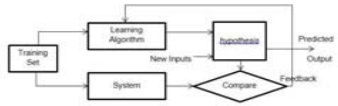
Adaptive test case selection for maximum functional coverage

Professor Morris Jones, Abhijith Byrappa, Bharath KV

Department of Electrical Engineering, San Jose State University, San Jose, California 95192.

Introduction

The increasing complexity of designs mandates the need for innovative and efficient methods of verifying designs in a short span of time. The objective of the project is to improve the time taken for verification by reducing the number of test cases used. Functional coverage is the metric used to determine the efficiency of design verification. A novel way of reducing functional verification time in a System Verilog verification environment is by using gradient descent algorithm(GDA). One of the ways of predicting the output of a system is to train an algorithm to understand the behavior of a system. For a given set of input values the algorithm predicts the output of the system, called the hypothesis. The algorithm compares the hypothesis to the actual output of the system that is based on a limited set of training examples [1]. From the differences in hypothesis and the actual system output the algorithm gradually learns to predict the correct output of the system, as shown in Figure below.



This technique can be applied to any complex design to reduce simulation/verification time. The training set is manually given for each design to train the algorithm. GDA can then be executed to determine the redundant/weak test cases that can be eliminated from subsequent re-runs [1]. A total of 27 test case runs out of 180 were saved in this project by running GDA on the design (SDRAM Controller core) that has been used for this project.

Gradient Descent Algorithm

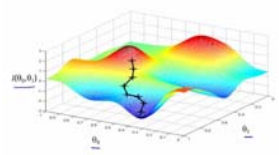
Gradient descent algorithm is a form of supervised learning algorithms which has been used in machine learning and artificial neural networks. Gradient descent algorithm can be used to solve linear regression problems

$$y = \theta 0 + \theta 1\ x1 + \theta 2\ x2 + \theta 3\ x3 \ldots\ldots\ldots \theta N\ xN$$

The x's are the input features, y is the output feature, N is the number of input variables, and θ's are the parameters for the corresponding input features. The algorithm learns to predict the output y for new input values of x1-xN. This is achieved by finding the minimum value of the function J(θ) defined as:

$$J(\theta) = \frac{1}{2}\sum_{i=0}^{n}(\theta_i\ xi\ -\ yi)^2$$

For one input variable, the minimum value for function J(θ) is reached by taking steps in the path of steepest descent as show in the next figure.



Implementation

In the project implementation N represents the number of test cases. The input variables 'x' are the number of times the corresponding test case is executed. 'y' is the total functional coverage. θ is the weightage which represents the ability of the corresponding test case to generate functional coverage. The System Verilog test-bench for the SDRAM memory controller consists of three test cases. Each test case does read/write to random memory locations in Bank 0, 1 & 2. Reads and writes are coded as tasks. The training data is generated by executing training sets which are coded as tasks. The training sets executes the test cases a random number of times using the repeat statement.

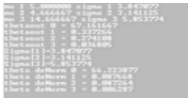
x1,x2 and x3 are the number of times test case 1, 2 and 3 are run respectively. A training example consists of random values of x1, x2 and x3 and its corresponding functional coverage 'y'. Training set data is developed by running 'm' training examples. When gradient descent algorithm is run on this training data the θ values are obtained, that represents how weak or strong the corresponding test case is in functional coverage. Test case with high θ value is run more number of times than the test case with low θ value. Running a test case with low θ value more number of times results in very less increase in functional coverage at the expense of a lot of clock cycles. GDA gives an indication as to which test case needs to be run more to achieve the target functional coverage. 'x' value corresponding to high θ value is increased by 10% while those corresponding to low θ values is dropped 2% while still achieving the target functional coverage. This process is iterated till the minimum values of 'x's' are found to achieve the same functional coverage.

In the top level test bench file, the top level design file of the memory controller to be tested is instantiated and all the parameters are initialized. The test bench contains a task for read and a task for write. The three test cases are coded as three tasks in the test bench. The test cases make a call to the read\write tasks to perform read\write operations. The main initial block of the test bench executes each of the training sets one after the other; the coverage data is collected after each training set is executed. Each training set is coded as a task and has calls to the test case tasks. 'Repeat' statement is used to run the test cases required number of times. The 'x' data corresponding to the number of times the test case is run is stored in 'x.dat' file and the coverage result corresponding to output 'y' for the entire training set is stored in 'y.dat' file. Gradient descent algorithm implemented in system Verilog code reads the data from these two files and outputs the theta values. The system Verilog code of gradient descent algorithm consists of three functions and a main function. The three functions are for normalization, gradient descent, and de normalization. The main function of the code creates a file handle to the 'x.dat' and 'y.dat' files and copies the data to an internal array for 'x' and 'y'. Gradient descent algorithm only works if the data does not have bias; hence normalization has to be performed on the input data. The main function calls the normalization function with the data arrays are arguments and the function returns the normalized values for the data.

Results

The total number of test case runs saved is 27 out of 180. On an average the total clock cycles spent on a test case is around 60,000. Thus the total clock cycles saved is 1.6 million clock cycles out of 10.8 million which is 14.8% savings for the reference design and test bench.

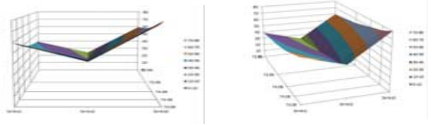
x1	x2	x3	y
1	1	6	66.44
2	2	11	66.7
3	3	14	66.78
5	6	16	67.31
8	7	18	67.57
11	9	23	68.17



When the test-bench is initialized, the initial functional coverage is zero, the test-bench first exercises training set 1, records the functional coverage, resets the functional coverage and executes training set 2. This process is repeated till all the training set is exercised and functional coverage recorded for each one of them. The 'x' and 'y' data are recorded in a text file. The gradient descent algorithm coded in System Verilog reads the 'x' and 'y' data file and outputs the normalized theta values. The theta values for the training set are as shown in Figure above. 'mu' represents the average value of the input 'x' data and sigma represents the mean squared difference. θ3 value is high, the algorithm indicates that test case 3 should be run more number of times and running test case 1 and test case 2 more times than test case 3 would not result in any better functional coverage. As a starting point x1, x2 and x3 are chosen to be 60 each. That is each test case is run 60 times which would result in a functional coverage of 74.08%. Test case 1 and 2 are dropped by 10% and test case 3 is increased by 2% and the functional coverage is monitored to remain at 74.08%. This process is repeated and for the reference design considered for x1 value of 49 , x2 value of 34 and x3 value of 70, the total number of times each test cases are run has reached its minimum value of 153 as against the initial 180 to the achieve the same target functional coverage of 74.08%.

x1	x2	x3	y
39	24	59	73.96
44	29	64	73.96
49	34	70	74.08
55	38	74	74.08
60	44	80	74.08

Figures above illustrates the 3D graph of the data around the minimum values for x1, x2 and x3, the data is as shown in Figure 14.



Summary

The project successfully demonstrated that gradient descent algorithm can be used to reduce the verification time for a design, where the method of verification is functional coverage involving more than one test case. 1.6 million clock cycles are saved with the implementation of gradient descent algorithm to optimize the number of test cases run to achieve a target functional coverage.

Key References

[1] Machine learning by Andrew NG (Stanford University)
[2] System Verilog for Verification – A guide to learning the test-bench language features by Chris Spear, 3e, 2012
[3] System Verilog Assertions and Functional Coverage by Ashok B. Mehta, 2014
[4] 8/16/32 bit SDRAM Controller – OpenCores

Acknowledgements

We wish to thank our Professor Morris Jones for his guidance throughout the semester , which enabled us to successfully implement and complete the project