Parallel Simulation Algorithms for Grid-Based Analog Signal Processors

L. Miguel Silveira Andrew Lumsdaine Jacob K. White

Research Laboratory of Electronics
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

ABSTRACT

In this paper, specialized algorithms for circuit-level simulation of grid-based analog signal processing arrays on a massively parallel processor are described and implementation results presented. In our approach, the trapezoidal rule is used to discretize the differential equations that describe the analog array behavior, Newton's method is used to solve the nonlinear equations generated at each time-step, and a block conjugate-gradient squared algorithm is used to solve the linear equations generated by Newton's method. Excellent parallel performance of the algorithm is achieved through the use of a novel, but very natural, mapping of the circuit data onto the massively parallel architecture. The mapping takes advantage of the underlying computer architecture and the structure of the analog array problem. Experimental results demonstrate that a full-size Connection Machine can provide a 1400 times speedup over a SUN-4/280 workstation.

1 Introduction

The recent success using one and two dimensional resistive grids to perform certain filtering tasks required for early vision [Mead 88] has sparked interest in general analog signal processors based on arrays of analog circuits coupled by resistive grids. As is usually the case, before fabricating these analog signal processors, substantial circuit-level simulation must be performed to insure correct functionality. Although desirable, simulation of *complete* signal processors has not been attempted because of the computational cost. Ambitious circuits consist of arrays of cells where the array size can be as large as 256×256, and each cell may contain up to a few dozen devices [Wyatt 88]. Therefore, simulation of a complete signal processor requires solving a system of differential equations with *hundreds of thousands* of unknowns.

The structure of grid-based analog signal processors is such that they can be simulated quickly and accurately with specialized algorithms tuned to certain parallel computer architectures. In particular, the coupling between cells in the analog array is such that a block-iterative scheme can be used to solve the equations generated by an implicit time-discretization scheme, and furthermore, the regular structure of the problem implies that the simulation computations can be accelerated by a massively parallel SIMD computer, such as the Connection Machine ®[1][Hillis 85]. In the next section of this paper, we describe an example grid-based analog signal processor, and in Section 3 we describe the simulation algorithm. In Section 4, the mapping onto the Connection Machine is detailed. Experimental results are presented in Section 5 and the conclusion and suggestions for further work are contained in Section 6.

[1] Connection Machine is a registered trademark of Thinking Machines Corporation.

2 Problem Description

Consider the circuit in Figure 1, an idealized version of a grid-based analog signal processor used for two-dimensional image smoothing and segmentation [Lumsdaine 90]. The node equation for a grid point i,j in the network is

$$\begin{aligned} c\dot{v}_{i,j} &= g_f(v_{i,j} - u_{i,j}) \\ &\quad + g_s(v_{i,j} - v_{i+1,j}) + g_s(v_{i,j} - v_{i-1,j}) \\ &\quad + g_s(v_{i,j} - v_{i,j+1}) + g_s(v_{i,j} - v_{i,j-1}) \end{aligned} \tag{1}$$

where $u_{i,j}$ represents the image data at the grid point i,j, $v_{i,j}$ is the output voltage at node i,j, g_f is the input source impedance, c is the parasitic capacitance from the grid node to ground, and $g_s(\cdot)$ is a nonlinear "fused" resistor. In this circuit, the g_s resistors pass currents in such a way as to force $v_{i,j}$ to be a spatially smoothed version of $u_{i,j}$, unless the difference between neighboring $u_{i,j}$'s is very large. In that case, g_s no longer conducts, there is no smoothing, and the image is said to be "segmented" at that point.

In a more complete representation of the image smoothing and segmentation circuit, the voltage source $u_{i,j}$ and the source impedance g_f are replaced with a subcircuit which typically contains operational amplifiers and a phototransistor. If such a subcircuit has M internal nodes and contains only voltage-controlled elements, then it can be described by a differential equation system of the form

$$\frac{d}{dt} q_{sub}(\tilde{v}_{i,j}(t), v_{i,j}(t), t) = f_{sub}(\tilde{v}_{i,j}(t), v_{i,j}(t), t) \tag{2}$$

where $\tilde{v}_{i,j} \in \Re^M$ is the vector of the i,j^{th} subcircuit's internal node voltages, and $q_{sub}(\tilde{v}_{i,j}(t), v_{i,j}(t), t)$, $f_{sub}(\tilde{v}_{i,j}(t), v_{i,j}(t), t) \in \Re^m$ are the vectors of sums of charges and sums of resistive currents, respectively, at each of the subcircuit's internal nodes. Incorporating the subcircuit's behavior into the equation for grid point i,j leads to

$$\begin{aligned} c\dot{v}_{i,j} &= i_{sub}(v_{i,j}, \tilde{v}_{i,j}) \\ &\quad + g_s(v_{i,j} - v_{i+1,j}) + g_s(v_{i,j} - v_{i-1,j}) \\ &\quad + g_s(v_{i,j} - v_{i,j+1}) + g_s(v_{i,j} - v_{i,j-1}), \end{aligned} \tag{3}$$

where $i_{sub}(v_{i,j}, \tilde{v}_{i,j})$ is the current entering subcircuit i,j from grid node i,j.

For our purposes, an $N \times N$ grid-based analog signal processor, or analog array, is any circuit that can be described by a system of equations generated by replicating Eqn. (2)

and Eqn. (3) for each $i, j \in 1, \ldots, N$. Note that this definition enforces a regular structure, and only allows for coupling between neighboring subcircuits through two-terminal nonlinear resistors. The nonlinear resistors are usually implemented with transistors, so our definition of an analog signal processor still represents an idealization, although the extension to the general case is straightforward.

3 Numerical Algorithms

For notational simplicity, the system of equations that describe an $N \times N$ grid-based analog signal processor, defined in the previous section, will be written compactly, and perhaps not very informatively, as

$$\frac{d}{dt}q(v) = f(v(t)), \tag{4}$$

where $q(v(t)), f(v(t)) \in \Re^{N^2 \times (M+1)}$ are the vectors of sums of node charges and node resistive currents.

The transient simulation of the analog grid involves numerically solving (4). To compute the s .ution, it is possible to use simple explicit or semi-implicit numerical integration algorithms, but for these types of circuits, experiments show that an implicit method like the trapezoidal rule is substantially more efficient [Silveira 90]. The trapezoidal rule leads to the following algebraic problem at each time step h:

$$q(v(t+h)) - q(v(t)) + \frac{1}{2h}[f(v(t+h)) + f(v(t))] = 0. \tag{5}$$

As is standard, the algebraic problem is solved with Newton's method,

$$J_F(v^m(t+h))[v^{m+1}(t+h) - v^m(t+h)] = -F(v^m(t+h)) \tag{6}$$

where

$$F(v(t+h)) = [q(v(t+h)) - q(v(t))] + \frac{1}{2h}[f(v(t+h)) + f(v(t))] \tag{7}$$

and the Jacobian $J_F(v(t))$ is

$$J_F(v(t+h)) = \frac{\partial q(v(t+h))}{\partial v} + \frac{1}{2h}\frac{\partial f(v(t+h))}{\partial v} \tag{8}$$

In "classical" circuit simulators such as SPICE [Nagel 75], the linear system of equations for each Newton iteration is solved by some form of sparse Gaussian elimination. When simulating grid-based signal processors, where the coupling between subcircuits is restricted to nonlinear resistors, the Newton iteration equation will be such that its solution can be efficiently computed by iterative algorithms like conjugate-gradient squared (CGS) [Sonneveld 89, Burch 89]. To demonstrate this, in Table 1, we compare the CPU time required to compute the transient analysis of the network in Figure 1 using several different matrix solution algorithms to solve the Newton iteration equation. This problem is hard for an iterative method because, though not described here, the transient analysis is performing a continuation on the nonlinear resistor elements that changes the conditioning of the matrix with time (see [Lumsdaine 90] for details). As the table indicates, sparse Gaussian elimination is much slower than CGS[2] or ILU preconditioned CGS, both of which perform almost identically. This is a fortunate result, because our goal is to develop an efficient parallel simulator, and unpreconditioned CGS is easiest to parallelize.

[2]This problem is symmetric, so CGS and standard conjugate-gradient are equivalent

Size	Direct	CG	ILUCG
16×16	16.53	11.72	10.27
32×32	156.57	60.72	50.75
64×64	1856.12	272.30	224.12

Table 1: Comparisons of serial execution time for direct, CG, and ILUCG linear system solvers when used for the transient simulation of the circuit in Figure 1, where $g_f = 3.0e-5$ and g_s has a conductance of $1e-3$ when linearized about zero.

4 Connection Machine Implementation

The Connection Machine model CM-2 is a single-instruction multiple data (SIMD) parallel computer consisting of 65,536 bit-serial processors and 2048 Weitek floating-point processors. The bit-serial processors are clustered together into groups of 16 to make a single integrated circuit, and these IC's are connected together in a 12-dimensional hypercube. Two IC's, or 32 processors, share a single Weitek IC. Since the CM-2 contains 2048 Weitek IC's, a speedup of a factor of 2048 over conventional computers containing a single Weitek IC (e.g., a SUN-4) is conceivable.

For an algorithm to approach this peak parallel performance on the CM, it must satisfy three requirements. First, the problem must have enough parallelism to use all the available processors. Second, the algorithm can depend only on local or infrequent interprocessor communication, like on any parallel machine. And third, the algorithm must be mostly *data-parallel* because of the SIMD nature of the Connection Machine. By data-parallel we mean:

- One can identically map individual pieces of data to individual processors for all relevant processors and
- One can operate identically on the data with all the relevant processors

The general circuit simulation problem violates all three of the above constraints, and previous attempts at circuit simulation on the Connection Machine have not yielded impressive results [Webber 87, Silveira 90]. As we will show in the rest of this section, simulation of grid-based analog signal processors *is* well suited to the CM. These circuits are large, and can be simulated with algorithms that are mostly data-parallel and which depend on mostly nearest-neighbor communication between processors.

4.1 Data to Processor Mapping

The two-dimensional nature of grid-based analog signal processing circuits naturally maps into a two-dimensional geometry on the CM, in such a way as to maintain data parallelism and locality. The circuit is divided into identical cells (as shown in Figure 1) and each processor is assigned the data associated with each cell, with nearest-neighbor grid cells being mapped to nearest-neighbor processors. It is an important point that the assigned data includes the node voltages, currents, charges and derivatives, *but not* a complete description of the cell, only the CM's front-end computer has that.

It can be seen from Figure 1 that some elements in each cell cross the cell boundaries, and the communication so implied must be organized carefully to maintain maximum data-parallelism. In our approach, copies are made of *shared-nodes*,

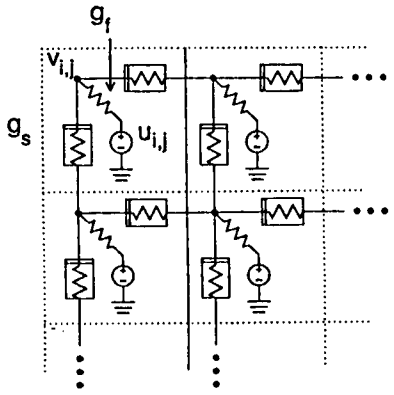

Figure 1: Grid of nonlinear resistors and its division into identical cells.

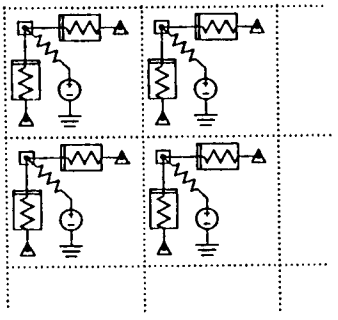

Figure 2: Separation of the cells by duplicating nodes. The shared nodes and pseudo-nodes are outlined with squares and triangles, respectively.

by which we mean nodes within each cell to which elements from other cells are connected. These copies are referred to as *pseudo-nodes*. As can be seen in Figure 2, using pseudo-nodes implies that the data for the cell devices is contained completely within the cell.

Two types of consistency between the shared-nodes and pseudo-nodes must be maintained through interprocessor communication, namely:

Voltage Consistency: Pseudo-Nodes must have the same voltage as their corresponding shared nodes.

Charge and Current Consistency: Charges and currents flowing into the pseudo-nodes are summed at the corresponding shared nodes.

This particular mapping of the circuit data insures that the many cells in a large grid can be simulated in a data-parallel fashion. That is, simulation of the entire grid is accomplished by simulating a simple cell using many copies of data, and then enforcing the voltage, charge, and current consistencies for the shared nodes and pseudo-nodes. Note that the cells on the east and south circuit boundaries respectively do not have east and south connecting elements. In order to model this properly on the CM, boundary processors are turned off whenever data corresponding to non-existent elements is manipulated. For reasons of clarity we will omit further discussion of this operation; it is performed in a straightforward manner for all the algorithms to be presented.

4.2 Device Evaluation

Evaluating the right-hand side and the Jacobian for the Newton iteration, equation (7), involves computing sums of device currents and charges. Given the previous discussion of the data to processor mapping, the device evaluation portion of the simulation is obvious:

1. Copy node voltages from shared nodes to pseudo-nodes (voltage consistency step)
2. Evaluate cell devices in parallel
3. Sum node charges and currents from pseudo-nodes to shared nodes (charge and current consistency step)

4.3 Linear System Solution

As mentioned in the previous section, for the case of grid-based analog circuits, solving the linear Newton iteration equation (6) using CGS is not only easy to parallelize, it is faster than using sparse Gaussian elimination, and nearly as fast as using ILUCGS. There are two parts of the CGS iteration which involve parallel data: the vector inner product and the matrix-vector product. The vector inner-product is accomplished with an in-place multiply and a global sum. The matrix-vector product $y = Ax$ is accomplished with the following sequence of operations:

1. Copy x values from shared nodes to pseudo-nodes (voltage consistency step)
2. Perform matrix-vector product with simple-cell matrix
3. Sum y values from pseudo-nodes to shared nodes (current consistency step)

That the operations involved in the matrix vector product are similar to those required for the device evaluation should come as no surprise. The communication steps are still required for consistency, and the device evaluation step is now replaced by an in-place matrix-vector product where the local matrix corresponds to the linearized conductance matrix of the simple cell circuit.

5 Experimental Results

In order to test our algorithms, a simulation program was written for the CM, using MIT's SIMLAB program [Simlab] as a base. The parallel portions of the code were written in C* Version 6.0, a CM superset of C. The front-end experiments were run on a conventional SUN-4 workstation and the CM results were obtained on a 16K CM-2 with double-precision floating point hardware. All computations were performed in double precision arithmetic.

In Table 2, the CPU times required on the CM and the SUN-4/280 to perform the DC and transient analysis of the nonlinear resistive grid of Figure 1 are compared. Only a 16k processor CM was available, but using the "virtual processor" feature of the CM, the 256 × 256 example was simulated as if the CM had 64K processors. A real 64K machine would have run the 256×256 example approximately four times faster, and have produced simulation results approximately 1400 times faster than a SUN-4/280 workstation.

To investigate how well simulation of more realistic circuits can be accelerated, the CM simulator was tested on an idealization of the Retina chip [Mead 88]. To generate the Retina-like circuit, the voltage source $u_{i,j}$ and the source impedance

Size	DC		Trans	
	Serial	CM	Serial	CM
64×64	147.47	3.96	268.47	6.28
128×128	632.80	3.99	7581.70	44.99
256×256	(2710.2)	10.84	(214110.4)	610.27

Table 2: Comparisons of serial and CM execution times (using CGS). Extrapolated serial results are contained in parentheses.

g_f for the circuit in Figure 1 are replaced with the Retina chip's follower-connected transconductance amplifier subcircuit. This subcircuit has eight internal nodes and consists of twelve MOS devices (SPICE MOS level 3 is used in SIMLAB). The coupling resistors, represented by g_s in Figure 1, were set a small value, $10K\Omega$, and a large value, $10M\Omega$, to test the simulator. The run times on the CM and the SUN-4 for simulating these two examples are given in Table 3. For the largest example shown, the best CM algorithm is over 285 times faster than the best serial algorithm.

There are two columns of CM results in Table 3, corresponding to two types of CGS algorithms. In the first column, the times using an unpreconditioned CGS algorithm are given, and in the second column, the times using a block-preconditioned CGS (CGSB) algorithm is given. The block preconditioner used here is easy to compute in parallel, it only involves factoring each of the subcircuit Jacobians and corresponds to solving that piece of the system described by equation (2) directly. As can be seen from the table, this preconditioner accelerates the CGS convergence enough to improve the CM run times by as much as a factor of four.

Circuit		Serial		CM	
g_s	Size	Direct	CGSIL	CGS	CGSB
1e4	4×4	26.00	28.92	201.03	139.37
	8×8	106.22	121.25	452.95	143.91
	16×16	435.38	496.70	604.89	143.19
	32×32	1991.33	2156.97	555.22	144.31
	64×64	(9107.9)	(9366.8)	623.33	144.43
	128×128	(41657.4)	(40676.5)	589.96	142.69
1e7	4×4	26.73	28.17	140.64	121.74
	8×8	106.03	113.98	142.28	121.81
	16×16	435.37	461.92	141.98	122.22
	32×32	2043.18	1954.42	150.86	150.73
	64×64	(9588.6)	(8269.3)	141.69	127.89
	128×128	(44999.6)	(34988.1)	144.51	122.37

Table 3: Comparisons of serial and CM execution times. Extrapolated serial results are contained in parentheses.

6 Conclusion

In this paper, we presented algorithms for simulating very large grid-based analog VLSI circuits on a massively parallel processor. By restricting our attention to this class of circuit problems, we were able to more fully exploit the capabilities of the Connection Machine, as demonstrated by the experimental results. It is perhaps remarkable to note that with our implementation on the CM, we were able to simulate a realistic vision circuit with almost 200,000 devices and 150,000 nodes in less than 10 minutes.

Our future work will be to extend the simulator to allow more general cell interconnection and to investigate whether further speed improvements can be obtained through the use of nonlinear Krylov subspace methods — the nonlinear analogue of CG.

Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency under Contract No. N00014-87-K-825 and by the Portuguese INVOTAN committee. The authors are grateful to Thinking Machines Corporation, especially Rolf Fiebrich, for providing hardware and software support for the development of the simulator. The authors would also like to thank Prof. John Wyatt and the MIT Vision Chip Project group for providing the motivation for this work.

References

[Burch 89] R. Burch, K. Mayaram, J.-H. Chern, P. Yang, P. Cox, "PGS and PLUCGS - Two New Matrix Solution Techniques for General Circuit Simulation," *Proc. ICCAD-89*, pp. 408 - 411, Nov. 1989.

[Hillis 85] W.D. Hillis, *The Connection Machine*, MIT Press, Cambridge, MA, 1985.

[Lumsdaine 90] A. Lumsdaine, J. Wyatt, and I. Elfadel, "Nonlinear Analog Networks for Image Smoothing and Segmentation," *Proceedings of the International Symposium on Circuits and Systems*, pp. 987 - 991, May, 1990.

[Mead 88] C. A. Mead, *Analog VLSI and Neural Systems*, Addison-Wesley, Reading, MA, 1988.

[Nagel 75] L. W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," Electronics Research Lab Report, ERL M520, Univ. of Calif., Berkeley, May 1975.

[Silveira 90] L. M. Silveira, "Circuit Simulation Algorithms for Massively Parallel Processors," S. M. Thesis, MIT, May 1990.

[Simlab] A. Lumsdaine, M. Silveira, and J. White, "Simlab Programmer's Guide," To be published as an MIT memo.

[Sonneveld 89] P. Sonneveld, "CGS, A Fast Lanczos-type Solver for Nonsymmetric Linear Systems," SIAM J. Sci. Stat. Comp., 10(1989), pp. 36-52.

[Tong 88] C. Tong, "The Preconditioned Conjugate Gradient on the Connection Machine," *Proceedings of the Conference on Scientific Applications on the CM*, Horst D. Simon, ed., pp. 188-213, World Scientific, Singapore, 1988.

[Webber 87] D. M. Webber, A. Sangiovanni-Vincentelli, "Circuit Simulation on the Connection Machine," *24th ACM/IEEE Design Automation Conf.*, pp. 108-113, June 1987.

[Wyatt 88] J. Wyatt, et al, *Smart Vision Sensors: Analog VLSI Systems for Integrated Image Acquisition and Early Vision Processing*, Proposal, MIT, 1988.