Matrix	Sparc2	CPU secs for PACE Processors			
	(secs)	1 Proc.	2 Proc.	3 Proc.	4 Proc.
s208	7.77	8.14	4.44	2.96	2.22
s510	13.6	17.68	9.18	6.80	4.76
s641	27.70	36.50	19.00	13.50	9.50
s953	25.60	39.20	20.40	14.00	10.80
s1196	47.84	68.64	36.40	24.96	18.72

Table 1: CPU time for model evaluation.

Matrix	Sparc2	CPU secs for PACE Processors			
	(secs)	1 Proc.	2 Proc.	3 Proc.	4 Proc.
s208	6.04	11.15	5.84	4.04	3.00
s510	14.34	24.79	13.00	8.87	6.71
s641	29.27	50.13	26.13	16.63	13.47
s953	33.60	47.60	24.80	17.20	13.20
s1196	58.86	100.20	52.42	35.78	27.25

Table 2: CPU time for finding DC operating points.

40MHz. The speedup relative to a single processor is the same for all circuits and is about 3.6. Of course, we cannot extrapolate these results to larger or more complex circuits without experimentation.

Table 2 shows the results from simulation runs for obtaining DC operating points for the same five benchmark circuits used in table 1. The table also shows the CPU seconds for ADVICE on a Sparc2 workstation. Since these are small circuits, we expect the simulation time to be dominated by model evaluation as opposed to the sparse matrix solution. This is clearly indicated in the results. The last column gives the speedup obtained with four PACE processors in relation to a single processor.

The prototype of PACE is operating at 20Mhz which is half the rated clock speed of the *i*860. In addition, the code running on PACE is not optimized for performance. We expect that we can at least triple the current performance by operating at the rated clock speed on a printed circuit board version of the system running optimized code.

5 Conclusion

This paper gives heuristics for binding the circuit elements such as resistors, capacitors and MOSFETS to processors for model evaluation and matrix stamping. The matrix rows are assumed to be statically distributed among the processors for efficient SOLVE operations. Our research investigated the contributions of processor load distribution and the interprocessor communication during the model equation formulations.

Model evaluation is a computationally and I/O intensive problem. A multiprocessor architecture can be exploited to yield the maximum performance by a clever partitioning of the work load amongst the processors. The network bandwidth, while it influences the SOLVE part of circuit simulation does not significantly affect the model evaluation part. The performance sensitive issues in model evaluation in a distributed or parallel processing environment are CPU processing power and the processor to memory bandwidth. This is because evaluation of complicated models needs frequent memory accesses for updating the model information. Study of communication requirements for circuit simulation under distributed environments is under study. While we used the direct method for solving the matrix equations, our research on parallel model evaluation becomes even more meaningful for relaxation-based algorithms on multiprocessors. This is because such algorithms are easy to parallelize and the bulk of the computation time is in model evaluation.

References

[Agra93] P. Agrawal, S. Goil, S. Liu and J. A. Trotter,. "PACE: A Multiprocessor System for VLSI Circuit Simulation," *Proc. SIAM Conference on Parallel Processing*,Norfolk, VA, Mar. 1993, pp. 573-581.

[Cox91] P. Cox, R. Burch, et al. "Direct circuit simulation algorithms for parallel processing," *IEEE Trans on CAD Vol. 10. No. 6*, 1991, pp. 714-725.

[Liu82] S. Liu and L. W. Nagel, "Small-Signal MOSFET Models for Analog Circuit Design," IEEE J. Solid-State Circuits, Vol. SC-17, No. 6, Dec. 1982, 983-998.

[Sada88] P. Sadayappan and V. Visvanathan, "Circuit Simulation on Shared Memory Multiprocessors," *IEEE Trans. on Computers,* Vol. 37, No. 12, December 1988, pp. 1634-1642.

[Teli91b] R. Telichevesky, P. Agrawal and J. A. Trotter, "A New *O(nlogn)* Scheduling Heuristic for Parallel Decomposition of Sparse Matrices," *IEEE Int. Conf. on Computer Design: VLSI in Computers and Processors*, Cambridge, MA, Oct. 1991, 612-616.

[Trot90] J. A. Trotter and P. Agrawal, "A Multiprocessor Architecture for Circuit Simulation," *IEEE Int. Conf. on Computer Design: VLSI in Computers and Processors*, Cambridge, MA, Oct. 1991, pp. 621-625.

average the model computation cost. The total LOAD cost per element on a processor in PACE is given by

Total Cost = Comp. Cost + Local FIFO Cost + Remote FIFO Cost + transmission Cost

In this formula, the computation cost reflects the model evaluation cost and the rest are communication-related costs. In PACE, the communication network is a wide, high speed broadcast bus with a FIFO queue attached to each of the processors. Thus, any information transmitted over the network is written into the transmitting processor's FIFO, transmitted over the network and read into the receiving processor's FIFO. The communication sequences required for transmitting the matrix data are precomputed and loaded into the PACE sequencer. Model evaluation is dominated by computing costs and the communication cost is comparatively negligible. However, even though the communication cost per transaction may be small, the number of transactions determines its contribution during model evaluation. We study different heuristics using a simulated PACE system and present results from each of the heuristics studied.

We experimented with three slightly differing heuristics. The first of these, *Locality-driven* heuristic, penalizes remote stamps while encouraging local stamps. Elements are assigned to that processor with the maximum number of rows of the elements Jacobian submatrix. The second one, *Load-balancing* heuristic, emphasizes load balancing during model evaluation and matrix stamping. For each type of element, a processor with the current minimum computational load is chosen and the element is allocated to it. The elements are processed in decreasing order of their computational load. The last one, *nodal-connectivity-driven* heuristic, addresses both the issues of load balancing and communication and appears to produce the best results. First, a processor with the minimum load is chosen, then a *best fit* element, with the maximum number of matching Jacobian submatrix rows with the rows already allocated to the processor, is chosen. A numerical measure, *the affinity index*, given by the number of non-zeros in the element's submatrix matching those in the rows already allocated to a processor, is used. For every processor, the elements are sorted in decreasing order of affinity indices. The heuristic examines the processors in the order of decreasing computing load. By examining the sorted list of elements on each processor, the element with the maximum affinity is assigned to the processor.

3.2 Scheduling Interprocessor Communication

Parallel processing of the LOAD part of circuit simulation can be made efficient by scheduling the model evaluation and Jacobian stamping tasks in an optimal fashion. In PACE, the processors can independently conduct the model evaluation as the memory is distributed and not shared. Nevertheless, the data required for model evaluation has to be communicated over the shared network. Thus the tasks of model evaluation and data sharing need to be carefully scheduled for efficiency. The overall schedule should be such that it permits each processor to continue with computations without waiting for other processors. A sequence of computing and communication tasks, using task dependencies, is generated by a task scheduler (see [Teli91b] for details). Fortunately, for a given circuit simulation session, this sequence is valid for every iteration. Thus, it can be generated statically and once for a complete simulation session. Since some models take a non-deterministic amount of time for evaluation, due to the device operating point dependency in some cases, the scheduler cannot accurately predict data arrival times. It therefore uses a cost model.

In PACE, each processor has a bidirectional FIFO of 48Kbytes associated with it. This buffering arrangement decouples the *transmissions* from a source processor from *receptions* by a destination processor. Thus the destination processor can continue with the computation of a model instead of waiting for data arrival.

4 Results

The PACE hardware is integrated with the ADVICE simulator. The algorithms for model evaluation and matrix solution and the user interface are the same as ADVICE. The only difference is that these algorithms have been modified using the parallel LOAD and SOLVE heuristics to run on the four processors of PACE. At present we are able to correctly execute small example circuits on PACE. This section will describe some preliminary results obtained from PACE with the nodal-connectivity driven model evaluation method.

Table 1 gives the result of model evaluation using the nodal-connectivity-driven element allocation scheme for five benchmark circuits. The table also shows the CPU seconds for model evaluation on a Sparc2 workstation. The speedup obtained with four PACE processors in relation to a single processor can be computed in a straightforward manner. The 4-processor PACE operating at a clock frequency of 20MHz is about 3 times faster than a Sparc2 at

$(t_n - t_{n-1})$. Using this value of $\dot{x}$ in equation 1 at timepoint t_n we get,

$$F(x_n, \frac{x_n - x_{n-1}}{h_n}, t_n) \quad = \quad 0 \qquad (2)$$

Equation 2 is solved using an iterative scheme such as Newton-Raphson (NR). Equation 3 corresponds to the kth iteration, for timestep t_n.

$$\left[\ \frac{\partial F}{\partial x}(x^{k-1}) \ \right] x^k = -F(x^{k-1}) + \left[\ \frac{\partial F}{\partial x}(x^{k-1}) \ \right] x^{k-1}. \quad (3)$$

$[\frac{\partial F}{\partial x} x^k]$ is the Jacobian matrix of F, k being the iteration counter. Starting from the initial guess, x^0, equation 3 is solved iteratively until convergence is reached at time t_n at which time, a new time step is selected and the process is repeated at time t_{n+1}.

At each iteration, the Jacobian matrix is assembled from the circuit equations. This is the LOAD step and involves the numerical computation of the terminal characteristics (and their derivatives) of the various semiconductor devices in the circuit. We shall refer to this step of LOAD as *model evaluation*. The individual contributions to the Jacobian matrix from the circuit elements are dictated by the nodal connectivity of the circuit and are stamped into the appropriate rows and to the right hand side of the equations. This constitutes the *matrix stamping* step.

The circuit elements are fully electrically characterized using model definitions. The elements could be as simple as a resistor and as complex as a MOSFET or a bipolar transistor. For example, a model for a resistor includes the first-order voltage coefficients, a quadratic dependence of resistance value on temperature, and a global scale factor that modifies the values of all elements with the given model name. Hence, the value of a resistor connected between the circuit nodes A and B, at temperature T, is determined by computing it from a model equation showing all the above dependencies.

Each circuit element has a Jacobian submatrix associated with it. These submatrices are used to formulate the global Jacobian matrix. The model evaluation routine uses several parameters to derive the submatrix. Regardless of the parameters chosen for modeling, the position of the submatrix and the the zero, non-zero structure in the global circuit matrix is determined by the circuit connectivity. However, different elements may have different sizes of submatrices, depending upon the number of nodes in the element. A MOSFET will stamp more locations than either a diode, capacitance or a resistor, since it is a multiterminal device. A complex circuit element such as a MOSFET may require nearly 60 parameters to characterize it accurately. Examples of parameters used are channel width and transconductance. Depending on the accuracy required the same circuit element can have several different models. For example, a MOSFET can be modeled, in ADVICE, using the CSIM [Liu82], ASIM, CFSIM, or IM models. ADVICE is the inhouse circuit simulator used for IC designs in Bell Laboratories.

3 Parallel Model Evaluation on PACE

In PACE, the matrix data and the model information are distributed among the processors. For achieving maximum parallelism, the computational load is balanced and the communication of Jacobian entries among the processors minimized. Since model evaluation is a far more expensive operation than transmitting evaluated submatrix stampings, duplication of model evaluation on processors is avoided. It is assumed that the rows of the matrix have been statically partitioned among the processors. This data partitioning results in a unique subset of the Jacobian matrix permanently allocated to each of the processors. Now, the circuit elements are distributed among the processors for model evaluation. This is the element allocation problem addressed in this paper.

An element's submatrix contributes to more than one row in the matrix depending upon the element's connectivity in the circuit. These rows may be locally present in the processor on which the evaluation is done or they may have been assigned to other processors. In the latter case, the results of model evaluation are communicated over the interconnection network. If the matrix stamp is communicated then its value along with its address in the destination processor must be transmitted. The receiving processor will read the value and store it at the appropriate location in its data structure. In the following sections, we present a set of heuristics for assigning the circuit elements to the processors.

3.1 Partitioning of Circuit Elements for LOAD

The objective of element allocation to processors during the LOAD part of circuit simulation is balancing the model evaluation load and minimizing the number of *remote* stamps. The latter is due to the cost of communicating the model evaluation data across the communication network. In this paper, a simple formula is used for assessing the cost of LOAD. The time a processor spends in model evaluation often depends on the nodal voltages and the operating region of the device. For example, it takes less time to evaluate a MOSFET transistor in the cutoff region than when it is in the saturation region. In our computations, we

Parallel Model Evaluation for Circuit Simulation on the PACE Multiprocessor

Prathima Agrawal, Sanjay Goil, Sally Liu and John A. Trotter
AT&T Bell Laboratories
Murray Hill, NJ 07974

Abstract

Device model evaluation, an essential part of a circuit simulator, is a compute-intensive task. A multiprocessor-based circuit simulator that ignores the parallelization of model equation formulation (LOAD), and just parallelizes the solution (SOLVE) of the equations will seriously degrade the simulation performance. This paper describes methods of parallelizing the LOAD part of a circuit simulator on PACE (Parallel Architecture for Circuit Evaluation) a distributed memory multiprocessor designed at AT&T Bell Laboratories. This is integrated with the parallel SOLVE algorithms given in our earlier work. Load balancing and minimization of interprocessor communication are used as the primary objectives of the parallel LOAD heuristics studied. Performance results, using the prototype PACE system, on benchmark circuits show the feasibility of our approach.

1 Introduction

Circuit simulators are used to model and predict the electrical behavior of VLSI designs. Due to the detailed analyses steps used the size of circuits simulated at the device level is limited to a few thousand devices. Circuit simulators using direct methods solve a set of tightly coupled non-linear circuit equations at a number of discrete timesteps within the simulation interval. In keeping with the literature, we shall refer to the equation formulation step as LOAD and the solution step as SOLVE. The LOAD and SOLVE form the inner loop of a circuit simulator. Parallelization of equations for multiprocessors is difficult due to the tightly coupled and sparse equations. A number of researchers have developed parallel circuit simulators on shared memory multiprocessors [Sada88, Cox91] and studied their performance for both LOAD and SOLVE.

A prototype multiprocessing system, PACE (Parallel Architecture for Circuit Evaluation), with four Intel *i860* microprocessors is designed with circuit simulation as the main application [Trot90, Agra93]. The PACE hardware is a VME plugin board to a workstation. It is a distributed memory system where each processor has 4Mbytes of high speed memory. One of them, the control processor, has 8Mbytes. The processors communicate through a 96-bit wide, high speed broadcast bus. The bus width is chosen to enable the simultaneous transmission of a double precision floating point value (64 bits) and an index value (32 bits). The sequences of the interprocessor communication are precomputed and loaded into a communication sequencer. This architectural feature permits fast communication with a complete transaction taking place in 2 processor memory cycles. PACE is integrated with a software circuit simulation system, ADVICE, at AT&T Bell Laboratories.

2 Model Evaluation

Transient analysis studies the behavior of a circuit, in a time interval [0,T], by modeling it as a system of non-linear differential equations,

$$F(x, \dot{x}, t) \quad = \quad 0, \qquad x(0) = x_0 \tag{1}$$

where x is an n-dimensional vector denoting voltages and currents and $\dot{x}$ is the derivative of x with respect to time. $x(0)$ is the value of x at time t = 0. F is a mapping function. In the simulation interval [0, T], the non-linear differential equations are converted to a system of non-linear algebraic equations using an integration scheme such as the Backward-Euler formula. A solution of equation 1 is obtained at a number of discrete timesteps, $t_1, t_2 ... t_n$. The time derivative $\dot{x}$ is replaced by a divided difference approximation,

$$\dot{x} = \frac{x_n - x_{n-1}}{h_n}.$$

Here, x_n and x_{n-1} are the values of x at timesteps $t = t_n$ and $t = t_{n-1}$, respectively. h_n is the nth timestep