A time warping algorithm is included in the waveform relaxation circuit simulation in a distributed environment in a multicomputer network. The algorithm feature, is that it can exploit parallelism in the spatial, iteration and temporal domains. The algorithm is characterized by a relative insensitivity to the interprocessor communication overhead. The simulator is ported on a network of Sun Sparc-2 workstations using a 10 Mbits/sec. communication link. Speedups results obtained are comparable for a shared memory architecture [1,5].

1. INTRODUCTION

The simulation of integrated circuits at the detailed electrical level continues to be an indispensable element in the overall IC design and verification process. However, the dramatic increase in the complexity of IC's over the last years has strained the capabilities of traditional circuit simulators such as SPICE.

With today technology it is possible to pack a few million transistors in a chip, and device densities continues to grow at a faster rate than the performance of the computer that can simulate them, and designers agree it is a bottleneck limiting their productivity. This situation has resulted in creation of different approaches to obtain acceleration in the simulation process for VLSI circuits, involving non-standard hardware resources and new algorithm methods.

Therefore, it is considered that a viable approach to improve the performance of VLSI circuit simulation is the use of parallel and distributed processing. Much attention has been focused in the last ten years on the mapping of circuit simulation algorithms to parallel architectures which use the shared-memory paradigm to facilitate parallel computation [1,2,3,4].

With the advent of high performance workstations and high speed interconnection networks, distributed processing offers a cost effective and attractive solution to the circuit simulation problem. Workstations available at Universities and Research Centers can be effectively used for distributed simulation, resulting in shorter simulation times with no additional investment. Also, workstations are more general purpose computers and are much easier to use and hence can be used for a wider variety of applications as compared to parallel dedicated machines. The major drawback of using such a network of workstations in circuit simulation is the high overhead involved in the interprocess communication, and therefore, a careful analysis is needed. The approach used in our research is the use of distributed algorithms for the acceleration of circuit simulation. Therefore, these distributed algorithms considered must guarantee efficient use of the hardware resources. This can be achieved by partitioning the problem and allocating them through the system, and minimizing their interactions.

2. PORTABLE PARALLEL SIMULATOR

A portable parallel simulator developed at the University of Illinois [5], was used to evaluate the WR algorithms in the network of Sun-Sparc2 workstations, using 8 workstations. The software allows to use different levels of relaxation methods, based on the coupling degree between subcircuits. Because this software evolved through a hierarchical architecture (the CEDAR machine) [1], WR methods is supported through a message-passing paradigm, and the Iterative Time Analysis (ITA) in conjunction with the Direct Methods (DM) are supported through the shared-memory paradigm. Because we are interested only in the WR algorithms, the top level platform of this simulator is used for our environment research study.

To improve the convergence speed of relaxation methods, WR involves partition in two domains: the spatial and temporal domain. The spatial domain partitioning involves grouping strongly-coupled components in the circuit under simulation into small subcircuits. These subcir-
circuits are solved using the DM and the relaxation is applied between subcircuits. Since WR works at the waveform level, it can involve substantial memory requirement and management. To make efficient use of these issues, most WR algorithms also partition the simulation interval into a number of subintervals called windows. This partitioning is then done in the temporal domain.

The simulation of a subcircuit in one WR iteration and in a given time window could be placed on a processor and several of these tasks can be done concurrently using multiple workstations. This imply that parallelism could be exploited in all domains: spatial, iteration and temporal domains.

The relationship between the subcircuits is defined by a partial ordering which is generated by extracting coupling impedance information from the subcircuit graph produced by the partitioner, as shown in Figure 1 for a circuit example of a chain of four inverters.

The graph in Figure 1 defines the relaxation scheme by specifying the data dependencies to be enforced within one iteration. This is the typical data dependence of a single Gauss-Seidel iteration. Therefore, subcircuit #1 must be solved first before subcircuit #2, and subcircuit #3 must be solved after subcircuit #2 has been solved, and so on. By allocating every subcircuit to a different processor, the enforcement of the strict ordering in Gauss-Seidel offer very little parallelism when the task graph is very narrow. In the other hand, the Gauss-Jacobi method is generated by placing no enforcement at all in the solving order of the subcircuits, therefore, maximum parallelism is obtained, but only one out of four processors is doing useful work for the example shown above.

The portable parallel simulator makes a trade-off between parallelism and convergence, by combining a data-flow scheduling approach with an event-driven technique using a priority queue mechanism [3]. Two separate task queues with a priority scheme are used. The Data-Flow (DF) queue has a higher priority than the Event-Driven (ED) queue. Task are taken from the DF queue, which maintain the subcircuit ordering of the Gauss-Seidel method. Processors will look for tasks in the ED queue only if the DF queue is empty. Tasks in the ED queue are not ready to fire but they do have at least one input with new information.

3. CONSERVATIVE SIMULATOR PERFORMANCE

The graph shown in Figure 2 represents the sequence of events generated by the parallel DF-ED WR simulator executed on 4 workstations for the circuit shown in Figure 1, with two windows. Each subcircuit is assigned to a different workstation. In the directed graph, each node (circle) represent an event generated from the subcircuit simulation and the edges correspond to messages between nodes (node voltage waveforms). Each node is assigned a weight corresponding to the execution time of the corresponding subcircuit simulation. Each arc is assigned a weight corresponding to a communication time. The longest path in the graph, the critical path, represents the shortest time in which this circuit can be simulated using this conservative WR algorithm (dark arrows). At the end of each window, there is a need to obtain the global convergence, which is represented by the gray block between the two dark horizontal lines. This synchronization overhead represents a big penalty for the size of the circuit in a multicomputer network using the 10 Mbits/sec communication link (ethernet).
4. SIMULATOR PERFORMANCE IMPROVEMENT

In the simulator presented above, the simulation time never advanced ahead for the next window, unless all the subcircuits converge in the current simulated window. This conservative technique has two main problems:

1. Subcircuit tasks become scarce when the average subcircuit task dependency graph width is less than the number of processors, creating processor starvation. And this situation worsens with the number of iterations, since more subcircuits have already converged, reducing the amount of work.

2. Global waveform convergence in a distributed environment consumes a significant amount of simulation time. This synchronization barrier gets worsened when the number of processors get larger, and the number of windows is increased.

To address the first concern, a strategy is proposed to add a third level of queues to the DF-ED technique, which will be used for scheduling tasks for future events. The mechanism to be used is based on the Time-Warping (TW) technique [6]. The strategy is to add a new queue to insert all the subcircuits that are available for simulation with all the future windows contained in the simulation interval. The future subcircuits events are ordered in increasing window time. The TW queue has the lower priorities of all the queues, thus events are taken from this queue only when the other DF and ED queues are exhausted. This implies that, instead of a processor staying idle as in the conservative methods, the processor proceeds by removing tasks from the TW queue, simulating subcircuits for subsequent future windows. This results in processors able to simulate different subcircuits in different iterations and in different windows.

To overcome the second limitation, it is proposed that processors synchronize by event-messages which contain the timestamp of the current simulation. If a processor receives a timestamp from the past of its current simulation, the processor must rollback its simulation to a time equal to the timestamp received, to recover from the causality error.

5. CRITICAL PATH ANALYSIS IN THE TW-WR DISTRIBUTED CIRCUIT SIMULATION

The same circuit example is used for the TW-WR analysis. The sequence of events generated from the TW-WR distributed algorithm is illustrated in Figure 3. The importance of this analysis is that it demonstrates that the TW-WR technique can reduce the lower bound of the critical path found in the conservative scheme. The TW-WR algorithm takes advantage of the lookahead computation only when the data-flow and event-driven queue have been exhausted. This optimistic method assumes that synchronization is not needed until it receives proof that it was. Thus, the TW-WR will proceed with the computation of the future windows on the premise that all messages with earlier timestamps have arrived. The length of the critical path shown by dark arrows in Figure 3 is only 272 ms, without considering the synchronization overhead at the end of the simulation, which is shorter than the one obtained with a conservative WR of 408 ms. This indicates that the critical path is not a lower bound with the TW-WR. An overhead in TW-WR needs to be added to allows the Global Virtual Time (GVT) computation to determine the end of simulation. This GVT is the time of the local clock that is farthest behind for all simulating subcircuits. Therefore, GVT gives the lower bound on the progress of a simulation in terms of simulation time. All messages and states saved prior to GVT are subject to garbage collection, which frees their storage space.

The main advantages of the TW-WR algorithm are the following:

First, it explores parallelism at all domains, giving first priority to the spatial domain. This implies that the TW-WR will proceed in the temporal domain only when all events in the spatial domain have been exhausted.
Second, synchronization overheads is reduced proportionally to how often the TW-WR algorithm recomputes the GVT.

Third, the lower bound of the critical path is reduced compared to the conservative WR simulation method, because of the window pipeline simulation effect.

6. EXPERIMENTAL RESULTS

Experiments were performed on a network of Sun Sparc-2 workstations using different benchmark industrial circuits. Simulations were executed with up to 8 workstations using a 10 Mbit/sec. ethernet communication link. Figure 4 presents the CPU runtimes for six industrial circuits versus the number of workstations, using the conservative WR (dashed lines) and the TW-WR method (solid lines).

Decpla is the example of a small circuit with a narrow task graph with an average width of only 2. This circuit is a real industrial circuit (critical portion of a PLA) and captures many aspects of standard circuits in a small example. Although this example shows the lower speedup (1.93) with the TW-WR method for all circuit examples, the relative speedup with respect to the conservative method is significant 2.8 higher.

The circuit which benefits the most with the TW-WR algorithm is Decpla_l, since each entire copy of Decpla is assigned to each workstation. The highest speedup of 7.36 correspond to this ideal case in which subcircuits are completely decoupled from each other, thus no rollbacks were necessary during the simulation.

7. CONCLUSIONS

The results of the experiments (using commercial benchmark circuits) prove that TW-WR performs better than the conservative WR algorithm. The price paid to improve the performance is the increase of memory size. The increase performance ranges from 1.6 to 3.2 (network of Sun Sparc-2 workstations with a 10 Mbit/sec intercommunication link). The memory requirements to simulate the commercial circuits using the TW-WR more than doubled in all cases in comparison with the conservative algorithm. The speedups obtained are similar to the one reported in a dedicated parallel hardware (Cedar machine) [1].

8. REFERENCES


