Predicting the Future: Resource Requirements and Predictive Optimism

Bradley L. Noble
Roger D. Chamberlain

B.L. Noble and R.D. Chamberlain, “Predicting the Future: Resource Requirements and Predictive Optimism,” in *Proc. of 9th Workshop on Parallel and Distributed Simulation*, June 1995, pp. 157-164.

Computer and Communications Research Center
Washington University
Campus Box 1115
One Brookings Dr.
St. Louis, MO 63130-4899

Predicting the Future: Resource Requirements and Predictive Optimism

Bradley L. Noble
brad@ccrc.wustl.edu

Roger D. Chamberlain
roger@ccrc.wustl.edu

Computer and Communications Research Center
Washington University, St. Louis, Missouri

Abstract. *The partitioning of systems for parallel simulation is a complex task, requiring consideration of both computational load requirements and communications activity. Typically, this information is not accurately known prior to execution. This paper investigates the use of historical information for the prediction of future requirements, both for computation and communications. In addition, for optimistic simulation algorithms, we present a novel technique (which we call* predictive optimism*) whereby binary prediction schemes can be used to increase the accuracy of optimistic assumptions, thereby decreasing rollbacks and potentially improving overall simulator performance.*

1 Introduction

Traditionally, the distributed simulation of systems has been carried out in a data parallel manner, in which the components of the simulated system are partitioned and assigned to processors for execution. It is clear that an inappropriate partitioning will seriously degrade the performance of any distributed simulation algorithm, yet good partitionings are, in general, difficult to determine. This is true primarily because the computational work associated with individual system components and the communications traffic required between components are typically unknown prior to the simulation execution.

This lack of *a priori* knowledge of computational workload and communications requirements has lead to two techniques for system partitioning. The first approach, pre-simulation, is to execute the simulation for some small time, measure the computational workload and/or communications requirements, and use this measured data to repartition the system for the remainder of the execution. The second approach is to use dynamic repartitioning, whereby system components are periodically migrated from one processor to another in response to measured resource requirements (either computation or communication).

A central assumption in both of the above approaches is that the measured historical data being used to guide a subsequent partitioning is indicative of future requirements. The main distinction between the two approaches is the time scale over which this assumption is made. For static partitioning guided by pre-simulation, we assume the resource requirements are fairly consistent over the entire course of the simulation execution. With dynamic migration, we rely on recent historical data to be a reasonable predictor for near-term future requirements.

This paper studies the predictive quality of historical data on future requirements, for both computational workload and communications volume. Empirical data is collected from VLSI logic simulations (using the ISCAS-89 sequential benchmark set [2] and a pair of locally available circuits) that shows high correlation between past resource requirements and future needs.

In addition, we investigate the standard assumptions made in optimistic synchronization protocols and show how binary prediction schemes can be used to increase the accuracy of optimistic assumptions, a technique we call *predictive optimism*. This leads us to question the use of communications volume as an appropriate metric for system partitioning. While the number of messages is a measure of the amount of communications volume across a processor set, it is not necessarily directly indicative of the amount of *information* communicated. Using predictive optimism, we contend that a potentially more appropriate metric is the entropy of the channel.

In the remainder of this section, we present the model of distributed discrete-event simulation (DES) used and motivate the study. In the following section, we describe the resource requirements prediction. Section 3 defines the predictive optimism technique. Section 4 describes the experiments performed. Section 5 presents the empirical results and the final section concludes and describes future work.

1.1 Model of Distributed DES

In parallel discrete-event simulation, system components are typically modeled via *logical processes* (LPs) that communicate by sending time stamped messages across explicit communications channels. The set of LPs is partitioned, and the partition blocks are assigned to processors for execution. At each point in simulated time, an LP may or may not send a message to a neighboring LP. The channel

[1] This work was supported in part by the National Science Foundation under grant number MIP-9309658.

can be viewed in much the same way as a binary communication channel where the presence of a message is represented by a "1" and the lack of a message by a "0." We will exploit this view when describing predictive optimism.

Upon receipt of a message, subject to an appropriate time synchronization protocol, an LP performs a functional evaluation which simulates the effect of the incoming message on the internal state of the LP and determines what (if any) outgoing messages are generated. The computational workload associated with each LP is assumed to be directly related to the number of functional evaluations performed.

1.2 Motivation

Resource Requirements. Essential to the efficient operation of any parallel application is a computational workload that is balanced across the set of processors. Additionally, if the application requires frequent interprocessor communication, performance can be diminished by communications traffic contention and latency. As a result, good partitioning algorithms must simultaneously balance the computational workload as well as minimize and balance the communications requirements. This is often accomplished by posing the partitioning problem as a combinatorial optimization problem, where the goal is to minimize a cost function of the form

$$C = f(COMP, COMM)$$

where $COMP$ represents the computational load balance, $COMM$ represents the communications requirements, and f is a combining function of some form (e.g., maximum, sum, etc.) [1].

One of the primary limitations of this approach for discrete-event simulation is the general lack of knowledge about the resource requirements (both computation and communications) prior to execution. This has motivated the use of data collected during simulation execution to guide subsequent system partitionings, either in the form of a pre-simulation run [6] or the dynamic redistribution of LPs during simulation execution [8].

A previous study [3] provided positive evidence for the use of past computational workload data as a predictor for future workload in VLSI systems simulation. This empirical study was limited, however, by the the use of a small set of benchmark circuits, the use of random input vectors to drive the circuits, and a focus on computational workload (ignoring communications requirements). We extend this previous work to address each of the above limitations.

Predictive Optimism. By their very nature, optimistic methods make causality errors. When a potential causality error has been detected (due to the arrival of a straggler message or an anti-message), the local state is rolled back to the time stamp of the arrival and the simulation execution proceeds from that point. Large numbers of rollbacks can have serious negative implications on the performance of the simulator, to the point of making the simulation behave erratically [5].

In all optimistic simulation methods to date, the standard optimistic assumption made is that if a message has not yet arrived, no message will arrive, and the simulation proceeds as if the communication channel were inactive. We propose to use the past history of messages communicated along a channel to enable a more sophisticated optimistic assumption, which we call predictive optimism.

Information theoretic techniques relating to communication channels have been available for several decades. These techniques can be applied to the communication channels between LPs to make more informed guesses as to the arrival and potential contents of future messages. Predicting the arrival of messages can be useful in throttling mechanisms (e.g., establishing appropriate window sizes), while predicting the content of messages can increase the probability that the optimistic assumption is correct and decrease the probability of eventual rollback.

In addition, measures of information content, such as entropy, could be used in partitioning algorithms, if the overhead of a channel crossing processor boundaries is less a function of the volume of messages and is more a function of the predictability of message content.

2 Predicting Resource Requirements

When using historical data to predict future resource requirements (either through pre-simulation or as part of a dynamic component migration algorithm), we are assuming that the historical data is reasonably indicative of future requirements. We empirically test this hypothesis (for both computational workload and communications traffic) on VLSI logic simulations.

To study computational workload, we measure the number of functional evaluations associated with each circuit component. Assuming that the time required for each evaluation is constant (a reasonable one for logic circuits), the evaluation count is a strong indicator of workload. To study the communications traffic, we measure the number of messages output from each latch. We limit the communications data collected to latch outputs for purely pragmatic reasons as described in Section 4.

For both computational workload and communications traffic, we are interested in whether requirements stay relatively constant during the course of the simulation execution. We assess this stability quantitatively by dividing the simulation execution into time intervals and measuring the coefficient of variation of the functional evaluation counts and message counts throughout the execution. Stable resource requirements will result in a low coefficient of variation, while instability in the requirements will manifest itself in a high coefficient of variation.

3 Predictive Optimism

All existing optimistic algorithms make the assumption that if a message has not arrived, none will arrive. Essentially, the assumption is that the current input to the LP will be the same as the previous input. This assumption amounts to making a prediction about the input state. For notational purposes, we will refer to this assumption as a Previous State Predictor (PSP).

The PSP can be described by $\hat{x}_{t+1} = x_t$, where x_t is the state of x at time t and $\hat{x}_{t+1}$ is the estimate of state x at time $t+1$. Since all future state is determined solely by the last known state, PSP does not effectively use historical information in prediction decisions. Intuitively, PSP will work well for fairly stable channels but poorly for active ones. This observation leads us to hypothesize that *discrete-event simulation contains historical information that can be used to improve the prediction of messages*.

If the hypothesis is true, then predictors can be designed to take advantage of this historical information. The prediction could be of two types: predicting the existence (or absence) of a message or predicting the contents of a message. The first type is beneficial when designing adaptive throttling techniques for optimistic algorithms (e.g., do not advance local time beyond when a message is predicted to arrive). The second type can be used to guide the optimistic assumption itself. Instead of predicting that the input state will stay constant, predict the next value of the input state and proceed to evaluate the resulting (conditional) event.

Predictors exist that predict at time t the next bit x_{t+1} based on sequential observations of an arbitrary deterministic binary sequence $x_1, x_2, \cdots$. To test the predictability of messages, we will focus on two predictors. The first is a simple finite-state predictor, and the second is a more sophisticated incremental parsing predictor [4].

3.1 Single-State Predictor

Of the class of finite-state predictors, the simplest, nontrivial one that retains historical information is the single-state predictor (SSP). The SSP can be described by

$$\hat{x}_{t+1} = \begin{cases} \text{“0”} & \text{if } N_0 > N_1 \\ \text{“1”} & \text{otherwise} \end{cases}$$

where N_0 and N_1 are counts of zeros and ones, respectively, observed from the entire sequence x_1 through x_t. At each time, t, the counts are updated and a guess of $\hat{x}_{t+1}$ is made based on the largest count. It is easily shown that this predictor has constant prediction time and constant memory usage.

3.2 Incremental Parsing Predictor

A much more sophisticated predictor based on the Lempel-Ziv incremental parsing algorithm used in compression has been developed by Feder et al. [4]. The incremental parsing predictor (IPP) builds a dictionary of distinct phrases where each phrase is the shortest observed phrase not previously parsed. The dictionary is structured as a binary tree and initially starts with a root and two leaves where each leaf corresponds to the phrases $0, 1$ respectively. Starting at the root, the branch corresponding to the current observed bit is traversed with each new observation. When a leaf is reached, the tree is extended at that point, making the leaf an internal node with two new leaves extended from it.

Each branch keeps a count of the number of traversals. The prediction $\hat{x}_{t+1}$ is made according to

$$\hat{x}_{t+1} = \begin{cases} \text{“0”} & \text{with probability } \phi_t(\hat{p}_t^{LZ}(0)) \\ \text{“1”} & \text{with probability } \phi_t(\hat{p}_t^{LZ}(1)) \end{cases}$$

where $\phi_t(\cdot)$ is a time-varying threshold function and $\hat{p}_t^{LZ}(\cdot)$ is calculated from the counts stored in the dictionary. The prediction time is constant for IPP, however the memory requirements are proportional to the number of nodes in the dictionary (i.e., logarithmic in the length of the input sequence).

4 Experimental Methods

The experimental setup uses VLSI logic circuits as the simulated system. We execute a unit-delay, gate-level simulation of all of the ISCAS-89 sequential benchmark set in addition to a pair of locally available circuits. The ISCAS benchmarks are exercised using random input vectors, and the remaining two circuits are exercised using “typical usage” input vectors. These typical usage vectors were chosen (by the designers of the circuit) to be indicative of the use of the circuit in normal operation. For all but one of the circuits, an input vector consists of one clock cycle, and the circuits were simulated for 5000 vectors. For the remaining circuit, a vector consisted of three clock cycles, and the circuit was simulated for 2500 vectors (or 7500 clock cycles).

For the purposes of resource requirements prediction, the simulation run was divided into intervals of 100 vectors each, and the resource usage counts (functional evaluations and messages) were accumulated within each interval. Raw data is presented for a pair of the circuits to illustrate the trends that were observed. To quantitatively assess the entire data set, the standard deviation and mean of the counts are calculated for each component (or channel) and plotted as a histogram of the coefficients of variation (the ratio of the sample standard deviations to the sample means). To address the important issue of random input vectors vs. typical usage input vectors, the aggregate results for the two local circuits are presented separately.

Due to data collection limitations in the simulator used, it was impractical to collect communications data on every signal line in the circuit. (The functional evaluation data *is* collected for every component.) To limit the number of signals that need to be monitored, we assume that the circuit is partitioned using a variant of the cones partitioning algorithm [7, 10], in which the only signals that are candidates for interprocessor communication are the latch outputs. As a result, an LP consists of one latch with all associated combinational logic gates at the latch input.

When investigating the predictive optimism technique, we exploit a property of the cones partitioning and observe that messages are constrained to have time stamps that coincide with clock edges (the only time a latch output can change). This implies that the predictors can be exercised once per clock, rather than once per simulated time step. We also exploit a property of the two-state logic simulation, predicting the existence of a message is equivalent to predicting the content of the message (since the message contents are constrained to be either “0” or “1”).

To assess the predictive optimism, the output of each latch is fed into each of the three predictors described above (PSP, SSP, and IPP). Prediction accuracy is then computed as the ratio of successfully predicted messages to the total number predictions.

5 Empirical Results

5.1 Resource Requirements

The raw functional evaluation counts for each interval for each component are presented in Figure 1 for circuit s510 of the ISCAS-89 set. The number of components in this circuit is 217. For each component, tracing the line across the time intervals gives an indication of the variability in the computational workload required for that component. The high-frequency bouncing indicates the degree to which neighboring time intervals reflect each other, and the similarity in height over the entire run indicates the stability throughout the simulation execution. As can be seen from the graph, the long term stability is excellent, and the neighboring time intervals have fairly good stability as well.

While this type of plot shows the most detail, it is impractical for presenting data from over 140,000 components. To reduce the data to manageable form, we calculate the sample standard deviation and mean of the counts for each component across the time intervals. The ratio of the standard deviation to the mean (the coefficient of variation) is then plotted as a histogram to show its distribution. To correlate the histogram plots with the raw data, Figure 2 shows the histogram for circuit s510. Note that the highest value in the distribution has a coefficient of variation of approximately 0.6, indicating that there are no components with a large variability in functional evaluations.

The variability histogram for all of the ISCAS-89 benchmark circuits is shown in Figure 3, and the histogram for the two local circuits (with typical usage input vectors) is shown in Figure 4.

For the ISCAS-89 set, under 0.05% of the components have a coefficient of variation over 1.0, indicating that the

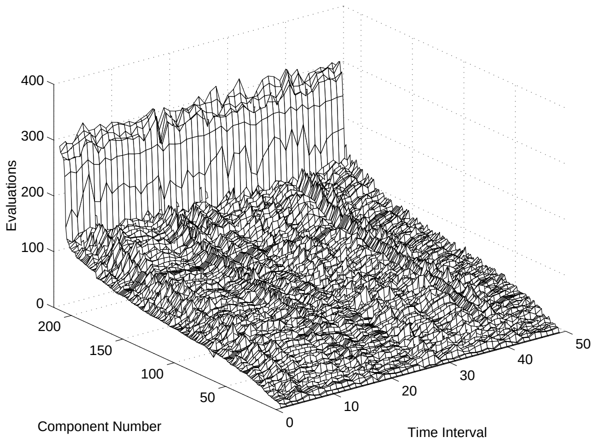

Figure 1: Functional evaluations for circuit s510

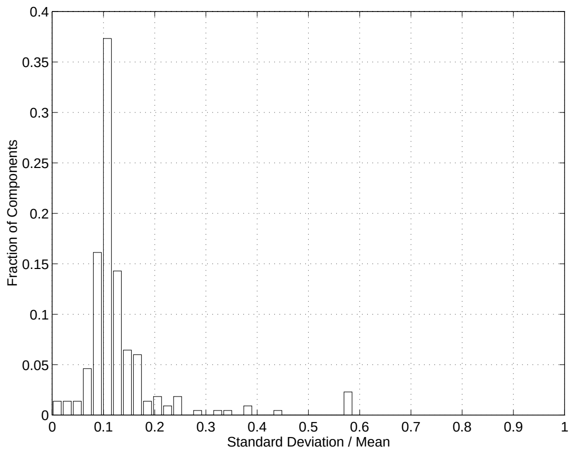

Figure 2: Variability of evaluations for circuit s510

vast majority have a limited degree of variability.

For the local circuits, just under 5% of the components have a coefficient of variation over 1.0. Of these components, however, all have a mean number of functional evaluations per interval below 2, indicating that they are relatively insignificant in terms of their impact on the total computational workload. Figure 5 shows the raw data for the local circuit with the highest variability. Although there are variations present (components with increasing evaluation counts and components with decreasing evaluation counts), generally, there is broad-based stability in the computational workload requirements, even with the utilization of typical usage input vectors.

Switching our consideration to the communications volume, the raw message counts are presented in Figure 6 for circuit s1196 of the ISCAS-89 set. The number of communications channels in this circuit is 18. For each channel, following a line across the time intervals gives an indication of the variability in the communications volume required for that channel. Again, we see fairly good stability across

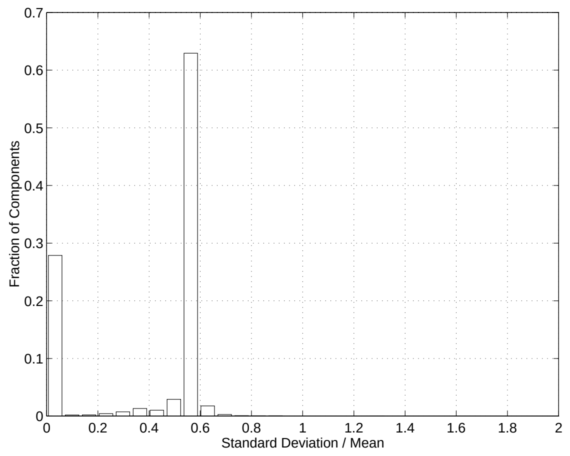

Figure 3: Variability of evaluations ISCAS-89 circuits

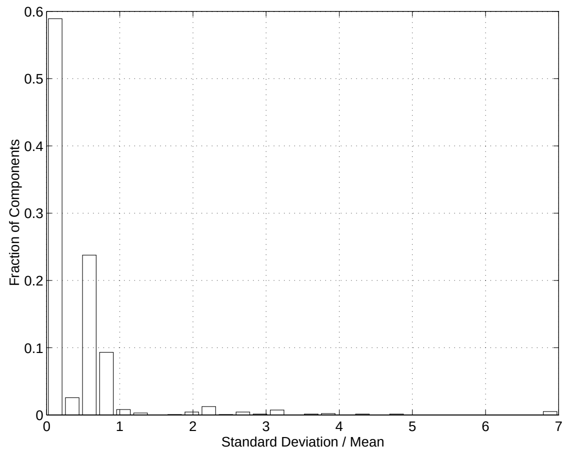

Figure 4: Variability of evaluations for local circuits

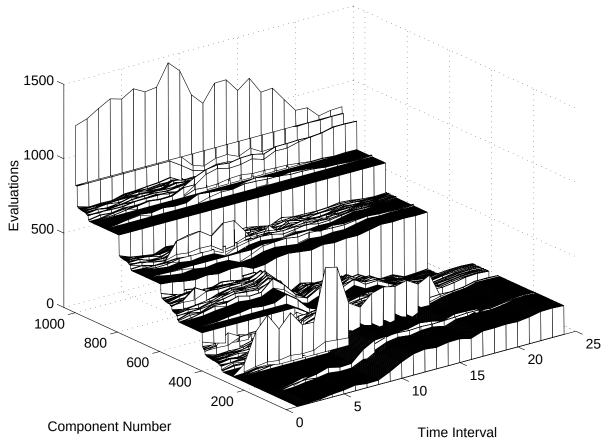

Figure 5: Functional evaluations for local circuit 2

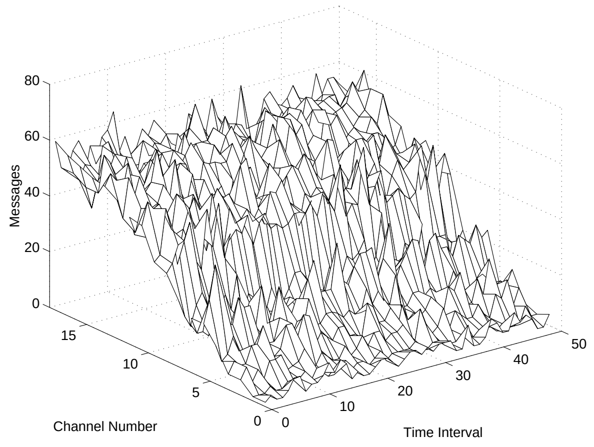

Figure 6: Message volume for circuit s1196

a wide time interval, and reasonably good stability across shorter time intervals.

As with the functional evaluation data, we calculate the coefficient of variation for each channel and plot a histogram of these values. Figures 7 and 8 are the plots for the ISCAS-89 set and the two local circuits, respectively. Across the ISCAS-89 benchmark set, a relatively small fraction of the channels have a coefficient of variation above 1.0, and for those that do, a significant fraction have a small mean (below 2 messages per interval).

For the local circuits, the results are very similar to the ISCAS-89 benchmark set. Approximately 5% of the communications channels have a coefficient of variation over 1.0, with virtually all of those over 1.0 having a very small mean (less than 0.5 messages per interval). The statistical outliers simply will not have a significant impact on the overall communications performance, so their high coefficient of variation should not be a significant problem.

The results above support the preliminary conclusions drawn in [3]. Historically measured resource requirements can be a reasonable predictor of future resource requirements. We have extended the work in [3] to include a wider range in circuits (some of which are considerably larger), test the sensitivity to random input vectors, and include measurements of communications requirements in addition to computational workload.

5.2 Individual Message Prediction

The entire ISCAS-89 benchmark set contains a total of 9696 latches, providing us with the same number of channels to predict. Figure 9 shows the distribution of messages values for all of the channels. Note that a channel that has a value of “0” 40% of the time and “1” 60% of the time will be counted as having an average value of 0.6. It is worth noting that over 40% of the channels are stuck at “0” or “1.” For these channels, we expect any reasonable binary predictor to perform with near perfect accuracy. Any improvements in accuracy beyond this will be gained for channels with average value between 0 and 1.

Figure 10 shows the accuracy of the PSP at predicting interprocess messages for the ISCAS-89 set. The mean ac-

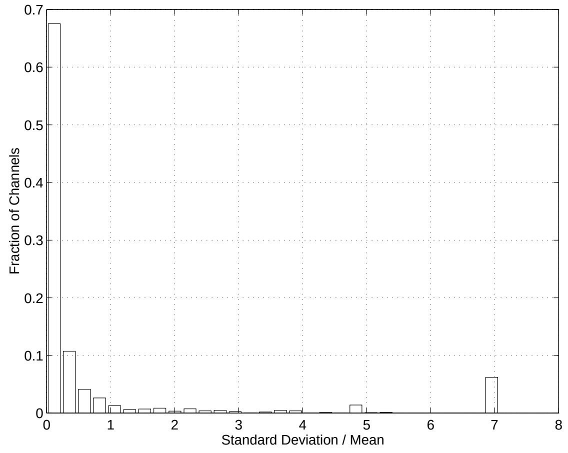

Figure 7: Variability of message vol. for ISCAS-89 circuits

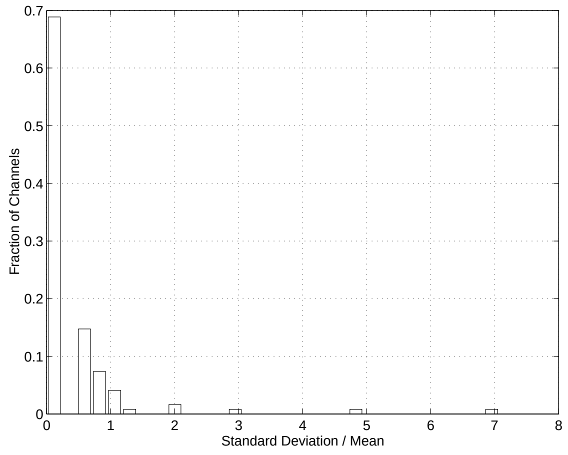

Figure 8: Variability of message vol. for local circuits

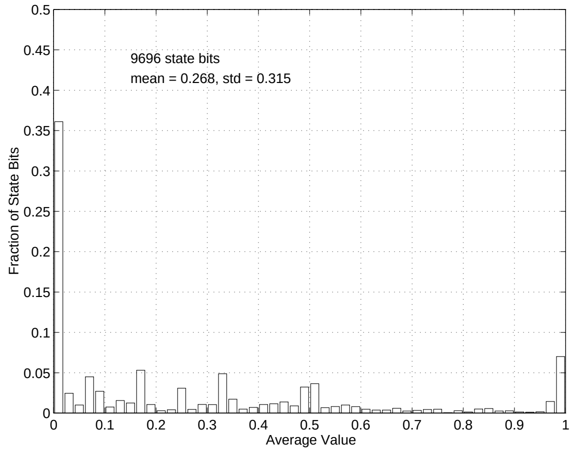

Figure 9: Dist. of channel values for ISCAS-89 circuits

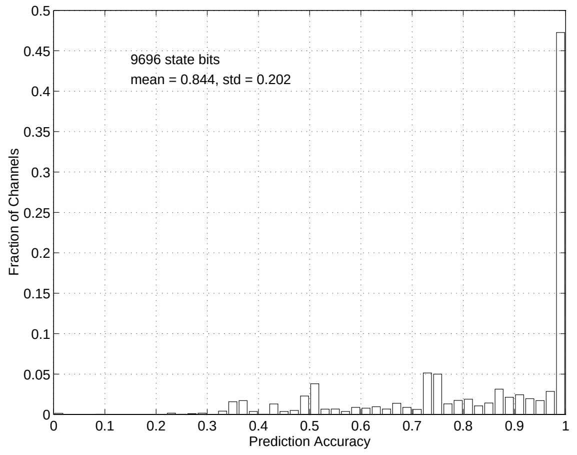

Figure 10: PSP results for ISCAS-89 circuits

curacy is 84.4% with a standard deviation of 20.2%. As expected, over 40% of the channels are predicted with near 100% accuracy. It is important to note, however, that over 9% of the messages are predicted with an accuracy of less than 50%, with minimum accuracy being 0%. This is significant because we do not expect the prediction accuracy for a reasonable predictor to be less than 50% (i.e., worse than a random guess). This implies that the optimistic assumption is poor for these channels.

The accuracy of the SSP is shown in Figure 11. Here we see that the mean has increased slightly to 84.9% and the standard deviation has dropped significantly to 17.8%. It too attains a near perfect accuracy for over 40% of the channels. Another important observation is that only 3% of the messages were predicted with less than 50% accuracy, however, the minimum accuracy is still 0%.

Comparing the performance of the PSP and SSP to the IPP shown in Figure 12, we first notice that it has the highest mean accuracy at 87% and lowest standard deviation at 15%. Again, a near perfect accuracy is attained for over 40% of the channels. In addition, only 1% of the messages are predicted with less than 50% accuracy as compared to the PSP at 9% and SSP at 3%. Another significant improvement is that the minimum accuracy of the IPP is 48% as compared to 0% for both the PSP and SSP. Summary results for the ISCAS-89 set are given in Table 1.

Table 1: Prediction statistics for ISCAS-89 circuits

Predictor	Mean	StdDev	$< 50\%$	Min
PSP	84.4%	20.2%	9.2%	0.0%
SSP	84.9%	17.8%	3.1%	0.0%
IPP	87.3%	15.2%	1.5%	48.4%

To test the sensitivity of these results to the use of random input vectors, we repeat the experiment on the local circuits exercised with typical usage vectors. Figure 13 shows the distribution of messages and Figures 14, 15, and 16 show the message prediction accuracy of the PSP,

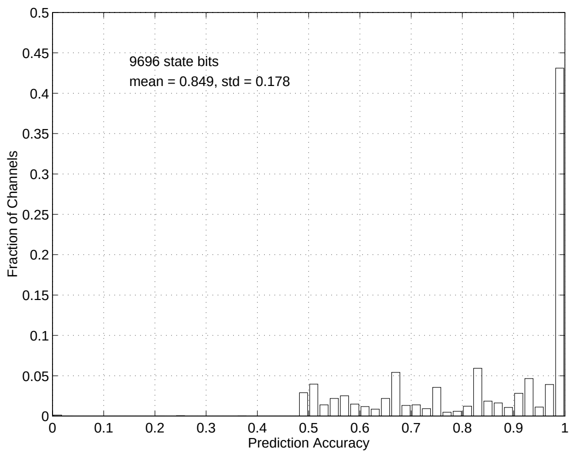

Figure 11: SSP results for ISCAS-89 circuits

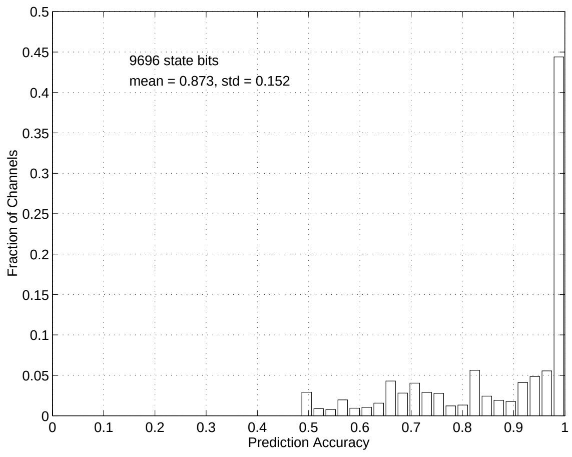

Figure 12: IPP results for ISCAS-89 circuits

SSP, and IPP respectively for the two circuits.

The summary results are given in Table 2. For these circuits, the SSP has the lowest mean and highest standard deviation. On the other hand, the PSP has a minimum accuracy of 0% as compared to 45% for SSP and 63% for IPP. For the IPP, the mean shows a slight improvement over the PSP but the standard deviation is significantly reduced.

Table 2: Prediction statistics for local circuits

Predictor	Mean	StdDev	$< 50\%$	Min
PSP	92.3%	14.0%	0.8%	0.0%
SSP	87.9%	16.5%	1.6%	45.0%
IPP	92.7%	11.5%	0.0%	62.7%

Due to it's sophistication, it is not surprising that the IPP performs better than both the PSP and SSP. What is more surprising is how poorly the PSP predicts some channels. It remains to be seen how the effects of poor predictions affect the performance of a simulation.

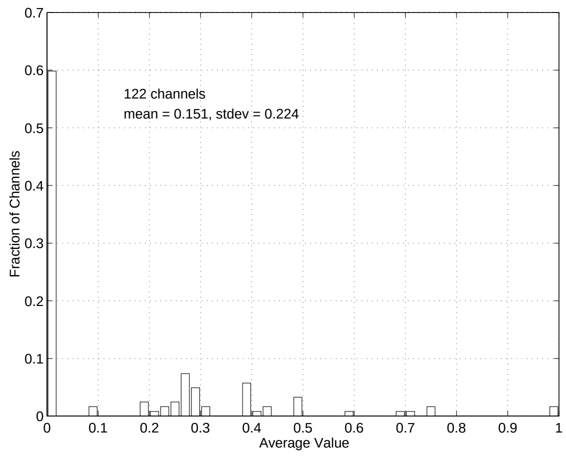

Figure 13: Distribution of channel values for local circuits

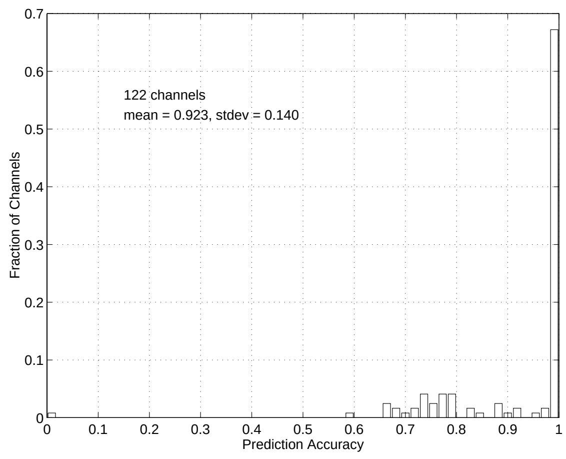

Figure 14: PSP results for local circuits

6 Conclusions

This paper investigates the use of historical information for the prediction of future computation and communications resource requirements. It has been shown that there is clear merit for incorporating historical information in partitioning decisions. Refining partitioning cost functions to include functional evaluations as well as communication rates between processors may offer improved performance.

We also present a novel technique called *predictive optimism* which uses binary prediction schemes to increase the accuracy of optimistic assumptions. By predicting the arrival of messages, efficient throttling mechanisms can be developed for optimistic algorithms. By predicting the content of messages, fewer rollbacks can be expected, potentially improving the overall performance of the simulator.

Merging these two techniques provides another dimension to the partitioning problem, by characterizing channels by their entropy or information content. Partitioning might then be encouraged across highly predictable channels. The incremental parsing predictor used in this experi-

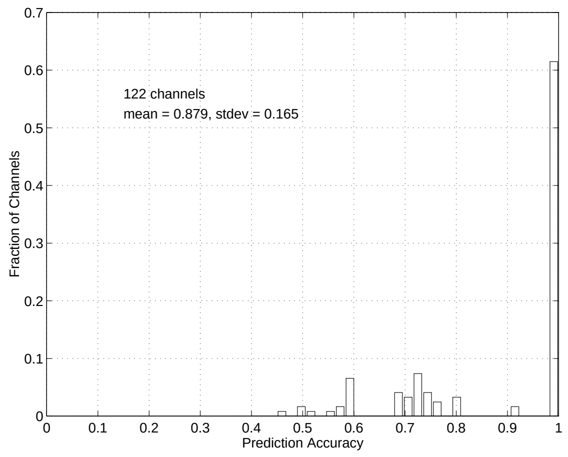

Figure 15: SSP results for local circuits

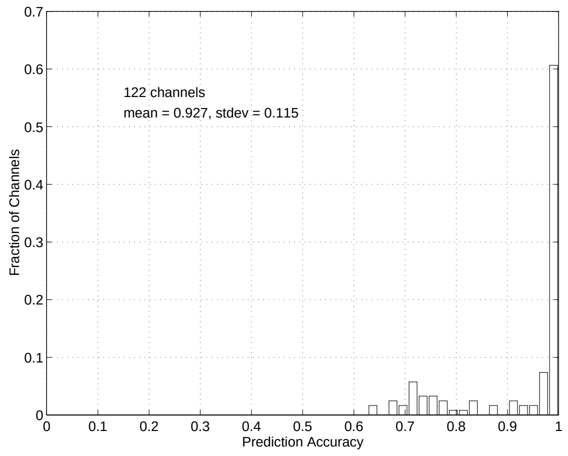

Figure 16: IPP results for local circuits

ment builds a prediction dictionary that also can be used to estimate the entropy of a channel [9].

There are, however, a number of caveats that must be stated. It is not clear if improving the prediction accuracy will significantly improve overall simulator performance. Also, we have compared the accuracy of a predictor that models existing optimistic algorithms (PSP) to that of a very simple predictor (SSP) and a fairly sophisticated predictor (IPP). While the accuracy of the IPP was better than both the PSP and SSP, it is the most costly to execute (primarily in terms of memory requirements). The tradeoffs between prediction accuracy and execution costs have not yet been examined.

Random input vectors were used on a reasonably large set of circuits, but tests using typical inputs were limited to only two circuits. These results need to be extended to a larger set of real input vectors before they can be safely relied upon.

There are other areas that still need investigation. Are channels where the optimistic assumption has a low prediction accuracy the major cause of erratic performance of optimistic time synchronization algorithms? For simulations with a larger communications symbol set, can the prediction algorithms be modified to incorporate these symbols so that prediction of message content is possible? Also, we must develop partitioning algorithms that effectively exploit the available historical information, be it computational workload, communications volume, and/or the information content of communications channels.

References

[1] A. Boukerche and C. Tropper. A Static Partitioning and Mapping Algorithm for Conservative Parallel Simulations. In *Proc. 8th Workshop on Parallel and Distributed Simulation*, pages 164–172, July 1994.

[2] F. Brglez, D. Bryan, and K. Kozminski. Combinational Profiles of Sequential Benchmark Circuits. In *Proc. Int'l Symp. on Circuits and Systems*, pages 1929–1934, May 1989.

[3] R. D. Chamberlain and C. L. Henderson. Evaluating the Use of Pre-Simulation in VLSI Circuit Partitioning. In *Proc. 8th Workshop on Parallel and Distributed Simulation*, pages 139–146, July 1994.

[4] M. Feder, N. Merhav, and M. Gutman. Universal Prediction of Individual Sequences. *IEEE Trans. on Information Theory*, 38(4):1258–1270, July 1991.

[5] Y.-B. Lin and E. D. Lazowska. Processor Scheduling for Time Warp Parallel Simulation. In *Proc. SCS Multiconference on Advances in Parallel and Distributed Simulation*, pages 11–14, January 1991.

[6] N. Manjikian and W. M. Loucks. High Performance Parallel Logic Simulation on a Network of Workstations. In *Proc. 7th Workshop on Parallel and Distributed Simulation*, pages 76–84, 1993.

[7] R. B. Mueller-Thuns, D. G. Saab, R. F. Damiano, and J. A. Abraham. VLSI Logic and Fault Simulation on General-Purpose Parallel Computers. *IEEE Trans. on Computer-Aided Design*, 12(3):446–460, March 1993.

[8] D. M. Nicol and P. F. Reynolds, Jr. A Statistical Approach to Dynamic Partitioning. In *Proc. SCS Multiconference on Distributed Simulation*, pages 43–51, 1985.

[9] N. T. Plotkin and P. P. Varaiya. The Entropy of Traffic Streams in ATM Virtual Circuits. In *Proc. IEEE Infocom*, pages 1038–1045, June 1994.

[10] S. P. Smith, B. Underwood, and M. R. Mercer. An Analysis of Several Approaches to Circuit Partitioning for Parallel Logic Simulation. In *Proc. Int'l Conf. on Computer Design*, pages 664–667, 1987.