Brigham Young University
BYU ScholarsArchive

All Theses and Dissertations

2005-08-08

Estimating the Discrepancy Between Computer Model Data and Field Data: Modeling Techniques for Deterministic and Stochastic Computer Simulators

Emily Joy Dastrup
Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Statistics and Probability Commons

BYU ScholarsArchive Citation

Dastrup, Emily Joy, "Estimating the Discrepancy Between Computer Model Data and Field Data: Modeling Techniques for Deterministic and Stochastic Computer Simulators" (2005). *All Theses and Dissertations*. 652.
https://scholarsarchive.byu.edu/etd/652

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

ESTIMATING THE DISCREPANCY BETWEEN COMPUTER MODEL DATA AND FIELD DATA: MODELING TECHNIQUES FOR DETERMINISTIC AND STOCHASTIC COMPUTER SIMULATORS

by

Emily J Dastrup

A Thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Statistics

Brigham Young University

August 2005

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a Thesis submitted by

Emily J Dastrup

This Thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Dr. Scott Grimshaw, Chair
Date	Dr. Shane Reese
Date	Dr. John Lawson

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the Thesis of Emily J Dastrup in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Dr. Scott Grimshaw
Chair, Graduate Committee

Accepted for the Department

G. Bruce Schaalje
Graduate Coordinator

Accepted for the College

G. Rex Bryce
Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

ESTIMATING THE DISCREPANCY BETWEEN COMPUTER MODEL DATA AND FIELD DATA: MODELING TECHNIQUES FOR DETERMINISTIC AND STOCHASTIC COMPUTER SIMULATORS

Emily J Dastrup

Department of Statistics

Master of Science

Computer models have become useful research tools in many disciplines. In many cases a researcher has access to data from a computer simulator and from a physical system. This research discusses Bayesian models that allow for the estimation of the discrepancy between the two data sources. We fit two models to data in the field of electrical engineering. Using this data we illustrate ways of modeling both a deterministic and a stochastic simulator when specific parametric assumptions can be made about the discrepancy term.

Acknowledgements

This work is a product of the efforts and collaboration of many individuals. Funding from the Deployable Adaptive Processing Systems (DAPS) project at Los Alamos National Laboratories provided the resources needed to collect the data. I would like to thank the many individuals at the laboratory who have dedicated time to this area of research.

I would also like to thank Dr. Michael Wirthlin and graduate students Eric Johnson and Keith Morgan from the department of electrical engineering that provided the expertise needed to choose appropriate prior distributions and formulate other parts of the statistical models.

Without the support of my advisor, Dr. Scott Grimshaw, this project would never have been completed. Thank you for taking the time to meet with me and helping me bridge the gaps that otherwise would have halted this research. Thanks also to Dr. Shane Reese, who provided invaluable help with the Bayesian methods used to fit and estimate the models. Finally, I would like to thank my husband Sam - my primary editor and number-one fan.

Contents

Chapter

1 Introduction 1

1.1 Field Programmable Gate Arrays 2

1.1.1 Flip Flop Upsets 2

1.1.2 Configuration Bit Upsets 3

1.2 Ground-Based Radiation Testing 4

1.3 Fault Injection Simulator 6

1.4 Comparing the Simulator and Accelerator Testing Results 8

1.4.1 Average Sensitivity 8

1.4.2 Validating the Simulator 8

1.4.3 Use of Bayesian Statistical Models 9

2 Literature Review 11

2.1 Statistical Methods involving Computer Models 11

2.1.1 Statistical Emulators of Complex Computer Models 12

2.1.2 Sensitivity or Uncertainty Analysis 13

2.1.3 Relating Data from Computer Models and Field Experiments 14

2.2 Models for Relating the Two Data Sources 15

3 Fitting the Deterministic Simulator Model 18

3.1 Data from Accelerator Testing 19

3.2 Data from the Fault Injection Simulator 20

3.3 Specification of the Likelihood 21

3.4 Elicitation of Prior Distributions and Hyperparameters 24

3.4.1 Sensitivity Analysis for τ^2 26

3.5 Implementation of MCMC methods of estimation 29

3.6 Assessing Model Convergence 30

3.7 Assessing Goodness-of-fit 31

3.8 Posterior Distribution Summaries 33

3.9 Model Prediction 34

3.9.1 Posterior Predictive Distribution of ϵ, the Model Discrepancy . 34

3.9.2 95% Credible Intervals for Design Behavior at Next Accelerator Testing 37

3.9.3 95% Credible Intervals for the True Accelerator Sensitivity . . 38

3.10 Results Using Adjusted Average Sensitivity Data 39

4 Fitting the Stochastic Simulator Model 42

4.1 Data from Accelerator Testing 43

4.2 Data from the Fault Injection Simulator 43

4.3 Specification of the Likelihood 44

4.4 Elicitation of Prior Distributions and Hyperparameters 47

4.4.1 Sensitivity Analysis for σ_d^2 49

4.5 Implementation of MCMC methods of estimation 51

4.6 Assessing Model Convergence 52

4.7 Assessing Goodness-of-fit 54

4.8 Posterior Distribution Summaries 55

4.9 Model Prediction 60

4.10 Results Using Adjusted Average Sensitivity Data 62

5 Summary 66

6 Appendix A: R® Code for the Deterministic Simulator Model 69

7 Appendix B: R® Code for the Stochastic Simulator Model 80

Bibliography 94

Tables

Table

3.1 Average Sensitivity Calculated from the Accelerator Data 20

3.2 Average Sensitivity Measured by the Simulator for Four Designs . . . 21

3.3 The Number of Flip Flops Utilized by Each of the Four Designs . . . 24

3.4 Listing of All Hyperparameters 27

3.5 Posterior Mean and Variance for τ^2 Under Three Proposed Priors . . 28

3.6 Prediction Intervals for True Accelerator Sensitivity Under Three Proposed Priors for τ^2 28

3.7 Cross-Validation Results 32

3.8 Posterior Predictive Checks Using Data from the First Accelerator Visit 33

3.9 Estimate of the Discrepancy Between the Simulator and The Accelerator for 4 Designs 36

3.10 95% Credible Intervals for Average Sensitivity at the Next Accelerator Tests 38

3.11 Credible Intervals for the True Accelerator Sensitivity 39

3.12 Estimate of the Discrepancy Between the Simulator and The Accelerator for 4 Designs, Adjusted Data 41

4.1 Design Failures at the Accelerator 44

4.2 Design Failures Recorded with the Fault Injection Simulator 45

4.3 Listing of All Hyperparameters 49

4.4 Posterior Mean and Variance for σ_d^2 Under Three Proposed Priors . . 50

4.5 95% HPD Intervals for True Accelerator Sensitivity Under Three Proposed Priors for σ_d^2 51

4.6 Results from Simulator Testing for the 3 Hold-out Designs 54

4.7 Posterior Predictive Checks Using Data from the First Accelerator Visit 55

4.8 Credible Intervals for the Average Sensitivity Due to Configuration Bits 59

4.9 Credible Intervals for the True Accelerator Sensitivity 60

4.10 Prediction Intervals for a Hypothetical New Design 61

4.11 95% Equal-Tailed Credible Intervals for Output Errors Observed at the Next Accelerator Tests 62

4.12 Estimate of the Discrepancy Between the Simulator and the Accelerator for Four Designs, Adjusted Data 63

Figures

Figure

1.1 Configuration bits and design architecture 3

1.2 Design change caused by a configuration bit upset 3

1.3 Accelerator Test Setup 5

1.4 Fault Injection Simulator 7

3.1 Prior Distribution for β_1 27

3.2 Convergence Plots for α, β, β_1, and τ^2 30

3.3 Convergence Plots of κ for the Four Designs Listed in Table 1 31

3.4 Prior and Posterior Distributions for β_1 34

3.5 Prior and Posterior Distributions for τ^2 35

3.6 Posterior Distributions for the κ Parameters 35

3.7 Posterior Distributions for the κ Parameters - Overlay 36

3.8 Distribution of the Discrepancy Term for Each Design 37

3.9 Distribution of the Discrepancy Term for Each Design, Adjusted Data 40

4.1 Convergence Plots for β_1, λ, and σ_d^2 53

4.2 Convergence Plots of μ_i for the Four Designs 53

4.3 Posterior Distribution for λ 56

4.4 Posterior Distribution for β_1 56

4.5 Posterior Distributions for the Total Discrepancy $\beta_1 w_i + \lambda$ 57

4.6 Prior and Posterior Distribution for σ_d^2 58

4.7 Posterior Distributions of the Configuration Bit Sensitivity for the Four Designs 58

4.8 Posterior Distributions of the Accelerator Sensitivity for the Four Designs 59

4.9 Posterior Distribution for λ, Adjusted Data 64

4.10 Posterior Distribution for β_1, Adjusted Data 64

4.11 Distribution of the Total Discrepancy Term for Each Design, Adjusted Data 65

Chapter 1

Introduction

In many fields computer models are used as an additional source of data about a physical system or process. Reasons for using computer models vary, but include considerations such as the time and cost of gathering field data, limitations on the ability to observe a process, ethical concerns about gathering experimental data, and the desire to extrapolate where field data does not exist. In some situations, the computer model may provide the only data used in an analysis. In other cases, data from both field observations and computer models are available to the researcher. One way the data from both sources is used is to estimate and quantify the differences between the two types of information. This can be a way to validate the accuracy of the computer model and provide a context for the interpretation of the simulated results. While a general model to relate the two data sources can be applied to many specific problems, a complete formulation of a statistical model requires detailed information about the data in order to apply appropriate distributional assumptions. This chapter gives background information about the simulator data and the field experiment data for a specific research application in electrical engineering that will be used as an illustrative example.

1.1 Field Programmable Gate Arrays

Field Programmable Gate Arrays (FPGAs) allow for logic functions to be integrated into hardware while providing a degree of flexibility because they can be reprogrammed to a different set of functions as needed. This flexibility makes FPGAs attractive for use in spacecraft because they can be reprogrammed after deployment (Caffrey et al. 2003a).

The higher levels of radiation to which FPGAs are exposed in orbit can affect their functioning in a variety of ways (Caffrey et al. 2003b). Single event upsets are of interest because they change the actual design but do not cause permanent damage because they can be easily repaired. Single event upsets can affect different structures on the FPGA: user flip flops, configuration bits, latches, or registers (Caffrey et al. 2003b). A single event upset can cause the FPGA device to malfunction.

1.1.1 Flip Flop Upsets

User flip flops are structures in a FPGA design whose state can be affected by radiation. A change of state in a user flip flop can cause an FPGA device to fail. Because the number of configuration bits is much larger than the number of flip flops, the vast majority of single event upsets caused by radiation in FPGAs are due to configuration bit upsets (Caffrey et al. 2003a).

1.1.2 Configuration Bit Upsets

A configuration bit can be in one of two states, represented by a 0 or a 1 (see Figure 1.1). A bit is upset when radiation (usually due to bombardment of the FPGA by a high-energy particle such as a proton) causes it to change states. Figure 1.2 is a replicate of Figure 1.1, however, one of its configuration bits has been changed by radiation from 1 to 0, which in turn has modified the design structure. A configuration bit upset does not necessarily cause an FPGA to fail. Whether a particular upset affects the functioning of a design depends on what logic functions the FPGA is asked to perform. The logic functions performed by an FPGA design are determined by the input vector given to the device. The input vector is a series of zeroes and ones that determines what calculations are performed. The length of the input vector depends on the individual design. For a 32-bit design there are 2^{32} unique input vectors available (Johnson 2005).

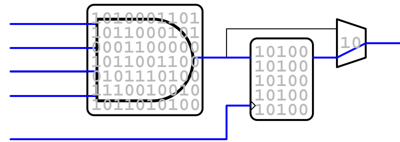

Figure 1.1: Configuration bits and design architecture

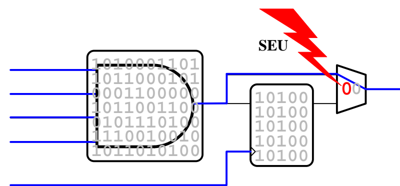

Figure 1.2: Design change caused by a configuration bit upset

Configuration bits that control essential parts of the design will be more likely to cause a design failure when upset than a configuration bit associated with a less important structure. In fact, a certain proportion of configuration bits are not utilized by a given design at all and therefore will never cause it to fail if upset. The probability that a particular configuration bit causes a device failure when upset given a randomly selected input vector is called its bit sensitivity.

1.2 Ground-Based Radiation Testing

Ground-based radiation testing attempts to simulate the radiation environment to which an FPGA might be exposed in orbit. One way to simulate radiation bombardment in space is through the use of a proton accelerator. To perform the test, an FPGA design is bombarded by a proton beam. At certain intervals, the output from this design is then compared to the output from an identical "golden design" that is shielded from the radiation (see Figure 1.3). A discrepancy in the output of the two designs indicates that a device failure has occurred. The design is also checked at intervals for configuration bit upsets. The time stamp at which each configuration bit upset or output error is observed is recorded. The configuration bitstream is then repaired and the device is reset (Caffrey et al. 2003a).

Protons are a good radiation source for ground-based testing because the proton beam can be adjusted to cause a very low upset rate. The goal is to adjust the beam intensity so that only one configuration bit is upset in the observation window. This is useful when applying the results from the testing to a space environment, since it is believed that only one configuration bit will be upset at a time in orbit

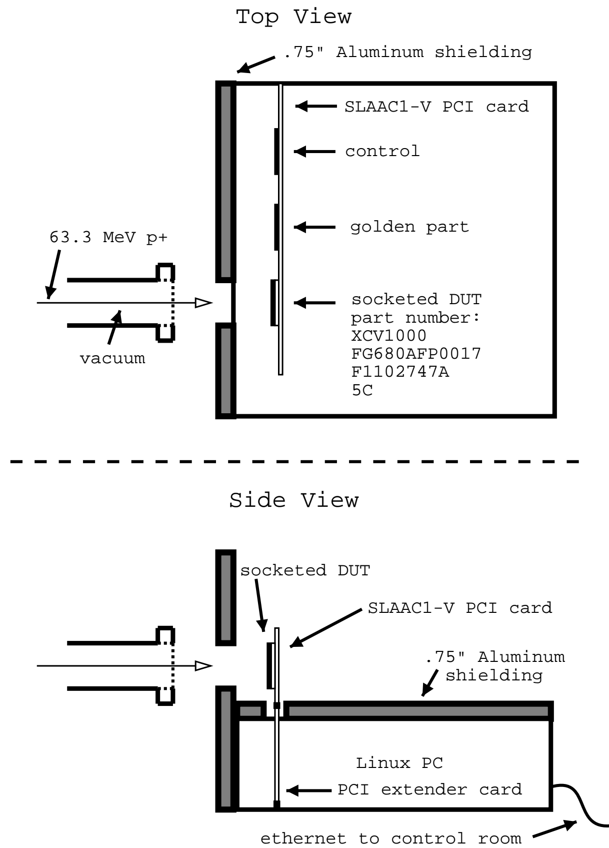

Figure 1.3: Accelerator Test Setup

conditions (Caffrey et al. 2003a). It is also important for comparison with the fault injection simulator results to be introduced later since the simulator also only looks at the effects of configuration bits that occur one at a time. Even with adjustments to the proton beam, there are still intervals where two or more configuration bits are upset during an observation cycle. These cases are excluded from the analysis (Caffrey et al. 2003a).

An important characteristic of ground-based radiation testing is that the proton bombardment causes both configuration bit, flip flop, and other device state errors. It is impossible to eliminate output errors caused by flip flops and other

structures (Caffrey et al. 2003b). A classification system has been employed to try to distinguish between output errors caused by configuration bit upsets and those caused by user flip flop errors. In essence, if a location at the accelerator associated with an output error never caused an error in simulator testing, the error is assumed to be due to a flip flop. This is a conservative adjustment, however, and may not completely remove the discrepancy between the two data sources. Two types of data therefore exist for the accelerator testing: total output errors and adjusted output errors. This analysis uconsiders both total and adjusted errors in order to examine how much this adjustment reduces the discrepancy between the simulator and accelerator results.

Proton accelerator testing is time-consuming and expensive. Additionally, the specific configuration bits that are upset during testing cannot be directly chosen by the researcher. Because of the limitations of time and cost, only a small fraction of the 5,810,024 configuration bits used in a design can be upset during the testing. In addition, each upset location is only tested with one input vector. The data about a specific design that can be gathered during ground-based radiation testing is very limited.

1.3 Fault Injection Simulator

The fault injection simulator attempts to study the behavior of an FPGA in a radiation environment without using radiation to induce configuration bit upsets. The setup of the fault injection simulator is shown in Figure 1.4. Note that two FPGAs of the same design are needed to run the simulator. Configuration bit upsets are manually inserted into one of the FPGAs. The output is then compared to the other

FPGA (the "golden design") that has not been altered. If there is a discrepancy in the output from the two FPGAs (detected by the comparator), the simulator records that the configuration bit upset caused a device failure (Johnson 2005).

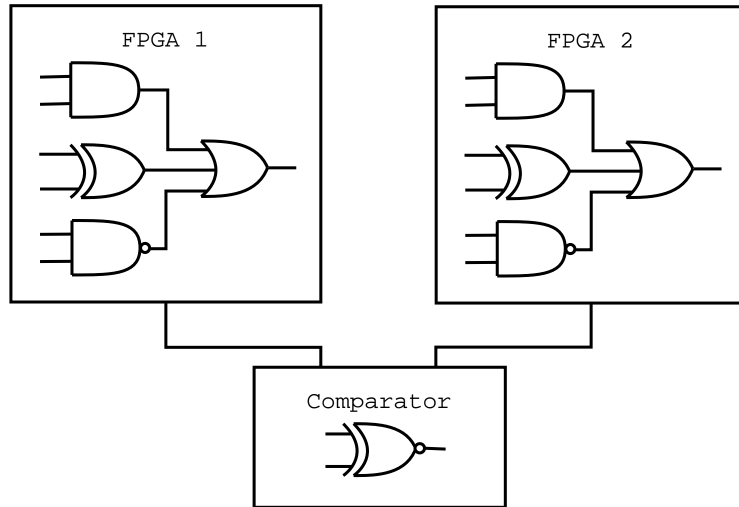

Figure 1.4: Fault Injection Simulator

There are many advantages to using the fault injection simulator in order to gain understanding about the effects of configuration bit upsets on FPGA design function. Unlike the proton accelerator testing, the simulator can test every configuration bit in the design. Because the testing is so much faster than ground-based testing, each configuration bit can also be tested with multiple input vectors. These input vectors are randomly selected among the population of all input vectors. The number of input vectors or locations to be tested can be controlled by the researcher (Johnson 2005). The simulator allows for much better coverage of both the configuration bit and the input vector space than proton accelerator testing.

The simulator is only designed to test the effects of configuration bit upsets on FPGA functionality. It is not able to test for device failures due to flip flop errors

or other single event upsets (Johnson 2005).

1.4 Comparing the Simulator and Accelerator Testing Results

In order to validate the results from the fault injection simulator, tests of several designs were performed using both fault injection simulator and the proton accelerator data. One way to compare the results from the simulator and ground-based radiation testing is to look at the average sensitivity as measured by both testing methods.

1.4.1 Average Sensitivity

The average sensitivity is defined as the total observed device failures divided by the total number of configuration bit upsets. The average sensitivity of a design can be interpreted as the probability that a randomly selected configuration bit causes an FPGA output error. If the probability that an upset configuration bit at location k causes a device failure (denoted by p_k) were known, we could alternatively calculate the average sensitivity by computing $\Sigma_k^n p_k/n$, where n is the total number of unique configuration bits in the sample. The average sensitivity is an aggregate measurement that represents the relative sensitivities of various designs to radiation. It can be easily computed from both the simulator and proton accelerator testing data.

1.4.2 Validating the Simulator

The question of interest is how the results from the simulator compare to results from ground-based radiation testing. Since both testing methods have the

objective of studying the sensitivity of FPGA designs in a radiation environment, it is of interest to calculate how close the estimates of the average sensitivity are for both methods. Since the accelerator necessarily measures the effects of more than just configuration bit upsets, we expect estimates of average sensitivity from the accelerator to be higher than those from the simulator. Specific goals of this analysis include:

- Determine if the data evidences a systematic discrepancy between the simulator predictions of the average sensitivity and the proton accelerator results.
- Quantify the magnitude of the discrepancy and the uncertainty associated with the estimate.
- Predict the outcome of accelerator testing for a new design using only data gathered from the simulator.

Ability to estimate the magnitude of the discrepancy between the two testing methods will provide information about how the simulator performs in relation to traditional testing methods. It will also allow for a more precise understanding of the differences in the two testing methods. By building a model that connects the accelerator results with the simulator results, we gain an idea of how a design will perform in ground-based testing using only data gathered from the simulator.

1.4.3 Use of Bayesian Statistical Models

Certain characteristics of the research question lend themselves to a Bayesian statistical methodology. Bayesian methods readily accommodate data from different

sources. They also allow for the incorporation of prior information about the problem. In addition, a large amount of work has been done in the area of combining data from computer and physical experiments using Bayesian methods. This work poses and fits two different Bayesian statistical models to quantify the discrepancy between the two sources. Both models provide metrics for the validation of the fault injection simulator performance and can be used to make predictions of new design behavior.

Chapter 2

Literature Review

2.1 Statistical Methods involving Computer Models

The increased development and use of computer models has introduced rich areas of research for the discipline of statistics. Statisticians are involved with both the collection and analysis of data from these models. The questions addressed by computer model researchers fall into a few basic categories (see Higdon et al. (2004)):

- Statistical Emulators of Complex Computer Models
- Designing Computer Experiments
- Sensitivity or Uncertainty Analysis
- Relating Data from Computer Models and Field Experiments.

A particular project may require the use of techniques from each of these areas. For example, a researcher may primarily be interested in calibrating a computer model. However, the computer model may be so complex that she needs to use data from a statistical emulator as a substitute for the computer output. Although there is considerable overlap between these categories they provide a basic framework for understanding some of the overarching issues in utilizing computer model data.

2.1.1 Statistical Emulators of Complex Computer Models

Among the types of computer models used in research are computationally intensive and complex models that require large amounts of time to generate output. For these models, it is not feasible to generate output at all the levels of interest of the input parameters. One statistical question of interest is how to choose the inputs at which to run the model in order to gain the most information about the response surface. The question of how to choose the input values and how many runs are necessary falls under the subdiscipline of computer experiments. This area of research tries to select designs that can give optimal information about the inference space subject to constraints on the number of runs that can be performed with the model. Santner et al. (2003) explores design issues specific to computer models in depth.

Once the observations from the complex model have been generated, the question becomes how to use the information to extend the inference to locations where the researcher has been unable to observe data. The data collected in the computer experiment is used to estimate a statistical emulator. A statistical emulator is a statistical smoother or interpolator that approximates the response surface over the inference space. The statistical emulator is able to quickly produce estimates of the response variable at locations where output is not observed from the computer model. Gaussian process models have been used extensively as statistical emulators ((Kennedy and O'Hagan 2001) ; (Bayarri et al. 2002)). Other types of approximators include Bayesian linear estimators ((Craig et al. 1997);(Craig et al. 2001)). The

intent of the statistical emulator is to provide a quick approximation of the computer model output.

2.1.2 Sensitivity or Uncertainty Analysis

A typical computer model will require values for a certain number of inputs in order to produce estimates. Some of these values may be fixed or known by the researcher. The values of other inputs, however, may be unknown or measured with error. In sensitivity analysis, the research will vary an unknown parameter across a range of reasonable values to see how much the output is affected by the value of that parameter. If a certain parameter value has a large impact on the final computer output, it can be said the model is highly sensitive to the value of that parameter. While the terminology is not standardized, uncertainty analysis is usually carried out more systematically, and can address the uncertainty associated with several parameters. This area again overlaps into the broader area of computer experiments. Here, the objective is to design an experiment that will allow for a good estimate of the uncertainty in the model outputs due to the uncertainty in the inputs. A researcher chooses factor levels for each input parameter of interest that capture the amount of uncertainty associated with the parameter. By holding the known parameters constant and varying the other inputs, the researcher can get an estimate of error or uncertainty in the computer output. While there may be many other sources of uncertainty associated with a computer model, this type of analysis attempts to address at least the uncertainty due to the inability to completely specify the input parameters. Bayarri et al. (2002) presents a detailed methodology

for computing "tolerance bounds" that reflect the accuracy that can be associated with the computer model based on a consideration of various sources of uncertainty. Oberkampf et al. (2002) discuss an approach of sampling among reasonable input parameters as part of their discussion of validation activities relating to computer experiments.

2.1.3 Relating Data from Computer Models and Field Experiments

A third area of research is how to use field data and computer data together (or alternatively, the data from two or more different computer models of a system) to gain information about the system as a whole. There are different ways that field data is used to connect the two systems. One important area is in calibration or tuning of the computer model. In calibration, the researcher uses data from field experiments to get estimates for certain parameters such that the parameters produce the best fit of the computer model data to the available field data. In this way, the researcher attempts to adjust the computer model so that it more accurately mimics the behavior of the actual system. A traditional approach is known as inverse modeling. In this approach the field data is used to find the best fitting of the calibration or tuning parameters (Hill 1998). Solving these inverse problems can be very difficult computationally. Bayesian approaches are discussed in detail in Kennedy and O'Hagan (2001), Higdon et al. (2004), and Goldstein and Rougier (2004).

Field data may also be incorporated into an analysis in order to borrow strength across data sources to get more accurate predictions. This is especially true in cases where it is much easier to get computer model observations than field data.

Combining data from both sources is an attempt to get more precise and accurate predictions than would be available with a separate analysis (Qian et al. 2004).

A final use of field data and computer experiment data is to validate or gain a better understanding of the accuracy of the computer model. The goal is to integrate information from both sources in order to get an estimate of the discrepancy between the computer model predictions and the observed data ((Oberkampf et al. 2002);(Higdon et al. 2004);(Bayarri et al. 2002);(Goldstein and Rougier 2004)). This allows the researcher to evaluate the accuracy of the model and interpret the computer model output in the context of the physical system. Accurate estimates of the discrepancy between the two sources also allow the researcher to transfer a computer model estimate into an estimate of the behavior of the actual system.

2.2 Models for Relating the Two Data Sources

This research focuses specifically on the question of how to simultaneously model field data and computer model data in order to understand the relationship between the two systems. This can be done in either a likelihood or Bayesian framework. This research focuses on the use of Bayesian models to connect the two systems. Bayesian models to estimate the discrepancy between the computer model output and field observations have been explored extensively. Higdon et al. (2004) present several alternative Bayesian statistical models that allow for estimation of a discrepancy term and values of tuning parameters. They use Gaussian process models both for the computer model output (as a statistical emulator) and the discrepancy term. A similar approach is discussed in Bayarri et al. (2002).

Goldstein and Rougier (2004) discuss a series of Bayesian models for a simple simulator, a complex simulator, and multiple simulators. Their approach can be applied using a variety of different likelihoods depending on the nature of the data. While their model for a direct simulator is presented in the context of combining computer model and field experiment data, it can also be viewed as a formulation of a measurement error model. This research will apply and adapt their general Bayesian model for a direct simulator to the FPGA sensitivity data. The research will be carried out according to the following steps:

- Selection of an appropriate likelihood
- Elicitation of prior distributions and hyperparameters
- Implementation of Metropolis-Hastings MCMC methods of estimation
- Assessment of convergence and goodness-of-fit
- Selection of posterior distribution summaries
- Model Prediction.

This work adapts their general model to the case of a deterministic simulator and later to the case of a stochastic simulator. It illustrates how to fit both types of models using the FPGA data.

There are certain characteristics of this case study from electrical engineering that allow for the exploration of some important issues in computer modeling research. Much of the research discussed above uses what is known as a "black box"

approach to formulating a model (Kennedy and O'Hagan 2001). The term black box refers to the fact that specific information about the simulator or the discrepancy term is not included in the modeling assumptions. Often, this information is simply unavailable, so a flexible approach such as Gaussian process modeling can provide a good fit without needing to specify the functional form of the discrepancy term or the relationship between the simulator inputs and outputs. This case study examines a situation where prior information is available about the relationship between the simulator and the field observations. Because of the simplicity of the example and the information available, we are able to formulate a statistical model that specifies a functional form for the discrepancy mean. This is known as "gray box" modeling.

The simulator under study is also fairly unique in that it is a stochastic simulator. While several of the papers listed above discuss how to generalize their models to the case of a stochastic simulator, concrete examples of statistical models involving a stochastic simulator are rare. We explain the differences in approach when moving from a deterministic to a stochastic simulator and illustrate how to fit the two types of models by applying distributional assumptions to the FPGA data. The deterministic simulator model uses a likelihood based on a normal approximation to sums of independent random variables, since the average sensitivity can be expressed as such a sum. The stochastic simulator model uses a likelihood based on the binomial distribution, since the raw data from both the simulator and the accelerator can be treated as a number of design failures in a fixed number of trials. Since we expect to see overdispersion due to correlation in the simulator results, we include an additive overdispersion parameter to better model the variance.

Chapter 3

Fitting the Deterministic Simulator Model

The deterministic simulator analysis is based on a simple additive model motivated by the direct simulator example in Goldstein and Rougier (2004). Let y be a vector of the true values under study. We observe z, a vector of field observations, with error e. In this discussion, the error is assumed to be additive. The relationship between the true values y and the observed data z can then be expressed as

$$z = y + e.$$

Similarly, the vector of true values y is believed to be related to the vector of simulator output measured at the same covariate values (denoted by C). In this model we assume that $y = C + \epsilon$, where ϵ is the discrepancy between the simulator output and the true value. Note that the simulator output C is treated as a constant. In reality the fault injection simulator is not deterministic. For simplicity, we ignore the stochastic component in this first model.

We can use the above model specification to relate the field data z to the simulator output C. The relationship can be expressed as

$$z = C + e + \epsilon.$$

Since the simulator output is treated as deterministic we do not jointly model the data from the accelerator and the simulator. Instead, we are essentially modeling the difference between the two sources $z - C$. This is a general model that can be used for any deterministic simulator where we have simulator data at the same levels as the field observations. To illustrate how to fit this type of model we use the FPGA application data and make specific distributional assumptions about e and ϵ.

3.1 Data from Accelerator Testing

This analysis considers the data from four designs that have been tested both at the accelerator and with the fault injection simulator. The field data comes from two ground-based radiation testing sessions at Crocker Nuclear Laboratory. Data also exists from an additional session; however, the testing procedure changed between this and subsequent visits. Because of the change in procedure, the data from this initial session is excluded from the likelihood. Some designs tested use TMR technology which is designed to reduce the sensitivity of the design. The effect of TMR technology is not the focus of the analysis and the data from the TMR designs is not as complete. For these reasons, any data involving TMR designs is also excluded. Table 3.1 shows the data for the four designs to be analyzed. For each design we report the number of locations upset and the average sensitivity (both total and adjusted as discussed in Section 1.2) computed for that run. The unadjusted data is the raw data from accelerator testing and is used to compute all results unless noted otherwise. Some designs were of greater interest for research purposes and were tested more extensively than others.

Obs	Design	Total Bits Upset (M)	Average Sensitivity (z)	Average Sensitivity (Adjusted)
1	Multiplier	4634	0.0993	0.0904
2	Multiplier	1293	0.1060	0.0959
3	Counter	1050	0.0400	0.0390
4	Counter	10036	0.0412	0.0391
5	Counter	9982	0.0379	0.0362
6	Snapshot Recorder	1909	0.0980	0.0906
7	Snapshot Recorder	5096	0.0995	0.0936
8	Snapshot Recorder	4307	0.0984	0.0924
9	Snapshot Recorder	10069	0.0967	0.0918
10	Snapshot Recorder	14953	0.0963	0.0912
11	Snapshot Recorder	5029	0.0929	0.0881
12	Snapshot Recorder	8893	0.0964	0.0912
13	Snapshot Recorder	34	0.1176	0.1176
14	Snapshot Recorder	2823	0.0886	0.0818
15	Snapshot Recorder	559	0.1252	0.1145
16	Snapshot Recorder	95	0.0526	0.0526
17	Snapshot Recorder	167	0.0719	0.0599
18	Snapshot Recorder	317	0.0631	0.0599
19	Snapshot Recorder	6135	0.0998	0.0936
20	Snapshot Recorder	924	0.1028	0.0942
21	Snapshot Recorder	5726	0.0959	0.0891
22	Snapshot Recorder	12641	0.0957	0.0904
23	Snapshot Recorder	719	0.0932	0.0904
24	Snapshot Recorder	2030	0.0857	0.0818
25	Snapshot Recorder	1274	0.0769	0.0738
26	Snapshot Recorder	1569	0.0911	0.0854
27	Snapshot Recorder	236	0.0932	0.0932
28	Snapshot Recorder	1957	0.0920	0.0874
29	Snapshot Recorder	4194	0.0882	0.0827
30	Snapshot Recorder	595	0.0958	0.0908
31	Snapshot Recorder	5196	0.0939	0.0893
32	Snapshot Recorder	328	0.0945	0.0884
33	Snapshot Recorder	571	0.1051	0.0963
34	Snapshot Recorder	371	0.0970	0.0970
35	Snapshot Recorder	8075	0.0977	0.0925
36	Snapshot Recorder	14309	0.0967	0.0906
37	Snapshot Recorder	1659	0.1013	0.0992
38	Snapshot Recorder	4001	0.0877	0.0992
39	Synthetic No TMR	106	0.0566	0.0566
40	Synthetic No TMR	3812	0.0795	0.0740
41	Synthetic No TMR	2372	0.0742	0.0708
42	Synthetic No TMR	4736	0.0758	0.0707
43	Synthetic No TMR	8929	0.0750	0.0708
44	Synthetic No TMR	1793	0.0742	0.0697
45	Synthetic No TMR	501	0.0579	0.0559
46	Synthetic No TMR	634	0.0662	0.0615
47	Synthetic No TMR	2698	0.0756	0.0723
48	Synthetic No TMR	489	0.0798	0.0695
49	Synthetic No TMR	2219	0.0685	0.0658
50	Synthetic No TMR	1065	0.0817	0.0761
51	Synthetic No TMR	9787	0.0743	0.0697
52	Synthetic No TMR	1466	0.0621	0.0600

Table 3.1: Average Sensitivity Calculated from the Accelerator Data

3.2 Data from the Fault Injection Simulator

This first model treats the data from the simulator as a known constant. The simulator tested most of the designs 150 times. The total number of configuration bits, 5,810,024, is constant for all designs. The data from the 150 tests was combined to compute one measure of the average sensitivity. In the case of the simulator, the average sensitivity can be expressed as:

$$Average\ Sensitivity = \frac{Total\ Output\ Errors}{n \times Total\ Configuration\ Bits}.$$

This sample size is believed to give accuracy to the third decimal place in repeated tests. Table 3.2 shows simulator estimates of the average sensitivity for the four designs used in the analysis.

Design	n	Average Sensitivity
Multiplier	150	0.0889
Counter	150	0.0312
Snapshot Recorder	150	0.0863
Synthetic No TMR	100	0.0614

Table 3.2: Average Sensitivity Measured by the Simulator for Four Designs

3.3 Specification of the Likelihood

We adapt the notation for the general deterministic simulator model to the FPGA data. For this model, let y_i be the true average sensitivity for the ith design under ground-based radiation testing. This sensitivity metric will include output errors due to configuration bit upsets and inevitably some flip flop upsets as well. We observe z_{ij}, the average sensitivity calculated for the jth test of the ith design. In addition, we observe the output C_i from the fault injection simulator for the ith design.

The general model relates the accelerator data to the simulator output, but has not made any distributional assumptions. To decide on a reasonable likelihood we need to look at the characteristics of the data z. The proportions calculated from the accelerator tests are sums of Bernoulli random variables, where each upset bit can take on a value of 0 (does not cause an output error) or 1 (causes an output error). The amount of data from the accelerator relative to the total number of configuration

bits is limited. Since the upset locations are sparse, it is reasonable to assume that the number of upset locations adjacent to each other is essentially zero. We would expect non-neighboring locations to be part of different structures on the circuit and therefore independent. If the data is independent (but not necessarily identically distributed) then by the Liapounov Central Limit Theorem the distribution of the sum of the Bernoulli random variables should be well-approximated by the normal distribution if certain conditions are satisfied. A version of the theorem is stated below, as found in Lehmann (1999).

Theorem 1

Let X_i (i=1,...,n) be independently distributed with means E(X_i)=ξ_i and variances σ_i^2, and with finite third moments. If

$$Y_n = \frac{\sqrt{n}(\bar{X} - \bar{\xi})}{\sqrt{(\sigma_1^2 + ... + \sigma_n^2)/n}},$$

then

$$Y_n \rightarrow N(0,1),$$

provided

$$(E(\Sigma|X_i - \xi_i|^3))^2 = o((\Sigma\sigma_i^2)^3).$$

The conditions of the theorem are not likely to be satisfied for designs with a large number of non-sensitive bits, since p_k (the probability that the configuration bit at location k produces an output error) is then not bounded away from zero.

Since the variance of the sum of independent random variables depends on the sum of the variances of each random variable we would expect the variance of the

proportion to differ from one design to the next. From the theorem, we also know the variance is dependent on the sample size. We expect a smaller variance for a proportion calculated from 50,000 upsets than one calculated from 10,000 upsets. The following specification of the sampling error accounts for the difference in variances across designs and sample size,

$$e_{ij} \sim N(0, \frac{\kappa_i^2}{M_{ij}}),$$

where κ_i is a parameter representing the measurement error variance of the observations z_{ij} for design i, and M_{ij} is the sample size of the jth replicate of the ith design.

Less information from statistical theory is available to indicate what distributional characteristics the model discrepancy term ϵ_i should have. However, information does exist in the form of expert opinion. The discrepancy should be some function of the proportion of flip flop structures present on a design. Keith Morgan, a graduate student in the department of electrical engineering at Brigham Young University, suggests that a linear relationship between the discrepancy and the number of flip flops is a reasonable modeling assumption. In this analysis we model

$$\epsilon_i \sim N(\beta_1 X_i, \tau^2),$$

where X_i is a constant equal to the number of design flip flops divided by the total number of configuration bits in the design. Table 3.3 lists the number of flip flops utilized by each design and the corresponding value for X_i.

The parameter β_1 can be thought of as the bias parameter - in other words, $\beta_1 X_i$ will indicate the degree by which the simulator is systematically over- or under-

Design	Flip Flops	Flip Flops/Total Configuration Bits (X)
Multiplier	14,832	0.002553
Counter	9,601	0.001652
Snapshot Recorder	9,187	0.001581
Synthetic No TMR	7,244	0.001247

Table 3.3: The Number of Flip Flops Utilized by Each of the Four Designs

predicting the true average sensitivity for the ith design. The other parameter τ^2 is the variance of this discrepancy term. It indicates how much uncertainty is associated with the discrepancy term across designs.

As noted above, we express $z_{ij} = C_i + e_{ij} + \epsilon_i$. Note that z_{ij} is simply a constant plus the sum of two independent normal random variables. From probability theory we know that $z_{ij} \sim N(C_i + \beta_1 X_i, \frac{\kappa_i^2}{M_{ij}} + \tau^2)$. The likelihood for all the observed data is

$$f(z|\beta_1, \kappa^2, \tau^2) = \Pi_i^4 \Pi_j^{n_i} \frac{1}{\sqrt{(2\pi(\frac{\kappa_i^2}{M_{ij}} + \tau^2))}} exp\left(-\frac{(z_{ij} - C_i - \beta_1 X_i)^2}{2\left(\frac{\kappa_i^2}{M_{ij} + \tau^2}\right)}\right).$$

3.4 Elicitation of Prior Distributions and Hyperparameters

This model uses informative priors for all parameters. We treat the variance parameters κ_i^2 as hierarchical parameters, where we assume there exists a mean variance for the population of all designs, and that the individual design variances are distributed about that mean as follows:

$$\kappa_i^2 \sim GAM(\alpha, \beta)$$

$$\alpha \sim GAM(a_\alpha, b_\alpha)$$

$$\beta \sim GAM(a_\beta, b_\beta).$$

We propose to use the gamma distribution as a prior for both the variance parameters κ_i^2 and τ^2 and the hierarchical parameters α and β because it preserves the parameter space by being defined on the interval (0,∞). As a two parameter distribution, it is fairly flexible in the kinds of shapes it can accommodate. All gamma distributions used in this model follow the form

$$p(\theta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \theta^{\alpha-1} exp(-\beta\theta),$$

which is sometimes referred to as the rate parametrization. To fully specify the model we make the following additional assumption:

$$\beta_1 \sim N(u, s^2).$$

The analysis requires hyperparameters for β_1, τ^2, α, and β. The hyperparameter values specified for α and β are motivated by Liapounov's Central Limit Theorem. From the central limit theorem we know that the variance of z_{ij} is the average variance of M_{ij} Bernoulli variables and can be expressed as $\Sigma_k \frac{(p_k)(1-p_k)}{M_{ij}^2}$, where k indexes the configuration bit locations. Each design should have an average variance κ_i that lies in the interval (0,0.25), since the variance of any one Bernoulli variable is $p_k(1-p_k)$. Of the population of all FPGA designs, we specify that the average variance should be about 0.10×0.90, with a variance of about 0.002. This variance is large enough for the distribution of $\kappa_i^2 s$ to cover the interval of possible variances (0,0.25) defined by the statistical theory, while giving less probability to extremely small or extremely large average variances. We use moment matching to find reasonable estimates for α and β that define the moments listed above but still express uncertainty as to their

exact values. The hyperparameters for the distributions of α and β are shown in Table 3.4.

The hyperparameters for the distributions of β_1 and τ^2 are harder to specify. In this case we need to rely on expert opinion and past experience rather than on statistical theory. Results from the first accelerator tests (not used in this analysis) suggest that the simulator and accelerator estimates differ in the third decimal place. Since the data at the proton accelerator tests includes a certain number of flip flop errors, the average sensitivity from the simulator should underpredict the average sensitivity seen in accelerator testing. Dr. Michael Wirthlin in the department of electrical engineering at Brigham Young University estimates that the magnitude of the underprediction, denoted in the model by $\beta_1 X$, will fall somewhere between 1 and 2 percent. Hyperparameters are chosen to concentrate most of the probability for β_1 between 0 and 20. This allows $\beta_1 X$ to vary between 0 and 3.2 percent. The prior distribution for β_1 is shown in Figure 3.1 and the hyperparameters are listed in Table 3.4 . The hyperparameters of the variance of the model discrepancy are even more difficult to specify with certainty. The distribution of τ^2 should reflect the degree of uncertainty regarding the model discrepancy term without being unreasonably large.

3.4.1 Sensitivity Analysis for τ^2

Because only limited information is available about τ^2, we perform a sensitivity analysis to evaluate whether the chosen prior exerts too much influence on the final results. The model estimates under the original prior specification are compared to

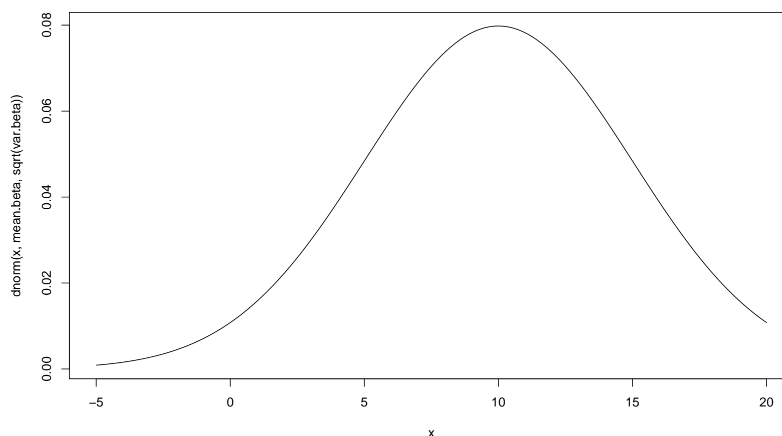

Figure 3.1: Prior Distribution for β_1

Parameter	Distribution	Hyperparameter	Value
β_1	Normal	u	10
β_1	Normal	s^2	25
α	Gamma	a_α	16.4
α	Gamma	b_α	4.05
β	Gamma	a_β	405
β	Gamma	b_β	9
τ^2	Gamma	a_τ*	33.33
τ^2	Gamma	b_τ*	3,333,333
*modified in Section 3.4.1			

Table 3.4: Listing of All Hyperparameters

estimates using two alternative priors. In the alternative priors, the hyperparameters are adjusted to increase the prior variance while holding the prior mean constant. Under the rate parametrization of a gamma distribution with parameters α and β the mean is defined to be $\frac{\alpha}{\beta}$ and the variance is defined as $\frac{\alpha}{\beta^2}$. By increasing the variance, the priors become flatter and the data is allowed to exert more influence on the estimated posterior distribution. Table 3.5 shows how the posterior mean and

variance of τ^2 change under the three different prior specifications. When the prior variance is increased, the data is able to pull the posterior mean down to a smaller estimate. The original prior chosen for the model constrains the posterior mean for τ^2 to be larger than the other less-informative priors. This adds uncertainty to the model predictions. The effect of the additional uncertainty on model predictions can be seen in the width of the prediction intervals listed in Table 3.6.

	Marginal Posterior Summaries	
Prior Distribution	Mean of τ^2	Variance of τ^2
$a_\tau = 33.33, b_\tau = 3,333,333$*	9.21×10^{-6}	2.53×10^{-12}
$a_\tau = 3.33, b_\tau = 333,333$	5.44×10^{-6}	1.01×10^{-11}
$a_\tau = 1.67, b_\tau = 166,667$	3.72×10^{-6}	8.18×10^{-12}
*original prior		

Table 3.5: Posterior Mean and Variance for τ^2 Under Three Proposed Priors

Prior Distribution	Design	95% HPD Interval for y_i
$a_\tau = 33.33, b_\tau = 3,333,333$*	Multiplier	0.0967-0.1097
$a_\tau = 3.33, b_\tau = 333,333$	Multiplier	0.0980-0.1085
$a_\tau = 1.67, b_\tau = 166,667$	Multiplier	0.0988-0.1079
$a_\tau = 33.33, b_\tau = 3,333,333$*	Counter	0.0343-0.0467
$a_\tau = 3.33, b_\tau = 333,333$	Counter	0.0357-0.0456
$a_\tau = 1.67, b_\tau = 166,667$	Counter	0.0365-0.0450
$a_\tau = 33.33, b_\tau = 3,333,333$*	Snapshot Recorder	0.0891-0.1014
$a_\tau = 3.33, b_\tau = 333,333$	Snapshot Recorder	0.0904-0.1003
$a_\tau = 1.67, b_\tau = 166,667$	Snapshot Recorder	0.0911-0.0995
$a_\tau = 33.33, b_\tau = 3,333,333$*	Synthetic No TMR	0.0666-0.0787
$a_\tau = 3.33, b_\tau = 333,333$	Synthetic No TMR	0.0678-0.0775
$a_\tau = 1.67, b_\tau = 166,667$	Synthetic No TMR	0.0687-0.0770
*original prior		

Table 3.6: Prediction Intervals for True Accelerator Sensitivity Under Three Proposed Priors for τ^2

Since we do not have definitive prior information about τ^2, it is undesirable

to constrain the posterior estimates to be large by using a prior that is too peaked. In order to avoid being too informative, the original hyperparameters are modified to $a_\tau = 1.67$ and $b_\tau = 166,667$. Unless otherwise specified, these will be the hyperparameters for τ^2 used for model estimation.

3.5 Implementation of MCMC methods of estimation

The specification of this first model requires the use of Metropolis-Hastings methods for posterior simulation nested within a Gibbs sampler. These methods are implemented as described in Gelman et al. (2004). The complete conditionals for each parameter form part of the Markov Chain Monte Carlo (MCMC) algorithm used to generate draws from the joint posterior distribution. The kernel of each parameter's complete conditional is listed below:

$$[\beta_1] \propto \Pi_i^4 \Pi_j^{n_i} exp\left(-\frac{(z_{ij} - C_i - \beta_1 X)^2}{2\left(\frac{\kappa_i^2}{M_{ij}} + \tau^2\right)}\right) exp(-\frac{(\beta_1 - u)^2}{2s^2})$$

$$[\kappa_i^2] \propto \Pi_j^{n_i} (2\pi(\frac{\kappa_i^2}{M_{ij}} + \tau^2))^{-\frac{1}{2}} exp\left(-\frac{(z_{ij} - C_i - \beta_1 X)^2}{2\left(\frac{\kappa_i^2}{M_{ij}} + \tau^2\right)}\right) \kappa_i^{2(\alpha-1)} exp(-\beta\kappa_i^2)$$

$$[\tau^2] \propto \Pi_i^4 \Pi_j^{n_i} 2\pi(\frac{\kappa_i^2}{M_{ij}} + \tau^2))^{-\frac{1}{2}} exp\left(-\frac{(z_{ij} - C_i - \beta_1 X)^2}{2\left(\frac{\kappa_i^2}{M_{ij}} + \tau^2\right)}\right) \tau^{2(a_\tau-1)} exp(-b_\tau \tau^2)$$

$$[\alpha] \propto \Pi_i^4 \kappa_i^{2(\alpha-1)} \alpha^{a_\alpha - 1} exp(-b_\alpha \alpha)$$

$$[\beta] \propto \Pi_i^4 exp(-\beta\kappa_i^2) \beta^{a_\beta - 1} exp(-b_\beta \beta).$$

3.6 Assessing Model Convergence

One hundred thousand realizations from the posterior distribution of each of the parameters are generated using Gibbs sampling and the Metropolis-Hastings algorithm as described in section 3.5. The number of generated realizations is chosen to provide accurate estimates without excess computation. Figures 3.2 and 3.3 show time series plots of the first 10,000 realizations of the chain. The first 4,999 observations are discarded as burn-in. The plots suggest that the algorithm has converged and has been tuned correctly. The algorithm is not staying at the same value too frequently and the variance of the candidate distribution appears to be large enough for the algorithm to adequately cover the parameter space.

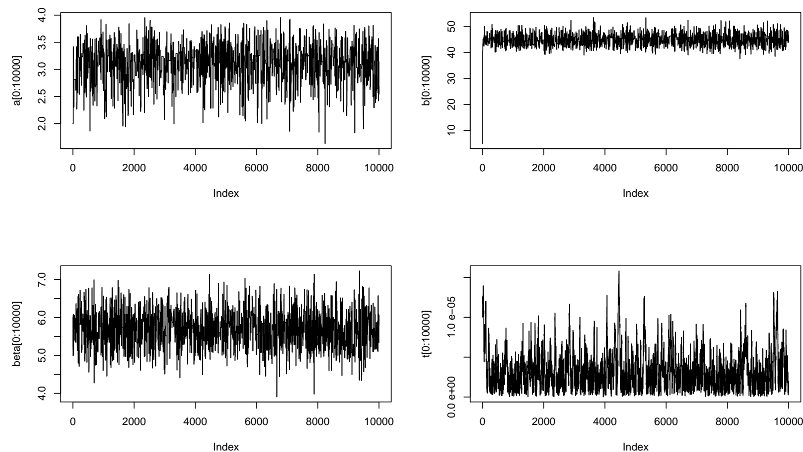

Figure 3.2: Convergence Plots for α, β, β_1, and τ^2

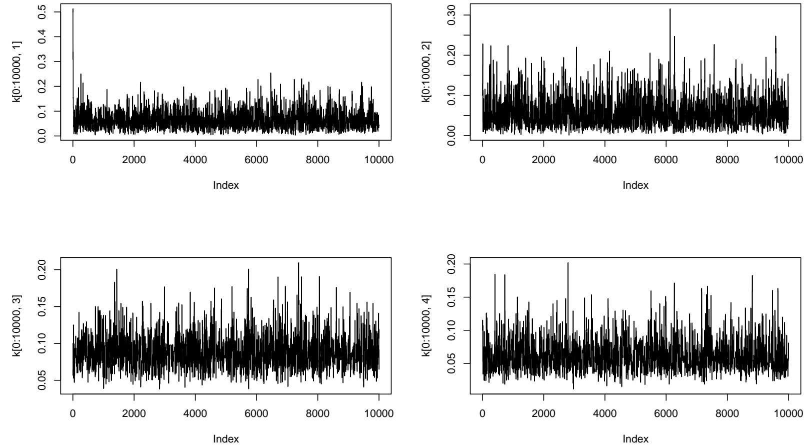

Figure 3.3: Convergence Plots of κ for the Four Designs Listed in Table 1

3.7 Assessing Goodness-of-fit

Each of the observations from the accelerator is systematically withheld and the model is fit to the remaining data. The withheld observation is compared to its posterior predictive distribution. Table 3.7 reports the percentile of the posterior predictive distributions corresponding to the withheld observation. The percentage of observations that fall outside the 95% equal-tailed prediction area after being withheld is compared to the expected 5% error rate. Out of the 52 total observations, one fell in the lower 0.025 tail of the posterior predictive distribution and two fell in the upper 0.025 tail. The 3 out of 52, or 5.8% of the observations that fell into these tails is very close to the 5.0% we would expect to see in a random sample. The rest of the observations lie in areas of high probability in their respective predictive distributions. The model is internally consistent with respect to its predictive ability.

In addition, data on the average sensitivity exists for three designs tested at an

Obs	Percentile	Obs	Percentile
1	0.8634	27	0.5467
2	0.3441	28	0.6846
3	0.5393	29	0.9342
4	0.4167	30	0.4762
5	0.8469	31	0.6209
6	0.3466	32	0.5219
7	0.1767	33	0.2176
8	0.2604	34	0.4573
9	0.3437	35	0.2643
10	0.3729	36	0.3180
11	0.7069	37	0.2161
12	0.3837	38	0.9427
13	0.3299	39	0.7592
14	0.8791	40	0.0555
15	0.0057*	41	0.3987
16	0.9271	42	0.2183
17	0.8514	43	0.2374
18	0.9778*	44	0.4107
19	0.1341	45	0.9161
20	0.2173	46	0.7485
21	0.4495	47	0.2859
22	0.4469	48	0.2625
23	0.5710	49	0.7876
24	0.9255	50	0.1199
25	0.9885*	51	0.3107
26	0.7142	52	0.9537
*outside interval			

Table 3.7: Cross-Validation Results

earlier session at Crocker Nuclear Laboratory. This data is excluded from the analysis due to testing changes implemented at the later sessions. While it is unclear how these changes will affect the average sensitivity measure, we expect some continuity across testing methods. Since this data is not used in likelihood of the model, it can be used as hold-out data to evaluate the fit. Table 3.8 shows the 95% Highest Posterior Density (HPD) credible prediction intervals for the behavior of each of the designs

with the corresponding observed accelerator value and sample size. These intervals are calculated using the same methods detailed in Section 3.9.2 for estimating the behavior of a new design at the accelerator. Note that five of the eight observed values fall within the prediction intervals.

Design	N	Observed Sensitivity	95% HPD Interval
VMULT72	57	0.2105	0.0974-0.2389
VMULT72	5754	0.1724	0.1606-0.1772
VMULT72	1919	0.1610	0.1558-0.1816
VMULT72	1733	0.1829*	0.1554-0.1825
VMULT72	2635	0.1283*	0.1578-0.1802
VMULT72	9139	0.1459*	0.1616-0.1759
VMULT36	3003	0.0493	0.0456-0.0665
LSFR72	1069	0.0496	0.0349-0.0684
*outside interval			

Table 3.8: Posterior Predictive Checks Using Data from the First Accelerator Visit

3.8 Posterior Distribution Summaries

One of the goals of the analysis is to determine how the data has updated our beliefs about the model discrepancy parameter. Figures 3.4 and 3.5 overlay the prior and posterior distributions for both β_1 and τ^2. The data from the proton accelerator is limited but we are able to significantly reduce our uncertainty about the magnitude of the bias. Our beliefs about τ^2, the model discrepancy variance, are updated by the data and the estimate of the magnitude of τ^2 is decreased.

Another posterior summary of interest is how the data has updated our beliefs about κ_i^2, the sampling error variance associated with the proton accelerator data for each design. Figure 3.6 plots the posterior distributions of the four variance terms.

Figure 3.7 overlays all four plots on the same graph for comparison. The updated estimates of the sampling error variance will be important in prediction. Note that the estimated sampling error is smallest for the Synthetic No TMR design. The large amount of data available for this design probably allows for a more accurate estimate of the sampling error. For the Counter and the Multiplier designs, where we only have two replicates, the prior seems to have dominated in producing the final estimate of the measurement error.

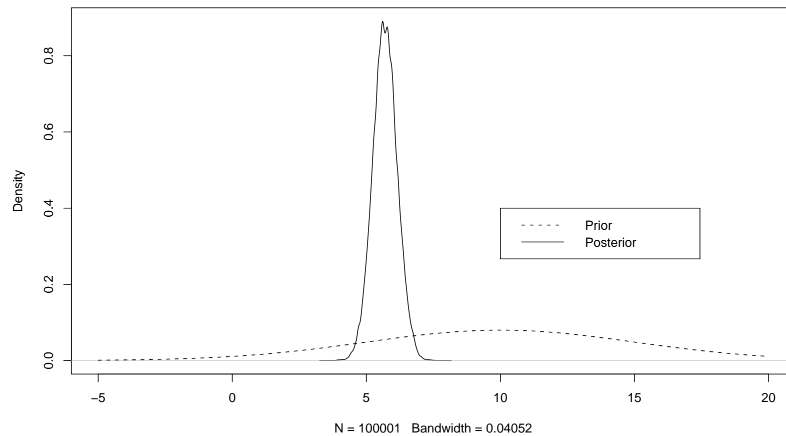

Figure 3.4: Prior and Posterior Distributions for β_1

3.9 Model Prediction

3.9.1 Posterior Predictive Distribution of ϵ, the Model Discrepancy

The magnitude of the discrepancy ϵ between the simulator and accelerator can be expressed in the form of a posterior predictive distribution. According to the model, the discrepancy term is normally distributed with mean $\beta_1 X$ and variance

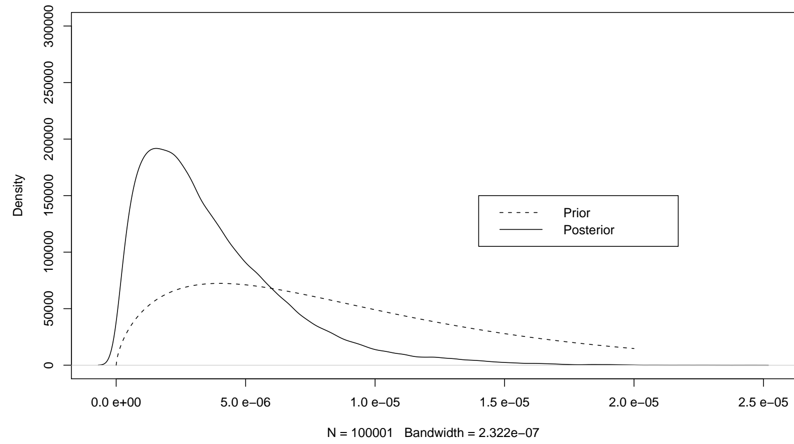

Figure 3.5: Prior and Posterior Distributions for τ^2

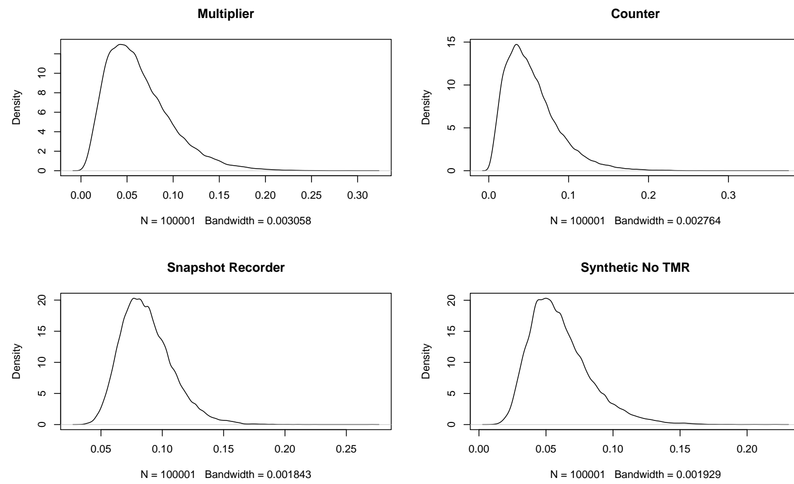

Figure 3.6: Posterior Distributions for the κ Parameters

τ^2. The posterior distributions for β_1 and τ^2 can be used to generate a predictive distribution of the bias for each design. The posterior predictive distribution for the discrepancy term given the data can be expressed as

$$f(\epsilon|z) = f(\epsilon|\beta_1, \tau^2)\pi(\beta_1, \tau^2|z).$$

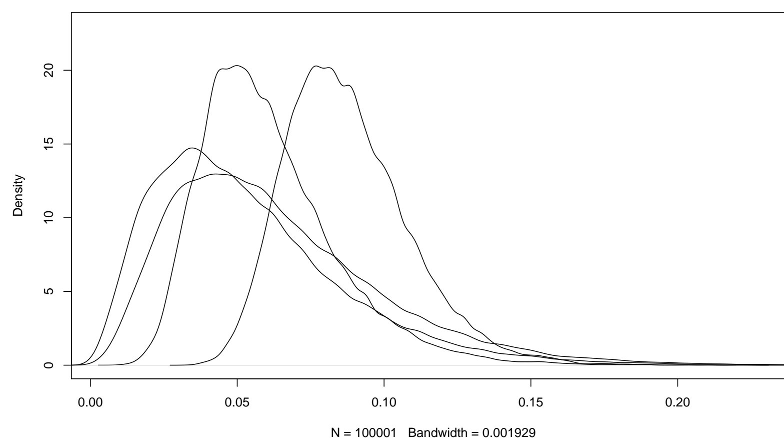

Figure 3.7: Posterior Distributions for the κ Parameters - Overlay

Each design will have a unique mean for its predictive distribution according to the number of flip flops present in the design. The four predictive distributions are shown in Figure 3.8, and 95% HPD credible intervals for the discrepancy estimated for each design are listed in Table 3.9. None of the predictive intervals for the discrepancy term include zero. This provides evidence of a systematic positive discrepancy between the simulator and accelerator sensitivity estimates.

Design	Mean	95% HPD Interval for the Discrepancy
Multiplier	0.0146	0.00984-0.0190
Counter	0.00941	0.00534-0.0137
Snapshot Recorder	0.00900	0.00478-0.0132
Synthetic No TMR	0.00710	0.00301-0.0113

Table 3.9: Estimate of the Discrepancy Between the Simulator and The Accelerator for 4 Designs

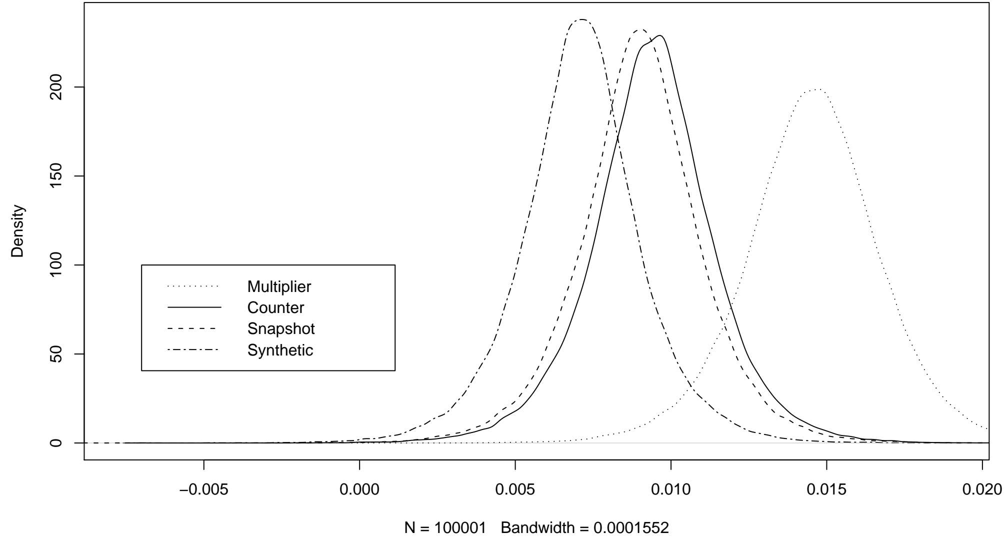

Figure 3.8: Distribution of the Discrepancy Term for Each Design

3.9.2 95% Credible Intervals for Design Behavior at Next Accelerator Testing

By computing the posterior predictive distribution for $z_{i(j+1)}$, we can predict the average sensitivity we would expect to see at the next proton accelerator test. The predictions are made conditional on observing the following numbers of upsets at the accelerator: 1,000, 5,000, and 10,000. This can be done for an existing design or for a design that has not been previously tested in a ground-based radiation environment. The posterior predictive distribution for the ith existing design with a sample size of $M_{i(j+1)}$, can be written as

$$f(z_{i(j+1)}|z) = f(z_{i(j+1)}|\beta_1, \tau^2, \kappa_i^2)\pi(\mu, \tau^2, \kappa_i^2|z).$$

The posterior predictive distribution for a new design is similarly expressed as

$$f(z_{new}|z) = f(z_{new}|\beta_1, \tau^2, \alpha, \beta)\pi(\mu, \tau^2, \alpha, \beta|z).$$

Here we take advantage of the hierarchical model specification for the measurement error of the new design κ^2_{new}, which, as specified, is distributed as $GAM(\alpha, \beta)$. Table 3.10 lists the design names, simulator value, and 95% HPD credible intervals for the four existing designs and a new, hypothetical design at desired sample sizes. The new design has the following characteristics: a simulator sensitivity of 0.0772 and 14,235 flip flops utilized in the design.

Design	Simulator Value	1,000	5,000	10,000
Multiplier	0.0889	0.0866-0.1200	0.0947-0.1116	0.0968-0.1102
Counter	0.0312	0.0250-0.0558	0.0328-0.0483	0.0343-0.0468
Snapshot Recorder	0.0863	0.0763-0.1139	0.0863-0.1047	0.0882-0.1024
Synthetic No TMR	0.0614	0.0569-0.0886	0.0650-0.0809	0.0664-0.0790
New Design	0.0772	0.0739-0.1084	0.0827-0.1001	0.0843-0.0980

Table 3.10: 95% Credible Intervals for Average Sensitivity at the Next Accelerator Tests

3.9.3 95% Credible Intervals for the True Accelerator Sensitivity

The model also allows us to form HPD intervals for the true accelerator sensitivity - i.e., the accelerator sensitivity without any sampling error. In other words, now that we have an estimate of the sampling error for each design, we can remove that uncertainty from the predictions to form an interval of the true accelerator sensitivity. This is done by forming a posterior predictive distribution not for the next observation $z_{i(j+1)}$, but for the true value y_i. From the original model specification

we know that

$$y_i = C_i + \epsilon.$$

This implies that

$$f(y_i|\beta_1, \tau^2) = \frac{1}{(2\pi\tau^2)^{\frac{1}{2}}} e^{-\frac{(y_i - C_i - \beta_1 X)^2}{2\tau^2}}.$$

The posterior predictive distribution for the true sensitivity of a design can be written as $f(y_i|z) = f(y_i|\beta_1, \tau^2)\pi(\beta_1, \tau^2|z)$. As more data is collected, we should be able to continue to reduce our uncertainty about parameters such as β_1 and τ^2. Gaining better estimates about these parameters will reduce the width of these HPD intervals (Table 3.11).

Design	Simulator Value	95% HPD Interval
Multiplier	0.0889	0.0988-0.1079
Counter	0.0312	0.0365-0.0450
Snapshot Recorder	0.0863	0.0911-0.0995
Synthetic No TMR	0.0614	0.0687-0.0787
New Design	0.0772	0.0846-0.0970

Table 3.11: Credible Intervals for the True Accelerator Sensitivity

3.10 Results Using Adjusted Average Sensitivity Data

This model can be re-fit to the adjusted average sensitivity data. One question of interest is how much the algorithm that filters out supposed flip flop errors is able to reduce the discrepancy between the simulator and the accelerator data. A positive discrepancy term would indicate that the adjustments made to the accelerator sensitivity results are still conservative and are not able to bring the data sources completely into agreement. Alternatively, if the algorithm is accurate, than a positive

discrepancy estimate would indicate that the simulator underpredicts the configuration bit sensitivity seen in accelerator testing. The normal likelihood model is applied to the adjusted data and 100,000 realizations of the joint posterior distribution are generated. This data is used to compute new posterior predictive distributions for the bias for each design.

The four predictive distributions are shown in Figure 3.9. 95% credible intervals for the discrepancy estimated for each design are listed in Table 3.12.

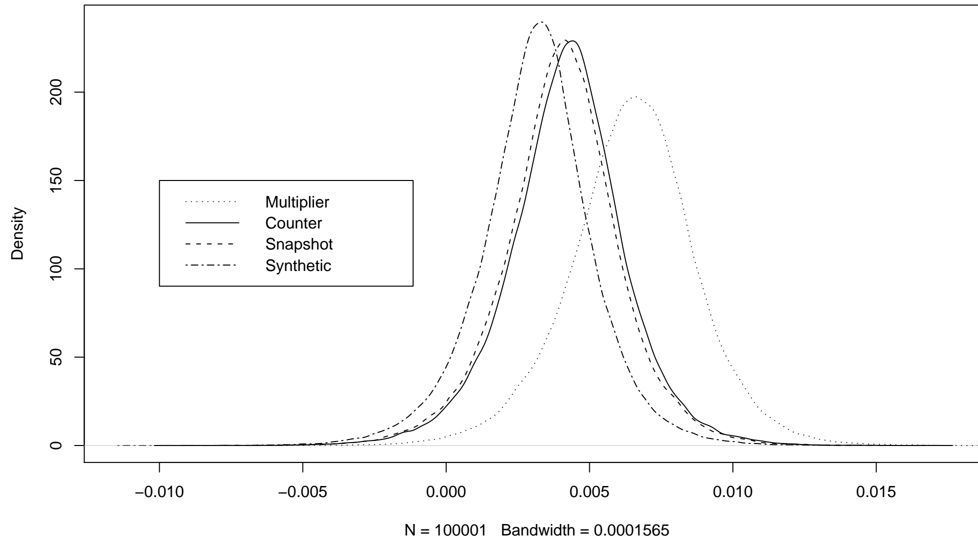

Figure 3.9: Distribution of the Discrepancy Term for Each Design, Adjusted Data

This data provides a better metric for the validation of the accuracy of the simulator since there has already been an attempt to remove the bias due to flip flop errors. When using the adjusted sensitivity numbers, the 95% HPD intervals for the discrepancy term include zero for only three of the four designs. This provides some

Design	Mean	95% credible interval for the bias
Multiplier	0.00651	(0.00172,0.0108)
Counter	0.00420	(-0.000108,0.00836)
Snapshot Recorder	0.00403	(-0.000289,0.000814)
Synthetic No TMR	0.00317	(-0.000940,0.00727)

Table 3.12: Estimate of the Discrepancy Between the Simulator and The Accelerator for 4 Designs, Adjusted Data

evidence that the discrepancy between the two sources is eliminated with the adjusted data. Note, however, that the posterior predictive means for the discrepancy term are still positive. If the posterior means were closer to zero, the evidence of a complete removal of the discrepancy between the two data sources would be more compelling.

Because this data is adjusted, it is not as useful for predicting actual design behavior at the accelerator, where flip flop errors are expected to occur.

Chapter 4

Fitting the Stochastic Simulator Model

If the output from the simulator is treated as stochastic, the likelihood must be specified differently. Instead of modeling the difference between the simulator and accelerator data, the data from both sources is modeled jointly. Let z be a vector of observations (y, x) where y represents the data from the accelerator and x represents the data from the simulator. Let the likelihoods of y and x be denoted by $f(y|\theta_1)$ and $f(x|\theta_2)$, where θ_1 and θ_2 are vectors of parameters. Some of the elements of θ_1 and θ_2 will be the same; the parameters that are common to both likelihoods allow us to relate the data from both systems. The parameters shared by both likelihoods are chosen such that the data (y, x) are independent conditional on these shared parameters. The likelihood of z then becomes:

$$f(z|\theta_1, \theta_2) = f(y|\theta_1)f(x|\theta_2).$$

This type of model can be used whenever the simulator output is stochastic. It can also be used for the case of a deterministic simulator where the simulator and accelerator output are not measured at the same levels, making explicit modeling of the difference between the two sources impossible. This joint modeling is described in

detail in Goldstein and Rougier (2004). To illustrate how to fit this type of model we again use the FPGA design data. Since the simulator data is treated as stochastic and requires a separate likelihood specification, we do not rely on the normal distribution for a likelihood as in the deterministic model. The data from the simulator is likely to be correlated since we observe data at neighboring locations; therefore, the average sensitivity at the simulator is a sum of dependent random variables and cannot be assumed to be approximately normal under the Liapounov Central Limit Theorem.

4.1 Data from Accelerator Testing

The raw data from accelerator testing is in the form of counts rather than proportions. Table 4.1 shows the number of design failures y_{ij} (both adjusted and unadjusted) for the j replicates of the i designs. It also lists m_{ij}, the total number of locations upset. The results presented in this section are based on the unadjusted counts of design failures unless otherwise specified.

4.2 Data from the Fault Injection Simulator

Since the simulator in this model is not treated as deterministic, for each design we have replicates of the number of design failures observed using the simulator. The total number of locations tested (M), is held constant at 290,501,200 for each trial, as each of the 5,810,024 individual configuration bits is tested 50 times with a randomly selected input vector. This is repeated two or three times for each design. The data from the simulator is shown in Table 4.2. This data is aggregated in the deterministic model to produce the proportions listed in Section 3.2.

Design	Total Bits Upset (m)	Design Failures (y)	Adjusted Design Failures
Multiplier	4634	460	419
Multiplier	1293	137	124
Counter	1050	42	41
Counter	10036	413	392
Counter	9982	378	361
Snapshot Recorder	1909	187	173
Snapshot Recorder	5096	507	477
Snapshot Recorder	4307	424	398
Snapshot Recorder	10069	974	924
Snapshot Recorder	14953	1440	1364
Snapshot Recorder	5029	467	443
Snapshot Recorder	8893	857	811
Snapshot Recorder	34	4	4
Snapshot Recorder	2823	250	231
Snapshot Recorder	559	70	64
Snapshot Recorder	95	5	5
Snapshot Recorder	167	12	10
Snapshot Recorder	317	20	19
Snapshot Recorder	6135	612	574
Snapshot Recorder	924	95	87
Snapshot Recorder	5726	549	510
Snapshot Recorder	12641	1210	1143
Snapshot Recorder	719	67	65
Snapshot Recorder	2030	174	166
Snapshot Recorder	1274	98	94
Snapshot Recorder	1569	143	134
Snapshot Recorder	236	22	22
Snapshot Recorder	1957	180	171
Snapshot Recorder	4194	370	347
Snapshot Recorder	595	57	54
Snapshot Recorder	5196	488	464
Snapshot Recorder	328	31	29
Snapshot Recorder	571	60	55
Snapshot Recorder	371	36	36
Snapshot Recorder	8075	789	747
Snapshot Recorder	14309	1383	1296
Snapshot Recorder	1659	168	153
Snapshot Recorder	4001	351	323
Synthetic No TMR	106	6	6
Synthetic No TMR	3812	303	282
Synthetic No TMR	2372	176	168
Synthetic No TMR	4736	359	335
Synthetic No TMR	8929	670	632
Synthetic No TMR	1793	133	125
Synthetic No TMR	501	29	28
Synthetic No TMR	634	42	39
Synthetic No TMR	2698	204	195
Synthetic No TMR	489	39	34
Synthetic No TMR	2219	152	146
Synthetic No TMR	1065	87	81
Synthetic No TMR	9787	727	682
Synthetic No TMR	1466	91	88

Table 4.1: Design Failures at the Accelerator

4.3 Specification of the Likelihood

Since the data is in the form of successes in a certain number of trials, it is natural to use a binomial likelihood to fit the data. Let z be a vector of observations (y, x) where y represents the data from the accelerator and x represents the data from the simulator. We express the kernel of the likelihood for an observation y_{ij} of m_{ij} trials from the accelerator as

$$f(y_{ij}|\pi_i) \propto exp\left(y_{ij}\ log\frac{\pi_i}{1-\pi_i} + m_{ij}\ log(1-\pi_i)\right).$$

Design	Total Bits Upset (M)	Design Failures (x)
Multiplier	290,501,200	25,308,206
Multiplier	290,501,200	25,775,373
Multiplier	290,501,200	25,780,362
Counter	290,501,200	9,635,835
Counter	290,501,200	8,777,447
Counter	290,501,200	8,776,247
Snapshot Recorder	290,501,200	25,106,702
Snapshot Recorder	290,501,200	25,045,368
Snapshot Recorder	290,501,200	25,044,093
Synthetic No TMR	290,501,200	19,049,461
Synthetic No TMR	290,501,200	19,105,479

Table 4.2: Design Failures Recorded with the Fault Injection Simulator

The binomial trials observed at the simulator may exhibit a degree of correlation. At the simulator we observe data at all locations in the design. Neighboring locations are often part of the same structure on the FPGA; therefore, the behavior of neighboring locations is expected to be correlated. Correlated binomial trials can often exhibit overdispersion (Kupper et al. 1986). We express the kernel of the likelihood for an observation x_{ij} of M trials from the simulator as

$$f(x_{ij}|p_i) \propto exp\left(x_{ij}\ log\frac{p_i}{1-p_i} + M\ log(1-p_i)\right).$$

To reflect the additional variance in the data we add a measurement error term in the mean structure of the likelihood. The model for the mean of the simulator is written as

$$log\frac{p_{ij}}{1-p_{ij}} = \mu_i + \gamma_{ij}$$

where μ_i is the logit of the average sensitivity due only to configuration bits for the ith design and γ_{ij} is the measurement error term. Here j is a pseudo-level added to model the extra variance for each observation, as described in Browne et al. (2005). In

this model the parameters γ_{ij} are assumed to follow a normal distribution with mean zero and variance σ_d^2. Under this assumption, the simulator provides an unbiased estimate of the average sensitivity due only to configuration bits. This assumption is consistent with expert opinion about the behavior of the simulator.

The measurement error terms γ_{ij} can also be referred to as overdispersion parameters. Browne et al. (2005) discusses in detail this method of modifying the binomial likelihood by incorporating additive dispersion in the mean structure. The addition of this parameter improves the overall fit of the model by allowing for a flexible relationship between the estimated means and variances of the number of design failures. If the overdispersion were ignored, resulting estimates would not reflect the actual uncertainty in the simulator data.

The relationship between the simulator and the accelerator data for the ith design is specified through the common parameter μ_i found in the structure placed on the mean of each likelihood. For the accelerator the relationship is expressed as

$$log\frac{\pi_i}{1-\pi_i} = \mu_i + \beta_1 w_i + \lambda.$$

Here μ_i is defined as discussed in the model for the simulator, w_i represents the proportion of flip flops for the ith design, β_1 controls the amount of discrepancy between the accelerator estimate and the simulator sensitivity due to flip flops, and λ estimates any other systematic discrepancy between the accelerator estimate and the simulator value that cannot be explained by the number of flip flops. Under this model, the average sensitivity estimated at the accelerator depends upon the average sensitivity due only to configuration bits, μ_i, but is allowed to vary from that value

through the parameters β_1 and λ.

The joint likelihood for all the data z can be expressed as the product of the likelihoods for each individual observation. By substituting the regression relationship for the logit of π_i and p_i into the likelihood the kernel becomes

$$f(y_{ij}, x_{ij}|\mu_i, \beta_1, \lambda, \gamma_{ij}) \propto \Pi_i^4 \Pi_j^{n_i} exp\left(y_{ij}(\mu_i + \beta_1 w_i + \lambda) - m_{ij}\ log(1 + exp^{\mu_i + \beta_1 w_i + \lambda})\right)$$
$$\Pi_i^4 \Pi_j^{q_i} exp\left(x_{ij}(\mu_i + \gamma_{ij}) - M\ log(1 + exp^{\mu_i + \gamma_{ij}})\right).$$

4.4 Elicitation of Prior Distributions and Hyperparameters

The parameter μ_i, the logit value of the ith design configuration bit sensitivity, is modeled hierarchically, where the $\mu_i \sim N(\theta, \tau^2)$. The hierarchical parameters θ and τ^2 are modeled as follows:

$$\theta \sim N(m, v^2)$$

$$\tau^2 \sim INVGAM(a, b).$$

The parameters β_1 and λ are defined such that

$$\beta_1 \sim N(m_\beta, v_\beta^2)$$

$$\lambda \sim N(m_\lambda, v_\lambda^2).$$

Finally, the measurement error parameters γ_{ij} are modeled as

$$\gamma_{ij} \sim N(0, \sigma_d^2).$$

The prior distribution for the overdispersion variance term σ_d^2 is specified as an inverse-gamma distribution with hyperparameters a_d and b_d.

This model requires the specification of hyperparameter values for λ, θ, τ^2, β_1, and σ_d^2. Since the model uses the logistic function when modeling the mean, the hyperparameters need to be chosen to reflect the transformed scale. Since *a priori* Dr. Wirthlin believes that no bias exists at the accelerator that is unexplained by the presence of flip flops, the model term that would capture any such bias, λ, is given a prior mean of zero. Choosing the magnitude of the variance is difficult due to the logit transformation of the average sensitivity. The variance is initially set to 0.04. This should make the prior flat enough to reflect the uncertainty associated with these terms and allow for a significant updating of prior beliefs due to the data.

To specify a mean for θ, the hierarchical mean parameter, we use moment matching and transform to the logit scale. We assume that the average sensitivity for the population of typical FPGA designs is 0.07. The mean for θ is then $log(\frac{0.07}{0.93})$. To specify the variance of this hierarchical mean, we assume the mean of the population of all designs falls in the interval (0.05,0.09), and estimate the prior variance as

$$\left(\left(log(\frac{0.09}{0.91}) - log(\frac{.05}{.95})\right)/6\right)^2.$$

This formula is motivated from properties of the normal distribution; the normal distribution captures the range of most probable values in six standard deviations. This same formula is used in combination with moment matching to calculate the hyperparameters for τ^2, except that it is augmented to reflect the additional uncertainty when considering the variance of the average sensitivity across all designs instead of the variance of the mean. Table 4.3 is a listing of the exact hyperparameter values used for all parameters.

As in Section 3.4, the hyperparameters chosen for the slope parameter β_1 reflect Dr. Wirthlin's belief that the bias due to flip flops falls between 1 and 2 percent; however, this statement must be transformed to have meaning in logit space. The range of reasonable values for β_1 is determined by solving for the parameter under a number of proposed average sensitivities, such that the increase due to flip flops ranges between 0 and 0.03.

The hyperparameters for the overdispersion variance parameter σ_d^2 are particularly hard to specify. The magnitude of the variance is set to be much larger than the mean to reflect the uncertainty associated with this parameter. The influence of the hyperparameters on the posterior estimates is examined in Section 4.4.1.

Parameter	Distribution	Hyperparameter	Value
λ	Normal	m_λ	0
λ	Normal	v_λ^2	0.04
θ	Normal	m	-2.59
θ	Normal	v^2	0.011
τ^2	Inverse-Gamma	a	2.022
τ^2	Inverse-Gamma	b	0.045
β_1	Normal	m_β	225
β_1	Normal	v_β^2	5500
σ_d^2	Inverse-Gamma	a_d	2.2*
σ_d^2	Inverse-Gamma	b_d	0.12*
*modified in Section 4.4.1			

Table 4.3: Listing of All Hyperparameters

4.4.1 Sensitivity Analysis for σ_d^2

While it is clear that the simulator data should manifest overdispersion, it is difficult to determine the magnitude of the parameter σ_d^2 because of the logit trans-

formation applied to the mean of the likelihood. It is important to select a prior that provides a true reflection of our uncertainty with respect to this parameter. The model estimates under the original prior specification are compared to estimates using two alternative priors. In the alternative priors, the hyperparameters are adjusted to decrease the prior mean and increase the prior variance. For an inverse-gamma distribution with parameters α and β the mean is defined to be $\frac{\beta}{\alpha-1}$ and the variance is defined as $\frac{\beta^2}{(\alpha-1)^2(\alpha-2)}$. Table 4.4 shows how the posterior mean and variance of σ_d^2 change for the three different prior specifications. The magnitude of the prior mean has a high influence in determining the magnitude of the posterior mean in the first prior specification. The original specification constrains estimates of σ_d^2 to be higher than the under the two alternative priors. This causes a marked difference in the width of some of the model predictions. In Table 4.5, the average sensitivity predictions at the accelerator are highly sensitive to the original model specification. In contrast, the two alternative priors yield very similar results.

	Marginal Posterior Summaries	
Prior Distribution	Mean of σ_d^2	Variance of σ_d^2
$a_d = 2.2, b_d = 0.12$*	2.14×10^{-2}	8.53×10^{-5}
$a_d = 2.0002, b_d = 0.010002$	2.49×10^{-3}	1.27×10^{-6}
$a_d = 2.000002, b_d = 0.001000002$	7.79×10^{-4}	1.23×10^{-7}
*original prior		

Table 4.4: Posterior Mean and Variance for σ_d^2 Under Three Proposed Priors

Since we do not have definitive prior information about σ_d^2, it is undesirable to constrain the posterior estimates to be large by using a prior mean that is too high. In order to avoid using a prior that is too influential, the original hyperparameters

Prior Distribution	Design	95% HPD Interval
$a_d = 2.2, b_d = 0.12$*	Multiplier	0.0970-0.1114
$a_d = 2.0002, b_d = 0.010002$	Multiplier	0.0986-0.1114
$a_d = 2.000002, b_d = 0.001000002$	Multiplier	0.0983-0.1107
$a_d = 2.2, b_d = 0.12$*	Counter	0.0361-0.0407
$a_d = 2.0002, b_d = 0.010002$	Counter	0.0349-0.0378
$a_d = 2.000002, b_d = 0.001000002$	Counter	0.0348-0.0367
$a_d = 2.2, b_d = 0.12$*	Snapshot Recorder	0.0939-0.0971
$a_d = 2.0002, b_d = 0.010002$	Snapshot Recorder	0.0945-0.0975
$a_d = 2.000002, b_d = 0.001000002$	Snapshot Recorder	0.0948-0.0977
$a_d = 2.2, b_d = 0.12$*	Synthetic No TMR	0.0717-0.0766
$a_d = 2.0002, b_d = 0.010002$	Synthetic No TMR	0.0715-0.0760
$a_d = 2.000002, b_d = 0.001000002$	Synthetic No TMR	0.0712-0.0752
*original prior		

Table 4.5: 95% HPD Intervals for True Accelerator Sensitivity Under Three Proposed Priors for σ_d^2

are modified to $a_d = 2.0002$ and $b_d = 0.010002$. Unless otherwise specified, these will be the hyperparameters for σ_d^2 used for model estimation.

4.5 Implementation of MCMC methods of estimation

The above model specification requires the use of Metropolis-Hastings methods of estimation nested within the Gibbs sampler for posterior simulation. For increased computational efficiency we reparametrize part of the model involving the likelihood for the simulator. Let $\mu_{ij}^* = \mu_i + \gamma_{ij}$. Under this reparametrization the prior $\pi(\mu_{ij}^*|\mu_i)$ is a normal distribution with mean μ_i and variance σ_d^2. The mean of the simulator likelihood can then be written as

$$log\frac{p_{ij}}{1-p_{ij}} = \mu_{ij}^*.$$

This parametrization is used to implement Gibbs sampling and Metropolis-Hastings estimation. This type of reparametrization is known as hierarchical centering and is discussed for this type of binomial-logit model in Browne et al. (2005). The complete conditionals for each parameter form part of Gibbs sampler used to generate draws from the joint posterior distribution. The kernels of the complete conditionals (or closed forms where available) for each parameter are listed below:

$$
\begin{aligned}
[\mu_i] &\propto \Pi_j^{n_i} exp\left(y_{ij}(\mu_i + \beta_1 w_i + \lambda) - m_{ij}\ log(1 + exp^{\mu_i + \beta_1 w_i + \lambda})\right) \\
&\times exp\left(-\frac{(\mu_i - \theta)^2}{2\tau^2}\right) exp\left(-\frac{\Sigma(\mu_{ij}^* - \mu_i)^2}{2\sigma_d^2}\right) \\
[\theta] &\sim N\left(\frac{\Sigma\mu_i \times v^2 + m \times \tau^2}{\tau^2 + N \times v^2}, \frac{\tau^2 v^2}{N \times v^2 + \tau^2}\right) \\
[\tau^2] &\sim INVGAM\left(N/2 + a, \frac{\Sigma(\mu_i - \theta)^2}{2} + b\right) \\
[\beta_1] &\propto \Pi_i^N \Pi_j^{n_i} exp\left(y_{ij}(\mu_i + \beta_1 w_i + \lambda) - m_{ij}\ log(1 + exp^{\mu_i + \beta_1 w_i + \lambda})\right) exp\left(-\frac{(\beta_1 - m_\beta)^2}{2v_\beta^2}\right) \\
[\lambda] &\propto \Pi_i^N \Pi_j^{n_i} exp\left(y_{ij}(\mu_i + \beta_1 w_i + \lambda) - m_{ij}\ log(1 + exp^{\mu_i + \beta_1 w_i + \lambda})\right) exp\left(-\frac{(\lambda - m_\lambda)^2}{2v_\lambda^2}\right) \\
[\mu_{ij}^*] &\propto \Pi_i^N \Pi_j^{n_i} exp\left(x_{ij}(\mu_{ij}^*) - M\ log(1 + exp^{\mu_{ij}^*})\right) exp\left(-\frac{(\mu_{ij}^* - \mu_i)^2}{2\sigma_d^2}\right) \\
[\sigma_d^2] &\sim INVGAM\left(11/2 + a_d, \frac{\Sigma\Sigma(\mu_{ij}^* - \mu_i)^2}{2} + b\right).
\end{aligned}
$$

4.6 Assessing Model Convergence

Five hundred and fifty thousand realizations from the posterior distribution of each of the parameters are generated after appropriately tuning the MCMC simulation algorithm. Figures 4.1 and 4.2 show time series plots of every hundredth realization of the first 300,000 realizations. The first 199,999 posterior realizations are discarded

as burn-in. The logistic structure in the likelihood of the model adds difficulty to the convergence of the algorithm and requires a large number of realizations to assure convergence.

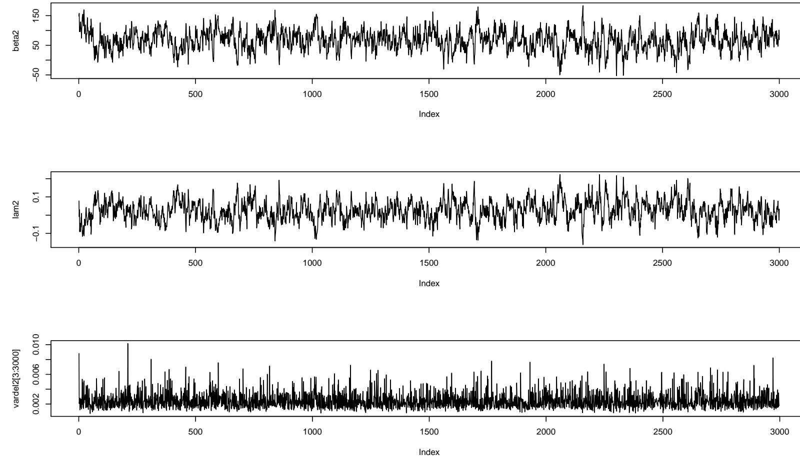

Figure 4.1: Convergence Plots for β_1, λ, and σ_d^2

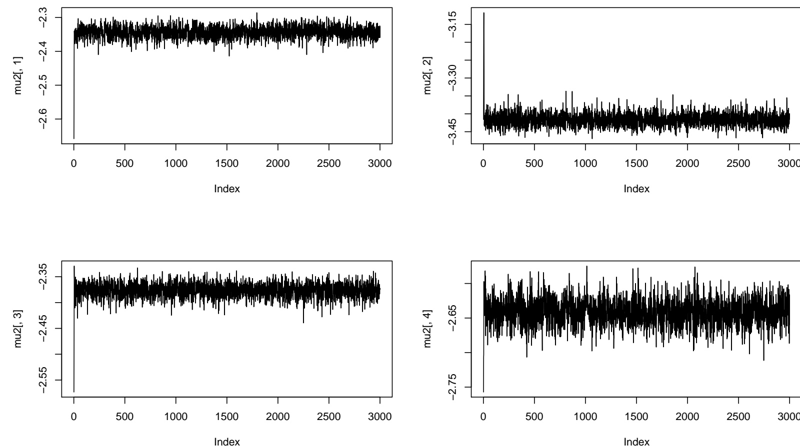

Figure 4.2: Convergence Plots of μ_i for the Four Designs

4.7 Assessing Goodness-of-fit

The data for three designs tested in an earlier session at the accelerator is again used as hold-out data in a cross-validation analysis. A test at the simulator is available for each design; the data can be found in Johnson (2005). The results of this testing are shown in Table 4.6.

Design	Total Locations Tested	Observed Output Errors
VMULT72	2,324,011,200	357,898,191
VMULT36	581,002,440	25,031,616
LSFR72	581,002,440	30,446,011

Table 4.6: Results from Simulator Testing for the 3 Hold-out Designs

Conditional on the data from the simulator, we will to predict the behavior of these designs at the accelerator and compare the predictions to the observed values. The sample sizes of the simulator testing for each of these designs is different than the sample sizes used to fit the model. The performance of the model in predicting for these new designs gives an idea of how the simulator performs in predicting not only for a new design, but also for a different number of simulator runs.

Table 4.6 shows the 95% equal-tailed prediction intervals for the behavior of each of the designs tested in the earlier session at the accelerator with the corresponding observed value and sample size. The methods to generate these intervals are described in more detail in Section 4.9. Note that six of the eight observations fall within the prediction intervals. The model seems to give good predictions even when using different sample sizes at the fault injection simulator.

Design	N	Observed Errors	95% Equal-Tailed Interval
VMULT72	57	12	5-16
VMULT72	5754	992	932-1172
VMULT72	1919	309	303-399
VMULT72	1733	317	272-362
VMULT72	2635	388*	420-544
VMULT72	9139	1333*	1488-1853
VMULT36	3003	148	112-167
LSFR72	1069	53	48-80
*outside interval			

Table 4.7: Posterior Predictive Checks Using Data from the First Accelerator Visit

4.8 Posterior Distribution Summaries

An examination of the marginal posterior distributions of several of the parameters allows us to assess how the data updates our prior beliefs about the accelerator discrepancy terms. The marginal posterior distributions of λ and β_1 allow us to assess the magnitude of the discrepancy between the average sensitivity at the accelerator and the average sensitivity due only to configuration bit errors on the logit scale. The prior and posterior distributions of these discrepancy parameters are shown in Figures 4.3 and 4.4 respectively.

We can use the posterior realizations of λ and β_1 to plot posterior distributions of the discrepancy at the accelerator for each of the four designs. The posterior distributions of the total discrepancy $\beta_1 w_i + \lambda$ are shown in Figure 4.5. Zero is not included as a probable value for the discrepancy for any of the designs. This is evidence of the existence of an overall systematic discrepancy between the simulator and the accelerator. The posterior distribution for λ includes zero with a high probability and the posterior distribution of β_1 includes zero with a low probability, indicating

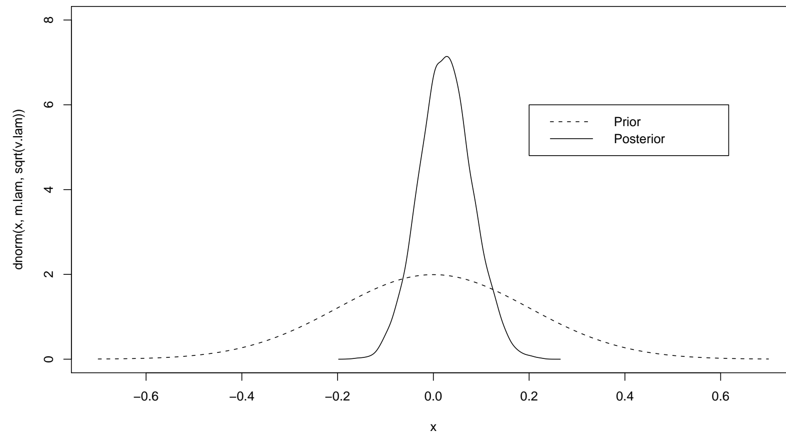

Figure 4.3: Posterior Distribution for λ

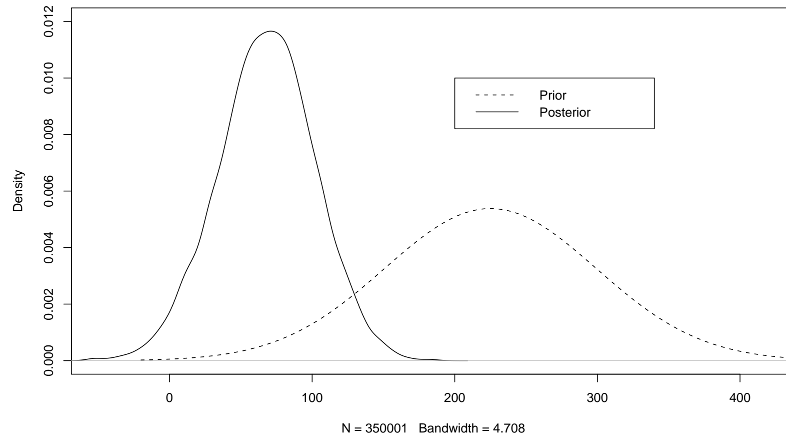

Figure 4.4: Posterior Distribution for β_1

that the overall discrepancy between the two sources is most likely due to design flip flops and not to some other unknown source of bias.

The posterior distribution of σ_d^2 allows us to assess the magnitude of the measurement error variance in the simulator data. Figure 4.6 plots both the prior and

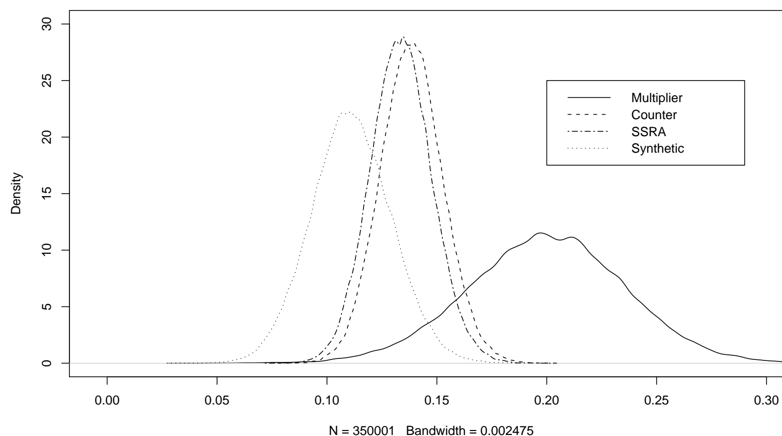

Figure 4.5: Posterior Distributions for the Total Discrepancy $\beta_1 w_i + \lambda$

posterior distributions of the overdispersion variance. The data has allowed us to significantly reduce our prior uncertainty about this quantity. For comparison, the posterior variance of λ is 0.0031 and the posterior mean of σ_d^2 is 0.0025. This comparison enables us to examine the relative precision of the accelerator and simulator data in estimating the average sensitivity. The estimates are very similar in magnitude. Although we get better coverage of the total number of locations and input vectors at the simulator, the presence of correlation among neighboring locations offsets the gain in precision that would be seen under independence.

The posterior realizations of μ_i for the ith design are transformed to

$$\frac{exp(\mu_i)}{1 + exp(\mu_i))}$$

to give posterior distributions for the average sensitivity due only to configuration bits as jointly estimated by the accelerator and the simulator. Figure 4.7 shows these distributions for each of the four designs included in the analysis and Table 4.8 lists

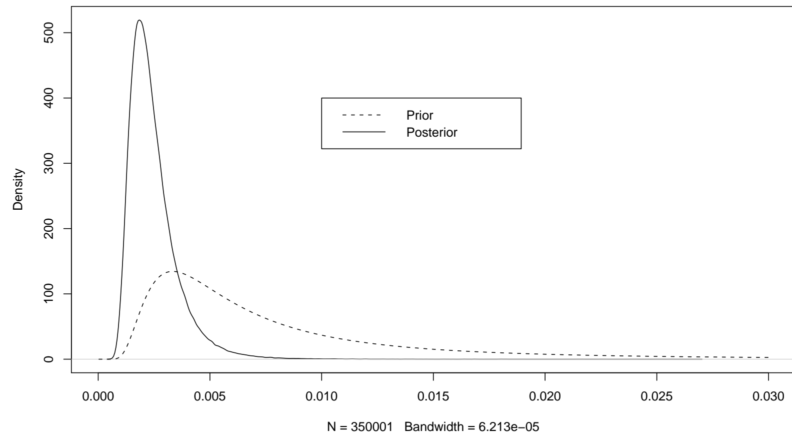

Figure 4.6: Prior and Posterior Distribution for σ_d^2

the corresponding 95% HPD intervals.

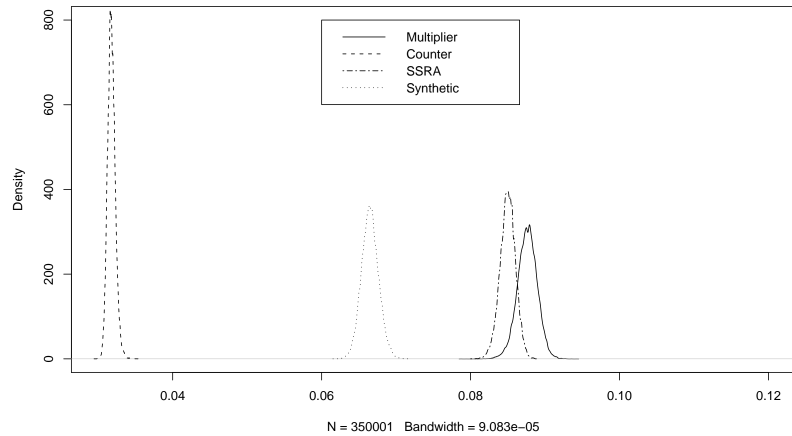

Figure 4.7: Posterior Distributions of the Configuration Bit Sensitivity for the Four Designs

We find the posterior distribution for the true average sensitivity observed at the accelerator (which would include flip flop errors and any other bias due to

Design	95% HPD Credible Interval
Multiplier	0.0851-0.0903
Counter	0.0308-0.0328
Snapshot Recorder	0.0829-0.0870
Synthetic No TMR	0.0643-0.0688

Table 4.8: Credible Intervals for the Average Sensitivity Due to Configuration Bits

accelerator testing) for the ith design by transforming the posterior realizations of $\mu_i + \beta_1 w_i + \lambda$ to

$$\frac{exp(\mu_i + \beta_1 w_i + \lambda)}{1 + exp(\mu_i + \beta_1 w_i + \lambda))}.$$

The posterior distributions for this quantity for each design are shown in Figure 4.8; Table 4.9 lists the corresponding 95% HPD intervals. The posterior distribution for the Multiplier design, which has the least amount of accelerator data, is the least precise.

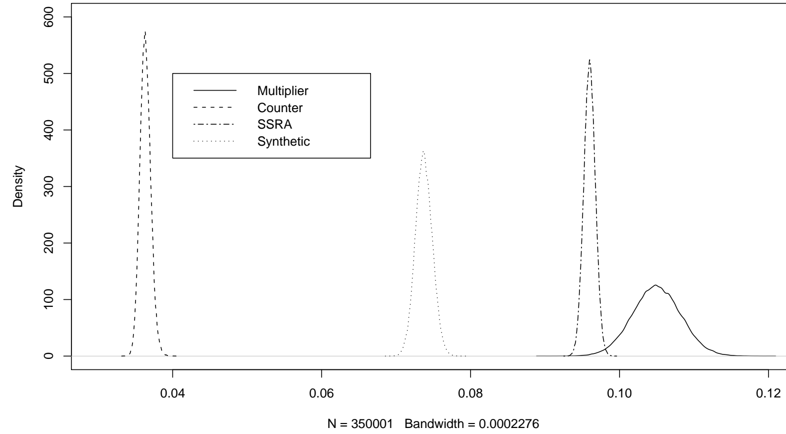

Figure 4.8: Posterior Distributions of the Accelerator Sensitivity for the Four Designs

Design	95% HPD Credible Interval
Multiplier	0.0986-0.1114
Counter	0.0349-0.0378
Snapshot Recorder	0.0944-0.0975
Synthetic No TMR	0.0715-0.0760

Table 4.9: Credible Intervals for the True Accelerator Sensitivity

4.9 Model Prediction

Since the simulator testing is much easier to perform, a statistical model comparing the two systems is especially useful if we can gain information about accelerator testing using only simulator data. We want to predict the sensitivity due to configuration bits using only fault injection data. We also want to predict the behavior of a design at the accelerator given data at the simulator as was done for the deterministic model. The stochastic simulator model allows us to compute posterior predictive distributions for both of these quantities. We consider the same hypothetical design presented in Section 3.9.2. This design has 14,235 flip flops; in simulator testing 22,426,693 output errors were observed in 290,501,200 trials, which corresponds to an observed average sensitivity of 0.0772.

We use the observation from the simulator for the new design to estimate $\mu_{new}|x_{new}$. We know from Sections 4.3 and 4.5 that the likelihood kernel for x_{new} in its reparametrized form is

$$exp\left(x_{new}(\mu^*_{new}) - M\ log(1 + exp(\mu^*_{new}))\right).$$

Using the Metropolis-Hastings algorithm and this likelihood kernel we generate realizations from $\pi(\mu^*_{new}|x_{new})$. By definition $\mu_{new} = \mu^*_{new} - \gamma_{new}$. Using the posterior

distribution of σ_d^2 we generate the posterior predictive distribution for $\gamma_{new}|z$ by generating from $f(\gamma_{new}|\sigma_d^2)\pi(\sigma_d^2|z)$. We subtract the realizations of γ_{new} from the realizations of μ^*_{new} to get posterior predictive realizations for $\mu_{new}|x_{new}$. These realizations are used to predict the average sensitivity due to configuration bits and to predict the behavior of this design at the accelerator.

As discussed in Section 4.8, the distribution of the average sensitivity due to configuration bits is estimated by transforming the posterior realizations of μ_{new} to

$$\frac{exp(\mu_{new})}{1 + exp(\mu_{new}))}.$$

To predict the average sensitivity at the accelerator we transform $\mu_{new} + \beta_1 w_{new} + \lambda$ to

$$\frac{exp(\mu_{new} + \beta_1 w_{new} + \lambda)}{1 + exp(\mu_{new} + \beta_1 w_{new} + \lambda))}.$$

Type	95% HPD Credible Interval
Average Sensitivity from Configuration Bits	0.0702-0.0843
Average Sensitivity at the Accelerator	0.0831-0.1047

Table 4.10: Prediction Intervals for a Hypothetical New Design

Using this model we can predict the counts of output errors we would expect to see at the next accelerator test for a given design and sample size. To predict the expected counts for design i given a sample of size 1,000, we simulate from the posterior predictive distribution

$$f(y_{i(j+1)}|\mu_i, \beta_1, \lambda)\pi(\mu_i, \beta_1, \lambda|z).$$

This is equivalent to simulating from a binomial likelihood with n=1,000 and vector

of probabilities

$$\frac{exp(\mu_i + \beta_1 w_i + \lambda)}{1 + exp(\mu_i + \beta_1 w_i + \lambda)},$$

where the parameters are substituted with their respective vectors of posterior realizations. To predict counts for the new design we replace the posterior realizations μ_i with the posterior predictive realizations of $\mu_{new}|x_{new}$.

Table 4.11 lists 95% equal-tailed credible intervals for the predicted counts for the four designs in the model and the hypothetical new design described earlier.

Design	1,000	5,000	10,000
Multiplier	85-126	473-578	963-1139
Counter	25-49	155-209	325-404
Snapshot Recorder	78-115	439-522	900-1020
Synthetic No TMR	58-91	331-407	682-794
New Design	74-116	404-539	820-1065

Table 4.11: 95% Equal-Tailed Credible Intervals for Output Errors Observed at the Next Accelerator Tests

4.10 Results Using Adjusted Average Sensitivity Data

The stochastic model is re-fit using the adjusted counts in order to see how the estimates of the discrepancy term change. A total discrepancy that includes zero with high probability would validate the effectiveness of the algorithm used at the accelerator to filter out flip flop errors. Three hundred and fifty thousand realizations of the joint posterior distribution are generated after burn-in. The posterior realizations are used to plot posterior distributions for the discrepancy parameters λ and β_1 (Figures 4.9 and 4.10), and for the total discrepancy for each design $\beta_1 w_i + \lambda$ (Figure 4.11). The 95% HPD intervals for the total discrepancy are shown in Table 4.12.

The marginal posteriors of both discrepancy components λ and β_1 include zero as a probable value using the adjusted data; both have shifted to the left as compared to the unadjusted results. The total discrepancy, however, does not include zero in the 95% HPD interval for any of the four designs. In contrast to the adjusted data results using the deterministic simulator model, the results under this model provide evidence of a discrepancy term even after using the adjusted data.

Two possible sources of the remaining discrepancy are failure of the algorithm to identify all flip flop errors and underprediction of the average sensitivity by the simulator. An incomplete identification of flip flop errors by the algorithm would reduce but not eliminate the discrepancy between configuration bit and accelerator sensitivity. If the algorithm is accurate in removing all flip flop errors at accelerator testing, then these results indicate that the simulator is underpredicting the sensitivity due to configuration bits. The assumption that the simulator is an unbiased estimator of errors would then be rejected.

Design	95% HPD credible interval for the bias
Multiplier	0.0406-0.1855
Counter	0.0419-0.0985
Snapshot Recorder	0.0396-0.0957
Synthetic No TMR	0.0133-0.0871

Table 4.12: Estimate of the Discrepancy Between the Simulator and the Accelerator for Four Designs, Adjusted Data

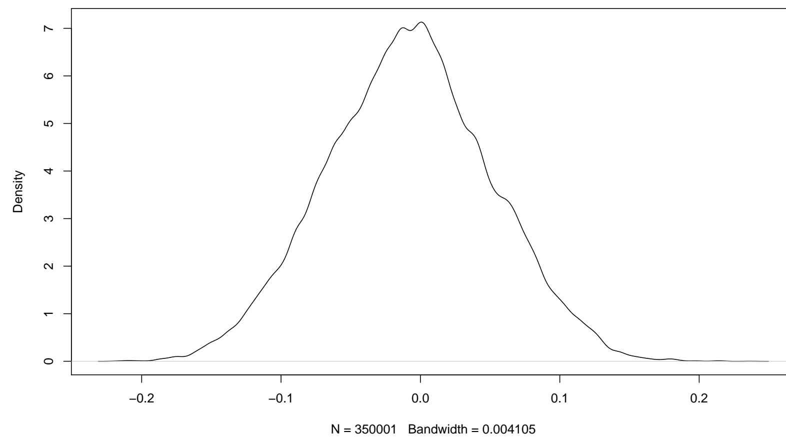

Figure 4.9: Posterior Distribution for λ, Adjusted Data

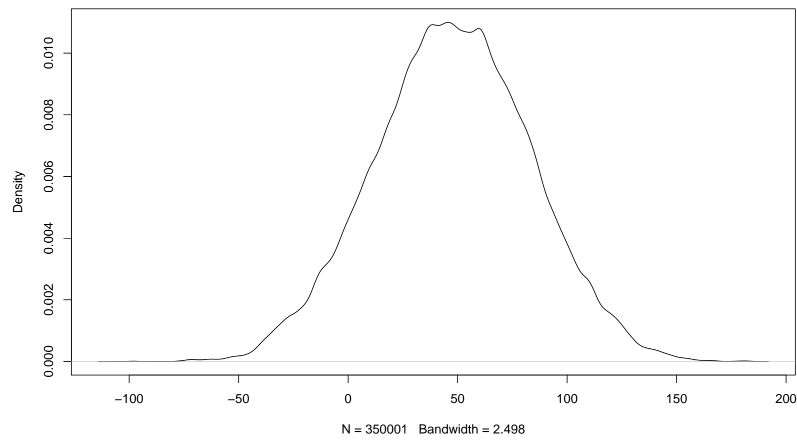

Figure 4.10: Posterior Distribution for β_1, Adjusted Data

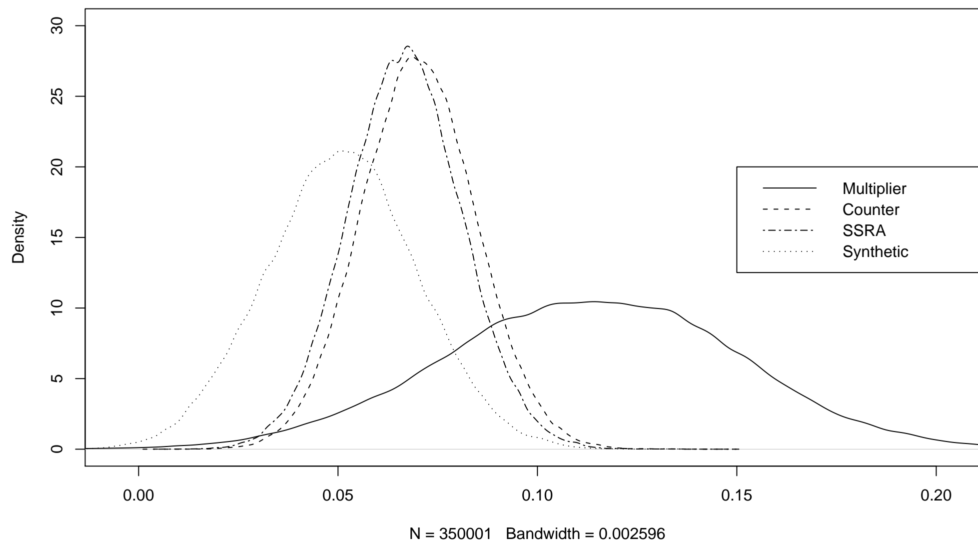

Figure 4.11: Distribution of the Total Discrepancy Term for Each Design, Adjusted Data

Chapter 5

Summary

The deterministic and stochastic simulator models use very different likelihoods and make some different modeling assumptions. While it is difficult to say which type of model is uniformly best, there are certain advantages and disadvantages to each approach. Whether the general deterministic or stochastic model should be used depends on the individual application area.

One possible research consideration when selecting a model is the importance of simplicity in interpretation and estimation. A deterministic simulator model is generally easier to understand and estimate because the field data provides the only source of randomness. In the FPGA example, when no probabilistic modeling of the simulator data is required, the problem is simplified and a normal likelihood can be used. This reduces the complexity of the Bayesian computation. Because of the use of a normal likelihood, the discrepancy term can be modeled on the same scale as the average sensitivity, so its magnitude is straightforward to interpret. Finally, it is simpler to make predictions conditional on the simulator output since the output is treated as a constant.

It is important to balance the desire for simplicity against the accuracy and

fit of the model. The estimates from both models are remarkably similar given their different structure and likelihood assumptions. Under both models, as expected, there is strong evidence of a positive discrepancy term which indicates that the proton accelerator data overestimates the configuration bit sensitivity, as seen in Figures 3.8 and 4.5. The models also concur on estimates for the true average sensitivity at the accelerator (Tables 3.11 and 4.9) and the design behavior at future accelerator testing (Tables 3.10 and 4.11). The prediction intervals from the deterministic model tend to be more precise. Since the deterministic model does not account for any uncertainty associated with the output from the fault injection simulator, this result is not surprising. If the stochastic component of the simulator is negligible, the simplicity of the deterministic model and the accuracy of its predictions for a new design would make it the better choice. If the stochastic component is considered relatively large, the more realistic incorporation of uncertainty justifies using the stochastic model.

Another consideration when choosing a general model is the ability to make good modeling assumptions within the application area. If choosing a likelihood that appropriately models the simulator data makes the model overly complex or poor-fitting as compared to selecting a likelihood for the field data, a deterministic model may be preferred. In the FPGA example, the Liapounov Central Limit Theorem motivates the use of a fairly simple likelihood for the accelerator data; however, the invocation of the theorem is harder to justify for smaller sample sizes. The limited data also makes precise estimation of the measurement error parameters impractical for the Multiplier and Counter designs. Finding an appropriate likelihood for the

simulator data is more challenging due to the correlation and overdispersion in the data. But despite the complexity of the binomial likelihood specification and logit link function, the model performs well in predictions and is more flexible than the deterministic model. For example, unlike the deterministic model, the stochastic model is able to pool data from both the accelerator and the simulator to get an estimate of the average sensitivity due only to configuration bits, as reported in Table 4.8. Were the binomial likelihood to provide a poor fit for either the accelerator or the simulator data, the deterministic model would be a better overall model choice. Since we are able to make better modeling assumptions for the simulator data, the stochastic model seems to be the best choice for the FPGA data.

There are other criteria that could be used to evaluate both types of models depending on the application area. These include considerations such as the relative time and cost of gathering field data versus generating simulator data, the end-users of the model estimates, and desired types of predictions. The considerations that should be given priority are ultimately motivated by the research environment.

Chapter 6

Appendix A: R® Code for the Deterministic Simulator Model

```
#Sensitive Bits Analysis#
#input the data#
Data<-read.csv("C:\\book1d.csv",header=TRUE)
attach(Data)
#Y=vector of accelerator average sensitivity, C=vector of simulator values,
#X=vector of fflops/5810024, M=vector of total ouput errors,#
#Design=vector of design numbers#
#defining functions#

#calculate HPD interval#
interval<-function(data){
ind<-seq(1,round(.05*length(data)))
yp<-sort(data)
hpd.width<-yp[length(yp)-length(ind)+ind]-yp[ind]
ind[hpd.width==min(hpd.width)]
hpd<-c(yp[ind[hpd.width==min(hpd.width)]],
yp[length(yp)-length(ind)+ind[hpd.width==min(hpd.width)]])
return(hpd)
}
```

```
#prediction of the next observation at the accelerator, given a certain number of observations#
preddata<-function(ind,nobs){
data<-rnorm(length(beta),C[ind]+beta*X[ind],sqrt(k[,Design[ind]]/nobs+t))
return(data)
}
#prediction for a new design#
prednew<-function(fflops,sim,nobs){
ratio<-fflops/5810024
data<-rnorm(length(beta),sim+beta*ratio,sqrt(rgamma(length(beta),a,b)/nobs+t))
return(data)
}
#prediction of the true accelerator sensitivity for existing designs#
predtrue<-function(ind){
data<-rnorm(length(beta),C[ind]+beta*X[ind],sqrt(t))
return(data)
}
#prediction of true accelerator sensitivity for a new design#
predtruenew<-function(fflops,sim){
ratio<-fflops/5810024
data<-rnorm(length(beta),sim+beta*ratio,sqrt(t))
return(data)
}
#specifying prior distributions#


#prior parameters#
#t#
mean<-.00001
```

```
var<-.00000000006
tb<-mean/var
ta<-tb*mean
x<-seq(0,.00002,len=1000)
test<-dgamma(x,ta,rate=tb)
plot(x,test)
#k#
mean<-.9*.1
var<-.002
kb<-mean/var
ka<-kb*mean
x<-seq(0,1,.001)
test<-dgamma(x,ka,rate=kb)
plot(x,test)
#b#
mean<-kb
var<-5
bb<-mean/var
ba<-bb*mean
x<-seq(0,100,.1)
test<-dgamma(x,ba,rate=bb)
plot(x,test)
#a#
mean<-ka
var<-1
ab<-mean/var
aa<-ab*mean
```

```
x<-seq(0,10,.1)
test<-dgamma(x,aa,rate=ab)
plot(x,test)
a.tau<-ta
b.tau<-tb
mean.beta<-10
var.beta<-25
x<-seq(-5,20, len=1000)
plot(x,dnorm(x,mean.beta,sqrt(var.beta)),type='l')
a.a<-aa
a.b<-ab
b.a<-ba
b.b<-bb
#complete conditionals#
lcck<-function(k,t,beta,a,b,id){-1/2*sum(log(k/M[Design==id]+t))-sum((Y[Design==id]
-C[Design==id]-beta*X[Design==id])^2/2/(k/M[Design==id]+t))+(a-1)*log(k)-b*k}
lcct<-function(krep,t,beta){-1/2*sum(log(krep/M+t))-
sum((Y-C-beta*X)^2/2/(krep/M+t))+(a.tau-1)*log(t)-b.tau*t}
lcca<-function(k,b,a,nd){nd*a*log(b)-nd**log(gamma(a))+(a-1)*sum(log(k))+
(a.a-1)*log(a)-a.b*a}
lccb<-function(k,b,a,nd){nd*a*log(b)-b*sum(k)+(b.a-1)*log(b)-b.b*b}
lccbeta<-function(krep,t,beta){-sum((Y-C-beta*X)^2/2/(krep/M+t))-
(beta-mean.beta)^2/2/var.beta}

#Initialize Vectors#
burn<-5000
length<-100000
```

```
nd<-length(unique(Design))
k<-matrix(.1,ncol=nd,nrow=(length+burn))
t<-numeric()
beta<-numeric()
a<-numeric()
b<-numeric()
#Starting Values#
k[1,1]<-.5
t[1]<-.00001
beta[1]<-5
a[1]<-2
b[1]<-5
candsig.k<-.1
candsig.t<-.000001
candsig.a<-3
candsig.b<-15
candsig.beta<-3
#Gibbs Sampler#
for(i in 2:(length+burn)){
  #update for k#
  for (j in 1:nd){
    k[i,j]<-k[i-1,j]
    old<-k[i-1,j]
    cand<-rnorm(1,old,candsig.k)
    if(cand>0){
    llo<-lcck(old,t[i-1],beta[i-1],a[i-1],b[i-1],j)
    lln<-lcck(cand,t[i-1],beta[i-1],a[i-1],b[i-1],j)
```

```
    uu<-runif(1,0,1)
    if(log(uu)<(lln-llo)){k[i,j]<-cand}}
   }

k1<-rep(k[i,1],sum(Design==1))
k2<-rep(k[i,2],sum(Design==2))
k3<-rep(k[i,3],sum(Design==3))
k4<-rep(k[i,4],sum(Design==4))
krep<-c(k1,k2,k3,k4)

  #update for t#
  t[i]<-t[i-1]
  old<-t[i-1]
  cand<-rnorm(1,old,candsig.t)
  if(cand>0){
  llo<-lcct(krep,old,beta[i-1])
  lln<-lcct(krep,cand,beta[i-1])
  uu<-runif(1,0,1)
  if(log(uu)<(lln-llo)){t[i]<-cand}}

  #update for beta#
  beta[i]<-beta[i-1]
  old<-beta[i-1]
  cand<-rnorm(1,old,candsig.beta)
  llo<-lccbeta(krep,t[i],old)
  lln<-lccbeta(krep,t[i],cand)
  uu<-runif(1,0,1)
```

```
  if(log(uu)<(lln-llo)){beta[i]<-cand}


  #update for a#
  a[i]<-a[i-1]
  old<-a[i-1]
  cand<-rnorm(1,old,candsig.a)
  if(cand>0){
  llo<-lcca(k[i,],b[i-1],old,nd)
  lln<-lcca(k[i,],b[i-1],cand,nd)
  uu<-runif(1,0,1)
  if(log(uu)<(lln-llo)){a[i]<-cand}}


  #update for b#
  b[i]<-b[i-1]
  old<-b[i-1]
  cand<-rnorm(1,old,candsig.b)
  if(cand>0){
  llo<-lccb(k[i,],old,a[i],nd)
  lln<-lccb(k[i,],cand,a[i],nd)
  uu<-runif(1,0,1)
  if(log(uu)<(lln-llo)){b[i]<-cand}}



}


#convergence diagnostics#
par(mfrow=c(2,2))
```

```
plot(a[0:10000],type='l')
plot(b[0:10000],type='l')
plot(beta[0:10000],type='l')
plot(t[0:10000],type='l')
plot(k[0:10000,1],type='l')
plot(k[0:10000,2],type='l')
plot(k[0:10000,3],type='l')
plot(k[0:10000,4],type='l')

#parameters of interest#
beta<-beta[burn:(length+burn)]
a<-a[burn:(length+burn)]
b<-b[burn:(length+burn)]
t<-t[burn:(length+burn)]
k<-k[burn:(length+burn),]

#plotting posterior distributions#
par(mfrow=c(1,1))

plot(density(beta),xlim=c(-5,20),main=" ")
x<-seq(-5,20,len=1000)
lines(x,dnorm(x,mean.beta,sqrt(var.beta)),lty=2, type='l')
legend(10,.4,c('Prior','Posterior'),lty=c(2,1))

plot(density(t),main=' ',ylim=c(0,300000))
x<-seq(0,.00002,len=1000)
test<-dgamma(x,ta,rate=tb)
```

```
lines(x,test,lty=2)

legend(.000014,150000,c('Prior','Posterior'),lty=c(2,1))


#model predictions#

#simulating the posterior predictive distribution of the discrepancy term#

d1<-rnorm(length(beta),X[1]*beta,sqrt(t))

d2<-rnorm(length(beta),X[3]*beta,sqrt(t))

d3<-rnorm(length(beta),X[6]*beta,sqrt(t))

d4<-rnorm(length(beta),X[39]*beta,sqrt(t))

plot(density(d4),main=" ", lty=6)

lines(density(d3), lty=2)

lines(density(d2),lty=1)

lines(density(d1),lty=9)

legend(-.01,150,c('Multiplier','Counter', 'Snapshot','Synthetic'),lty=c(9,1,2,6))

interval(d1)

interval(d2)

interval(d3)

interval(d4)


#prediction of behavior at next accelerator tests#

mult<-preddata(1,5000)

interval(mult)

count<-preddata(3,5000)

interval(count)

snap<-preddata(6,5000)

interval(snap)

synth<-preddata(39,5000)
```

```
interval(synth)

new<-prednew(14385,0.0772,5000)
interval(new)

#predictions for hold-out data#
lsfr<-prednew(8640,0.0431,1069)
interval(lsfr)
vmult36<-prednew(3744,0.0524,3003)
interval(vmult36)
vmult72<-prednew(15264,0.1540,57)
interval(vmult72)
vmult72<-prednew(15264,0.1540,5754)
vmult72<-prednew(15264,0.1540,1919)
vmult72<-prednew(15264,0.1540,1733)
vmult72<-prednew(15264,0.1540,2635)
vmult72<-prednew(15264,0.1540,9139)

#prediction of the true accelerator sensitivity#
mult<-predtrue(1)
interval(mult)
count<-predtrue(3)
interval(count)
snap<-predtrue(6)
interval(snap)
synth<-predtrue(39)
interval(synth)
```

new<−predtruenew(14385,0.0772)

interval(**new**)

Chapter 7

Appendix B: R® Code for the Stochastic Simulator Model

```
attach(Counts)
#F=vector of fflops/5810024, X=vector of output errors, N=vector of Total Upsets,#
#Design=vector of design Numbers#

#formatting the data for coding purposes − separating simulator and accelerator data.
#Assumes simulator data is first in all data vectors.#
F<−F[1:52]
XA<−X[1:52]
DesignA<−Design[1:52]
N1<−N[1:52]
XB<−X[53:63]
DesignB<−Design[53:63]
N2<−N[53:63]

#defining functions#
#HPD interval#
interval<−function(data){
ind<−seq(1,round(.05*length(data)))
yp<−sort(data)
```

```
hpd.width<-yp[length(yp)-length(ind)+ind]-yp[ind]

ind[hpd.width==min(hpd.width)]

hpd<-c(yp[ind[hpd.width==min(hpd.width)]],

yp[length(yp)-length(ind)+ind[hpd.width==min(hpd.width)]])

return(hpd)

}

#equal-tailed interval#

interval1<-function(data){

equaltailed<-quantile(data,c(.025,.975))

return(equaltailed)

}

#predicting for a new design#

prednew<-function(xnew,nnew,fnew){

fnew<-fnew/5810024

out<-numeric()

candsig<-.003

out[1]<--2.3

for (i in 2:(length+burn)) {

out[i]<-out[i-1]

cand<-rnorm(1,out[i-1],candsig)

accept<-g(cand,xnew,nnew)-g(out[i-1],xnew,nnew)

u<-runif(1,0,1)

if (log(u)<accept) {out[i]<-cand}

}

out1<-out[burn:(length+burn)]

munew<-out1-rnorm(length(out1),0,sqrt(vardel1[burn:(length+burn)]))

pnew<-(exp(munew+beta1[burn:(length+burn)]*fnew+lam1[burn:(length+burn)])
```

```
/(1+exp(munew+beta1[burn:(length+burn)]*fnew+lam1[burn:(length+burn)])))
return(pnew)
}


#defining parameters needed for likelihood specification#
phi.a<-1
phi.s<-1


#specification of priors#
#hyperparameters for u#
m<-log(.07/.93)
v<-((log(.09/.91)-log(.05/.95))/6)^2
x<-seq(-3,3,.01)
lines(x,dnorm(x,m,sqrt(v)))
#hyperparameters for s#
mean<-4*((log(.09/.91)-log(.05/.95))/6)^2 #we have four designs#
var<-8*((log(.09/.91)-log(.05/.95))/6)^2
a<-mean^2/var+2
b<-mean*a-mean
#hyperparameters for beta#
m.beta<-225
v.beta<-5500
x<-seq(-20,470,1)
plot(x,dnorm(x,m.beta,sqrt(v.beta)),type='l',ylim=c(0,.006))
#hyperparameters for lamda#
m.lam<-0
v.lam<-0.04
```

```
x<-seq(-.7,.7,.01)

plot(x,dnorm(x,m.lam,sqrt(v.lam)),type='l', ylim=c(0,2))

#hyperparameters for vardel - overdispersion parameter#

#inverse gamma#

mean<-.01

var<-.5

a.vardel<-mean^2/var+2

b.vardel<-mean*a.vardel-mean

#pvardel<-1/rgamma(1000000,a.vardel,rate=b.vardel)#


#log complete conditionals#

lccmu<-function(mu,beta,lam,u,s,mustar,vardel,id){sum((XA[DesignA==id]*
(mu+beta*F[DesignA==id]+lam)-N1[DesignA==id]*
log(1+exp(mu+beta*F[DesignA==id]+lam)))/phi.a)-
(mu-u)^2/2/s-sum((mustar[DesignB==id]-mu))^2/2/(vardel)}

lccmustar<-function(mustar,u,s,vardel,mu,obs){sum((XB[obs]*(mustar)-N2[obs]*
log(1+exp(mustar)))/phi.s)-(mustar-mu)^2/2/(vardel)}

lccbeta<-function(murepA,beta,lam){sum((XA*(murepA+beta*F+lam)-N1*
log(1+exp(murepA+beta*F+lam)))/phi.a)-(beta-m.beta)^2/2/v.beta}

lcclam<-function(murepA,beta,lam){sum((XA*(murepA+beta*F+lam)-N1*
log(1+exp(murepA+beta*F+lam)))/phi.a)-(lam-m.lam)^2/2/v.lam}


#Initialize Vectors#

burn<-200000

length<-350000

mu<-matrix(0,ncol=4,nrow=(length+burn))

mustar<-matrix(0,ncol=11,nrow=(length+burn))
```

```
u<-numeric()
s<-numeric()
beta<-numeric()
lam<-numeric()
del<-numeric()
vardel<-numeric()

#Starting Values#
mu[1,]<-rep(m,4)
mustar[1,]<-rep(m,11)
u[1]<-m
s[1]<-mean
beta[1]<-30
lam[1]<-.1
del[1]<-0
vardel[1]<-.01
candsig.mu<-.2
candsig.mustar<-.003
candsig.beta<-40
candsig.lam<-.1

#Gibbs Sampler#
for(i in 2:(length+burn)){

 #update for mu#
  for (j in 1:4){
    mu[i,j]<-mu[i-1,j]
```

```
    old<-mu[i-1,j]
    cand<-rnorm(1,old,candsig.mu)
    llo<-lccmu(old,beta[i-1],lam[i-1],u[i-1],s[i-1],mustar[i-1,],vardel[i-1],j)
    lln<-lccmu(cand,beta[i-1],lam[i-1],u[i-1],s[i-1],mustar[i-1,],vardel[i-1],j)
    uu<-runif(1,0,1)
    if(log(uu)<(lln-llo)){mu[i,j]<-cand}
   }

mu1<-rep(mu[i,1],sum(DesignA==1))
mu2<-rep(mu[i,2],sum(DesignA==2))
mu3<-rep(mu[i,3],sum(DesignA==3))
mu4<-rep(mu[i,4],sum(DesignA==4))
mu1B<-rep(mu[i,1],sum(DesignB==1))
mu2B<-rep(mu[i,2],sum(DesignB==2))
mu3B<-rep(mu[i,3],sum(DesignB==3))
mu4B<-rep(mu[i,4],sum(DesignB==4))
murepA<-c(mu1,mu2,mu3,mu4)
murepB<-c(mu1B,mu2B,mu3B,mu4B)

   #update for mustar#
  for (j in 1:11){
    mustar[i,j]<-mustar[i-1,j]
    old<-mustar[i-1,j]
    cand<-rnorm(1,old,candsig.mustar)
    llo<-lccmustar(old,u[i-1],s[i-1],vardel[i-1],murepB[j],j)
    lln<-lccmustar(cand,u[i-1],s[i-1],vardel[i-1],murepB[j],j)
    uu<-runif(1,0,1)
```

```
    if(log(uu)<(lln−llo)){mustar[i,j]<−cand}

  }

#update for u#
newmean<−(sum(mu[i,])*v+m*s[i−1])/(s[i−1]+4*v)
newvar<−s[i−1]*v/(s[i−1]+4*v)
u[i]<−rnorm(1,newmean,sqrt(newvar))

#update for s#
anew <− a + 4/2
bnew <− (b + sum((mu[i,]−u[i])^2)/2)
s[i] <− 1/rgamma(1,anew,rate=bnew)

#update for beta#
 beta[i]<−beta[i−1]
 old<−beta[i−1]
 cand<−rnorm(1,old,candsig.beta)
 llo<−lccbeta(murepA,old,lam[i−1])
 lln<−lccbeta(murepA,cand,lam[i−1])
 uu<−runif(1,0,1)
 if(log(uu)<(lln−llo)){beta[i]<−cand}

#update for lam#
 lam[i]<−lam[i−1]
 old<−lam[i−1]
 cand<−rnorm(1,old,candsig.lam)
```

```
  llo<-lcclam(murepA,beta[i],old)

  lln<-lcclam(murepA,beta[i],cand)

  uu<-runif(1,0,1)

  if(log(uu)<(lln-llo)){lam[i]<-cand}


 #update for vardel#

 anew <- a.vardel + 11/2

 bnew <- (b.vardel + sum((mustar[i,]-murepB)^2)/2)

 vardel[i] <- 1/rgamma(1,anew,rate=bnew)



}


#convergence diagnostic plots#

beta2<-numeric()

lam2<-numeric()

mu2<-matrix(0,3000,4)

vardel2<-numeric()

#convergence plots#

for (i in 1:3000){

beta2[i]<-beta[100*i]

lam2[i]<-lam[100*i]

vardel2[i]<-vardel[100*i]

mu2[i,]<-mu[100*i,]

}


par(mfrow=c(3,1))
```

```r
plot(beta2,type='l')

plot(lam2,type='l')

plot(vardel2[3:3000],type='l')

par(mfrow=c(2,2))

plot(mu2[,1],type='l')

plot(mu2[,2],type='l')

plot(mu2[,3],type='l')

plot(mu2[,4],type='l')


#removing the burn#

mu1<-mu[burn:(length+burn),]

mustar1<-mustar[burn:(length+burn),]

u1<-u[burn:(length+burn)]

s1<-s[burn:(length+burn)]

beta1<-beta[burn:(length+burn)]

lam1<-lam[burn:(length+burn)]

vardel1<-vardel[burn:(length+burn)]


#posterior distributions#

par(mfrow=c(1,1))

x<-seq(-.7,.7,.01)

plot(x,dnorm(x,m.lam,sqrt(v.lam)),lty=2, type='l', ylim=c(0,8), main=' ')

lines(density(lam1,adjust=2), lty=1)

legend(.2,6,c('Prior','Posterior'),lty=c(2,1))


plot(density(beta1,adjust=2),main=' ',ylim=c(0,.012), xlim=c(-50,420))

x<-seq(-20,470,1)
```

```
lines(x,dnorm(x,m.beta,sqrt(v.beta)),lty=2)
legend(200,.01,c('Prior','Posterior'),lty=c(2,1))


plot(density(vardel1),main=' ',xlim=c(0,.03))
library(MCMCpack)
x<-seq(0,.03,length=1000)
lines(x,dinvgamma(x,a.vardel,rate=b.vardel),lty=2)
legend(.01,400,c('Prior','Posterior'),lty=c(2,1))
plot(density(mu1[,1]))
plot(density(s1))
plot(density(u1))


#posterior distribution of accelerator discrepancy terms#
plot(density(beta1*F[1]+lam1), main=' ',xlim=c(-.005,.3),ylim=c(0,30))
lines(density(beta1*F[3]+lam1), lty=2)
lines(density(beta1*F[6]+lam1), lty=6)
lines(density(beta1*F[39]+lam1), lty=9)
legend(.2,25,c('Multiplier','Counter','SSRA','Synthetic'),lty=c(1,2,6,9))


interval(beta1*F[1]+lam1)
interval(beta1*F[3]+lam1)
interval(beta1*F[6]+lam1)
interval(beta1*F[39]+lam1)


#for the true proportion of sensitive configuration bits#
plot(density(exp(mu1[,1])/(1+exp(mu1[,1]))),
xlim=c(.03,.12),ylim=c(0,800),main=' ')
```

```
lines(density(exp(mu1[,2])/(1+exp(mu1[,2]))),lty=2)
lines(density(exp(mu1[,3])/(1+exp(mu1[,3]))),lty=6)
lines(density(exp(mu1[,4])/(1+exp(mu1[,4]))),lty=9)
legend(.06,800,c('Multiplier','Counter','SSRA','Synthetic'),lty=c(1,2,6,9))

interval(exp(mu1[,4])/(1+exp(mu1[,4])))

#for the true proportion of sensitive bits at the accelerator#
plot(density(pmult<-(exp(mu1[,1]+beta1*F[1]+lam1)/(1+exp(mu1[,1]+beta1*F[1]+lam1)))),
xlim=c(.03,.12),ylim=c(0,600),main=' ')
lines(density(pcount<-(exp(mu1[,2]+beta1*F[3]+lam1)/(1+exp(mu1[,2]+
beta1*F[3]+lam1)))),lty=2)
lines(density(pssra<-(exp(mu1[,3]+beta1*F[6]+lam1)/(1+exp(mu1[,3]+
beta1*F[6]+lam1)))),lty=6)
lines(density(psynth<-(exp(mu1[,4]+beta1*F[39]+lam1)/(1+exp(mu1[,4]+
beta1*F[39]+lam1)))),lty=9)
legend(.04,500,c('Multiplier','Counter','SSRA','Synthetic'),lty=c(1,2,6,9))

interval(pmult)
interval(pcount)
interval(pssra)
interval(psynth)

#predicts expected counts of accelerator behavior at next test#
pmult1000<-rbinom(length(lam1),1000,pmult)
pmult5000<-rbinom(length(lam1),5000,pmult)
pmult10000<-rbinom(length(lam1),10000,pmult)
```

```
interval1(pmult1000)

interval1(pmult5000)

interval1(pmult10000)


pcount1000<-rbinom(length(lam1),1000,pcount)

pcount5000<-rbinom(length(lam1),5000,pcount)

pcount10000<-rbinom(length(lam1),10000,pcount)


interval1(pcount1000)

interval1(pcount5000)

interval1(pcount10000)


pssra1000<-rbinom(length(lam1),1000,pssra)

pssra5000<-rbinom(length(lam1),5000,pssra)

pssra10000<-rbinom(length(lam1),10000,pssra)


interval1(pssra1000)

interval1(pssra5000)

interval1(pssra10000)


psynth1000<-rbinom(length(lam1),1000,psynth)

psynth5000<-rbinom(length(lam1),5000,psynth)

psynth10000<-rbinom(length(lam1),10000,psynth)


interval1(psynth1000)

interval1(psynth5000)
```

```
interval1(psynth10000)


#prediction for a new design with the following characteristics...#
#14,235 flip flops and a simulator sensitivity of 0.0772#
fnew<-14235/5210024
nnew<-290501200
xnew<-round(nnew*0.0772)


#simulate mustar for this observation#
g<-function(mustar,xnew,nnew){xnew*(mustar)-nnew*log(1+exp(mustar))}
out<-numeric()
candsig<-.003
out[1]<--2.3
for (i in 2:(length+burn)) {
out[i]<-out[i-1]
cand<-rnorm(1,out[i-1],candsig)
accept<-g(cand,xnew,nnew)-g(out[i-1],xnew,nnew)
u<-runif(1,0,1)
if (log(u)<accept) {out[i]<-cand}
}
out1<-out[burn:(length+burn)]
munew<-out1-rnorm(length(out1),0,sqrt(vardel1))
pnew<-(exp(munew)/(1+exp(munew)))
pnew<-(exp(munew+beta1*fnew+lam1)/(1+exp(munew+beta1*fnew+lam1)))
interval(pnew)
pnew1000<-rbinom(length(pnew),1000,pnew)
pnew5000<-rbinom(length(pnew),5000,pnew)
```

```
pnew10000<-rbinom(length(pnew),10000,pnew)


interval1(pnew1000)

interval1(pnew5000)

interval1(pnew10000)


#prediction of accelerator behavior for hold-out data#

length<-100000

burn<-20000

plsfr<-prednew(30446011,581002440,8640)

pvmult36<-prednew(25031616,581002440,3744)

pvmul72<-prednew(357898191,2324011200,15264)

plsfr1069<-rbinom(length(plsfr),1069,plsfr)

interval1(plsfr1069)

pvmult363003<-rbinom(length(pvmult36),3003,pvmult36)

interval1(pvmult363003)

pvmult72a<-rbinom(length(pvmul72),9139,pvmul72)

interval1(pvmult72a)
```

Bibliography

Bayarri, M., Berger, J., Higdon, D., Kennedy, M., Kottas, A., Paulo, R., Sacks, J., Cafeo, J., Cavendish, J., Lin, C., and Tu, J. (2002), "A Framework for Validation of Computer Models," Technical Report 128, National Institute of Statistical Sciences.

Browne, W. J., Subramanian, S., Jones, K., and Goldstein, H. (2005), "Variance Partitioning in Multilevel Logistic Models that Exhibit Overdispersion," *Journal of the Royal Statistical Society Series A*, 168, 599–613.

Caffrey, M., Graham, P., Johnson, E., Rollins, N., and Wirthlin, M. (2003a), "Accelerator Validation of an FPGA SEU Simulator," *IEEE Transactions on Nuclear Science*, 50.

— (2003b), "The Reliability of FPGA Circuit Designs in the Presence of Radiation Induced Configuration Upsets," in *Proceedings of the 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*.

Craig, P., Goldstein, M., Seheult, A., and Smith, J. (1997), *Case Studies in Bayesian Statistics III*, New York: Springer, chap. Pressure Matching for Hydrocarbon Reservoirs: A Case Study in the Use of Bayes Linear Strategies for Large Computer Experiments, with Discussion.

Craig, P. S., Goldstein, M., Rougier, J. C., and Seheult, A. H. (2001), "Bayesian Forcasting for Complex Systems using Computer Simulators," *Journal of the American Statistical Association*, 96.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2004), *Bayesian Data Analysis*, Boca Raton: Chapman and Hall, 2nd ed.

Goldstein, M. and Rougier, J. (2004), "Probabilistic Formulations for Transferring Inferences from Mathematical Models to Physical Systems," *SIAM Journal on Scientific Computing*, 26, 467–487.

Higdon, D., Kennedy, M., Cavendish, J., Cafeo, J., and Ryne, R. (2004), "Combining Field Data and Computer Simulations for Calibration and Prediction," *SIAM Journal on Scientific Computing*, 26, 448–466.

Hill, M. C. (1998), “Methods and Guidelines for Effective Model Calibration,” Water-Resources Investigations Report 98-4005, U.S. Geological Survey.

Johnson, E. (2005), “Estimating the Dynamic Sensitive Cross Section of an FPGA Design Through Fault Injection,” Master’s thesis, Brigham Young University.

Kennedy, M. C. and O’Hagan, A. (2001), “Bayesian Calibration of Computer Models,” *Journal of the Royal Statistical Society B*, 63, 425–464.

Kupper, L. L., Portier, C., Hogan, M. D., and Yamamoto, E. (1986), “The Impact of Litter Effects on Dose-Response Modeling in Teratology,” *Biometrics*, 42, 85–98.

Lehmann, E. L. (1999), *Elements of Large Sample Theory*, New York: Springer-Verlag.

Oberkampf, W. L., Trucano, T. G., and Hirsch, C. (2002), “Verification and Validation for Modeling and Simulation in Compuational Science and Engineering Applications,” in *Foundations for Verification and Validation in the 21st Century Workshop*.

Qian, Z., Seepersad, C., Joseph, V., Allen, J., and Wu, C. (2004), “Building Surrogate Models Based on Detailed and Approximate Simulations,” Discussion Papers - Georgia Institute of Technology, School of Industrial and Systems Engineering, Statistics Group.

Santner, T. J., Williams, B. J., and Notz, W. (2003), *The Design and Analysis of Computer Experiments*, New York: Springer.