Enhancing the Efficiency of Bayesian Network Based Coverage Directed Test Generation

Markus Braun
STZ Softwaretechnik
Esslingen, Germany
email: braun@stz-softwaretechnik.de

Shai Fine Avi Ziv
IBM Research Laboratory in Haifa
Haifa, Israel
email: {fshai,aziv}@il.ibm.com

Abstract

Coverage Directed Test Generation (*CDG*) is a technique for providing feedback from the coverage domain back to a generator, which produces new stimuli to the tested design. Recent work showed that CDG, implemented using Bayesian networks, can improve the efficiency and reduce the human interaction in the verification process over directed random stimuli. This paper discusses two methods that improve the efficiency of the CDG process. In the first method, additional data collected during simulation is used to "fine tune" the parameters of the Bayesian network model, leading to better directives for the test generator. Clustering techniques enhance the efficiency of the CDG process by focusing on sets of non-covered events, instead of one event at a time. The second method improves upon previous results by providing a technique to find the number of clusters to be used by the clustering algorithm. Applying these methods to a real-world design shows improvement in performance over previously published data.

Introduction

To date, functional verification using directed random simulation is one of the most commonly used verification techniques. Figure 1 shows the general setup of a directed random simulation environment. The verification environment is responsible for generating random stimuli to the Design Under Verification (*DUV*). Directives provided by the user are used to direct the stimuli to hit different areas in the design and create interesting scenarios. Coverage is used to measure the quality of the verification process. During a simulation run, monitors observe the behavior of the design and feed coverage information into coverage tools [5]. These tools compute the current coverage state and identify holes in the coverage model. Such holes usually indicate that there are untested or lightly tested areas in the design.

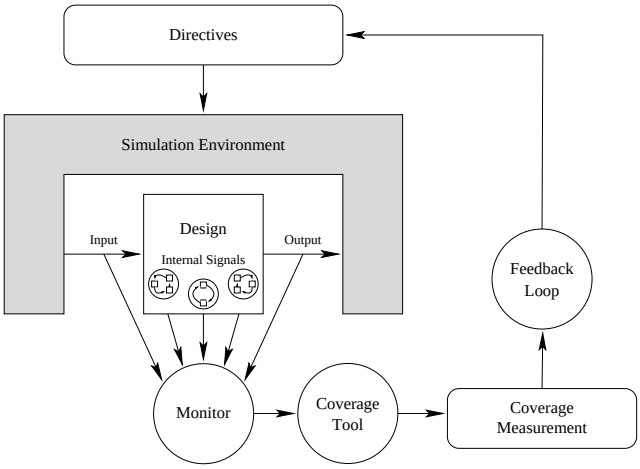

Figure 1. Structure of the Simulation Environment with a Feedback Loop

While the directed random simulation approach is highly automated, analyzing the coverage data and changing the directives to the verification environment to cover the holes in the coverage is still mostly manual. A verification engineer needs to observe the progress of the verification process and has to adjust the directives to guide the simulation to the previous non-covered cases. To be able to do this, the engineer needs to be an expert in the area of verification, as well as in the DUV. In addition, even an experienced veri-

fication engineer is only able to provide a very limited number of directive sets that target a specific area in the design. Therefore, the manual feedback can be a major bottleneck in the verification process.

To address this bottleneck, Coverage Directed Generation (*CDG*), an automated feedback loop between the coverage measurement and test directives, has been proposed [7]. In general, this feedback loop gets information about the simulation environment, the DUV, and the coverage model. This information is used to design simulator directives that cover previously non-covered events in the coverage model. The feedback loop generates a set of directives for each single non-covered event. These directives are used to run the simulator, and all tasks that occurred during this run are marked as covered. This "prediction – simulation" cycle is repeated until either all events in the coverage model are covered or a time limit is reached.

Several approaches that implement such feedback loops have been proposed. One approach uses Bayesian networks [8] to build a model of the coverage process [3]. It models the whole process, from the simulator directives to the coverage events. Other approaches include data mining techniques [2] and genetic algorithms [1].

This paper presents two techniques that enhance the efficiency of a Bayesian network based CDG. The first technique uses data gathered during simulations to adapt a Bayesian network. Smart sampling of the coverage data during simulation focuses on the weak regions of the network (i.e., where the network cannot provide directives or provides directives with low certainty). This additional data is used to "fine tune" the Bayesian network. As a result, the adapted network is able to provide directives to areas that it was previously unable to reach.

The second technique is a method to calculate the number of clusters for clustering algorithms [4]. Clustering techniques enhance the efficiency of the CDG process by creating one set of directives adequate for covering a set of non-covered tasks, thus reducing the number of directives (and simulation runs) needed to cover the coverage model. In this paper, we propose two criteria to measure the quality of a specific clustering, and use these criteria to calculate the number of clusters.

For both techniques, we provide experimental results on a real-life design. The experimental results indicate that both adaptation and calculation of the number of clusters, can be used to enhance the quality of Bayesian network based CDG.

The Hardware Design

For the experiments described in this paper, we used the Storage Controller Element (*SCE*) of an IBM zSeries processor. This is the same design that was used in previous work on Bayesian network based CDG [3, 4]. Figure 2 shows the block diagram of the design and parts of its simulation environment.

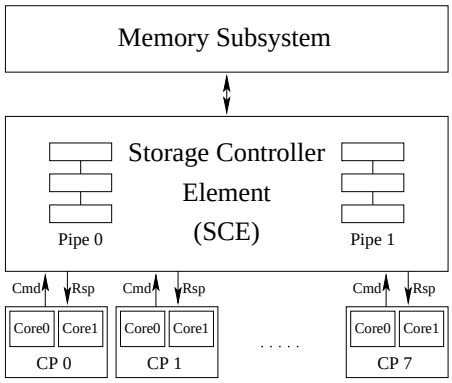

Figure 2. Storage Controller Element Design

The design implements a coherent switch with integrated L2 cache, which connects 8 CPs with the memory subsystem. Each CP consists of two cores, which generate commands to the SCE independently. The coverage space is the cross-product of the core (2 values) that generated the command, the CP (8 values) on which the core is located, the node (4 values) where the CP is located, the pipeline (2 values) that handled it, the command (31 values) and the response to a command (14 values). The coverage model consists of all 18048 valid transactions between the CPs and the SCE.

Adaptation of a Bayesian Network

Building the Bayesian network for the feedback loop consists of two steps [3]. The first step is defining the structure of the Bayesian network. This is done by using domain knowledge of the simulation environment and the DUV. The structure encodes the supposed relations between the directives to the simulator

and the coverage model. After the structure is defined, the parameters of the probability distribution functions within the network are trained based on a set of training data, which consists of directives to the simulator and the respective coverage tasks that were hit during simulation. After the network has been trained it can be used to predict directives for the simulator. Specifically, the Bayesian network is instantiated with an evidence that represents a coverage event, and the simulator directives that are supposed to hit this particular coverage event are calculated using posterior probabilities.

The resulting Bayesian network may have a low confidence in the directives it provides for some of the coverage events; it may not be able to provide directives at all for some other events. A commonly used measure for the confidence of the Bayesian network is the log-likelihood of predicted covering probability. That is, the predicted probability by the Bayesian network of covering the event using the provided directives. The lower the log-likelihood, the lower the confidence in a prediction. The top level chart in Figure 3 shows the log-likelihood values for a subspace of the coverage model of the initial Bayesian network. It is clearly visible that some of the log-likelihood values are very low.

While the Bayesian network cannot provide quality directives (or any directives at all) for events with low log-likelihood, some of these events are being hit during the verification process. The basic idea in adaptation is to extract data where the simulator hit a low log-likelihood coverage event and use this data to improve the training of the Bayesian network. The goal of the adaptation process is to improve the quality of the directives and the confidence of the network for events with low confidence, without deteriorating the results for other coverage events. We applied two different adaptation methods to an existing Bayesian network of the SCE and compared the performance of the adapted networks. Both adaptation methods modify merely, the Bayesian network's parameters; namely the network's structure remains unchanged.

The first applied adaptation method had been the so-called batch adaptation. The batch adaptation method uses all extracted data sets at once to train a new network from scratch. This is done using the same training algorithms used to train the initial network. After the training of the new network is completed, the conditional probabilities of the new network are merged with the conditional probabilities of the initial network, weighted according to the sizes of the corresponding training sets.

In contrary to the batch adaptation method, where a new network is trained using all data sets at once, at each training iteration the on-line adaptation method use, a single data set, resulted from a single simulation run, to train a network from scratch. Then, the conditional probabilities of this new network are merged with the conditional probabilities of the initial network. This process is repeated until all extracted data sets have been used to train a new network and have been accumulated to the initial Bayesian network.

Figure 3 compares the log-likelihood of the two adapted networks and the log-likelihood of the original network. The figure shows that in both adaptation methods, the resulting adapted networks contain no coverage events with a low log-likelihood. The differences between the log-likelihood values of the two adapted networks are very small.

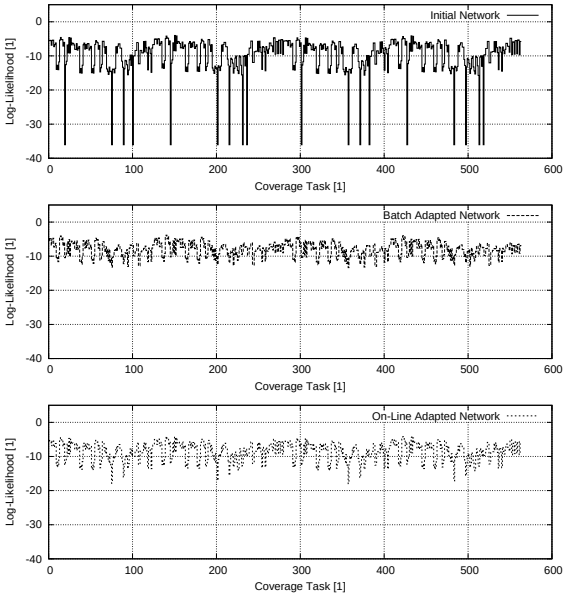

Figure 3. Comparison of Log-Likelihood Values

To validate the quality of the adapted networks, we used both networks to cover the entire SCE coverage model (18048 events). Figure 4 shows the resulting coverage progress of the adapted networks and the initial network. The performance of the two adapted networks is fairly close, and both adaptations methods

gain a decisive improvement over the initial network. For example, the initial Bayesian network reached a level of 98.96% after 5105 simulation runs, while the batch and on-line adapted networks needed only 2420 and 2091 runs, achieving a speed-up of 2.1 and 2.4, respectively. By comparison, the simulation using a directed random environment without an automated feedback loop, needed 53930 simulation runs to reach the coverage level of only 95.24%.

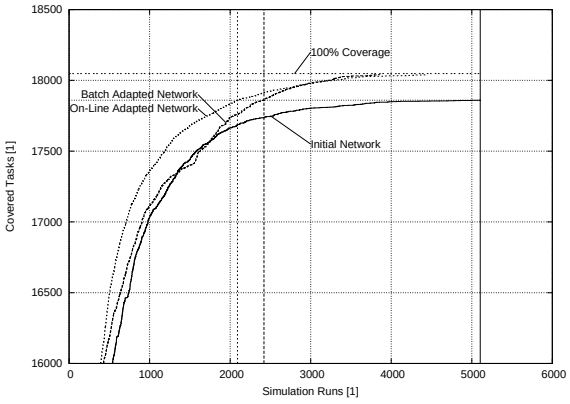

Figure 4. Performance Comparison of the Initial and the Adapted Bayesian networks

While the performance of the two adapted networks is similar, the difference in the computation complexity of batch and on-line methods is substantial. Because the on-line method breaks the training into small steps, the overall training time is much shorter. For the SCE network, we observed a speedup of more than twice, when moving from batch to on-line adaptation. In addition, the on-line adaptation can be used in an earlier stage of the verification process, before the entire data for the adaptation is ready. Therefore, the on-line method is preferable to the batch adaptation.

Overall this technique is beneficial, because the additional effort needed to adapt the network is compensated favorably by the gain achieved in saving simulation time needed to provide full coverage.

Clustering Coverage Events

Clustering enhances the efficiency of the CDG process by focusing on sets of coverage events instead of one event at a time. In [4], a clustering algorithm that is based on similarity of the directives was proposed. In that algorithm, there are several "free" parameters that can be used, such as the distance measure between directives and the method used to cluster directives. In the work described here, we used the k-Medoids algorithm [6] to cluster directives, and Euclidean distance measure:

$$d(x,y) = \sqrt{\sum_i (x_i - y_i)^2} \qquad (1)$$

The results presented here can be easily extended to other clustering methods and other distance measures.

One issue regarding clustering that is not addressed in [4] is the number of clusters to use. This number is left as a free parameter. In this section, we define two measures for the quality of clustering and use these measures to find the number of clusters for the clustering algorithm described in [4]. The main requirement for the measures was the ability to calculate the criteria off-line; that is, use only information provided by the Bayesian network, without running simulations.

Local versus Global (LvG) Criterion

The *Local versus Global (LvG)* criterion uses information about the compactness of the individual clusters and the diversity of the clusters themselves. The calculation uses a given number of clusters N, a number of points I to be clustered and M_n points within a cluster n. Furthermore, the predicted log-likelihood values for each point, ll_i, and for each cluster, llc_n, are required. The log-likelihood values for clusters are computed using soft evidence. This means that all coverage events in a cluster serve as evidence to the Bayesian network simultaneously, and the log-likelihood for all of them is computed. The local compactness d_{local} is defined as the weighted sum of the difference between the predicted log-likelihood of the entire cluster llc_n and the average of the log-likelihood of the points within the cluster.

$$d_{local} = \sum_{n=1}^{N} M_n llc_n - \sum_{i=1}^{M_n} ll_i \qquad (2)$$

The global compactness d_{global} is defined as the difference between the average log-likelihood of the

clusters and the average log-likelihood of all points.

$$d_{global} = I(\frac{1}{N}\sum_{n=0}^{N} llc_n - \frac{1}{I}\sum_{i=0}^{I} ll_i) \quad (3)$$

The criterion is defined as the difference between d_{local} and d_{global}

$$Criterion_{LvG} = d_{local} - d_{global} \quad (4)$$

For this criterion the optimal number of clusters is the one that minimizes the criterion (starting with 2 clusters).

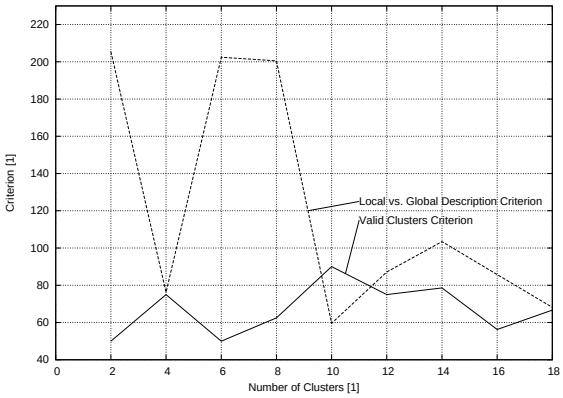

Figure 5. Criteria to Find the Best Number of Clusters

Valid Clusters Criterion

The *Valid Clusters* criterion divides the clusters into two sets, valid and invalid clusters, and uses their ratio to measure the quality of the clustering. We define a cluster as valid if it significantly improves the probability of hitting the tasks in it. Specifically, a cluster is defined as valid, if the predicted log-likelihood for the cluster is 20% higher than the average log-likelihood of the cluster members. The Valid Clusters criterion is computed as follows:

$$Criterion_{VC} = \frac{n_{valid}}{N} \quad (5)$$

The optimal number of clusters according to this criterion is the number of clusters that maximizes it (starting with 2 clusters).

To evaluate both criteria, we used a subspace of the SCE coverage model that covers unrecoverable errors (*UE*) and includes 98 coverage events. Figure 5 shows the calculated values for both criteria, as a function of the number of clusters for this subspace. The figure shows that both criteria predict that 10 clusters will provide the best performance for this coverage space.

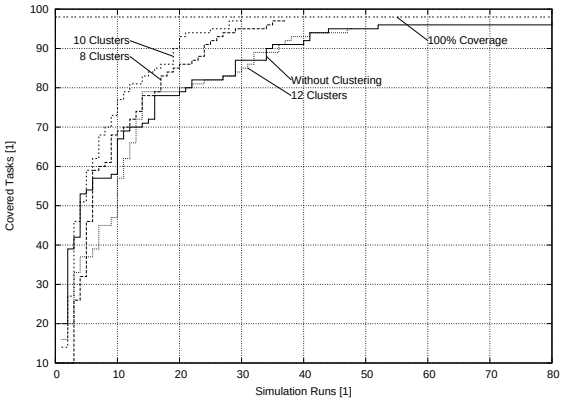

Figure 6. Coverage Progress on UE Subspace

In Figure 6, the resulting coverage progress is shown. The results show that indeed 10 clusters provide the best performance, leading to speed-up by a factor of 3 over the base case (without clustering) and significant improvement over clustering with other numbers of clusters.

Summary and Future Work

In this paper, we presented two techniques to enhance the efficiency of CDG based on Bayesian networks. The adaptation of a Bayesian network improves the quality of the directives produced by the network and the confidence in its predictions. This, in turn, leads to faster coverage. We observed a speedup of more than a factor of 2 in a real-life application.

For clustering, we proposed two criteria to measure the quality of the given clustering configuration; this enabled us to find the number of clusters. Both criteria can be rapidly calculated from predictions provided by the Bayesian network, without the need to run simulations.

For future work, we plan to apply the proposed methods to other verification environments. In addition we are investigating other techniques to enhance the performance and capabilities of Bayesian networks based CDG.

References

[1] M. Bose, J. Shin, E. M. Rudnick, T. Dukes, and M. Abadir. A Genetic Approach to Automatic Bias Generation for Biased Random Instruction Generation. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 442–448, May 2001.

[2] M. Braun, W. Rosenstiel, and K.-D. Schubert. Comparison of Bayesian Networks and Data Mining for Coverage Directed Verification. In *IEEE International High-Level Validation and Test Workshop (HLDVT 2003)*, pages 91–95. Omnipress, 2003.

[3] S. Fine and A. Ziv. Coverage Directed Test Generation for Functional Verification using Bayesian Networks. In *Proceedings of the 40th Design Automation Conference (DAC 2003)*, pages 286–291, New York, 2003. Association for Computing Machinery.

[4] S. Fine and A. Ziv. Enhancing the Control and Efficiency of the Covering Process. In *IEEE International High-Level Validation and Test Workshop (HLDVT 2003)*, pages 96–101. Omnipress, 2003.

[5] R. Grinwald, E. Harel, M. Orgad, S. Ur, and A. Ziv. User Defined Coverage: A Tool Supported Methodology for Design Verification. In *Proceedings of the 35th Design Automation Conference (DAC 1998)*, pages 158–163, New York, 1998. Association for Computing Machinery.

[6] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, Berlin, July 2001.

[7] G. Nativ, S. Mittermaier, S. Ur, and A. Ziv. Cost Evaluation of Coverage Directed Test Generation for the IBM Mainframe. In *International Test Conference (ITC 2001)*, pages 793–802, Washington - Brussels - Tokyo, 2001. IEEE Computer Society Press.

[8] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Network of Plausible Inference*. Morgan Kaufmann, 1988.