Optimization of Parallel Multilevel-Newton Algorithms on Workstation Clusters

Robert Gräb[1], Michael Günther[1] *, Utz Wever[2] and Qinghua Zheng[2]

[1] Technische Hochschule Darmstadt, Fachbereich Mathematik,
Schloßgartenstr. 7, D-64289 Darmstadt, Germany
[2] Siemens AG, Corporate Research and Development,
D-81730 München, Germany

Abstract. The design of dynamic memories and digital circuits demands the numerical simulation of networks with more than 10 000 transistors. Only a parallel implementation of industrial simulation packages allows short design cycles required today. Multilevel-Newton algorithms (MLN) based on circuit partitioning techniques have proved to be very efficient in parallel circuit simulation on workstation clusters. In industrial environments, the performance of these algorithms depends essentially on an optimal partitioning and adaption to actually achieved transmission performance of the network. Recent results with the circuit simulator TITAN at Siemens' research laboratories are discussed.

1 Introduction

Circuit simulation programs are standard tools for computer aided design of electric circuits. At Siemens AG a major application is the verification of dynamic memories and digital circuits with more than 10 000 transistors. Due to the desired accuracy and reliability, classical circuit simulation algorithms such as solutions of nonlinear equations, numerical integration and exploition of latency are used. They are implemented in the circuit simulator TITAN (SPICE [5] compatible input language), which was developed at Siemens' research laboratories, see [1].

In the past supercomputers have been used for simulating circuits. Nowadays the floating point performance of modern workstations approaches those of small vector computers. Thus it makes sense to utilize the overall power of a workstation cluster by running additional parallel software, see [4, 7]. But as the network performance is very low, only those algorithms guaranteeing a minimum of communication overhead can reach good speedup results. Domain decomposition methods split up the problem in the physical domain. Each domain is calculated by a separate process and can thus run in parallel. The boundaries of the domains are equalized by a serial master process. In [7, 8], an improvement of multilevel Newton methods was developed, which takes into account the low communication of workstation clusters. A further decrease of communication by

* Corresponding author. E-Mail: guenther@mathematik.th-darmstadt.de

introducing Quasi-Newton techniques was considered in [8]. In this paper we aim at an additional optimization with respect to parallel computation costs.

In section 2, multilevel Newton methods for circuit simulation are introduced. They exploit iteration latency and can be optimized in three directions: in case of low network performance, alternative tangential corrections can be used, see section 3. The optimal partitioning of an electric circuit is discussed in section 4. Additionally in section 5, the number of inner iterations is adapted to the effective transfer rate of data.

2 Parallel Multilevel-Newton Algorithms

By applying the scheme of Modified Nodal Analysis to a given circuit, a system of N nonlinear differential algebraic equations is setup:

$$f(\dot{q}(x(t)), x(t), t) = \begin{pmatrix} f_1(x(t), t) + \dot{q}(x(t)) \\ f_2(x(t), t) \end{pmatrix} = 0.$$

f_1 and f_2 describe the static part of the network equations, q the charges and magnetic fluxes and $x \in \mathbb{R}^N$ the node potentials and the branch currents through voltage sources and inductors. An implicit time discretization scheme such as the Backward Differentiation Formula yields to nonlinear equations, see [1]:

$$F(x) = 0. \tag{1}$$

A standard method to handle this equation is Newton's method, where the linear system is solved by Gauss' algorithm for sparse matrices. By applying domain decomposition methods, equation (1) can be decoupled into the nonlinear system

$$F_i(x_i, x_{m+1}) = 0 \quad \text{for} \quad i = 1, \ldots, m, \tag{2}$$

$$G(x_1, \ldots, x_m, x_{m+1}) = 0 \tag{3}$$

with $F_i \in \mathbb{R}^{N_i}$, $x_i \in \mathbb{R}^{N_i}$, $x_{m+1} \in \mathbb{R}^{N_{m+1}}$ and $G \in \mathbb{R}^{N_{m+1}}$. F_i describes the i-th sub-circuit, depending only on the inner variables x_i and coupling quantities x_{m+1}. The subsystems are coupled by the system G of generally small dimension. An efficient multilevel Newton algorithm for solving (2,3) reads, see [8]:

Multilevel-Newton algorithm — MLN(n)

start: $(x_1^0, \ldots, x_m^0, x_{m+1}^0)$; *latency* = false
do $l = 0, 1, 2, 3, \ldots$
  do parallel $j = 1, \ldots, m$
    $x_j^{l,0} = x_j^l$
    do $i = 0, 1, 2, \ldots, n-1$
      $\frac{\partial F_j}{\partial x_j} \Delta x_j^i + F_j(x_j^{l,i}, x_{m+1}^l) = 0$; $x_j^{l,i+1} = x_j^{l,i} + \Delta x_j^i$
    end do$_i$
    if (*latency* = false) **then** $S_j^l = \frac{\partial G}{\partial x_j} (\frac{\partial F_j}{\partial x_j})^{-1} \frac{\partial F_j}{\partial x_{m+1}}$
  end do$_j$
  if (*latency* = false) **then** $S^l = \frac{\partial G}{\partial x_{m+1}} - \sum_{j=1}^m S_j^l = L^l U^l$

else $(S^l, L^l, U^l) = (S^{l-1}, L^{l-1}, U^{l-1})$
$L^l U^l \Delta x^l_{m+1} + G(x^{l,n}_1, \ldots, x^{l,n}_m, x^l_{m+1}) = 0$
do parallel $j = 1, \ldots, m$
$x^{l+1}_j = x^{l,n}_j - (\frac{\partial F_j}{\partial x_j})^{-1} \frac{\partial F_j}{\partial x_{m+1}} \Delta x^l_{m+1}$ (*)
end do$_j$
$x^{l+1}_{m+1} = x^l_{m+1} + \Delta x^l_{m+1}$
if ($\|G(\ldots, x^{l+1}_{m+1})\| * const \leq \|G(\ldots, x^l_{m+1})\|$) **then** *latency* = true
else *latency* = false
end do$_l$

To guarantee quadratic convergence, the inner nodes are tangentially corrected [3]. The nonlinearity is shifted to the local systems to reduce the number of outer iterations (index l). This decrease of outer iterations is paid by n additional inner iterations (index i), which are performed in parallel. In section 5 strategies are discussed that yield optimal n depending on the actual network performance. To reduce communication and computation costs for the master process, the outer Newton step is replaced by a Quasi-Newton algorithm, which is controlled by the norm decrease of the correction. If there is a sufficient decrease, the slave processes do not need to evaluate the outer derivatives S^l_j, $j = 1, \ldots, m$. Sending these matrices to the master makes up the main part of the communication. Therefore these costs are reduced drastically in case of latency. The computational bottleneck of the master process remains the solution of the interconnect system $S^l \Delta x^l_{m+1} = -G(\ldots)$. In case of latency, the Jacobian S^l of the last iteration is used. Thus, also the old LU-decomposition can be used and only a forward-backward substitution has to be performed.

The algorithm is implemented in TITAN using FORTRAN and C. The communication is performed by the public domain PVM software package, see [2]. First results are discussed in [7, 8]. In this paper we will concentrate on the HP workstation cluster at the TITAN research laboratory, consisting of one HP 9000/755 and three HP 9000/725.

Distributed simulation suffers from the fact that both the floating point performance and the effective network bandwidth may suddenly change in a multiuser environment. To consider these effects, we introduce an alternative tangential correction and optimize the number of sub-circuits and inner Newton iterations.

3 An alternative tangential correction

An alternative to the common tangential correction is to replace (*) in the multilevel-Newton algorithm described in the previous section by

$$x^{l+1}_j = x^{l,n}_j - (\frac{\partial F_j}{\partial x_j})^{-1} F_j(x^{l,n}_j, x^{l+1}_{m+1})$$

with slightly better convergence properties. However, this alternative correction implies $l(n)$ additional evaluations of F_j for $j = 1, \ldots, m$ in parallel. Hence the

performance of the alternative approach depends on the number $l(n)$ for each n. In table 1, the average number of outer Newton iterations $l(n)$ for each time step as well as the total simulation time is given for the PATH circuit, using pure Quasi-Newton outer iterations. The PATH circuit, part of a critical path in a 16 MBit DRAM memory, consists of 13 015 transistors and 32 639 nodes. One notes a large drop of $l(n)$ from $n = 1$ to $n = 2$; for larger values of n, only a slight decrease is observed. The simulation times show that the alternative approach is preferable only, if many inner Newton iterations have to be performed, e.g. if communication costs are very high.

method	n=1	n=2	n=3	n=4	n=9
tangential correction	6.4 / 47:08	3.4 / 46:36	2.9 / 50:28	2.8 / 54:23	2.7 / 60:42
alternative correction	4.8 / 57:15	3.3 / 51:04	2.9 / 52:06	2.8 / 55:09	2.7 / 58:38

Table 1. Average number of outer iterations $l(n)$ per time step and total simulation time in min:sec for PATH circuit with $m = 4$ slaves

4 Optimal number of sub-circuits

Until now, there are no tools available for an optimal partition of an electric circuit. Dynamic load balancing is not practicable, since a change of the partitioning during simulation is too expensive in TITAN. Hence only a static load balancing is possible, i.e. an optimal distribution of the sub-circuits on the workstations. To implement such a load balancing, the computation and communication costs for each sub-circuit have to be estimated.

As a first step, we derived a model for the simulation time. The total cost T^{total} for the parallel multilevel-Newton algorithms consists of four main parts:

$$T^{\text{total}} = l(n) \cdot \left\{ \max_{j=1,\ldots,m} T(j) + \sum_{j=1}^{m} T^{\text{comm}}(j) + T^{\text{master}} \right\}. \tag{4}$$

Here $l(n)$ is the number of outer Newton iterations, $T(j)$ the computation and communication time for the j-th subproblem, T^{master} the computation time for the interconnect system in the master process, and $T^{\text{comm}}(j)$ the communication overhead that cannot be performed in parallel.

If the floating point performance of each workstation, as well as the transmission performance of the network are known, parameters can be obtained experimentally for each partial task. This yields a rough formula for the total cost which depends on the dimension of the subproblems and the interconnect systems. Hence the simulation time can be estimated a priori for different circuit partitions.

As a test example we have derived such an estimate for the PATH circuit. The results for a multilevel-Newton method with two inner Newton iterations are given in table 2 for the HP cluster. The values for $m = 4$ with Ethernet communication performance are the experimental basis, the other values are estimated: a slower network with performance factor 0.1, and a faster one with factor 10.

In any case, $m = 8$ slaves are optimal. The bad values for $m = 9, 10$ correspond to disadvantageous partitions with large interconnect systems. One notes that faster networks improve the results only slightly. However, a lower communication performance drastically deteriorates the simulation times. Hence an extension to the algorithm is necessary, which reacts to dynamic changes in the transmission performance of the network.

	m							
Network performance	3	4	5	6	7	8	9	10
0.1	6521	5766	5363	4815	4820	4299	6523	5837
1	3857	3050	2578	2091	2010	1632	3339	2893
10	3590	2778	2299	1819	1729	1365	3021	2599

Table 2. Estimated simulation time in sec for different circuit decompositions

5 Optimal number of inner Newton iterations

Since the workstation cluster is connected by Ethernet, the effective transfer rate of data may change permanently. Hence the optimal number n of inner Newton iterations depends on the actual status of the network.

Formula (4) teaches us that a larger n may be advantageous in the case of high communication costs, since more inner yield less outer iterations. However, the table 1 shows that only $n \leq 3$ inner iterations should be used. This suggests the following optimization of n at every multilevel-Newton step:

n_{opt}-multilevel-Newton algorithm

1. *Estimate $l(n)$.* For the first iteration in the actual time step, the estimate is based on information of the preceding time step. Otherwise, the estimate is based on experimental data and an error inequality derived from convergence analysis.
2. *Determine communication and computation costs.* In principle, communication and computation times can be determined as described in section 4. However, parameters of the time model must be adapted permanently to changing network and workstation performance. To avoid this expenditure, we just measure the corresponding times by using `gettimeofday` and send them to the master together with the remaining data.

3. *Estimate total cost* T^{total} *for* $n = 1, 2, 3$. For the next iteration step, we choose the n with the smallest time T^{total}.

The n_{opt}-algorithm was tested at different times of the day, and showed a robust behavior with respect to changing conditions. In table 3, the multilevel-Newton algorithm with $n = 1$ (MLN(1)) is compared with the n_{opt}-algorithm MLN(n_{opt}) under different network conditions.

fast network		slow network	
MLN(1)	MLN(n_{opt})	MLN(1)	MLN(n_{opt})
51:08	50:12	1:28:26	1:14:48

Table 3. MLN(1) compared with MLN(n_{opt}) for $m = 4$ slaves (PATH circuit)

Conclusion

Distributed circuit simulation on workstation clusters using multilevel-Newton methods is an efficient and comparatively cheap tool in industrial VLSI design. However, in a multiuser system one has to consider changes in both the floating point and network performance. The demonstrated results show that these methods can be optimized to react efficient and robust to changing conditions.

References

1. Feldmann, U., Wever, U., Zheng, Q., Schultz, R., Wriedt, H.: "Algorithms for Modern Circuit Simulation." *AEÜ Vol. 46, No. 4*, pp 274–285 (1992)
2. Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., Sunderam, V.: "PVM 3.0 User's Guide and Reference Manual." *ORNL*, Tennessee (1993)
3. Hoyer W., Schmidt J.W.: "Newton-type decomposition methods for equations arising in network analysis." *ZAMM Vol. 64*, pp 397–405 (1984)
4. Kleis, U., Wallat, O., Wever, U., Zheng, Q.: "Domain Decomposition Methods for Circuit Simulation." *Proceedings of the 8th Workshop on Parallel and Distributed Simulation.* Edinburgh (1994)
5. Nagel, L.W.: Spice2: "A Computer Program to Simulate Semiconductor Circuits." *Mem. No. ERl-M520,* University of California, Berkeley (1975)
6. Rabbat, N. B. G., Sangiovanni-Vincentelli, A. L., Hsieh, H. Y.: "A multilevel Newton algorithm with macromodeling and latency for the analysis of large-scale nonlinear circuits in the time domain." *IEEE Trans. CAS 26,* pp 733–741 (1979)
7. Wever, U., Zheng, Q.: "Parallel circuit simulation on workstation cluster." *8th Conference of the European Consortium for Mathematics in Industry (ECMI'94),* Kaiserslautern (1994)
8. Wever, U., Zheng, Q.: "Parallel Transient Analysis for Circuit Simulation". *Proceedings of the 29 Annual Hawaii International Conference on System Sciences, Vol. 1,* pp 442–447 (1996)