Designing VLSI Circuit Masks with the Software Agents2

EVANDRO DE ARAÚJO JARDINI[1]
DILVAN DE ABREU MOREIRA[2]

1 - Fundação Educacional de Fernandópolis
Av. Teotônio Villela s/n
15600-000 Fernandópolis - SP
eajardini@yahoo.com.br

2 – ICMC/USP – Instituto de Ciências Matemáticas e Computação
Universidade de São Paulo
13560-970 São Carlos - SP
dilvan@icmc.sc.usp.br

Abstract

The traditional way for creating layout for Application Specific Integrated Circuits (ASICs) demands that a human designer interacts with a computer program. This program usually uses a layout generation methodology based in standard cell libraries. This methodology is effective for designing ASICs circuits because the processes for layout generation can be automated, reducing the time spent in designing circuits and increasing their reliability. However, this methodology has problems related to the maintenance of the cell libraries and to the total number and variety of cells available, as some cells required in a project may not exists in the library forcing design adaptations.

The Agents2 system was developed, using the software agent technology and the Java language, to automate the generation of standard cell libraries. Many agents (implemented as servers) compose the system: one Placer (that places the components of a circuit) and many Routers (that wires the circuits' components). The servers have the capacity to explore parallel computers (SMP) and to work distributed over a computer network. System scalability increases as new CPUs are added to the computers or new computers are added to the network.

1 Introduction

The project of integrated circuits is usually based on a hierarchy of modules. Modules in the superior levels are composed by submodules, which are formed by other submodules and so on. At the end of the hierarchy, there are modules formed only by transistors. Those modules are denominated leaf-cells [1].

The traditional way of creating an ASIC (*Application Specific Integrated Circuit*) layout requires a human planner interacting with a CAD (*Computer Aided Design*) program. It uses a layout methodology based on libraries of leaf-cells. That methodology is good for ASIC projects because the processes performed for layout generation (mainly positioning and routing of components) can be automated, thus reducing the time spent in the project and increasing its reliability [1]. However, that methodology has inconveniences in relation to the maintenance of the leaf-cell library (for instance updating of the manufacture process) and to the availability of specific cells (the variety of existing library cells is limited). As a consequence performance of the designed circuit may be sacrificed [1].

One solution for those problems is the generation of specific leaf-cells as needed by circuits, using tools for automatic generation of leaf-cell layout. The tools should be able to generate layouts for a great number of SSI (*Small Scale Integration*) circuit cells using different manufacturing processes. Since those tools would produce cells that could exactly fit the needs of the project for different process technologies, they would solve both of the main inconveniences mentioned.

The *Agents2* system has exactly the goal of generating leaf-cell layout for integrated circuits automatically (for different process technologies). This system is an improvement over the *Agents* system that was originally developed by Dr. Dilvan Moreira [2]. The *Agents2* system was developed using the Java language and is formed by two kinds of programs:

- **The Placer Server:** It positions the components that belong to a circuit and generate several layouts with different placings.

0-7695-1333-6/01 $10.00 © 2001 IEEE

- **The Router Servers:** They wire the trails that interconnect the components positioned by the Placer in a layout.

This new system design improves greatly over the old *Agents* design (written in C++ and Lisp). It is a complete rewriting of the program using the Java language, making it portable across many computer platforms and operating systems, without the need for recompiling the code. The use of the Java language facilities for multithreaded programming makes the new system capable of exploiting parallel processing in Symmetric Multiprocessing (SMP) machines. The networking system was improved and the programs simplified (for instance, the Java system now takes care of freeing unused memory) allowing the *Agents2* system to work with only two kinds of programs, Placers and Router servers (The former system used four).

This article describes the technologies and operation of the *Agents2* system and shows some results of tests performed generating leaf-cells.

2 The *Agents2* System

The *Agents2* system is a program created to generate leaf-cells layouts automatically in the CMOS, BICMOS and mixed digital/analogue circuit technologies. A key characteristic added to the version 2 of the system *Agents* is the capacity of the Router servers to explore the parallelism provided by multiprocessor computers (SMP), while it performs the wiring of the circuit trails.

The circuit layout generated by *Agents 2* does not need a virtual grid. It is a mask layout ready to be used in an integrated circuit layout. The program does not need virtual grid either during the positioning of the components or during their routing. All operations are executed at mask level.

2.1 Positioning of Components

The fabrication rules that the layout has to follow (which vary according to each production process) are read from a file by the Place server and sent to the Router servers, before any routing takes place. The circuits to be generated are received by the Placer server in the EDIF format (Electronic Design Interchange Format).

After receiving a circuit, the Placer server executes the following tasks:

- Generation of columns of FET (Field Effect Transistor) transistors that have their gates interconnected or groups of other interconnected components (bipolar transistors, for instance).
- Union of the FET columns in order to form groups. To be united, FET columns have to share a certain number of connections through drain or source. The idea is to unite FETs to layout them in the same diffusion line.
- Positioning of this component groups using the genetic algorithm. The resulting placed circuit is then sent to a Router server for routing. Several circuits are placed and sent for routing until one of them is successfully routed.

Since the positioning of components in a digital circuit is a faster task than routing them, the system uses only one *Placer* server and several *Router* servers working together. Each placed layout is sent to a *Router* that will try to connect the positioned components. If successful the *Router* sends the finished circuit back to the *Placer*.

2.2 Wiring of Circuits

The *Router* server was developed imitating the way human designers use a CAD system (Computer Aided Design) for routing circuits. The designer makes all the important decisions about the project, such as: where the wires will be drawn, the quality of the routing, the need of rewiring, etc. The CAD system supplies the designer with tools to manipulate the project. In the *Router* server, the role of the designer is performed by the *RouterExpert* and *Connect* software agents and the role of the CAD system is performed by the *Design* object.

The *Design* object keeps the data of the project as well as the methods to analyze or change it. The *Design* provides the necessary ways to collect information about the project (if two wires are colliding, etc.) and the methods to implement changes in the project (to insert or destroy wires, etc.). The *Design* object's resources are used by the agents *RouterExpert* and *Connect*.

2.2.1 *RouterExpert* and *Connect* Agents

The *Router* server was developed using the software agent technology. Agents are software components that communicate with their pairs through messages using a communication language[3]. That characteristic allows the agents to cooperate among themselves. They can be as simple as a subroutine or bigger entities such as autonomous programs[1]. The software agent technology is used at two levels:

- One *RouterExpert* agent is in charge of creating and eliminating *Connects*

agents, verifying the quality of the routing, determining the wires cost, etc.

- Many *Connect* agents are in charge of exploring and finding interconnection routes among the circuit components.

The behavior of each *Connect* agents is simple. But the idea is to have several agents working together and in a parallel way (instead of having one complex program full of production rules). The more *Connect* agents can be executed in parallel, the less time it will take to route a circuit.

The agents *RouterExpert* and *Connect* execute the routing by using a variation of the maze routing algorithm[4]. That algorithm is based on a rectangular grid that represents the circuit. The circuit is mapped on the grid and it is determined which cells are free or blocked. To each cell a cost is attributed, forming a structure similar to a search tree. Then an algorithm is used to analyze and expand the vertexes of the tree generated to find the objective.

A variation of that algorithm introduces the concept of interesting points[4]. Instead of expanding the vertexes to all the directions, *Connect* agents expand them directly to an interesting point. A point is considered interesting when it is aligned with the objective point or with the extremities of an obstruction. Actually to the extremity of an obstruction is added a security margin proportional to the minimum distance from the obstruction to the wire being routing plus half of the wire width.

2.2.2 *Router* Agent Operation

The *Router* server receives the circuit serialized from the *Placer* server and instantiates an object *Design* containing the circuit. Starting from this point, the *RouterExpert* and *Connect* agents will use this object to perform the circuit routing.

First the *RouterExpert* will do the simple routings, connecting the points that are aligned in polisilicon or diffusion.

To connect the other components, the remaining circuit nodes are put in a list, ordered by importance and size (the smallest ones first). From that point on, only the polisilicon, metal1 and metal2 layers will be used for connection.

In the *Design* object, each node contains a list of subnets that partially connect the node components. A subnet is the group of wires that interconnect part of the components that belong to the same circuit node. The method *connectNode* of the *RouterExpert* agent uses this list to connect the subnets. The method *connectSubnet* does this job. If the method does not manage to connect a subnet, it puts it at the end of the list (to give it a second chance later) and tries to wire the next one. If at the end, the method does not manage to connect all subnets, an error condition occurs and this placed circuit is considered unwirable.

The *subnet* connections are based on the interesting points and are performed by the *Connects* agents. They analyze the interesting points that are closer to them running in parallel as Java execution threads (one agent, one thread). Starting from this analysis, several operations can be performed by *Connects*: they can connect the wire with its destination (and offers it to the RouterExpert as a possible best wire), create a new wire section or replicate. When they replicate a new *Connect* agent is created for each new interesting point. The agents will reproduce until they find their objective, another the subnet in the same node, or they run out of interesting points.

The *Connect* agent's goal is to reach a point in another subnet in the same node to join the two. In this process, they should join all subnets belonging to a node. This node is them considered wired. If all nodes of a placed circuit are wired, the routed circuit is sent back to the *Placer* agent.

3 Results

The motivation of this work was to help the developing of better processes for circuit mask generation using the inherent parallelism of distributed systems and multiprocessor SMP computers. The idea is that, as the program uses these resources, the time spent in the process of circuit cell generation decreases. The goal of the tests was to prove that the program scales up in distributed environments and in computers with multiple processors.

Two groups of tests were performed:

- The first group tested the *Router* server in relation to its use with different numbers of *Connect* agents and processors. The server was executed in only one machine at a time. Those tests analyzed its scalability in relation to the number of available processors.
- The second group of tests analyzed the *Agents2* system executing in a distributed way in a network of heterogeneous computers.

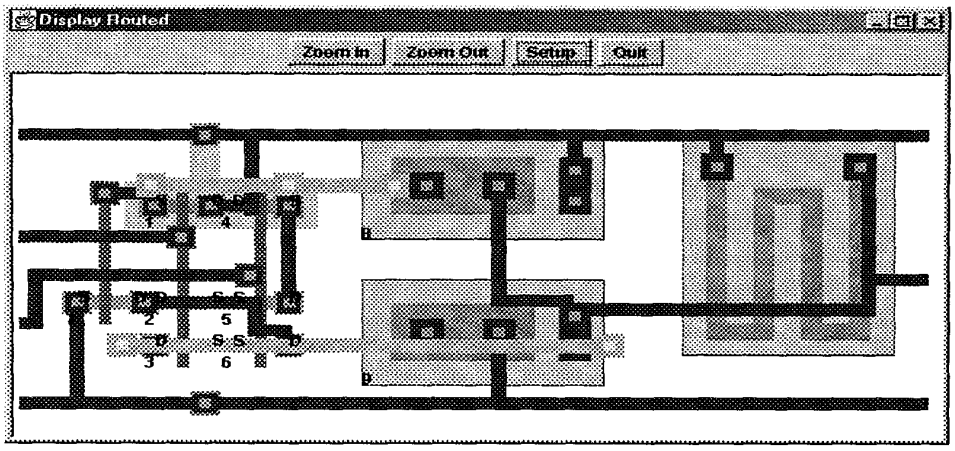

Figure 1: NAND Gate with Analog Cell

3.1 Tests in Individual Machines

The following machines were used to perform the tests:

- Computer 1 - Intergraph computer with 4 Pentium Pro processors of 200 MHz using Windows NT 4.0.
- Computer 2 - Computer with 2 Pentium Pro processors of 200 MHz using Windows NT 4.0.
- Computer 3 – Computer with 1 K6 II processor of 400 MHz using Windows 98

The tests in individual machines were performed with circuits whose components were already positioned to test just the *Router* server acting in different computers. The time taken to wire a NAND gate with an analog cell (figure 1) is shown in the graphic of figure 2 routed in the computers 1, 2 and 3 respectively.

The graphics presented in figure 2 show the *Router* server scalability in relation to the number of available *Connect* agents working in parallel. With the analysis of the graphics, it is possible to show that, as the number of parallel *Connect* agents increases, the circuit routing time decreases down to a limit point. Also, if the graphics for the computers 1, 2 and 3 are compared with each other, it can be shown that the *Router* agents scales up in relation to the number of available processors.

It is worth emphasizing that the difference among the execution time between the machines was not bigger, because only one *Router* generating just only one circuit does not need a lot of computer power.

Another problem that should be remembered is that the machines involved in all the tests do not differ just in numbers of processors, but also in speed, memory, motherboards, etc. So the values cannot be considered precise measurements, but they are good enough to show scalability. The Java Virtual Machine, that run the tests, uses a JIT technology developed by Sun Microsystems to speed up execution.

3.2 Tests in Distributed Computers

This work also tested the system scalability in distributed environments. The goal of the tests is to show that the cell circuit generation time decreases as new computers are added to the setup.

The following computers were used to perform the tests:

- Computer 1 - Intergraph computer with 4 Pentium Pro processors of 200 MHz - Windows NT 4.0.
- Computer 2 - Computer with 1 Pentium II processor of 266 MHz - Windows NT 4.0.
- Computer 3 - Computer with 2 Pentium II processors of 350 MHz - Windows NT 4.0.

The graphics of figure 3 show the obtained results. They demonstrate that, besides being able to scale up in SMP computers, the *Agents2* system still maintains the original *Agents* [2] system ability to scale up in distributed environments. The results of the performed tests can vary a little from a program execution to another due to factors linked to the computer configuration differences and the workload of the network.

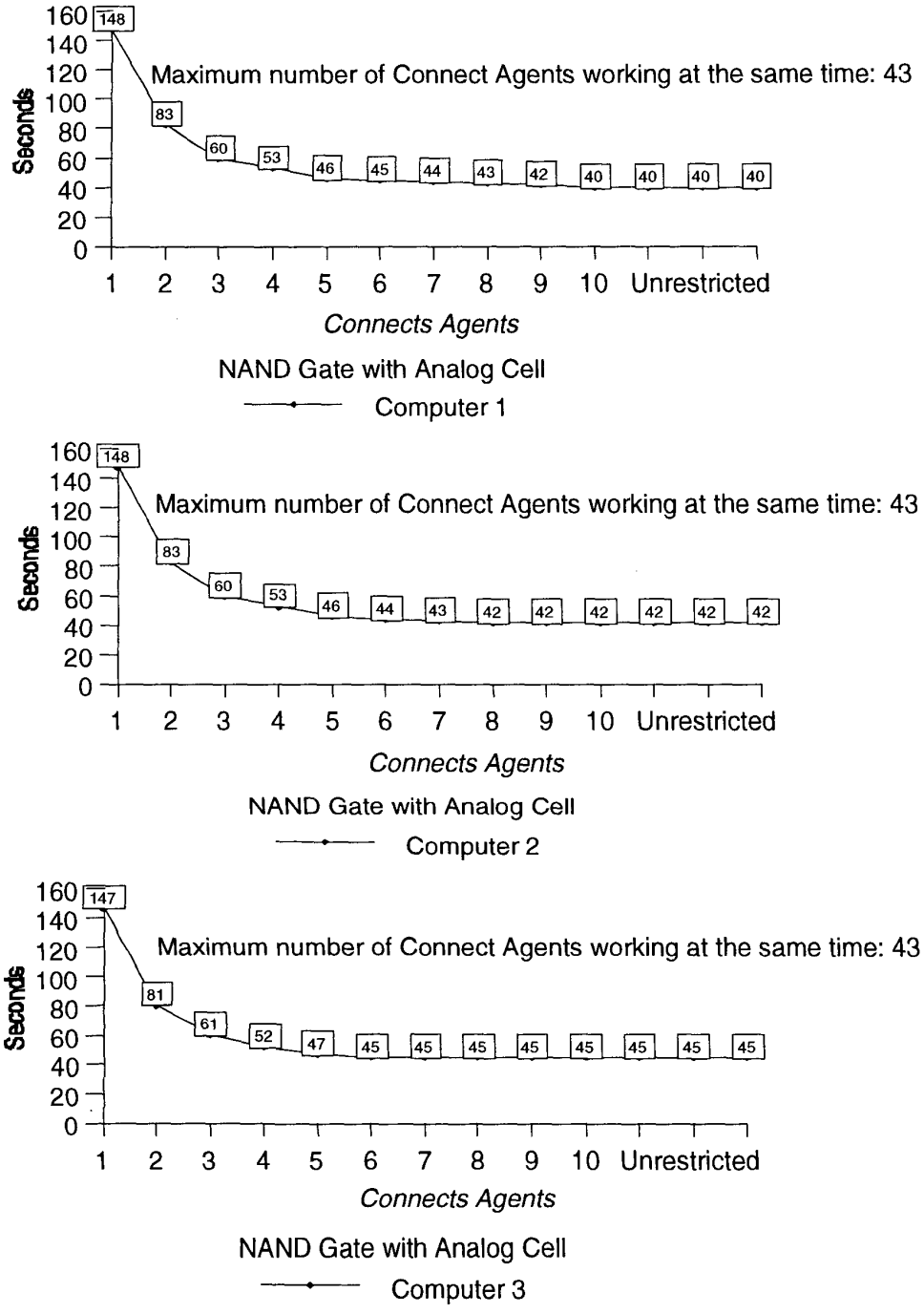

Figure 2: Results for routing a NAND gate with an analog cell.

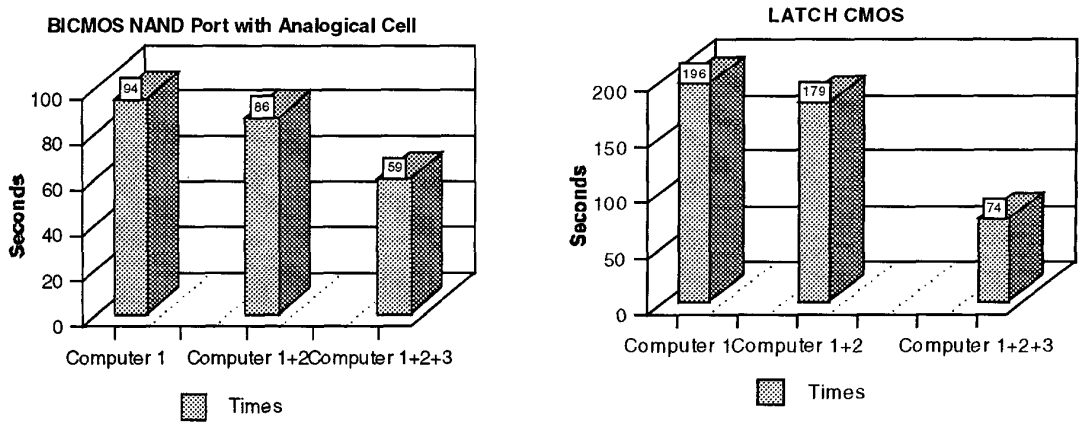

Figure 3: Results of the times in a distributed environment.

Unfortunately it was not possible to test the system in other architectures (Sparc, PowerPc, Alpha, etc), as the machines to do so were not available. Also, it was not possible to test using the Linux operating system, because all available Java Virtual Machines for Linux (at the time of the tests) where not able to work with Linux native threads, instead they created a new process for each new Java thread. They artificially degraded performance.

4 Conclusion

This article presents the new *Agents2* system: A tool for automatic generation of leaf-cell mask layout. The system has two parts, the *Placer* and *Router* servers. The servers can work together in a computer network or in a single computer.

Besides generating mask layout for a variety of technologies, the system goal was to show scalability in parallel SMP computers and distributed environments. The tests results shown here demonstrate the system scalability in both distributed and multiprocessor SMP systems. Also the system technology seems very appropriate for pure analog cell generation. No work has been done to test this idea, but the *Agents2* system is freely available for download (under the GNU Public License) at the site http://java.icmc.sc.usp.br/research. The system is still a prototype, and many improvements have to be made to it to integrate the system in commercial layout generation tools.

The authors gratefully acknowledge the São Paulo State Research Foundation (FAPESP) for financially supporting this work.

5 References

[1] D.A Moreira, and L.T. Walczowski, "Using Software Agents to Generate VLSI Layouts," *IEEE Expert Intelligent Systems,* Vol. 12, No. 6, November/December 1997, pp. 26-32.

[2] D.A Moreira and L.T. Walczowski, "The Development of Very Large Scale Integration Systems", *Knowledge Based Systems Techniques and Applications,* Vol IV, Chapter 37, pp. 1227 - 1271, Academic Press, January 2000, ISBN 012443875X.

[3] M.R. Geneserethand S.P. Ketchpel, "Software Agents," *Communications of the ACM,* Vol. 37, n.7, July 1994, pp. 48-53, 147.

[4] M.H. Arnold and W.S. Scott, "An Interactive Maze Router with Hints," *Proc. Of the 25th ACM/IEEE Design Automation Conference,* 1988 IEEE, pp. 672-676.