

Electrical Design Space Exploration for High Speed Servers

Caleb Wesley¹

ECE Department, North Carolina State University

Bhyrav Mutnury², Nam Pham³, Erdem Matoglu⁴, Moises Cases⁵

IBM Systems and Technology Group

11400 Burnet Road, Austin, TX – 78758

Phone: (512) 823-6918; Fax: (512) 823 - 5938

¹cjwesley@ncsu.edu, ²bmutnury, ³npham, ⁴matoglu, ⁵cases@us.ibm.com

Abstract

Today's high-speed systems are characterized by a multitude of design parameters due to their complexity. Therefore, performing a thorough design space exploration is computationally exhaustive. This is primarily the result of the number of variables (electrical parameters) needed to do a comprehensive analysis. In this paper, Genetic Algorithm and Design Space Exploration (GADSE) is described as a solution to address the above problem. GADSE constitutes features like generating spice decks automatically to perform brute-force Monte-Carlo simulation. Multiple sockets and cores in a system can be leveraged to run multiple simulations in parallel using GADSE. GADSE also features statistical approaches like Taguchi and Central Composite Design (CCD) and Genetic Algorithm based optimization methods to dramatically reduce the number of simulations needed to explore the design space.

1. Introduction

Today's high-speed systems are characterized by a multitude of design parameters due to their complexity. Therefore, performing a thorough design space exploration is computationally exhaustive. This is primarily the result of the number of variables (electrical parameters) needed to do a comprehensive analysis. Approaches such as Monte Carlo and design of experiment (DoE) methods[1] such as Taguchi and Central Composite Design (CCD) are currently used by designers for high-speed system analysis. Even though these methods reduce the design space, they are mostly used in a non-automated fashion. Generating the limited set of Spice decks, simulating the decks, and post-processing the data are tedious tasks by themselves and when done manually can be very tedious. There is a need to automate the entire process from Spice circuit generation to post-processing the output files and performing Sensitivity Analysis. All these steps today can be very exhaustive. Genetic Algorithm and Design Space Exploration (GADSE) is a proposal to address the problem.

A memory subsystem in a high-end server platform is used in this paper to show the features of GADSE. GADSE is a tool for creating, running, and analyzing whole sets of decks for design space exploration. It supports DoE as a way of running small sample sets. It also supports Genetic Algorithms (GA) for shrinking the design space. Sensitivity Analysis (SA) is used with all three methods to give the designer a sense of important factors in the design.

In section 2, DoE techniques like Taguchi and CCD are discussed, along with Genetic Algorithms. In section 3, GADSE tool is described in detail with focus on its benefits to the designer through automation. A test case using GADSE is described and the results are shown in section 4.

2. Statistical Approaches

Figure 1 shows a typical DDR-2 memory subsystem with drivers, transmission lines, and receivers. Such a system has numerous variables. Variations can include transmission line impedance, transmission line length, driver and receiver strengths, etc. As the number of variations increase the search space grows exponentially. Thus, there is a need for methods to find the worst case corners without having to search the entire space.

One common method used by current designers is that of Monte Carlo statistical approach. This is a brute force exhaustive search technique where a simulation is randomly selected and ran until the entire space has been traversed. The obvious problem with this approach is that it scales linearly with the size of the search space making it unsuitable for problems with a large design space.

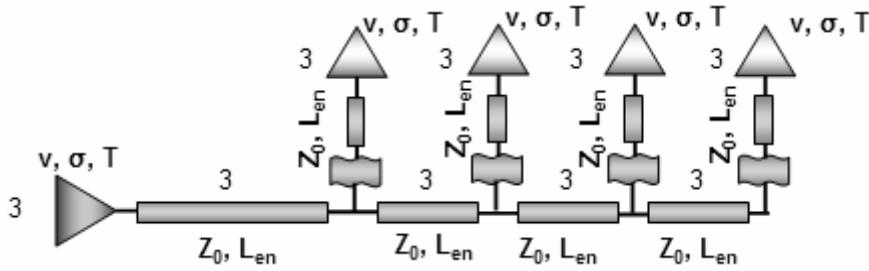


Figure 1 – DDR-2 Memory Example

A reductive method involves estimating points across a linear space that will give a good description of the entire space. Design of Experiment approaches like Taguchi and CCD use this for a set number of factors (variables) and levels (variations). A typical example involves the designer choosing high, typical, and low settings for a group of parameters. This amounts to a multi-factor 3-level design. Figure 2 shows an example matrix for a 4-factor 3-level design. Pre-made normalized matrices are used that describe which experiments are necessary to run for that number of factors and levels. The matrices include a set of orthogonal arrays. The creation of which is a complex procedure that most designers do not have the knowledge and time to go through. Thus they are limited to a handful of factor and level variations that they have access to.

Another problem with the Taguchi and CCD DoE approaches are that they rely on a linear or weakly non-linear design space. Thus it becomes complicated for non-linear systems. For non-linear systems, Genetic Algorithms is a solution that works well[2][3]. They rely on natural genetics and survival of the fittest to achieve the desired result. A typical execution of a GA starts with an initial group of variable settings called a population. Each population is considered a generation. The variable settings for one member of a population is called a chromosome. These chromosomes are typically converted into a string of bits (1s and 0s) called genes for easier manipulation. After the initial population is set up, a fitness function is used to determine which chromosomes are the “best” or closest to the optimal solution. Each chromosome is fed into the function to obtain an objective ranking of each chromosome. The fitness function can be suited to lean the GA toward the desired results.

After the chromosomes have been ranked, a method for determining which chromosomes to retain is used to create a new population. One common method and the one used in this paper is a roulette wheel, which gives each chromosome a percentage value based on their output from the fitness function. Those chromosomes with a better fitness value are given a larger percentage of the wheel. A generator is used to spin the wheel and randomly select two or more of the chromosomes. To mimic typical genetics these chromosomes can be interchanged in various ways in what is called crossover. There is also a chance that a chromosome can acquire a small random change to its genes to mimic mutation. The chance of mutation is usually low, but is necessary to keep the GA from attaining to a local minimal or maxima rather than the overall global minima or maxima. Certain improvements like Master-to-Slave Nonlinear Genetic Algorithm[4] which focuses on a nonlinear mutation operator that resides only in the global and not local search space can be made improve nonlinear accuracy. An example of crossover and mutation is shown in Figure 3. This process is repeated until the population for the new generation is filled. A common addition to the GA is an idea called elitism where the best chromosome in each generation is always carried over to the next generation without any change even at the expense of other chromosomes. By doing this it ensures that the GA will gradually converge on a value rather than oscillating.

+1	-1	-1	+1
-1	+1	-1	+1
+1	+1	-1	+1
-1	-1	+1	+1
+1	-1	+1	+1
-1	+1	+1	+1
+1	+1	+1	+1
0	0	0	0
+1	0	0	0
-1	0	0	0

Figure 2 - CCD Matrix Template

<u>Crossover</u>	<u>Mutation</u>
001 010 100	001 010 100
110 010 111	011 010 101
100 110 010	
001 010 010	
110 110 100	
100 010 111	

Figure 3 – Crossover/Mutation Example

3. Genetic Algorithm for Design Space Exploration (GADSE)

GADSE contains a full set of tools to aid the designer with Spice simulations. It takes, as input, a deck template with certain parameters signaled as variables by the designer by surrounding them with two question marks. The variables are read into the GUI as displayed in Figure 4. The main focus is on a Genetic Algorithm for quickly finding the best and/or worst case corner values. Figure 5 shows the GA settings within the GUI. Options within the GUI allow for changing probabilities for crossover and mutation and also for using elitism. The fitness function for the best case eye and worst case eye are as follows:

$$\begin{aligned} & \text{Best Case: Height} + \text{Width}; \\ & \text{Worst Case } (\text{maxHeight} - \text{Height})^2 + (\text{maxWidth} - \text{Width})^2 \end{aligned}$$

Squaring in the worst case calculation ensures positive values. Maximum and minimum height and width values are required to be set in the GUI. These values are used for determining the worst case fitness function and also to calculate corner violations which are outputted to a file for the designer to analyze. To stop the GA a value is given for the number of consecutive generations without any increase in the best fitness value.

A GA theoretically improves each generation up to the optimal point. Biasing the initial population can reduce the number of simulations ran even further. GADSE provides this facility within the GUI. It allows setting or randomizing variables for the initial population. If a certain setting for one variable is known to give a good result, it can be set, leaving all other variables randomized.

Output of the population within each generation is included in a log file and also visually displayed in a graph showing the steady improvement. After the GA finishes each simulation output is analyzed using Analysis of Variance (ANOVA)[5][6]. ANOVA is a technique for comparing the variance among different populations. It is used to determine the interaction within variables and from the variables to the output. Combined with a count of each variable in the number of best/worst case corner violations (eye values that do not fit within the minimum and maximum height/width), the designer can get a good picture of each variables influence on the overall design.

GADSE can also perform traditional techniques for creating and running decks. Within the tab that sets the parameters there are options for only creating the decks or creating and running those decks in a parallel fashion. This feature can be used for small design spaces or if designers want to do their own analysis separately.

Taguchi and CCD analysis can be run in separate tabs. A picture of the GUI for CCD is shown in Figure 6. Template matrices are provided for three to nine factors with two and three levels. An option exists for the designer to load their own templates for higher numbers of factors. These simulations can likewise be ran in parallel and there are options to only create the decks, to only create and run the decks, and to do the whole ANOVA analysis based on the best/worst case corner value.

Calculating the best/worst case corner is important for all previous analysis techniques. A tab in the GADSE GUI gives an intuitive, highly configurable way of setting up the corner detection automation. Parameters for setting the fineness of the calculation are included along with a way to calculate the corner for multiple output parameters from a spice circuit.

4. Results

A DDR-2 memory subsystem shown in Figure 1 is used as a test case. The subsystem contains many transmission lines that can vary in length and impedance. For this case each transmission line has three different variations defined with pre-made models. The template loaded into GADSE turns the model numbers into variables and GADSE simulates all the different settings to find the best and worse case eye. Figure 3 shows a picture of the gui with the different settings defined. With eight factors at three levels each the size of the design space is

$$3 * 3 * 3 * 3 * 3 * 3 * 3 * 3 = 6561$$

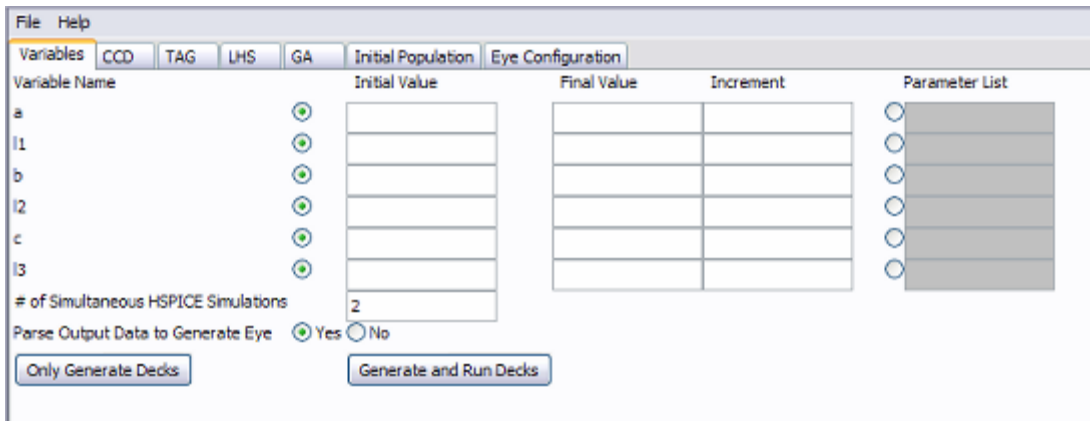


Figure 4 – Variables Tab of GADSE's GUI

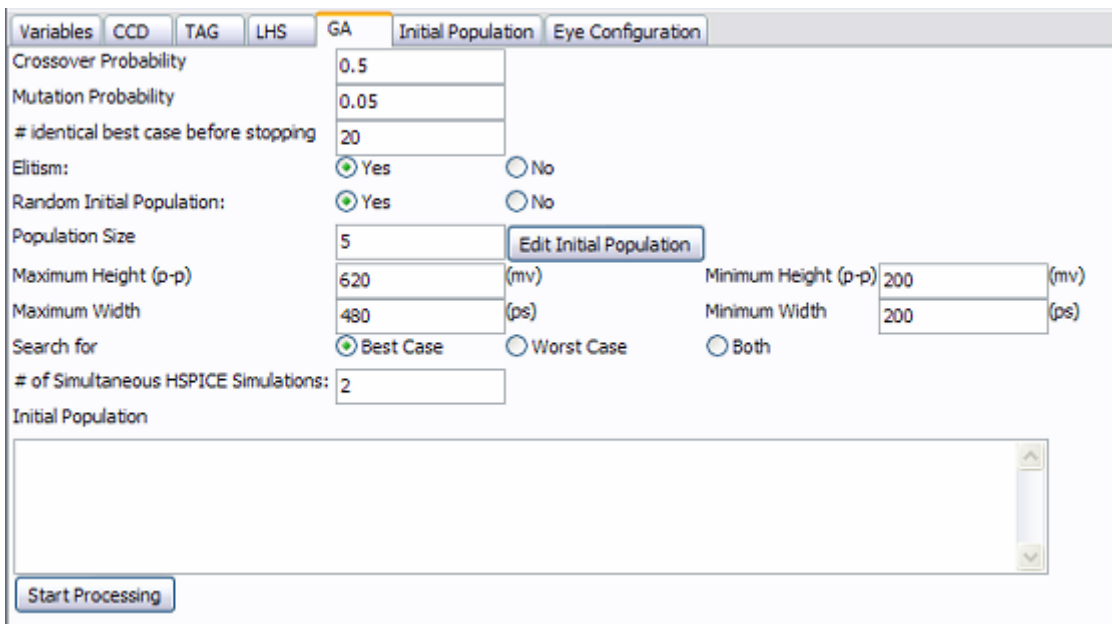


Figure 5 – GA Tab of GADSE's GUI

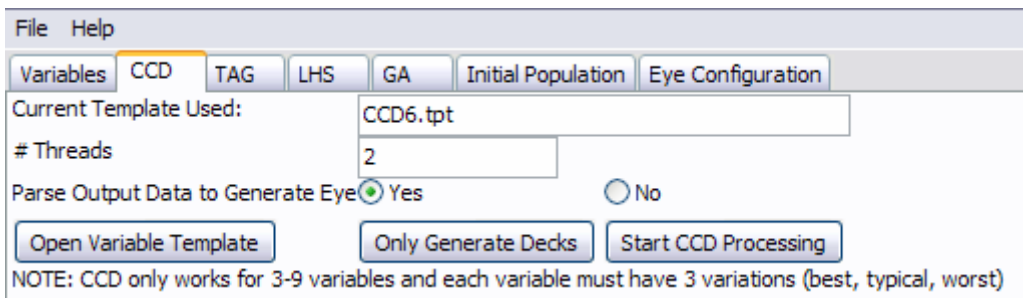


Figure 6 – CCD Tab of GADSE's GUI

Traversing this search space using a Monte Carlo strategy takes hours or even days. If the results are gone through by hand the analysis could take even longer. This test ran two threads in a parallel fashion. The speedup with running more than one simulation at once depends on the availability multiple simulations. As the number of generations rise and the GA closes in on an optimal result, many of the simulations that result from the crossover and mutation have already been completed in previous generations. These values are saved so the simulations do not have to be run again. This sometimes results in generations where fewer than two simulations are run which limits advantages of running in parallel.

Four separate GA simulations were ran in this case with a requirement of 20 consecutive generations with the same best fitness value for the first three and 60 for the final run as a signal for completion. Mutation was also increased for the final 60 iteration run. The bar graphs for run #3 (20 iterations) are shown in Figures 7 and 8. It is easily observed that since elitism is set the overall eye results gradually improved and while the height (black bar and left y-axis) actually went down, the gain in the width (lighter bar and right y-axis) of the corner overwhelmed the decrease. A CCD simulation was also ran using all eight factors and three levels for the high, typical, and low settings. The highest and lowest corner results of all simulations are shown in Table 1. The final GA simulation with 60 iterations beats both the best and worst case corners of the thorough orthogonal search space of CCD even with a random starting population. Priming the population with an orthogonal space can lead to even more precise results. The effect of mutation is seen in the number of extra decks that need to be ran for the final GA simulation.

An analysis of variance was done on the CCD results. The three most significant parameters are shown in Table 2. The resulting corner is shown for each parameter when all other parameters are kept at their nominal settings and only that parameter is varied from best to worst corner. Both C_comp and Dim_tline show a big dropoff between best and worst settings giving credence to their influence on the overall result.

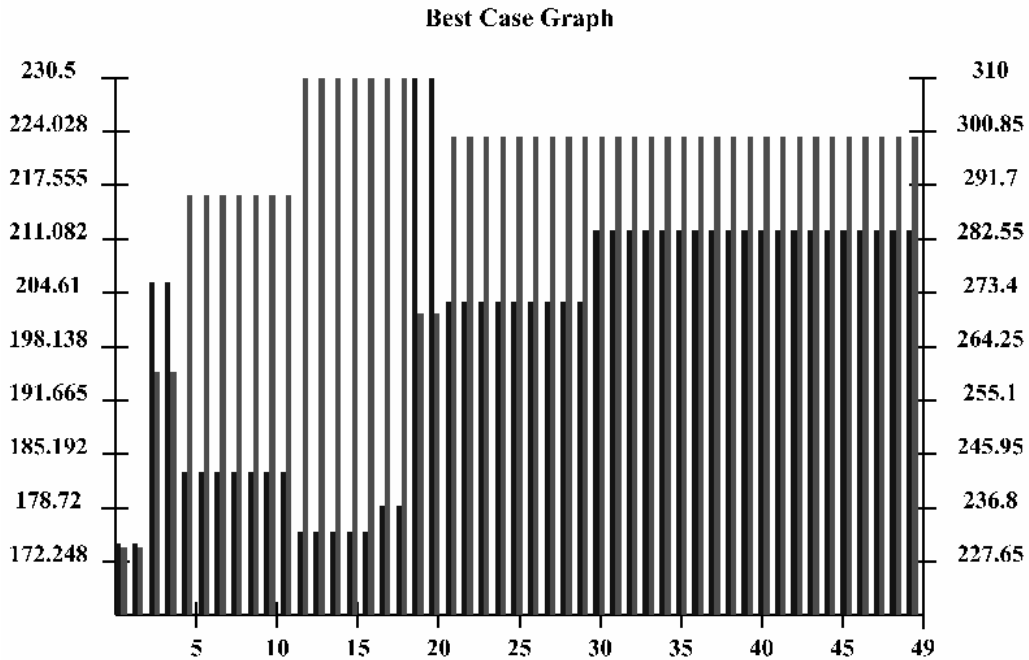


Figure 7 – Best case graph for GA Run # 3. Darker bar matches height on left y-axis. Lighter bar matches width on right y-axis.

Worst Case Graph

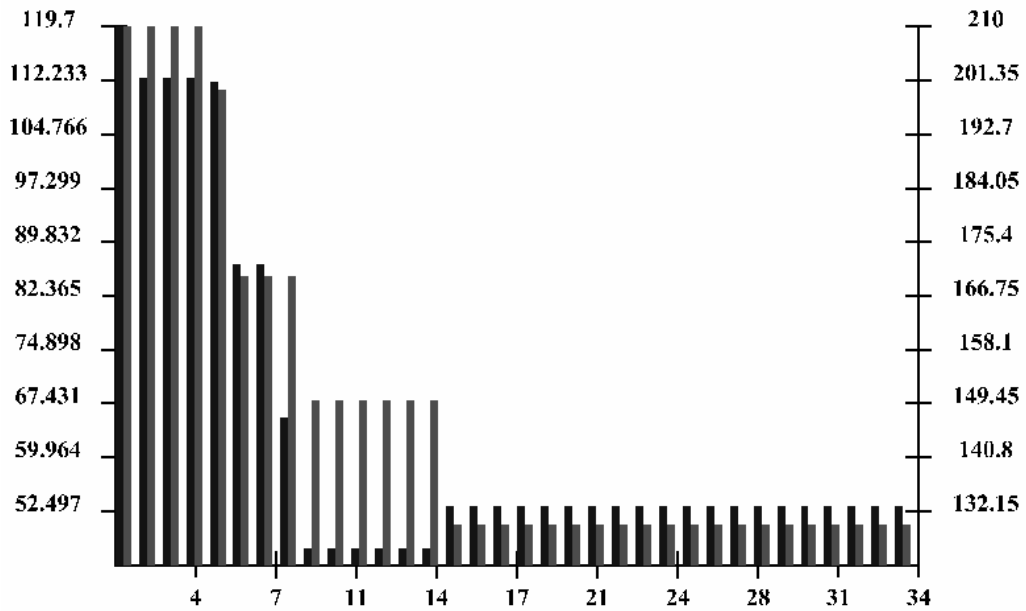


Figure 8 – Worst case graph for GA Run # 3. Darker bar matches height on left y-axis. Lighter bar matches width on right y-axis

Table 1 – Results of simulation runs

Experiment	Largest	Smallest	Simulation Time
GA #1 – Mutation: 0.05 20 generations to exit	Height: 211.7 Width: 300	Height: 87.9 Width: 150	Generations: 76 # decks ran: 215
GA #2 – Mutation: 0.05 20 gen. to exit	Height: 215 Width: 290	Height: 68.6 Width: 140	Generations: 55 # decks ran: 179
GA #3 – Mutation: 0.05 20 gen. to exit	Height: 212 Width: 300	Height: 53.2 Width: 130	Generations: 83 # decks ran: 242
GA #4 – Mutation: 0.25 60 gen. to exit	Height: 190 Width: 320	Height: 50.3 Width: 120	Generations: 158 # decks ran: 705
CCD – 8 Factors 3 Levels	Height: 230.5 Width: 270	Height: 43.8 Width: 130	# decks ran: 81

Table 2 – Sensitivity Analysis results from CCD run

Sensitive parameter settings	C_comp	Dim_tline	P_emp
Low	Height: 189.4 Width: 280	Height: 153 Width: 250	Height: 152.1 Width: 220
Typical	Height: 165 Width: 260	Height: 155.7 Width: 250	Height: 155.7 Width: 250
High	Height: 117.0 Width: 210	Height: 87.5 Width: 220	Height: 158.4 Width: 260

Summary

In this paper GADSE has been shown to be a beneficial tool for designers. It supports statistical techniques such as Taguchi and CCD and automates the tedious aspects of deck generation, deck simulation, and output data analysis. Along with this it adds a novel and highly configurable Genetic Algorithm to strongly reduce the design search space for highly nonlinear systems. GADSE analyzes and outputs the data from all techniques into simple, elegant formats useful to the designer.

References

- [1] Wong, K.Y.; Singh, V.P.; Rustagi, J.S.: "Statistical methods in manufacturing" Electronic Manufacturing Technology Symposium, 1993, Fifteenth IEEE/CHMT International 4-6 Oct. 1993 Page(s):215 – 218
- [2] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs" Springer Verlag; 3rd Revision edition, March 1996.
- [3] Goldberg, David E., "Genetic Algorithms in Search, Optimization & Machine Learning" Addison Wesley Longman, Inc. 1989
- [4] Cui Zhi-Hua; Zeng Jian-Chao; Xu Yu-Bin; "Master-to-slave nonlinear genetic algorithm" Decision and Control, 2003. Proceedings. 42nd IEEE Conference on Volume 4, 9-12 Dec. 2003 Page(s):3830 - 3833 vol.4
- [5] Mandel, John, "The Statistical Analysis of Experimental Data" Dover Publications Inc.; Dover edition, 1984.
- [6] Kachigan, Sam Kash, "Multivariate Statistical Analysis" Radius Press; 2nd edition, June 1991.