which all hierarchy has been removed. This set of gates is drawn from a library provided by a commercial silicon foundry, where the finished device can be fabricated. For this work the silicon foundry chosen was that of European Silicon Structures (ES2) in Aix-en-Provence, France and their 0.7μm ECPD07 library was selected for the synthesis procedure [ES296]. An incremental approach was taken to this phase of the work, with initial synthesis of submodules such as the control unit, the multiplier, the register file and the ALU, followed by a synthesis of the complete design. The details of the outcome of this step are now reported for each module.

6.2.1 Control Unit

For the combinational control unit, the input to the synthesis procedure was a high level VHDL description of the required functionality. The output netlist resulting from the synthesis process using the ECPD07 library is shown in Figure 6.14. The circuit contains 56 gates using 14 different gate types drawn from the library. The estimated area of this module is 24810 μm^2 with a maximum delay of 7.82ns. This critical path is indicated in bold on the schematic.

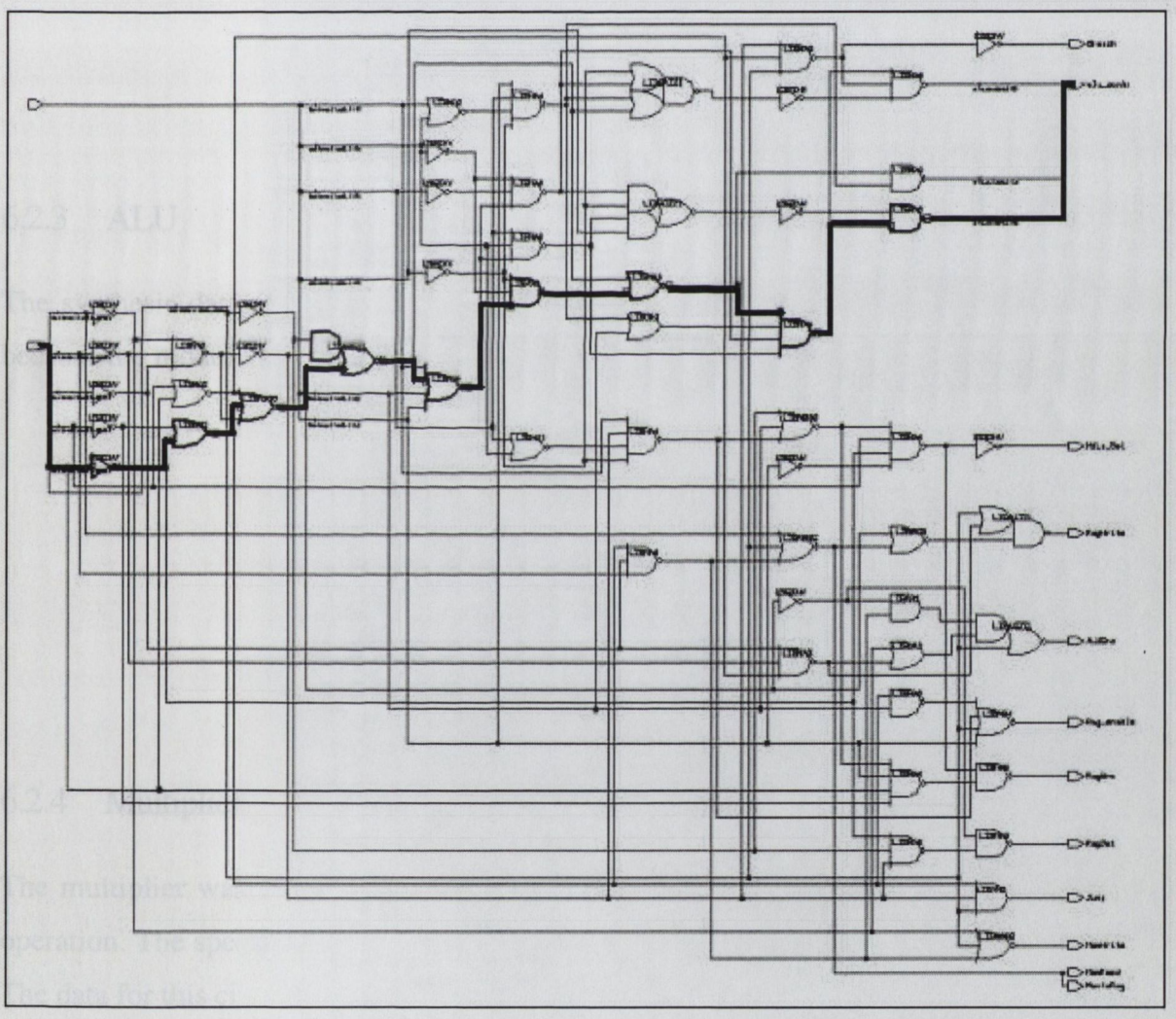


Figure 6.14: Control Unit Schematic

6.2.2 Register File

The synthesis data for the register file is reported in Table 6.3. The netlist contains 3571 cells and so the schematic is too large for inclusion here. This module contains sequential and combinational modules and the longest combinational path is estimated as 10.82ns.

Number of cells	3571
Number of cell types	18
Number of ports	247
Number of nets (nodes)	4101
Combinational area	1603634μm^2
Sequential area	1741555μm^2
Total area	3345189μm^2
Maximum delay	10.82ns

Table 6.3: Register File Synthesis Results

6.2.3 ALU

The synthesis data for the ALU is given in Table 6.4. The reported sequential area is zero because the module has been implemented entirely with combinational cells.

Number of cells	654
Number of cell types	26
Number of ports	100
Number of nets (nodes)	752
Combinational area	432686μm^2
Sequential area	0
Total area	432686μm^2
Maximum delay	46.9ns

Table 6.4: ALU Synthesis Results

6.2.4 Multiplier

The multiplier was the largest submodule in terms of area while also providing the fastest operation. The speed is due to the short inter-register path lengths in the synthesised module. The data for this circuit is summarised in Table 6.5.

Number of cells	2532
Number of cell types	17
Number of ports	100
Number of nets (nodes)	3562
Combinational area	2445885μm^2
Sequential area	112358μm^2
Total area	2558243μm^2
Maximum delay	3.61ns

Table 6.5: Multiplier Synthesis Results

6.2.5 Datapath

The last module to be synthesised was the datapath. This comprised the ALU, register file and multiplier, together with multiplexers, a sign extension unit and some glue logic. The final form of the processor then consisted of the synthesised datapath and controller, together with behavioural models of RAM and ROM, as indicated earlier in Figure 6.1. The implementation data for the datapath are reported in Table 6.6.

Number of cells	11248
Number of cell types	35
Number of ports	301
Number of nets (nodes)	12195
Combinational area	5642578μm^2
Sequential area	1910093μm^2
Total area	7552671μm^2
Maximum delay	96.09ns

Table 6.6: Datapath Synthesis

The maximum delay of 96ns allows a 10MHz clock cycle and confirms the observation made earlier that the absence of pipelining considerably reduces the maximum clock speed. The relative contributions of datapath and control to the processor area and the contribution of each of the modules to the datapath area is indicated in the pie chart of Figure 6.15.

In the next section, the power estimation techniques presented so far are drawn together, to demonstrate how the power dissipation of a complex system may be efficiently estimated using a combination of the methods discussed previously. The synthesised processor is used as the architectural vehicle for verification of the methodology.

In the next section, the power estimation techniques for these submodules are verified in the context of an overall approach to the power estimation of complex systems. Traces of real

signals captured during program operation are used to verify both probabilistic and Markov data synthesis techniques developed in earlier chapters. UWN power estimates of the modules are also provided for reference.

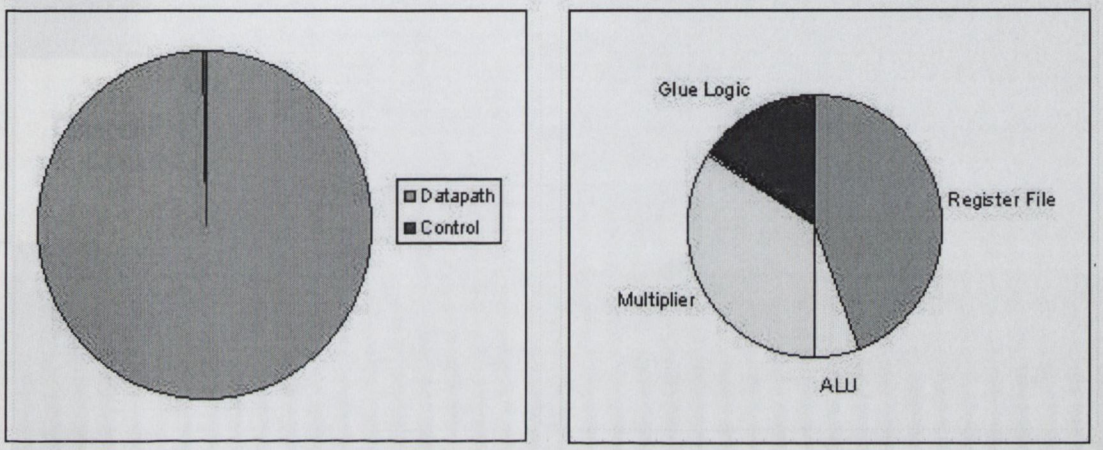


Figure 6.15: Processor and Datapath Area

6.3 Power Estimation in Complex Systems

Just as a digital hardware design may be viewed as a hierarchical structure with a tree-like topology, so too may the signals within a design be considered hierarchically. At the top level of the design, there is the interface between the system and its environment. Below that, there are vertical interface signals between each level of the tree structure, in addition to horizontal signals between each submodule at any given level. VHDL reinforces this notion by giving a limited scope to the data type "signal", which cannot extend above or below its immediate level in the structure.

In the context of this work, the notion of a signal hierarchy provides a convenient framework for the analysis of the data models for power estimation discussed in previous chapters. The system-environment level represents the processor-memory level, the next layer down is the datapath-control interface, and finally the signals between datapath submodules may be considered to be on a third level, for example register file-ALU communication. The models examined in previous chapters are now applied to these layers, to demonstrate the application of the techniques presented to power estimation at each layer in the design tree. A combination model is also proposed for system level power estimation. Figure 6.16 illustrates the proposed relationship between power models and system levels.

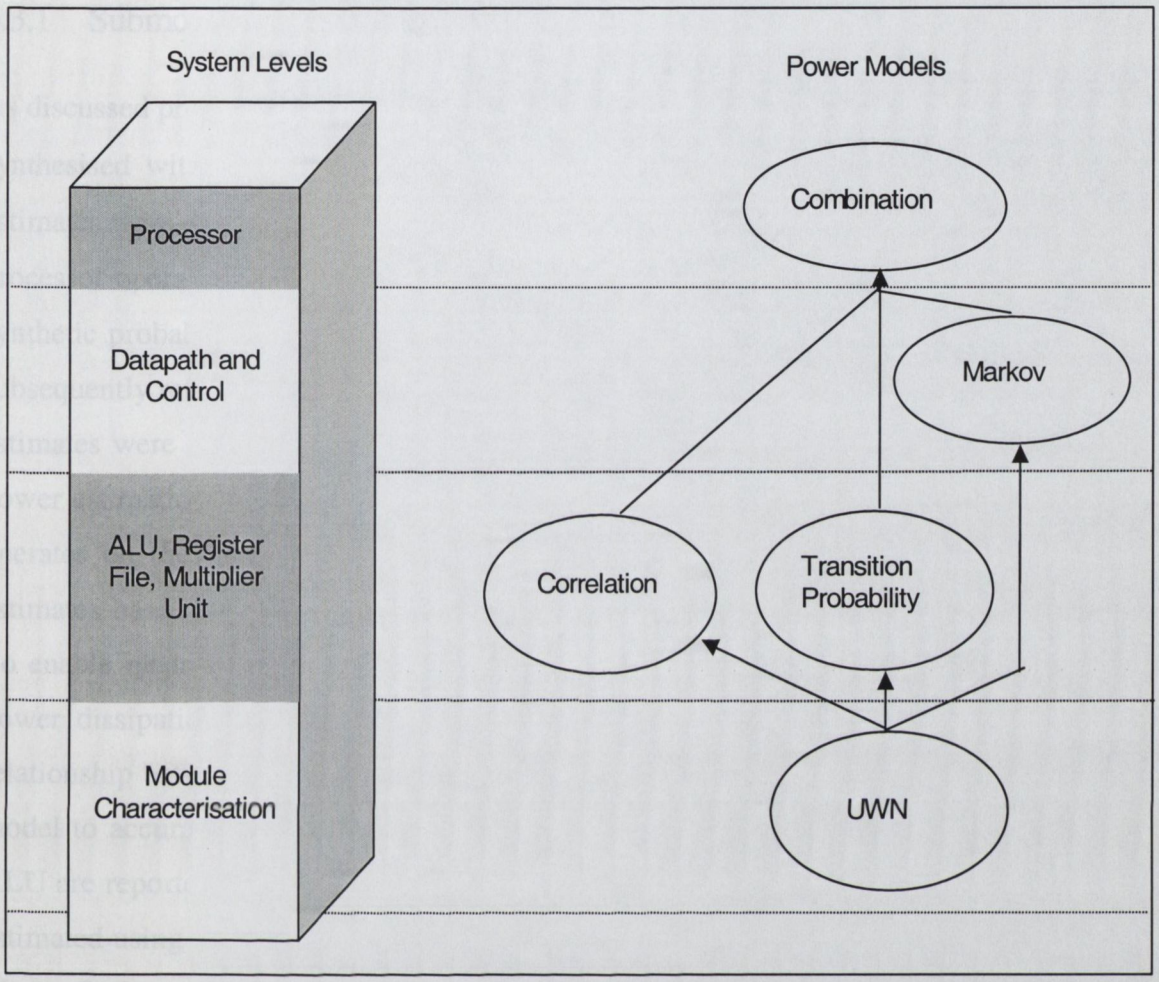


Figure 6.16: Appropriate Power Models for Hierarchical Approach

At the top level of the design, the processor power during program operation is estimated with a combination model, consisting of contributions from each of the techniques discussed. At the next level down, the datapath control signals may be modelled with a Markov model, as discussed in Chapter 5. At the lowest level, the transition probability model is used to accurately predict the power dissipation of individual submodules, such as the ALU and the register file. In addition, UWN testing of all modules provides a worst-case datum to which power-sensitive architectural modifications are referenced. The results of this approach are now presented bottom up, starting with an analysis of the transition probability model for submodule power estimation.

6.3.1 Submodule Power Analysis

As discussed previously, the ALU, register file, multiplier block, controller and datapath were synthesised with a 0.7μm CMOS technology. For each synthesised module, three power estimates were produced: the actual or real power dissipation for the module recorded during processor operation, the estimated power dissipation of the module due to the application of synthetic probability-based data and the power dissipation with UWN input data. These are subsequently referred to as real power, synthetic power and UWN power, respectively. Power estimates were obtained for all three types of input data using PowerCount, a Monte Carlo power estimation tool for Synopsys developed by A. T. Schwarzbacher [Schw01]. This tool operates on the synthesised netlist of standard cells and produces highly accurate power estimates based on the switched capacitance recorded during each of the three simulations. To enable meaningful comparisons between the real, synthetic and UWN power estimates, power dissipation was also recorded for a range of input sample sizes. This allows the relationship between power estimate and sample size to be factored out and the ability of the model to accurately track the real power can be better observed. The power results for the ALU are reported in Figure 6.17, for sample sizes of 1 to 750 input vectors. The power was estimated using a single iteration in each case because the data may not be Gaussian and so a Monte Carlo analysis has no significance.

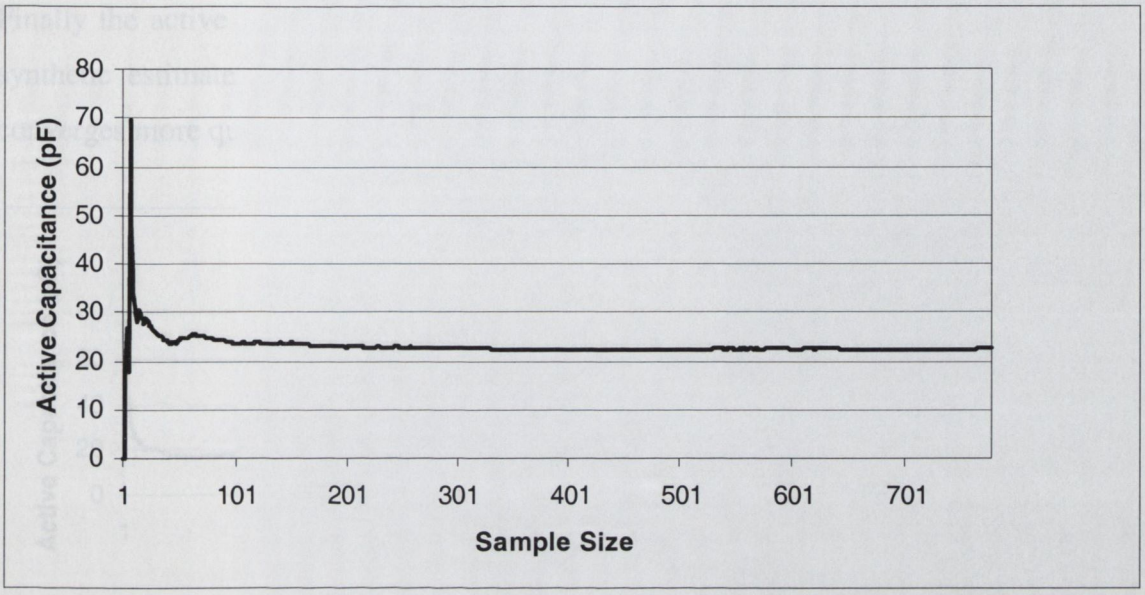


Figure 6.17: ALU Active Capacitance for UWN Input Data

The UWN active capacitance converges to a value of approximately 22pF and represents an upper bound on the likely power dissipation. As discussed in Chapter 2, the power dissipation is obtained from the active capacitance by multiplying it by the clock frequency and the square of the supply voltage. It provides a convenient measure of power dissipation while abstracting from both supply voltage and clock frequency. The real active capacitance, recorded during the execution of the test program is given in Figure 6.18. It converges to approximately 18pF, which is 22% lower than the UWN estimate.

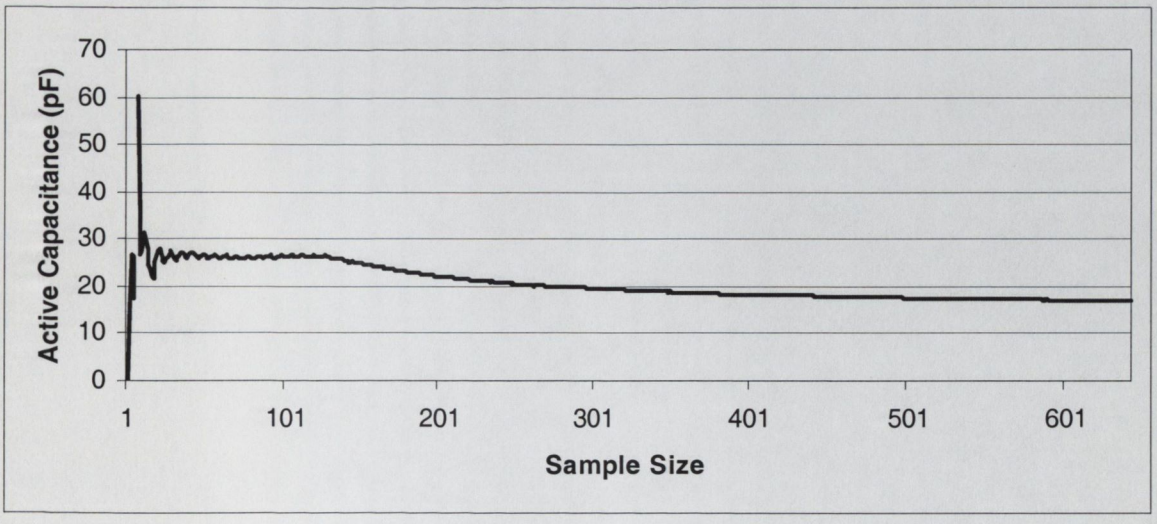


Figure 6.18: ALU Active Capacitance for Real Input Data

Finally the active capacitance produced by the transition probability model, referred to the synthetic estimate, is given in Figure 6.19, were it can be seen that the synthetic data converges more quickly than the real input data.

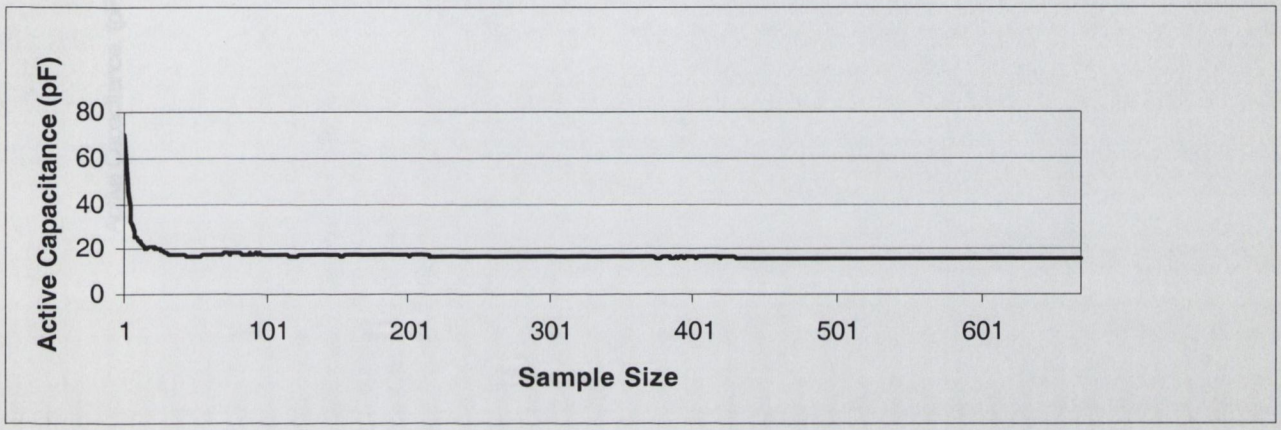


Figure 6.19: ALU Active Capacitance for Synthetic Input Data

The estimate here converges to approximately 19pF, which is in close agreement with the real active capacitance. Similar results are now reported in summary form for the other modules. Figure 6.20, for example, shows three traces, the real capacitance, the UWN capacitance and the synthetic value obtained with the transition probability model. The convergence of the synthetic power and real power with increasing sample size is evident and demonstrates the usefulness of this approach.

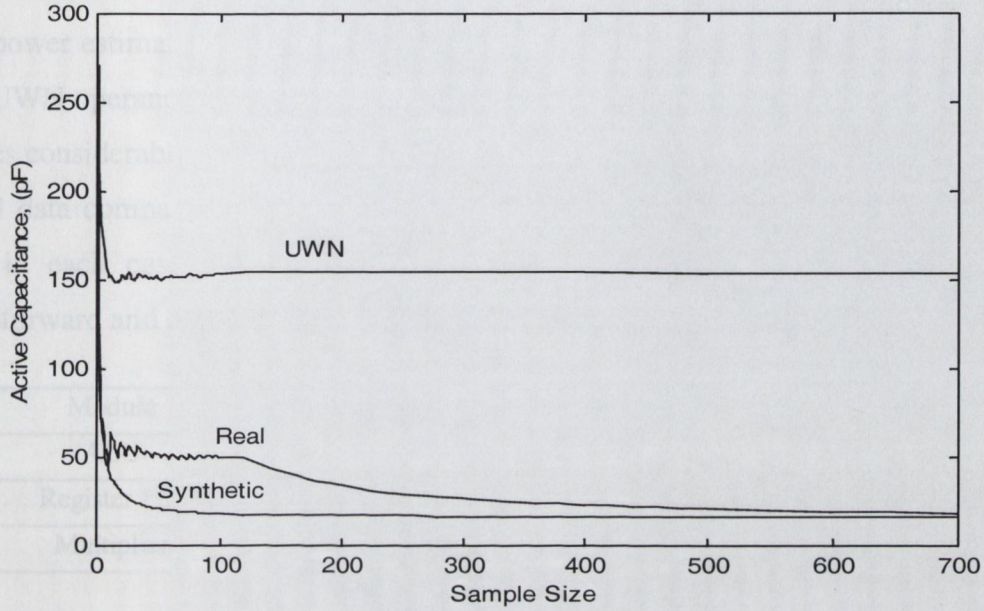


Figure 6.20: Multiplier Active Capacitance

Finally the other major component in the datapath is the register file. Power estimates for this component are also adequately determined using the synthetic data sequence. The results are presented in Figure 6.21.

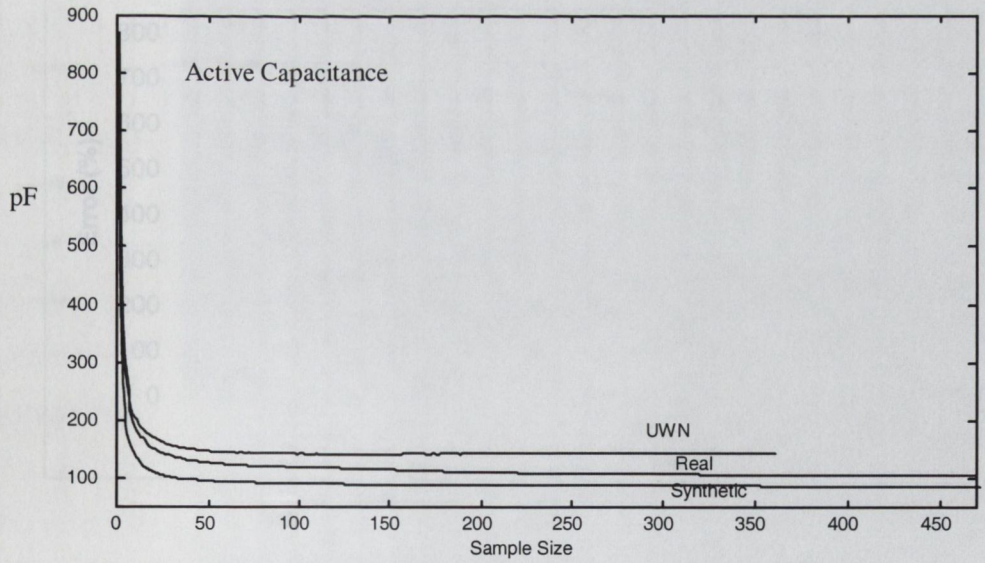


Figure 6.21: Register File Active Capacitance

The errors in the power estimate relative to the real power when using UWN and synthetic data are reported in Table 6.7 for the three modules. These were calculated using estimates for the largest sample size in each simulation. The improvement ratio quoted is the relative

improvement in error when comparing UWN and synthetic estimates. For the multiplier, the UWN power estimate is over eight times larger than the real power. This is due to the much larger UWN operands. The real data, in this example, contains much smaller data values and occupies considerably less of the dynamic range. The multiplier is therefore less exercised by the real data compared to the UWN inputs. The synthetic data slightly underestimates the power in each case. The probability model used to obtain the synthetic is however, straightforward and easily implemented and is satisfactory in the context of this work.

Module	UWN Error (%)	Synthetic Error (%)	Improvement Ratio
ALU	33.2	5.9	4.62
Register File	141.8	20.9	5.78
Multiplier	823.5	4.7	174.21

Table 6.7: Module Active Capacitance

The relationship between the three power dissipation evaluation methods is also presented graphically in Figure 6.22. For each datapath module the synthetic power estimate is significantly better than the UWN estimate.

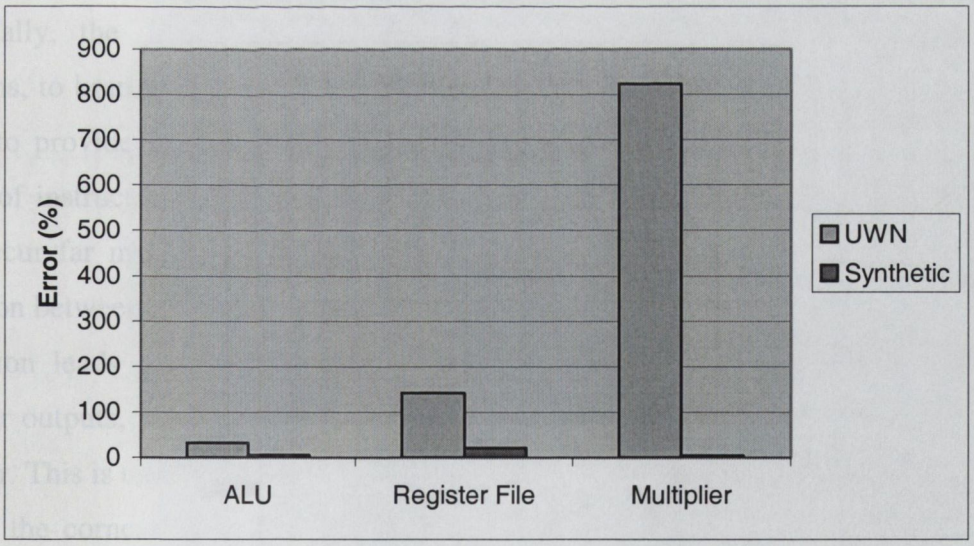


Figure 6.22: Power Estimates of Submodules

6.3.2 Modelling Control Signals

The application of a Markov model to the synthesis of finite state machine output data was discussed in detail in Chapter 5, where results were presented for real, SPICE level circuits. This model, as indicated in Figure 6.16, is used to model the datapath controller interface

when an FSM control circuit is used. Additionally, the Markov model is incorporated into a combination model for modelling the complete processor or datapath as required. Therefore, as the FSM contributes only some of the input signals to the datapath, those results will not be duplicated here but will be utilised in the next section where the combination model is applied to the power estimation of the entire datapath.

6.4 System-level Power Estimation with a Combination Model

At the top level of the system hierarchy as defined in this work, the datapath communicates with instruction and data memory and also receives controller signals. For the system used as the architectural test in this chapter, 70 bits of data are transferred to the datapath each clock cycle comprising a 32b instruction, 32b of data and a 6b control word. Realistic representation of such a complex sequence of vectors requires a correspondingly high level approach. Simple probabalistic patterns, while useful at lower submodule level, are not appropriate at a system level. For example, a synthetic sequence of instructions must still contain only valid patterns to ensure correct system operation. Likewise, bit fields representing resister selects are constrained to avoid some values, depending on the context. Additionally, the format of instructions varies from memory accesses to arithmetic operations, to branches and jumps. All these must be realistically represented in an efficient manner to provide realistic and functionally correct input sequences for the datapath. The mixture of instructions also varies with the application. Some instructions, such as *add* and *store*, occur far more frequently than for example, *jump* and *shift* [Patt98]. There is also correlation between one instruction and the next, as will be demonstrated in this Section. That observation leads to the conclusion that the Markov model, successfully applied to the controller outputs, may also be of relevance in modelling the stream of instructions into the processor. This is demonstrated to be the case and the instruction stream model proposed here provides the cornerstone of a system-level power estimator, which is now described. The model is referred to as the combination model.

The aim of the combination model is to provide realistic system-level datapath input sequences, without the overhead of running a complete software environment around the processor. The relative frequencies and correlation between instructions and data, if any, must also be present and preserved. Data is represented as either correlated or random

patterns, depending on the amount of prior knowledge available about the data. The constraints on the instruction and control sequences are, however, much more onerous. No new instructions are introduced into the synthetic sequence, as this would represent an erroneous processor state. Additionally the format of each instruction must be correctly reconstructed, with valid entries in all bit fields. To produce such a complex sequence, all the models previously discussed are utilised. For example, the 32b *add* instruction has five active fields, reproduced in Table 6.8 for convenience, with an offset of zero.

(31:26)	(25:21)	(20:16)	(15:11)	(10:5)	(5:0)
op-code	register	register	register	offset	function code

Table 6.8: Add Instruction Format

Each field of this instruction is modelled with one of the techniques presented. The 70-bit datapath input, consisting of instruction, data and control is then formed from an aggregate of the 32-bit Markov instruction model, a 32-bit UWN or correlated data model and a 6-bit control model. The control model is Markov for sequential control or is derived directly from the instruction model for single cycle combinational control. The combination is represented graphically in Figure 6.23.

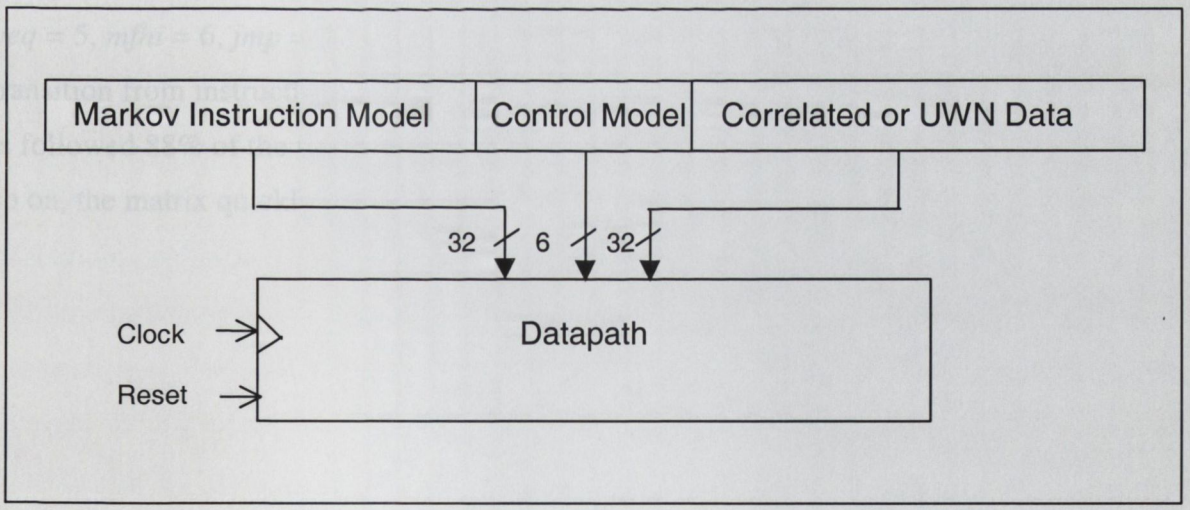


Figure 6.23: Combination Power Estimation Model

The instruction synthesis procedure is based on an underlying Markov model present in the instruction stream. The suitability of this procedure was verified as follows. A long system simulation was performed for 5 million clock cycles. The instruction stream was monitored

during the complete simulation. Software was written to extract and test a transition matrix from the instructions recorded. This matrix was then tested for Markov properties, specifically the convergence of the matrix to the relative probabilities, as discussed in Chapter 5. The relative frequencies of the instruction were calculated and were found to be in close agreement with values predicted by the Markov model. An example of a transition matrix obtained by running the Karplus -Strong algorithm is given in (90).

$$T = \begin{pmatrix} 0.0250 & 0 & 0 & 0 & 0 & 0 & 0 & 0.9750 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3091 & 0.6909 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0231 & 0 & 0 & 0 & 0.0564 & 0.8671 & 0.0535 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.0007 & 0.0284 & 0.0007 & 0 & 0 & 0 & 0.4624 & 0.4539 & 0.0270 & 0.0270 \\ 0.0562 & 0 & 0 & 0.0562 & 0 & 0 & 0 & 0 & 0 & 0.8876 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \quad (90)$$

Each instruction was assigned a row in the matrix as follows: *add* = 1, *slt* = 2, *sll* = 3, *mult*=4, *beq* = 5, *mfhi* = 6, *jmp* = 7, *addi* = 8, *lw* = 9 and *sw* = 10. Entry t_{ij} denotes the probability of a transition from instruction i to instruction j during program operation. For example, a load *lw* is followed 88% of the time by a save *sw*. By forming higher powers of the matrix, T^2, T^4 and so on, the matrix quickly converges to the relative probabilities P^n, given in (91).

$$P^n = \begin{pmatrix} 0.0080 \\ 0.1384 \\ 0.0110 \\ 0.0108 \\ 0.1384 \\ 0.0108 \\ 0.1304 \\ 0.2820 \\ 0.1352 \\ 0.1350 \end{pmatrix} \quad (91)$$

This indicates that, for example the most common instruction is the *add immediate* instruction *addi*, with a 28% likelihood of occurrence. Based on this transition matrix, code was written to generate a stream of instructions. This ensures that the same mix of instructions is placed in the data and that no illegal instructions can occur. The register fields of each instruction are filled by bit patterns derived from the transition and signal probabilities of the appropriate signals, while the larger address and constant fields for load, store, branch, etc., are similarly filled. In this manner any constraints on address space, not present in this work, are easily incorporated into the model. The method is demonstrated for an *add* instruction in Figure 6.24.

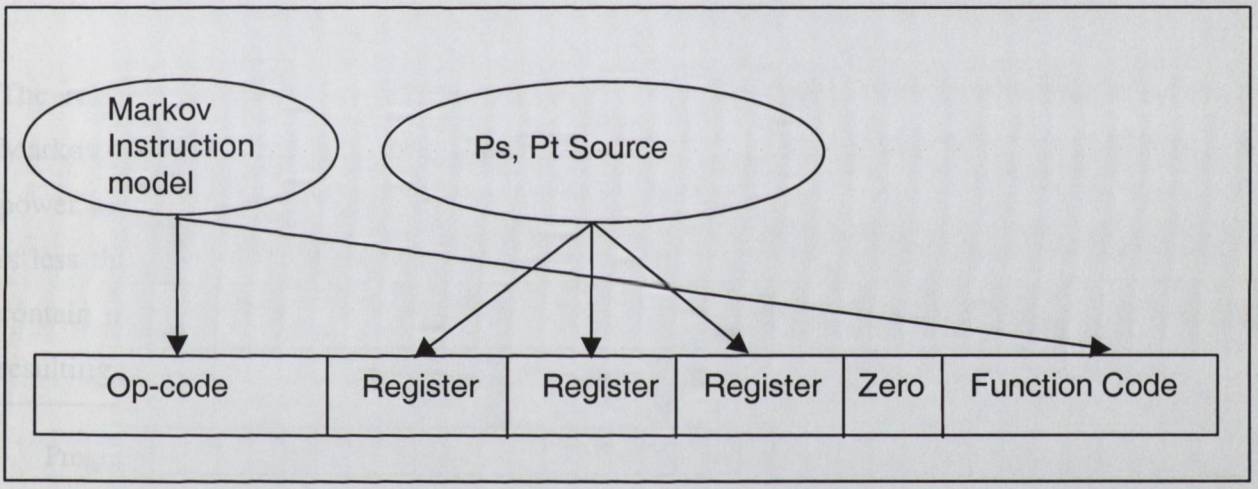


Figure 6.24: Instruction Level Model

Power estimates were taken for the synthesised datapath under three conditions: real program execution, operation with data from the combined model and operation with UWN input data. This methodology was applied to a number of programs of varying amounts of code. Each program was developed from sections of the larger audio processing routine previously referred to and consists of a mix of instruction types and operands. For each program a Markov model was developed and the three power estimates, referred to as real power, Markov power and UWN power were obtained. Figure 6.25 shows the results obtained for five such programs of varying length. In each case the model tracks the actual processor power with good accuracy.

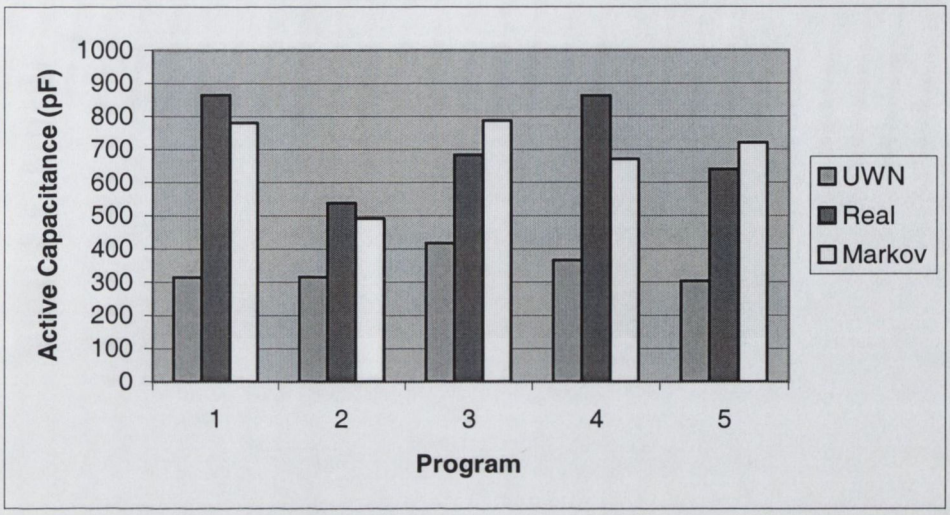


Figure 6.25: Processor Power Estimates

The results are summarised in Table 6.9 where a maximum relative error between the Markov and Real estimates of 22% is reported, with satisfactory agreement between the power estimate provided by the model and that of the real data. In all cases the UWN power is less than the real power. This is because a sequence of 32b UWN patterns is unlikely to contain many valid instructions and so the processor is under-exercised by the UWN input, resulting in lower power estimates.

Program	Number of Lines of Code	Active Capacitance (pF)			
		UWN	Real	Markov	Relative Error (%)
1	50	314.6	864.9	780.1	9.8
2	100	316.4	538.6	491.4	8.7
3	10	417.6	683.9	787.4	15.1
4	20	365.5	862.6	670.7	22.2
5	200	303.7	640.4	721.5	12.6

Table 6.9: System-level Power Estimates

Finally the power consumption for the audio application was estimated. This represents the long-term power dissipation of the processor when, for example, it is configured as an embedded system running the algorithm as a real-time application. It also represents a challenging application for the power estimation model for the following reason. The audio application requires tens of thousands of clock cycles to process even a small fragment of a real audio signal. This in turn means that the real program execution also consists of tens of thousands of lines of code, with each line requiring a clock cycle to execute. Thus a VHDL simulation of a significant portion of the real program operation would require months of

simulation which is not feasible. The model must therefore be verified by some means other than a direct comparison with a simulation of real data. The method chosen uses a theoretical calculation of the actual or real processor power, which is used to verify the Markov estimate. This approach is described below. The Markov model parameters for this program are based on a realistically long sample of 5000 lines of code. During such a long sample, all loops in the code are visited and all instruction types are represented in sufficient quantities to provide a meaningful long-term power estimate. The Markov properties are therefore a good reflection of the real program parameters from a power perspective. Simulations of varying lengths, up to the practical limit of the simulation environment, were then performed for the three data types of UWN, real data and Markov data. The power estimates resulting from the inclusion of segments of real data are also given for comparison. The power estimates provided from such a small fraction of the real program completely underestimate the long term power dissipation as provided by the theoretical calculation. This is to be expected as for example after 500 clock cycles of real program operation only a handful of instructions have been used due to a loop at the beginning of the code. The operands too are very small as the first audio samples, representing the initial portion of the sound envelope, are of low magnitude. These factors result in power estimates from the real data, which are of a similar magnitude to the UWN estimates, which possibly may contain no valid instructions at all.

The approximate theoretical estimate of the power consumption was calculated using the relative frequencies of each instruction calculated from a long sample and the power consumption associated with a single execution of each instruction type. This exercise yielded a figure of approximately 1700pF, which will be used as the reference figure for the long term real power dissipation of the processor. Markov, real and UWN estimates were then performed with a typical sample size of 400. The resulting power estimates are indicated in Table 6.9. The theoretical and Markov values agree while the UWN and real values are considerably lower. As indicated above, the real value is such a small fraction of the actual program that it cannot possibly provide a meaningful estimate while the UWN value does not use real instructions and is therefore also invalid.

Data Source	Active Capacitance (pF)	Relative Error (%)
Theoretical Estimate	1700	0
Markov	1621	4.6
UWN	298	82.4
Real	484	71.5

Table 6.10: Power Estimates for Audio Program

The Markov model has an error when compared with the theoretical estimate, of approximately 4% which provides sufficiently close agreement. Both the UWN estimate and the real estimate fail to accurately predict the long-term power dissipation because neither adequately reflects the underlying large scale distributions of instructions that appear when sample sizes of tens of thousands are considered.

The behaviour of the model was further explored by varying the sample size up to the limit of practical simulation times. The results are given in Figure 6.26 where it is observed that a minimum sample size of 300 is required for satisfactory results from the Markov model. Below this number of input samples, the error is significant. For both UWN data and real segments of the program the increase in sample size does not improve the estimate, up to the limit of realistic simulation times.

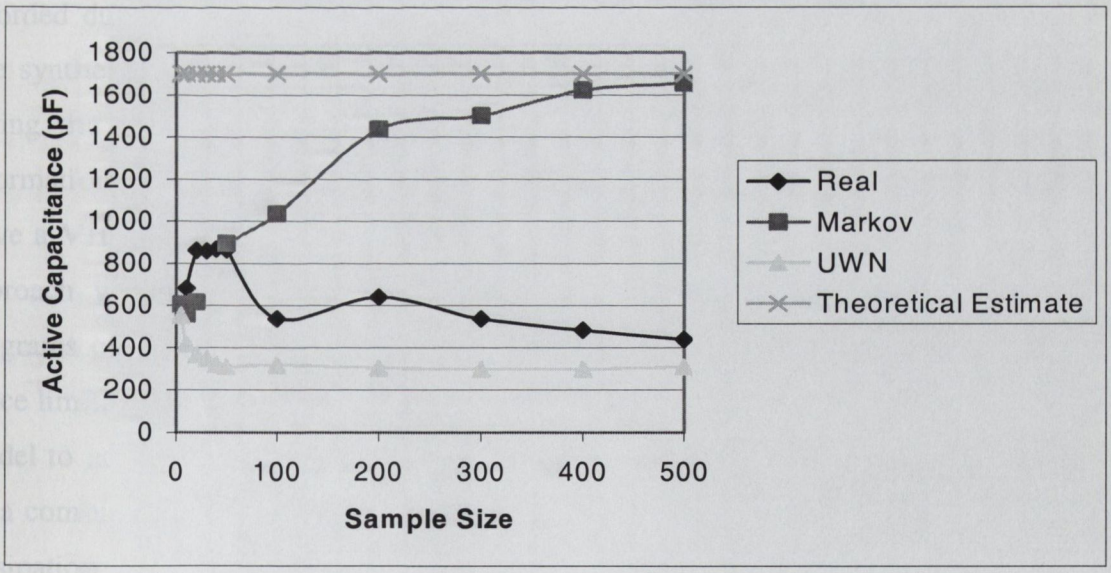


Figure 6.26: Long-term Processor Power Estimates

While no great significance may be attached to the approximate theoretical calculation of power, it is nonetheless significant that the Markov power is of the same order of magnitude. A simulation with real data containing 5000 vectors would require approximately 50 days of

simulation time and is therefore not practical. Some suggestions to address this issue in future work are proposed in the next Chapter.

6.5 Summary and Conclusions

A case study in power analysis was performed using a processor design as the vehicle. A number of application programs were written, to allow the collection of samples of real signals during the program operation. The data was then modelled using the various methods discussed in earlier chapters. A combination model employing all earlier work was also presented for capturing the behaviour of the instruction stream. Results for power estimation show that the techniques operate satisfactorily, while providing an efficient method of generating system and module data. The power estimates were taken at three levels of the design hierarchy, the system level, datapath-control level and at the level of datapath submodules. At all levels the appropriate technique was applied to a synthesised circuit with satisfactory and conclusive results. The instruction sequence was shown to be Markov, allowing the characteristics of the instruction memory to datapath interface to be captured and reproduced easily. Three power estimates were taken for the datapath, the actual power recorded during program operation, the synthetic power and a UWN estimate for reference. The synthetic data sequences were synthesised by first recording all relevant signal traces during the execution of the algorithm. This data was then analysed for the statistical information required to construct an appropriate model. Synthetic data streams were used to drive a VHDL simulation while the resulting power dissipation was estimated. The general approach was demonstrated to be a useful technique for efficient power estimation for programs of varying sizes. The ability to include constraints on processor operation, address space limitations and data correlation is easily incorporated into the model. The ability of the model to adequately predict the long-term power dissipation of the processor was indicated by a combination of simulation results and a first order theoretical calculation of the power dissipation.

As the work presented in this Chapter is entirely novel and not a development of other published studies, the overall context of the work and the conclusions to be drawn from it are discussed in the next Chapter.

7 Conclusions

This Chapter presents a synopsis of the work undertaken, a summary of the results and conclusions reached as a consequence. The specific conclusions consist of those findings that may be deduced directly from an analysis of the simulation studies undertaken in earlier chapters. Additionally, general conclusions are drawn from the work, which are intended to be of use to the wider community of researchers currently active in the fields of CMOS power estimation, data modelling and sequence compaction. The final Section addresses those conceptual discontinuities caused by the finite nature of such a thesis by indicating how future research in this area might be directed.

7.1 Specific Conclusions

In Chapter 1, it was stated that this thesis aimed "...to investigate the application of power estimation techniques in a comprehensive and systematic manner, using three stated goals of adaptability, efficiency and relevance". These were defined as the ability to encompass and traverse system hierarchy when estimating power, the compaction of data patterns so as to minimise the computation required to achieve such estimates and the production of models which realistically reflect the circuit conditions under which the system will operate. This latter objective is a key concept which underpins the entire work. The simulation speeds resulting from realising the first two objectives are significant and when incorporated with the production of meaningful sequences of actual code result in a significant development in CMOS power estimation. Therefore, a system model for compacting the input data sequence to a typical processor was developed, without prejudice to the general case. This model provided a sequence of input data for each field of the complete instruction set. Based on the observation that the flow of instructions between memory and processor could be modelled by a Markov process, a synthetic stream of valid instructions may be produced. This has the advantage in the early stages of the system design that no software tools for the new system need to be developed, resulting in useful power estimates early in the design cycle. Other portions of the data flow to the processor were modelled using the statistics of the binary data, specifically the bit transition and signal probabilities. The relevance of this model to real

software applications was demonstrated by the calculation of the long-term power dissipation for the system, which was then accurately obtained using the combined Markov model, while the real data required an unrealistically large sample size to provide the same result. Several smaller programs of varying size were also successfully modelled with power predictions from the model typically within 15% of the actual power dissipation. When no such program data is available the technique allows a general exploration of the likely power dissipation associated with a candidate architecture, rapidly leading to an accurate estimate of the maximum power dissipation together with the instruction mix that produced it. This valuable information may then be employed to selectively minimise the power dissipation of heavily utilised blocks during the most common instructions. Thus the model is of general application for power exploration, becoming an accurate prediction technique with a minimum of information regarding the intended functionality.

To arrive at this outcome several additional concepts were explored. Firstly a reliable reference standard for both the data model and the power estimates was required. The use of uniform white noise in this regard is well established. However, the production of such data was often regarded as a potential source of error. Therefore, a novel method for the production of UWN generators was developed with an *a priori* determination of the data quality. This technique employed a genetic algorithm to produce parameters for the linear congruential equation, with the spectral test as an objective function. The resulting binary vectors were extensively tested and a new measure of data quality, based on the statistical properties of the power spectral density was proposed. The data generation technique was used throughout the work, providing a reference point by which power estimates might be compared. To demonstrate the application of the genetic UWN source and to obtain preliminary power estimates of CMOS circuits at the layout level, a Monte Carlo based tool, llama, was developed. A significant new approach to the monitoring of the power estimates of transistor level circuits was also proposed. This involved the estimation of the energy consumed during each simulation cycle. Simulation studies revealed that the energy dissipated on each cycle is normally distributed when the circuit is driven by UWN. It was also recognised that after the first iteration through a hardware design cycle there is a considerable amount of information regarding the power performance of a submodule. These two observations were combined to provide a novel approach to early power estimation. In

particular, incremental designs were shown to be capable of an efficient and early estimate of power by monitoring the convergence of the confidence intervals for the simulation.

Next the methodology was explored for adaptability, initially with regard to modelling input data streams to processor submodules such as the ALU, multiplier and register file. A model based on the signal statistics of transition and signal probability was found to be sufficient at this level. Power estimates so obtained corresponded closely with those produced from real data and therefore the technique was incorporated into the general model. Results were produced from different stages of the design process, ranging from full custom adder and multiplexer circuits to synthesised standard cell netlists. In all cases the probability model was found to be a useful technique at the lower levels of the system design.

Ascending to the next level of system complexity required a means of dealing with the datapath and control interface of the processor. The Markov process was found to be a very useful approach to modelling this interaction. Analysis of the instruction stream demonstrated that the pattern of instruction execution over time was a Markov process. This allowed the construction of a synthetic model of the instruction sequence for a given set of program statistics. The development of a VHDL testbench based on a real set of assembly language instructions for a new processor was a considerable but necessary undertaking, if the power performance of a processor is to be determined. Therefore, a technique was developed whereby such a set of instructions could be obtained with minimum effort, given the basic instruction set of the processor and some general information about the mix of instructions to be used. Initially simple adder circuits were used as dummy loads for a comparative test of the Markov and the real program data. The results were significant with close agreement between the two sets of input data. This technique was incorporated into the overall strategy for the development of an adaptable efficient power estimation technique, resulting in a combined input data model for the processor.

The final aspect of the work consisted of the development and test of a combined data model for processor power estimation. A RISC processor design was completed and synthesised as a case study for the technique. A prototype instruction set for the processor was developed, based on academic models previously developed. The instruction set was then modelled by a combination of the data models described in previous chapters. For each

instruction, bit patterns for the fields of the instruction were provided by one of the methods developed during this work. For example, the 6-bit op-code field was generated by sampling an appropriate Markov process while the 5-bit register fields were represented using an appropriate transition probability or UWN bit stream. Data and address fields were simulated using a statistical model based on a high level model of the processor busses. Testing of this approach took place in two phases. Firstly, Markov models were extracted from sample programs of varying length, representative of embedded processor applications, in that they were short repeating segments of code. Using the extracted model parameters, samples of synthetic code were produced. UWN input data samples were also prepared, giving three different types of input data. For such programs, there was good agreement between the power estimates from the real data and the combined Markov model. As expected, the UWN estimates contained fewer valid instructions and so the resulting power dissipation was considerably lower than for the other two as the processor is under exercised by a sequence of invalid input patterns. An audio processing algorithm was coded to provide a typical example of the system operation. Using a high level model of this program running in RTL-level VHDL, a large sample of code was quickly analysed to produce the required Markov parameters such as the transition matrix and the relative probabilities of the instructions executed. Using the prototype instruction set, three types of input data, real, Markov and UWN were produced. Due to the reduced simulation speed of the synthesised circuit, sample sizes were practically limited to several hundred input samples or lines of code. An approximate theoretical estimate of the long-term power dissipation when running for thousands of cycles was also calculated. There was agreement between the Markov prediction of the power and the theoretical calculation of the dissipation, giving a reasonable estimate of the steady state power dissipation after only several hundred cycles. Simulation with real data would have required thousands of such simulation cycles to adequately provide the processor with the same long-term statistical characteristics. For example, in real program execution, loops may require hundreds of clock cycles, executing only a few instructions during that period. Other less frequent instructions may occur in the instruction stream only after thousands of cycles. These statistical relationships are present in the synthetic data while requiring less cycles. Thus the method also provides a useful speed-up when estimating the steady-state power dissipation of large systems.

7.2 General Conclusions

When considering the important problem of power estimation for CMOS circuits and systems, there are two useful approaches to the task of providing speedy estimates with the required confidence and accuracy. The first focuses on the input sequence to the simulation model and seeks to provide meaningful patterns in an energy sense, such that the power estimate reflects the actual power dissipation under normal operating conditions. The second approach examines the simulation model of the circuit and attempts to minimise the processing required to model the power conscious aspects of the system operation. This work concerns itself mainly with the first aspect, that of the input sequence. As the power dissipation in CMOS circuits is always pattern-dependent, a successfully formulated input sequence captures only those characteristics that determine the long-term or operational power dissipation. The obvious initial choice of input sequence is a sample of real input data from, for example, a high-level functional simulation of the system in its environment. That this choice is often incorrect is a significant, if counter-intuitive, conclusion of this work. It represents the first general conclusion of this work. Such patterns reflect only the transient, local conditions from a power perspective and cannot produce steady state or global estimates of power consumption. The identification and synthesis of appropriate global input characteristics were proposed and verified for a generic processor. The appropriateness of this technique of modelling the instruction flow with a combined Markov and probability model provides a novel approach to the extraction of the most important power characteristics of the input sequence. Its significance therefore represents a second general conclusion to this work.

A second aspect of the input sequence as an important factor in the power estimation process attempts to produce speedier estimates by compacting the sequence. One simple measure of this is to compare the amount of synthetic data required to obtain the estimate produced using real data. In this regard the model proposed in this work incorporates a natural compression by requiring a considerably smaller data set to estimate the operational power dissipation of a processor when compared with the sample of real data that would be needed to get the same result. The conclusion was reached that when a model of input data captures the power performance in the way the combined Markov model was observed to do, time efficiency and sequence compaction result. This is because much larger samples of real

program code are required to provide data with the same global characteristics. In particular, for the case study undertaken, the real data applied to the processor considerably underestimated the power dissipation as verified by a theoretical calculation of power consumption. Additionally, the use of an input data model can result in reduced amounts of development time. To test a processor design requires code typical of the intended application. However, this code need not reflect the functionality of the system when power estimation is the objective. This work also demonstrated that by replacing time consuming software development with a code model targeted at power consumption, the design cycle can be speeded up considerably. This approach also allows the designer to explore architectural alternatives within the system, by observing the effect on power dissipation of altering the model parameters. Thus power intensive code sequences may be identified, enabling a more targeted design of individual modules within the system.

7.3 Future Work

Consideration of future directions for this work fall into two categories according to the scale of the idea. The most significant large scale challenge concerns the development of a high level simulation model for processor systems. The work undertaken in this thesis has underlined the limits of the post-synthesis circuit when estimating power. The amount of time required to simulate such circuits is extremely large due to the nature of event driven functional simulators such as Synopsys. The combination of a compact input data model as developed in this work with a speedy high level system model, represents the next step in the development of power estimation techniques for VLSI systems.

Further enhancements of the Markov approach to data modelling should be possible. In particular, higher order models may be possible that incorporate the interaction between adjacent instructions. By analogy with simple combinational circuits where the transition between input vectors is the prime determinant of power dissipation, so too could the transition between instructions be of importance when estimating the power. Markov models with higher orders are possible, whereas a first order model was applied in this work.

The related systems design issues of hot software, instruction design for low power and low power processor design are currently active research topics that would benefit from the work undertaken here, particularly if a high level processor power model was investigated.

In terms of small scale ideas, the novel application of a genetic algorithm to the production of UWN generators suggests the investigation of other forms of optimisation. In particular, simulated annealing might prove to be an interesting candidate. Other approaches might include the use of cellular automata for pattern generation while the work of the Russian mathematician Illya Sobol [Sobo94], who advocated highly deterministic sequences with a measure termed low-discrepancy, could also be of interest.

At the lower circuit levels of the design, modules such as standard cells and customised modules are routinely tested with UWN sources in a Monte Carlo configuration. The observation made during this thesis that the energy distribution of a single clock cycle is often normally distributed and that the confidence intervals of the simulation follow a dynamic trajectory on an energy axis may allow a speedier approach to this type of power estimation. This aspect of the work could well reward further investigation.

In conclusion, the extent of the overall degree of success of this work in providing a new approach to CMOS power estimation may be critically evaluated in a number of aspects. The application of Markov models to the instruction sequence has proved to be a very interesting approach, which should reward further investigation. In addition, the proposal for a combined model is novel and has given good results for the systems investigated. The work undertaken thus far allows for the inclusion of other possibly more effective components within such a combined model. In particular, this work utilised a first order Markov model while higher orders are possible and might prove useful. The verification of small program sizes was demonstrated adequately while larger programs, given the current speeds of VHDL simulation, were slightly more problematic. A future approach to this issue was suggested, in the form of a high level power estimation tool. Overall, it might be reasonably asserted that a novel and significant new avenue to the goal of fast and accurate high level power estimation has been opened up, leaving it to future researchers to provide a completely smooth surface.

8 References

[Akers78] S. Akers, "Binary decision diagrams, *IEEE Transactions on Computers*," vol. c-27, June 1978, pp. 509-516.

[AMS02] *http://www.vhdl-ams.com*

[Atki80] A. Atkinson, "Tests of pseudo-random numbers," *Applied Statistics*, vol. 29, no. 2, February 1980, pp. 164-171.

[Baas99] B. Baas, "A low-power, high-performance, 1024-point FFT processor," *IEEE Journal of Solid State Circuits*, vol. 34, no. 3, March 1999, pp. 380-387.

[Bella95] A. Bellaouar and M. Elmasry, *Low-Power Digital VLSI Design, Circuits and Systems*, Kluwer Academic, Boston, 1995.

[Beni99] L. Benini, G. DeMicheli, A. Macii et al., "Automatic selection of instruction op-codes of low-power core processors," *IEE Proc. Comput. Digit. Tech.* vol. 146, no. 4, July 1999, pp. 173-178.

[Beni00] L. Beni L., G. DeMicheli and E. Macii, "A multilevel engine for fast power simulation of realistic input streams," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 4, April 2000, pp. 1118-1124.

[Bhar01] M. Bhardwaj and A. Chandrakasan, "Quantifying and enhancing power awareness of VLSI systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 6, December 2001, pp. 757-771.

[Bind88] K. Binder and D. Heermann, *Monte Carlo Simulation in Statistical Physics*, 2nd edition, Springer Verlag, Berlin, 1988.

[Birn52] Z. Birnbaum, "Numerical tabulation of the distribution of Kolmogarov's statistic for finite sample size," *Journal of the American Statistical Association*, vol. 47, 1952, pp. 425-441.

[Blair94] G. Blair, "Designing low-power digital CMOS," *Electronics and Communication Engineering Journal*, October 1994, pp. 229-236.

[Bori83] J. Borish and J. Angell, "An efficient algorithm for measuring the impulse response using pseudo-random noise," *Journal of the Audio Engineering Society*, vol. 31, no. 7, August 1983, pp. 478-487.

[Bowm95] R. Bowman, "Evaluating pseudo-random number generators," *Comput. and Graphics*, vol. 19, no. 2, 1995, pp. 315-324.

[Brac87] L. Brackenbury, *Design of VLSI Systems, A Practical Introduction*, MacMillan, London, 1987.

[Brya86] R. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Transactions on Computer-Aided Design*, August 1986, pp. 677-691.

[Burc93] R. Burch, F. Najm, P. Yang et al., "A Monte Carlo approach for power estimation," *IEEE Transactions on Very Large Scale Integartion (VLSI) Systems*, vol. 1, no. 1, March 1993, pp. 63-71.

[Burd94] T. Burd., *Low-Power CMOS Library Design Methodology*, M.Sc. Thesis, University of California, 1994.

[Burd02] T. Burd, *private communication.*

[Cham67] R. Chambers, "Random number generation," *IEEE Spectrum*, February 1967, pp. 48-56.

[Chan92] A. Chandrakasan, S. Sheng and R. Broderson., "Low-power CMOS digital design," *IEEE Journal of Solid State Circuits*, Vol.27, no. 4, April 1992, pp. 473-483.

[Chan95a] A. Chandrakasan, M. Potkonjak, R. Mehra et al., "Optimising power using transformations," *IEEE transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 14, no. 1, January 1995, pp. 12-30.

[Chan95b] A. Chandrkasan and R. Broderson, "Minimising power consumption in digital CMOS circuits," *Proceedings of the IEEE*, vol. 83, no. 4, April 1995, pp. 498-523.

[Chan96] J. Chang and M. Pedram, "Module assignment for low power," *Proceedings of EuroDAC-96:IEEE European Design Automation Conference*, Geneva, 1996.

[Chen00] Z. Chen, K. Roy and E. Chong, "Estimation of power dissipation using a novel power macromodelling technique," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol.19, no. 11, November 2000, pp. 1363-1367.

[Chil97] D. Childers, *Probability and Random Processes*, Irwin, Chicago, 1997.

[Chou94] T. Chou, K. Roy and S. Prasad, "Estimation of circuit activity considering signal correlations and simultaneous switching," *ACM/IEEE International Conference on Computer-Aided Design, November 1994*, pp. 300-303.

[Chou96] T. Chou T. and K. Roy, "Accurate estimation of power dissipation in CMOS sequential circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 1996, pp. 369-380.

[Conte00] T. Conte, K. Menzes, S. Sathaye et al., "System-level power consumption and modelling and tradeoff analysis techniques for superscalar processor design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 2, April 2000, pp. 129-138.

[Cove67] R. Coveyou and R. MacPherson, "Fourier analysis of uniform random number generators," *Journal of the ACM*, vol. 14, no. 1, January 1967, pp. 100-119.

[Dag88] J. Dagpunar, *Principles of Random Variate Generation*, Clarendon Press, London, 1988.

[Darm02] *http://www.rs.e-technik.tu-darmstadt/TUD/res/dlxdocu/SuperscalarDLX.html.*

[Deng88] A. Deng, Y. Shiayi and K. Loh, "Time-domain current waveform simulation of CMOS circuits," *IEEE International Conference on Computer-Aided Design*, Santa Clara California, 1988, pp. 208-211.

[Diet75] U. Dieter, "How to calculate the shortest vectors in a lattice," *Mathematical Computing*, vol. 29, 1975, pp. 78-84.

[Ding98a] C. Ding and C. Tsui, "Gate-level power estimation using tagged probabilistic simulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 11 November 1998, pp. 135-142.

[Ding98b] C. Ding, Q. Wu, C. Hsieh et al., "Stratified random sampling for power estimation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 6, 1998, pp. 465-471.

[Ding00] C. Ding, C. Hsieh and M. Pedram, "Improving the efficiency of Monte Carlo power estimation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 5, October 2000, pp. 584-593.

[Espe02] Esperan, *VHDL Application Workshop Manual*, 2002

[ES296] European Silicon Structures, *ECPD07 Library Databook*, 1996, Aix-en-Provence, France.

[Eyre01] J. Eyre, "The digital signal processing derby," *IEEE Spectrum*, June 2001, pp. 62-68.

[Feld94] J. Feldman and C. Retter, *Computer Architecture, A Designer's Text Based On A Generic RISC*, McGraw-Hill, New York, 1994.

[Ferra97] F. Ferrandi, "Power estimation of behavioral descriptions," *DATE-98: IEEE Design, Automation and Test in Europe*, France 1998, pp. 75-85.

[Freu90] J. Freund, *Probability and Statistics for Engineers*, Prentice Hall, New York, 1990.

[Gibb96] W. Gibbs, "Programming with primordial ooze," *Scientific American*, October 1996, pp. 34-35.

[Giva01] T. Givargis, F. Vahid and J. Henkel, "Evaluating power consumption of parameterised cache and bus architectures in system-on-a-chip designs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 4, August 2001, pp. 500-508.

[Gold76] E. Golder, "The spectral test for the evaluation of linear congruential pseudo-random number generators," *Applied Statistics*, vol. 25, 1976, pp. 46-51.

[Gold89] D. Goldberg, *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison-Wesley, Reading, Mass.,1989.

[Gold94] D. Goldberg, "Genetic and evolutionary algorithms come of age," *Communications of the ACM*, vol. 37, no. 3, 1994, pp. 113-119.

[Gold97] A. Goldstein, "Tackling tyranny, a profile of Jack Kilby," *The Solid-State Century*, Scientific American Publications, 1997, pp. 62-71.

[Gold01] H. Goldstein, "Packages go vertical," *IEEE Spectrum*, August 2001, pp. 46-51.

[Gonz77] T. Gonzalez, S. Sahni and W. Franta, "An efficient algorithm for the Kolmogarov-Smirnov and Lilliefors tests," *ACM Transactions on Mathematical Software*, vol. 3, no. 1, March 1977, pp. 60-64.

[Gonz96] R. Gonzalez and M. Horowitz, "Energy dissipation in general purpose microprocessors," *IEEE Journal of Solid State Circuits*, vol. 31, no. 9, September 1996, pp. 1277-1283.

[Gupt97] S. Gupta and F. Najm, "Power macromodelling for high-level power estimation," *Proceedings of DAC-34:ACM/IEEE Design Automation Conference*, California 1997, pp. 365-370.

[Gupt00a] S. Gupta and F. Najm, "Analytical models for RTL estimation of combinational and sequential circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 7, July 2000, pp. 808-814.

[Gupt00b] S. Gupta and F. Najm, "Power modelling for high-level power estimation," *IEEE Transactions On Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 1, February 2000, pp. 18-29.

[Hamb02] *http://tech-www.infomatic.uni-hamburg.de/vhdl/models*

[Hamm67] J. Hammersley and D. Handscomb, *Monte Carlo Methods*, Methuen, London, 1967.

[Haye96] M. Hayes, *Statistical Digital Signal Processing and Modelling*, Wiley, New York, 1996.

[Haye98] J. Hayes, *Computer Architecture and Organisation*, McGraw-Hill, New York, 1998.

[Heal00] R. Heald, K. Aingaran, C. Amir et al., "A third-generation SPARC V9 64-b microprocessor," *IEEE Journal of Solid-State Circuits*, pp. 1526-1535.

[Henn96] J. Hennessy and D. Patterson, *Computer Architecture, A Quantitative Approach*, 2nd edition, Morgan Kaufmann, San Francisco, 1996.

[Holl75] J.H. Holland, *Adaption in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, USA, 1975.

[Houc97] C. Houck, J. Joines and M. Kay, "A genetic algorithm for function optimisation: a Matlab implementation," *Technical Report NCSU-IE, North Carolina State University*.

[Hsia00] M. Hsiao, E. Rudnick and J. Patel, "Peak power estimation of VLSI circuits: new peak power measures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol.8, no. 4, August 2000, pp. 435-439.

[Hsie97] C. Hsieh and M. Pedram, "Profile-driven program synthesis for evaluation of system power dissipation," *Proceedings of DAC-34, ACM/IEEE Design Automation Conference*, California, 1997, pp. 1080-1089.

[Hsie98] C. Hsieh and M. Pedram, "Microprocessor power estimation using profile-driven program synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 11, November 1998, pp. 576-581.

[Huiz90] C. Huizer, "Power dissipation analysis of CMOS VLSI circuits by means of switch-level simulation," *IEEE Eur. Solid State Circ. Conf.,* Grenoble France,1990, pp. 61-64.

[Hull62] T. Hull and A. Dobell, "Random number generators," *SIAM Review*, vol. 4, no. 2, July 1962, pp. 230-254.

[Jaeg97] R. Jaeger, *Microelectronic Circuit Design*, McGraw-Hill, New York, 1997.

[Jian00] Y. Jiang, A. Krstic and K. Cheng., "Estimation for maximum instantaneous current through supply lines for CMOS circuits", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 1, February, 2000, pp. 61-73.

[Jone01] W. Jones, "EEs' gifts for the holidays," *IEEE Spectrum,* December 2001, pp. 48-53.

[Kain96] R. Kain, *Advanced Computer Architecture*, Prentice Hall, 1st edition, 1996.

[Kang96] S. Kang and Y. Lablebici, *CMOS Digital Integrated Circuits, Analysis and Design*, McGraw-Hill, New York, 1996.

[Karp83] K. Karplus and A. Strong, "Digital synthesis of plucked string and drum timbres," *Computer Music Journal*, vol. 7, no. 2, Summer 1983, pp. 43-55.

[Karr91] C.L. Karr, "Genetic algorithms for fuzzy controllers," *AI Expert*, vol. 6, no. 2, 1991, pp. 26-33.

[Kart76] S. Kartalopoulos, "Response time measurement using Walsh functions," *IEEE Transactions on Instrumentation and Measurement*, vol. 25, no. 2, June 1976, pp. 116-119.

[Katz94] R. Katz, *Contemporary Logic Design*, Benjamin Cummings, Redwood, Calif., 1994.

[Kenn87] J. Kennedy and A. Neville, *Basic Statistical Methods For Engineers And Scientists*, 3rd edition, Harper and Row, New York, 1987.

[Kirk83] S. Kirkpatrick, C. Gelatt and M. Vecchi, "Optimisation by simulated annealing," *Science*, vol. 220, no. 4598, May 1983, pp. 671-680.

[Kozh01] J. Kozhaya and F. Najm, "Power estimation for large sequential circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 2, April 2001, pp. 400-407.

[Knut81] D. Knuth, *The Art of Computer Programming*, Volume 2, Semi-numerical Algorithms, 2nd edition, Addison-Wesley, Reading, Mass., 1981.

[Kolm33] A. Kolmogarov, "On the empirical determination of a distribution law", *Journal of Italian Actuarial Institute*, vol. 4, 1933, pp. 1-11.

[Komo91] J. Komo and W. Park, "Decimal pseudo-random number generator," *Simulation*, October 1991, pp. 228-230.

[Kouk01] A. Koukab, P. Deval, C. Dehollain et al., "Low noise power supply technique for RF amplifiers," *IEEE Transactions on Circuits and Systems II: Analog and Digital Processing*, vol. 48, no. 3, March 2001, pp. 291-296.

[Krau95] T. Krauss, *Signal Processing Toolbox*, Mathworks, 1st edition, 1995.

[Kris89] B. Krishnamurty and I. Tollis, "Improved techniques for estimating signal probabilities," *IEEE Transactions on Computers*, vol. 38, no. 7, July 1989, pp. 1041-1045.

[Krus01] L. Kruse, E. Schmidt, G. Jochens et al., "Estimation of lower and upper bounds on the power consumption from scheduled data flow graphs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 1, February 2001, pp. 3-14.

[Kuma95] N. Kumar, S. Katoori, L. Rader et al., "Profile-driven behavioral synthesis for low-power VLSI systems," *IEEE Design and Test of Computers*, vol. 12, no. 3, 1995, pp. 70-84.

[Laks99] G. Lakshiminarayana and A. Raghunathan, "Power management in high-level synthesis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 1, March 1999, pp. 7-13.

[Land93] P. Landman and J. Rabaey, "Power estimation for high-level synthesis," *EDAC-93: IEEE European Conference on Design Automation*, Paris 1993, pp. 361-366.

[Land94] P. Landman and J. Rabaey, "Black-box capacitance models for architectural power analysis," *Proceedings of the International Workshop on Low Power Design*, California, 1994.

[Land95] P. Landman and J. Rabaey, "Activity-sensitive architectural power analysis for the control path," *Proceedings of ISLPD-95:ACM/IEEE International Symposium on Low-Power Design*, California, 1995, pp. 93-98.

[Lecu90] P. Lecuyer and S. Tezuka, "Structural properties for two classes of combined random number generators," *Mathematics of Computation*, vol. 57, no. 196, October 1991, pp. 735-746.

[Lee75] C. Lee, "Representation of switching circuits by binary decision diagrams," *Bell System Technical Journal*, vol. 38, July 1959, pp. 985-999.

[Lehm51] D. Lehmer, "Mathematical methods in large-scale computing units," *Annual Report of the Computing Laboratory*, vol. 26,Harvard 1951, pp. 141-146.

[Leme98] C. Leme, J. Silva, P. Rodrigo et al., "A low-power CMOS nine-channel 40MHz binary detection system with self-calibrated 500μV offset," *IEEE Journal of Solid State Circuits*, vol. 33, no. 4, April 1998, pp. 565-571.

[Levi02] P. Levin and R. Ludwig, "Crossroads for mixed-signal chips," *IEEE Spectrum,* March 2002, pp. 38-43.

[Liu94] D. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 6, June 1994, pp. 10-17.

[Maci96] G. Macii and M. Poncino, "Estimating power consumption of CMOS circuits modelled as symbolic neural networks," *IEE Proc. Comput. Digit. Tech.*, Vol 143, no. 5, September 1996, pp. 331-337.

[Maci98] E. Macii and M. Pedram, "High level power modelling, estimation and optimisation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 11, November 1998, pp. 1061-1079.

[Maci01] A. Macii, E. Macii M. Poncino et al., "Stream synthesis for efficient power simulation based on spectral transforms," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 3, June 2001, pp. 417-426.

[Marc96] R. Marculescu, D. Marculescu and M. Pedram, "Information theoretic measures for power analysis," *IEEE Transactions on Computer-Aided Design*, vol. 15, no. 6, 1996.

[Marc99] R. Marculescu, D. Marculescu and M. Pedram, "Sequence compaction for power estimation: theory and practice,"*IEEE Transactions on Computer-Aided*

Design of Integrated Circuits and Systems, vol. 18, no. 7, July 1999, pp. 973-993.

[Marc00] D. Marculescu, R. Marculescu and M. Pedram, "Theoretical bounds for switching activity analysis in finite-state machines," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 3, June 2000, pp. 335-339.

[Mars68] G. Marsaglia, "Random numbers fall mainly in the planes, *Proceedings of the National Academy of Science*, vol. 61, September 1968, pp. 25-28.

[Mars94] G. Marsaglia and A. Zaman, "Some portable very long-period random number generators," *Computers in Physics*, vol. 8, no. 1, January/February 1994, pp. 117-121.

[Mass51] F. Massey, "The Kolmogarov-Smirnov test for goodness of fit," *Journal of the American Statistical Association*, vol. 46, 1951, pp. 68-78.

[Math96] *Matlab Reference Guide*, The Mathworks Inc., 1995.

[McCl99] J. McClellan, *DSP First, A Multimedia Approach*, Prentice Hall, New Jersey, 1999.

[Mehr94] R. Mehra and J. Rabaey, "Behavioral level power estimation and exploration," *Proceedings of the First International Workshop on Low Power Design*, April 1994.

[Micr96] *MicroSim PSpice Reference Manual*, MicroSim Corporation, 1996.

[Mill72] J. Millman and C. Halkias, *Integrated Electronics: Analog and Digital Circuits and Systems*, McGraw-Hill, New York, 1972.

[Mitr98] S. Mitra, *Digital Signal Processing, A Computer-Based Approach*, McGraw-Hill, New York, 1998.

[Mont94] J. Monteiro and S. Devedas, " A methodology for efficient estimation of switching activity in sequential circuits," *ACM/IEEE Design Automation Conference*, 1994, pp.131-138.

[Murp97] A. Murphy, S. Tsutumi and P. Gaussen, "A low-power, low-cost bipolar GPS receiver chip," *IEEE Journal of Solid State Circuits*, vol. 32, no. 4, April 1997, pp. 587-591.

[Nage75] L. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," *Memo ERL-M520*, Berkeley California,.: University of California, May 1975.

[Najm93] F. Najm, "Transition density: a new measure of activity in digital circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 3, February 1993, pp. 310-323.

[Najm94] F. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, December 1994, pp. 446-455.

[Najm 95] F. Najm, "Feedback, correlation and delay concerns in the power estimation of VLSI circuits," *32nd ACM/IEEE Design Automation Conference*, 1995.

[Namg00] W. Namgoong, S. Reader and T. Meng, "An all-digital low-power IF GPS synchroniser," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 6, June 2000, pp. 856-863.

[Nand95] K. Nand, "Profile-driven behavioral synthesis for low-power VLSI systems," *IEEE Design and Test of Computers*, Autumn 1995, pp. 70-84.

[Nema96] M. Nemani and F. Najm, "Toward a high-level power estimation capability," *IEEE Tranactions on Computer-Aided Design*, vol. 15, no. 6, 1996, pp. 588-589.

[Nema97a] M. Nemani and F. Najm, "High-level area prediction for power estimation," *Proceedings of CICC-97:Custom Integrated Circuits Conference*, California, 1997, pp. 483-486.

[Nema97b] M. Nemani and F. Najm, "High-level area and power estimation for VLSI circuits," *Proceedings of ICCAD-97:IEEE/ACM International Conference on Computer-Aided Design*, California, 1997, pp. 114-119.

[Nema99] M. Nemani and F. Najm, "High-level area and power estimation for VLSI circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, Vol.18, no. 6, June 1999, pp. 697-713.

[Niko00] S. Nikolaidis and E. Karaolis, "Estimation of signal transition activity in FIR filters implemented by a MAC architecture," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 1, January 2001, pp. 164-167.

[Nico01] N. Nicolici and B. Al-Hashimi, "Minimising power dissipation in partial scan sequential circuits," *IEE Proc. Comput. Digit. Tech.*, vol. 148, no. 4, July 2001, pp. 313-318.

[Nied92] H. Niederreiter, "Random number generation and quasi-Monte Carlo methods," *Society For Industrial and Applied Mathematics*, 1st edition 1992.

[Nose00] K. Nose and T. Sakurai, "Analysis and future trend of short-circuit power," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 9, September 2000, pp. 1023-1028.

[Oliv81] W. Oliver, "White noise loading of multi-channel communication systems," *Electronic Engineering*, November 1965, pp. 714-719.

[Park75] K. Parker and E. McCluskey, "Probabilistic treatment of general combinational networks," *IEEE Transactions on Computers*, June 1977, pp. 668-670.

[Park88] S. Park, "Random number generators: good ones are hard to find," *Communications of the ACM*, vol. 31, no. 10, October 1988, pp. 1192-1201.

[Patt98] D. Patterson and J. Hennessy, *Computer Organisation and Design*, 2nd edition, Morgan Kaufmann, San Francisco, 1998.

[Pick91] C. Pickover, "Picturing randomness on a graphics supercomputer," *IBM Journal Res. Development*, vol. 35, no. 1/2, January/March 1991, pp. 227-230.

[Pipp77] N. Pippenger, "Information theory and complexity of boolean functions," *Mathematical Systems Theory*, vol. 10, Springer-Verlag, 1997.

[Pirs98] P. Pirsch, *Architectures for Digital Signal Processing*, Wiley, New York, 1998.

[Powe90] S. Powell and P. Chau, "Estimating power dissipation of VLSI signal processing chips: The FA techniques," *Proceedings of IEEE Workshop on VLSI Signal Processing*, 1990.

[Pric97] K. Price and R. Storn, "Differential evolution," *Dr. Dobb's Journal*, April 1997, pp. 18-24.

[Qing01] W. Qing, "Estimation of peak power dissipation in VLSI circuits using the limiting distributions of extreme order statistics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 8, August 2001, pp. 778-789.

[Qiu01] Q. Qiu, Q. Wu and M. Pedram, "Stochastic modelling of a power-managed system, construction and optimisation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 10, October 2001, pp. 1200-1211.

[Raba95] J. Rabaey, L. Guerra and R. Mehra, "Design guidance in the power dimension," *ICASSP*, May 1995.

[Raba96] J. Rabaey, *Digital Integrated Circuits, A Design Perspective*, Prentice Hall, New Jersey, 1996.

[Ragh99] A. Raghunathan, S. Dey and N. Jha, "Register transfer level power optimisation with emphasis on glitch analysis and reduction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 9, August1999, pp. 1114-1130.

[Reiz91] M. Reizenmann, "*Wanlass's CMOS circuit*," IEEE Spectrum, May 1991.

[Robe82] C. Roberts, "Implementing and testing new versions of a good, 48-bit pseudo-random number generator," *Bell System Technical Journal*, vol. 61, no. 8, October 1982, pp. 2053-2063.

[Roy00] K. Roy and S. Prasad, *Low-Power CMOS VLSI Design*, Wiley Interscience, New York, 2000.

[San96] R. San Martin and J. Knight, "Optimising power in ASIC behavioral synthesis," *IEEE Design and Test of Computers*, vol. 13, no. 3, 1995.

[Sato95] T. Sato, "Evaluation of architecture-level power estimation for CMOS RISC processors," *Proceedings of the Symposium on Low Power Electronics*, 1995, pp. 132-139.

[Saty00] J. Satyanarayana and K. Parhi, "Power estimation of digital datapaths using HEAT," *IEEE Design and Test of Computers*, April 2000, pp. 101-110.

[Savi83] J. Savir, "A new empirical test for the quality of random integer generators," *IEEE Transactions on Computers*, vol. C-32, no. 10, October 1983, pp. 960-961.

[Sber97] M. Sbert, "Error and complexity of random walk Monte Carlo radiosity," *IEEE Transactions on Visualisation and Computer Graphics*, vol. 3, no. 1, January/March 1997, pp. 23-39.

[Schw01] A. Schwarzbacher, "*Algorithmic level low-power VLSI design applied to RGB to HSI conversion*," Ph.D. Thesis, Trinity College, Dublin, 2001.

[Scie97] *Scientific American*, Special Solid State Issue, 1997.

[Shan48] C. Shannon, "The synthesis of two-terminal switching circuits," *Bell Systems Technical Journal*, 1948.

[Sher93] Y. Sherif and R. Dear, "The development and testing of a new composite random number generator," *Computers Math. Applications*, vol. 25, no. 1, 1993, pp. 3-16.

[Sher95] Y. Sherif and R. Dear, "An efficient algorithm for testing the quality of the output of random number generators," *Advances in Engineering Software*, vol. 22, no. 5, 1995, pp. 69-77.

[Sher01] R. Sherwin, "Memory on the move," *IEEE Spectrum*, 2001, pp. 55-59.

[Simm01] T. Simunic and L. Benini, "Event-driven power management," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 7, July 2001, pp. 15-26.

[Simo00] T. Simon and A. Chandrakasan, "An ultra low-power adaptive wavelet video encoder with integrated memory," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 4, April 2000, pp. 572-581.

[Simu01] T. Simunic, L. Benini and G. DeMicheli, "Energy-efficient design of battery-powered embedded systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 1, February 2001.

[Sjoh97] S. Sjoholm and L. Lindh, *VHDL for Designers*, Prentice Hall, New York, 1997.

[Smir48] N. Smirnov, "Table for estimating the goodness of fit of empirical distributions," *Annals of Mathematical Statistics*, vol. 19, 1948, pp. 279-281.

[Smit71] C. Smith, "Multiplicative pseudo-random number generators with prime modulus," *Journal of the ACM*, vol. 18, no. 4, October 1971, pp. 586-593.

[Sobo94] I. Sobol, *A Primer for the Monte Carlo Method*, CRC Press, Boca Raton, 1994.

[Spec01] Editorial, "Wristwatch tells wearer's location," *IEEE Spectrum*, February 2001, pp. 83-84.

[Spie88] T. Spiegel, *Introduction to Statistics*, Schaum, New York 1988.

[Star90] T. Starkweather, D. Whitley and K. Mathias, "Optimisation using distributed genetic algorithms," *Proc. Parallel Problem Solving From Nature*, vol. 1, Springer-Verlag, 1990.

[Stei96] K. Steiglitz, *A DSP Primer with Applications to Digital Audio and Computer Music*, Addison Wesley, Reading, Mass., 1996.

[Syno02] Synopsys, *VHDL Compiler Reference Manual*, 2002.

[Tang96] K. Tang, K. Man, S. Kwong et al., "Genetic algorithms and their applications, *IEEE Signal Processing Magazine*," November 1996, pp. 22-37.

[Theo00a] G. Theodoris, S. Theoharis, D. Soudris et al., "Switching activity estimation under real gate delay using timed boolean functions," *IEE Proc. Comput. Digit. Tech.*, vol. 147, no. 6, November 2000, pp. 444-449.

[Theo00b] S. Theoharis, G.Theodoris, D. Soudris et al., "Accurate datapath models for fast RT-level power estimation," *IEE Proc. Comput. Digit. Tech.*, vol. 147, no. 4, July 2000, pp. 209-214.

[Tiwa96] V. Tiwari, "Instruction level power analysis and optimisation of software," *Journal of VLSI Signal Processing*, vol. 13, no. 2, 1996, pp. 223-238.

[Turg98] S. Turgis and D. Auvergne, "A novel macromodel for power estimation in CMOS structures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 11, November 1998, pp. 1090-1098.

[Tyag97] A. Tyagi, "Entropic bounds on FSM switching," *IEEE Transactions on VLSI Switching*, vol. 5, no. 4, 1997, pp. 1240-1248.

[Uchi96] T. Uchino, F. Minami, M. Murakate et al., "Switching activity analysis for sequential circuits using the boolean approximation method," *IEEE Symposium on Low Power*, 1996, pp. 79-84.

[Vatt95] I. Vattulainen and T. Ala-Nissila, "Mission impossible: find a random pseudo-random number generator," *Computers in Physics*, vol. 9, no. 5, September/October 1995, pp. 500-504.

[Veen84] H. Veendrick, "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *IEEE Journal of Solid-State Circuits*, vol. Sc-19, No.4 August 1984, pp. 468-473.

[Vini98] Y. Viniotis, *Probability and Random Processes For Electrical Engineers*, McGraw-Hill, New York, 1998.

[Wang00] J. Wang, W. Tseng and H. Li, "Low-power embedded SRAM with the current-mode write technique," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 1, January 2000, pp. 119-132.

[Weym01] L. Wehmeyer, M. Jain, S. Steinke et al., "Analysis of the influence of register file size on energy consumption, code size and execution time," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 11, November 2001, pp. 1329-1337.

[West93] N. Weste and K. Eshraghian, *Principles Of CMOS VLSI Design*, Addison Wesley, Reading Mass.,1993.

[Widr86] B. Widrow and S. Stearns, *Adaptive Signal Processing*, Prentice Hall, New York, 1985.

[Wolf94] W. Wolf, *Modern VLSI Design, A Systems Approach*, Prentice Hall International, New York, 1994.

[Wolf98] D. Wolfe and P. Comiskey, "Genetic optimisation for random vector generation," *DIT Internal Report*, Dublin Institute of Technology,1998.

[Wu01] Q. Wu, Q.Qiu and M. Pedram, "Estimation of peak power dissipation in VLSI circuits using the limiting distributions of extreme order statistics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 8, August 2001, pp. 942-955.

[Xake94] M. Xakellis and F. Najm, "Statistical estimation of the switching activity in digital circuits," *31st ACM/IEEE Design Automation Conference*, 1994, pp. 728-733.

[Xant00] T. Xanthopolous and A. Chandrakasan, "A low-power DCT core using adaptive bitwidth and arithmetic activity exploiting signal correlations and quantisation," *IEEE Journal of Solid-State Circuits*, Vol.35, no. 5, May 2000, pp. 740-750.

[Xie99] A. Xie and P. Beerel, "Accelerating markovian analysis of asynchronous systems using state compression," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 7, July 1999, pp. 869-884.

[Ye96] Y. Ye and K. Roy, "Energy recovery circuits using reversible and partially reversible logic," *IEEE Transactions on Circuits and Systems I*, vol. 43, no. 9, September 1996, pp. 769-778.

9 Authors Publications

P.A. Comiskey, A.Th. Schwarzbacher and J.B. Foley, "Power estimation in CMOS circuits using genetically optimised input patterns," *Electronic Systems and Devices Conference,* Bruno, Czech Republic, pp. 1-4, June 1998.

P.A. Comiskey, A.Th. Schwarzbacher and J.B. Foley, "Random binary vector generation and analysis using genetic optimisation," *Irish Systems and Signals Conference*, Dublin, Ireland, pp. 519-525 June 1998.

A.Th. Schwarzbacher, P.A. Comiskey and J.B. Foley, "Powercount: measuring the power at the VHDL netlist level," *Electronic Devices and Systems Conference,* Bruno, Czech Republic, pp. 70-73, June 1998.

A.Th. Schwarzbacher, P.A. Comiskey and J.B. Foley, "Reduction of the power consumption at the algorithmic level of CMOS circuits," *Electronic Systems and Devices Conference,* Bruno, Czech Republic, pp. 5-8, June 1998.

P.A. Comiskey, A.Th. Schwarzbacher and J.B. Foley, "The effect of input lattice structure in image processing algorithm," *UK Low Power Forum,* Sheffield, Untied Kingdom, pp. 11.1–11.6, September 1998.

A.Th. Schwarzbacher, P.A. Comiskey and J.B. Foley, "High level power estimation powercount," *Irish Systems and Signals Conference*, Dublin, Ireland, pp. 101-108, June 1998.

A.Th. Schwarzbacher, P.A. Comiskey and J.B. Foley, "Improving the power consumption in image processing algorithms," *UK Low Power Forum,* Sheffield, Untied Kingdom, pp. 4.1–4.5, September 1998.

A.Th. Schwarzbacher, P.A. Comiskey, J. Neves Rodrigoues and J.B. Foley, "Design of integrated circuits for the power domain," *First International Postgraduate Research Conference*, Dublin, Ireland, November 1998.

P.A. Comiskey and A.Th. Schwarzbacher, "Classification of Uniform White Noise Sources using the Spectral Test," *Fling High - Magazine for Supercomputing,* no. 15, Spring 1999.

A.Th. Schwarzbacher and P.A. Comiskey, "Power estimation at the higher levels of integrated circuit design" *Irish Scientist Year Book*, p. 112, 1999.

A.Th. Schwarzbacher, P.A. Comiskey, J.B. Foley, J. Rodrigoues and F. Klemenz, "Rapid estimation of the active node capacitance of VLSI circuits," *Programmable Devices and Systems 2000,* Ostrava, Czech Republic, pp. 85-88, February 2000.

A.Th. Schwarzbacher, P.A. Comiskey and J.B. Foley, "Low-Power CMOS implementation of an image processing algorithm," *Irish Machine Vision and Image Processing Conference,*

Maynooth, Ireland, p. 257, September 2001.

A.Th. Schwarzbacher, J.P. Silvennoinen and P.A. Comiskey, "Comparison of VLSI implementations of adder circuits," accepted for *3rd International Symposium on Communication Systems, Networks and Digital Signal Processing 2002,* Stafford, Untied Kingdom, July 2002.

A.Th. Schwarzbacher and P.A. Comiskey, "LLAMA: a circuit level power estimation tool," *Electronic Devices and Systems Conference,* Bruno, Czech Republic, pp. 177-182, September 2002.

P.A. Comiskey, A.Th. Schwarzbacher, J.B. Foley, "High Level Markov Power Estimation," *Accepted for publication for the Irish Systems and Signals Conference,* Limerick, Ireland, July 2003.