REDUCING COMMUNICATION OVERHEAD IN DISTRIBUTED LOGIC SIMULATION OF VLSI CIRCUITS

Amar Guettaf, Pirouz Bazargan-Sabet

Université Pierre et Marie Curie (Paris VI), Laboratoire LIP6/ASIM
Tour 55-65, 2ème étage - 4, place Jussieu
75252 Paris Cedex 05, France

Keywords: partitioning, node replication, discrete event, distributed simulation, VLSI

ABSTRACT

Distributed simulation represents an attractive and smart way of improving the verification speed of large VLSI circuits. Unfortunately, this inexpensive approach suffers from the low performance of the communication networks used to connect local workstations. In this paper, we present a partitioning algorithm that attempt to find a suitable balance between the communication and the execution load in a distributed simulator to enhance its speedup. The main features of this method are the use of logic replication to reduce the communication overhead and a realistic cost function that takes into account the activity of signals. Signals' activity can be obtained through a probabilistic evaluation. A distributed simulator implementing a conservative synchronization method has been used to measure the efficiency of this algorithm.

1 INTRODUCTION

Today, the verification of a complex VLSI may represent the major part of the development time of the circuit. In this process, the logic simulation is certainly the tool the most often invoked. Simulation is called whenever it is necessary to check that, at a certain level, a description continues to fit within the initial specifications. According to designers, near 50% of the development time is spent in logic simulation. Therefore, improving the simulation speed may results in a significant profit regarding the time-to-market constraint.

Distributed simulation seems to be an appropriated answer to make the verification of such high complexity circuits be faster. Several processors are used to perform the simulation of a single system. In this approach, which exceeds the technical bounds of a single machine, a great amount of memory, disk space and other computational resources are provided to enhance the simulation.

The efficiency of a distributed simulation can be measured using a **speedup** value. The speedup is the ratio of the time spent in achieving some process using a standard program by the time required by an enhanced program. In distributed simulation, the speedup depends on 4 main parameters: (1) the intrinsic parallelism of the system, (2) the number of systems involved in the simulation process, (3) the simulator's synchronization strategy and (4) the efficiency of the partitioning method.

Several works have already been driven in the field of parallel logic simulation (a detailed overview is given in Meister 1993 and in Bailey *et al*. 1994) but no general method has been found which gives reasonable speedup under general conditions. All previous works deal with specific techniques that are promising for some synchronizing strategies and disappointing for others (Bailey 1992; Matsumoto and Taki 1992; Soulé 1992).

The main method used in logic simulation is the event-driven trace-selective approach where events are associated with a transition of a signal. In this method, a signal is re-evaluated each time an event occurs on, at least, one of its inputs. In addition to the simulation method, in a distributed simulator, different sub-simulators have to be synchronized among each other to maintain the global coherence of the system. Most of the researches carried out in this domain concern this last issue.

The main synchronization strategies for distributed event-driven simulation are the conservative (Misra 1986) and the optimistic approaches (Jefferson and Sowizral 1984). Briefly, the first one avoids out-of-order execution of events by waiting for messages that maintain the global simulator in a coherent state. The second class risks sequencing errors by processing events as soon as possible. In case of erroneous computation, the state of the simulated system is rolled back to a former state to correct the order of events. This requires that the state of sub-simulators being periodically saved. There exists a variety of combinations of these basic strategies.

Partitioning is used to split a complex design into multiple smaller blocks. Making the partition, some important topics have to be considered: respect of a certain balance between the different blocks in terms of execution and reduction of the communication through the network.

Finding an optimal partition for a given circuit is a tricky task due to the NP-completeness of the problem (Johnson and Garey 1979). Thus, heuristics must be applied. A large number of algorithms has been proposed. Four different classes can be pointed out: move-based methods, combinatorial formulations, geometric representations and clustering approaches (Alpert and Khang 1995). In VLSI tools, the move-based algorithms are the most commonly used.

In this paper, we present a partitioning algorithm that belongs to the move-based approaches' class. We propose an improvement of the basic Fiduccia and Mattheyses (Mattheyses and Fiduccia 1982) algorithm by calling logic replication to exchange communication load against execution. Also, the concept of signal activity has been introduced to build a more realistic cost function for the event-driven simulation method.

The next section gives a brief description of the original algorithm proposed by Fiduccia and Mattheyses. Section 3 presents the logic replication method. The concept of signal activity and the cost function is depicted in section 4. Sections 5 and 6 describe the distributed simulator used to measure the efficiency of the proposed method as well as some results on a set of benchmark circuits. In section 7, a conclusion is given and future works are put in perspective.

2 MOVE BASED PARTITIONING METHODS

Logic partitioning is a common task in digital circuit design. It helps designers to apply the divide-and-conquer technique to decrease the complexity of problems. Many of the researches, in this domain, address the problem of partitioning for applications other than logic simulation: place & root, testability, system prototyping, etc.

Despite the diversity of the domains where the partitioning is applied, it is possible to give a unique formulation for this problem. A VLSI circuit may be represented by a hyper-graph $H(V,E)$. $V=\{v_1, v_2, \ldots, v_n\}$ is a set of n vertices representing the internal nodes (containing a Boolean function) and $E=\{e_1, e_2, \ldots, e_m\}$ a set of m hyper-edges representing the internal signals that connect the nodes. The partitioning problem consists in defining k subsets - or blocs - $V_1, V_2, \ldots, V_k$ of V such as:

$$\bigcup_{i=1}^{k} V_i = V \text{ and } \bigcup_{\substack{i=1 \\ j\neq i}}^{k} (V_i \cap V_j) = \varnothing$$

The aim is to reach an optimized partition for some specific application. As it is unreasonable to run the application for each intermediate solution, a cost function is defined to give a simplified model of the target application. The cost function is supposed to have a behavior as close as possible to the target application. The cost function traditionally used to measure the quality of a partition is the **cutsize**. It represents the number of signals that cross the partition.

The move-based algorithms belong to the class of iterative improvement techniques. These algorithms start with an initial partition. Then, local changes are applied to get into a better partition. The efficiency of the solution is then measured and the procedure of local changes is repeated as long as an improvement is possible.

One of the most known algorithms, in this domain, has been proposed by Kernigan and Lin (Lin and Kernigan 1970): a graph bisection

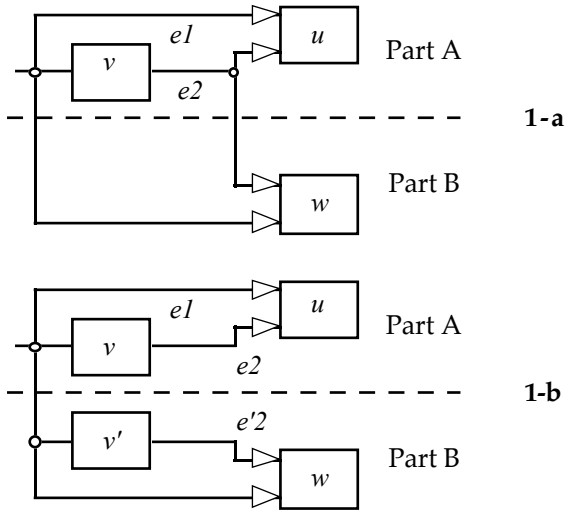

Figure 1: Node replication to improve the cutsize

technique. It starts with a random initial partition and uses pairwise swapping of vertices between partitions as local change.

An improvement of the K-L algorithm which reduces the time complexity to $O(n)$ (where n is the number of vertices in the graph) was proposed by Fiduccia and Mattheyses (Mattheyses and Fiduccia 1982). In this algorithm, only one vertex is moved in a single move that allows the handling of unbalanced partitions. The concept of cutsize is extended to hyper-graphs and a well suited data structure is defined to make a fast selection of the vertices to be moved. As in K-L algorithm, a vertex is locked when it is tentatively moved. When no further move is possible, only those vertices that result in the best cutsize are effectively moved.

3 REPLICATION PARTITIONING

A vertex replication technique has been proposed by Kring and Newton (Newton and Kring 1991) for circuit partitioning applied to mapping on FPGAs. It can substantially reduce the cutsize by allowing some vertices to be replicated in two or more partitions. Figure 1-a shows the partition of a circuit without vertex replication. However, when the node v is replicated, as in Figure 1-b, the cutsize is reduced. When a node is replicated, it is present in both sub-circuits and its outputs do not contribute to the cutsize.

Once a node has been replicated, it tends to remain so and nets connected to it remain in both sub-circuits. This may reduce the possibility of further improvements of the partition. For this reason, the number of replicated nodes must be limited to achieve an optimal partition. Table 1 shows the results obtained on a set of sequential circuits from ISCAS89 where the number of replicated nodes has been limited to 5%. Net cutsize reduction is the cutsize obtained from the application of node replication algorithm compared to the cutsize resulted from the classical F-M partition. The node replication is the percentage of the replicated nodes to the total number of nodes. The table shows clearly that node replication can significantly reduce the cutsize without increasing the size of the initial circuit.

circuit	cutsize by FM	cutsize by KN	cutsize reduction	node replication
s5378	94	59	37.2%	4.3%
s9234	56	41	26.8%	4.4%
s13207	75	49	34.7%	4.3%
s15850	87	48	44.8%	4.2%
s35932	128	97	24.2%	3.1%
s38584	81	49	39.5%	3.3%
s38417	144	38	73.6%	4.3%

Table 1: Improvement of the cutsize by K-N

4 IMPROVEMENT OF THE BASIC ALGORITHM

As mentioned in the above sections, the original K-L algorithm as well as the F-M and the K-N methods define a cost function based on the number of signals that cross the partition. However, in a distributed simulation, the cutsize does not give a realistic representation of the simulation method. In particular, the execution load is not taken into account. Moreover, in K-N, to reduce the cutsize through the node replication, a maximum number of replicated nodes must be settled. Therefore, an inappropriate replication rate may lead to a sub-optimal simulation speed.

4.1 The cost function

In the node replication method, finding out the optimal value for replication rate may be hazardous. Moreover, replicated nodes increase the

execution load of the simulation. In this method, the cost function has been defined to minimize the communication load and would be:

$$\sum_i \|c_i\|$$

Where c_i is a signal crossing the cut and $\|c_i\|$ the number of blocs that read the signal c_i.

Actually, the key point underlying the replication method, is to obtain a smaller communication load by increasing the execution load. Knowing that sending a message over the network requires much more time than an execution, we propose a function that attempt to reduce the global load of a simulation: the execution and the communication load. For a conservative distributed simulation, the following cost function seems to be convenient:

$$\max_k\left(\sum_i F_{out}(x_i)\right) + R_{ce}\sum_j \|c_j\|$$

$F_{out}(n)$ is the fanout of a node n. x_i are the internal nodes of a bloc and c_j are the hyper-edges that cross the cut. R_{ce} is the ratio of the time required for one message to be transmitted from one bloc to another by the time needed for the execution of one node. To ensure that the partition is well balanced in terms of events' execution, our partitioning algorithm has to fit into the following constraint :

$$Min\_Bound \le \left(E_k = \sum_{x_i \in V_k} F_{out}(x_i)\right) \le Max\_Bound$$

The proposed algorithm is based on the K-N partitioning algorithm but uses a more realistic cost function. However, in the above function and constraint all signals have the same weight. In fact, all the signals do not generate the same number of messages on the interface of the partitions. Communications due to "silent" signals are not as important as the messages generated by "active" signals. Yet, a more appropriate cost function, that exploits the concept of signal's activity, is conceivable.

4.2 Signals' activity calculation

These recent years, a great interest has been given to the concept of signal's activity, in VLSI design. The most important application of this concept is certainly in the evaluation of the power consumption. In a CMOS digital circuit, the most part of the power consumption is due to the transitions of signals. Therefore, knowing the mean number of transitions of signals per second - or the transition density - it is possible the estimate of power consumption.

Number of researches have been initiated in this domain. Basically, there are two classes of methods to evaluate the transition density. Statistical approaches use logic simulation and count the number of transitions during the simulation of a long sequence of patterns. Probabilistic methods try to calculate the transition density of signals through a single-pass symbolic simulation, given the probability and the transition density of the circuit's inputs.

The main problem, in the second class is the correlation between signals which can impair the probability calculation. We have developed a method based on symbolic simulation to detect potential sources of correlation. For each node, statistically independent inputs are identified and the Boolean expression of the node is expanded to these signals. Then, the transition density of signals is calculated in a second pass. The method of calculation is detailed in Dunoyer *et al*. 1995 and 1996. Briefly, the calculation uses the concept of Boolean difference and computes the transition density of a signal from the transition density and the probability of its inputs (Najm 1991):

$$D(s) = \sum_j D(e_j) P\left(\frac{\partial s}{\partial e_j}\right)$$

Where s is the output signal of the node, $D(s)$ is the transition density of the signal s, e_j is an input of s, $\frac{\partial s}{\partial e_j}$ is the Boolean difference of s in regard of e_j and $P(F)$ the probability of the Boolean function F to take the value one.

However, in the partitioning problem, unlike the power evaluation tools, the activity of a signal is not directly related to its transition density. In a distributed event-driven simulator, the value of a signal is re-evaluated whenever, at least, one of its

direct inputs has received an event or a transition. Thus, we propose a proper definition of the concept of **activity** of a signal, well suited to the partitioning problem. For a node i the activity is defined as:

$$A_i = \sum_j D(e_{ij})$$

Where e_{ij} is a direct input of the node i and $D(x)$ is the transition density of a signal x.

Using the concept of signal's activity the proposed cost function becomes:

$$\underset{k}{Max}\Big(\sum_i A(x_i)\Big) + R_{ce}\sum_j A(c_j)\cdot\|c_j\|$$

and the constraint:

$$Min\_Bound \le \Big(E_k = \sum_{x_i \in V_k} A(x_i)\Big) \le Max\_Bound$$

5 SIMULATION ENVIRONMENT

A distributed simulation tool has been developed to measure the efficiency of the proposed algorithm. It has been built upon an already existing set of CAD tools called Alliance (Greiner and Pêcheux 1993) and developed at University of Paris VI. The distributed simulator is derived from the Alliance's sequential event-driven simulation tool which supports a subset of VHDL. This simulator has been used as a reference to check the correctness of the results and to compare the performance of the prototype distributed simulator.

circuit	# of nodes	S.up by F-M	% of replic. nodes	S.up proposed approach	gain in S.up
s5378	3 137	0.14	13.4	0.16	21%
s9234	6 019	0.17	12.1	0.20	19%
s13207	9 227	0.37	8.8	0.49	33%
s15850	10 840	0.38	7.2	0.45	20%
s35932	19 841	1.22	6.5	1.48	22%
s38584	22 105	1.31	4.6	1.63	25%
s38417	25 451	1.41	4.7	1.73	23%

Table 2: Improvement of the speedup

The distributed simulator comprises two main parts: a central task and several sub-simulators. The central task reads the hierarchical description of the circuit, produced by the partitioning tool. Each bloc is distributed to one simulation task and the communications between sub-simulators are defined. The simulation process can start once the test pattern file is loaded. The simulator implements a conservative synchronization strategy. Using this technique, deadlocks may appear because of cyclic dependencies. This is solved by sending Null messages as proposed by Chandy and Misra (Chandy and Misra 1981).

6 RESULTS

This section presents some results using the proposed partitioning technique. The platform was a network of Sun ULTRA SPARC 1 connected through an Ethernet 10Mbits/sec. A maximum of 8 machines with equivalent computing performance were involved. The experiments were performed using the sequential logic circuits from the ISCAS89 benchmark suite. Since the exact function of these circuits is not known, random test patterns have been applied during 1000 clock cycles.

To measure the speedup, the time spent in sequential simulator was compared to the time required by the distributed simulation. In theory, a speedup of n is expected for a distributed simulator running on n processors. However, this theoretical value is never reached because of the communication overhead. This overhead depends on many parameters: the test patterns, the circuit, the partition, the synchronization strategy and the network's load.

Table 2 shows the gain of speedup on two machines using the proposed partitioning method. It should be noted that the results presented in this section take only into account the simulation time excluding the partitioning and the initialization time.

7 CONCLUSION AND FUTURE WORKS

A new partitioning method based on a realistic cost function has been investigated. A reasonable speedup for a distributed simulator can be reached. The experiments show that a certain gain may be

obtained over the conventional partitioning even with a conservative synchronization method. This algorithm is able to reduce significantly the amount of messages on the network, but the speedup remains smaller then what would be expected. The difference can be explained by: (1) the usage of heuristics that may fall into local minima, (2) the overhead introduced by the synchronization and (3) the network's load.

Future works will focus on implementing other synchronization strategies such as Time Warp to study the relationship between synchronization and partitioning and its impact on the speedup. Also, the simulator will be implemented on a high performance distributed computer, based on several workstations connected through a 1 Gbyte/sec. network.

REFERENCES

Alpert C.J., A.B. Khang, 1995, "Recent Developments in Netlist Partitioning: A Survey", *Integration: the VLSI Journal*, pp. 1-81.

Bailey M., 1992, "How Circuit Size Affects Parallelism", *IEEE Trans. on Computer-Aided Design*, Vol. 11, pp. 208-215.

Bailey M., Jr. Briner, R. Chamberlain, 1994, "Parallel Logic Simulation of VLSI Circuits", *Computing Surveys*, Vol. 24, pp. 255-294.

Chandy K., J. Misra, 1981, "Asynchronous Distributed Simulation Via a Sequence of Parallel Ccomputations", *Communication of the ACM*, Vol. 24 pp. 198-206.

Dunoyer J., N. Abdallah, P. Bazargan-Sabet, 1995, "A New Generation of Digital CAD Tools Based on Probability", *27th Southeastern Symposium on System Theory*, pp. 348-352, 1995.

Dunoyer J., N. Abdallah, P. Bazargan-Sabet, 1996, "A Symbolic Approach in Resolving Signal's Correlation", *29th Annual Simulation Symposium*, pp. 203-211.

Greiner A., F. Pêcheux, 1993, "Alliance: A Complete Set of Cad Tools for Teaching VLSI Design", *Proc. 3rd eurochip workshopon VLSI design Training*, pp. 230-237.

Jefferson D., H. Sowizral, 1984, "Fast Concurent Simulation Using Time Warp Mechanism", *Proc. of the Conference on Distributed Simulation*, pp. 63-69.

Johnson D.S., M. Garey, 1979, "Computer and Intractability: A Guide to the Theory of NP Completeness", San Fransisco, CA: Freeman.

Lin S., W. Kernigan, 1970, "An Efficient Heuristic Procedure for Partitionning Graphs", *Bell System Technical Journal, Vol 49*, pp. 291-307.

Matsumoto Y., K. Taki, 1992, "Parallel Logic Simulation on a Distributed Memory Machine", *Proc. of European Conference on Design Automation*, pp. 76-80.

Mattheyses R.M., C.M. Fiduccia, 1982, "A Linear Time Heuristics for Improving Network Partitions", *Proceedings of the 19th Design Automation Conference*, pp. 175-181.

Meister G., 1993, "A Survey on Parallel Logic Simulation", *Technical Report No. TR 14-1993, SFB 124, University of Saarland, Department of Computer Science,*

Misra J., 1986, "Distributed Discrete-Event Simulation", *Computing Surveys*, Vol. 18 pp. 39-65.

Najm F., 1991, "Transition Density a Stochastic Measure of Activity in Digital Circuits", *28th Design Automation Conference,* pp. 644-649.

Newton A.R., C. Kring, 1991, "A Cell-Replicating Approach to Mincut-Based Circuit Partitioning", *International Conference on Computer Aided Design*, pp. 2-5.

Soulé L., 1992, "Parallel Logic Simulation: An Evaluation of Centrelized-Time and distributed Time Algorithms", *PhD thesis, Stanford University,* Technical Report CSL-TR-92-527.