MASTER

Gate array placement by stimulated annealing

Jongen, R.J.

Award date:
1985

5530

TECHNISCHE HOGESCHOOL EINDHOVEN

VAKGROEP AUTOMATISCH SYSTEEM ONTWERPEN (ES)

GATE ARRAY PLACEMENT BY SIMULATED ANNEALING.

DOOR R.J.JONGEN

Verslag van een afstudeer periode verricht in de vakgroep ES,
in de periode van december 1983 tot oktober 1984,
in opdracht en onder leiding van Prof.Dr.-Ing. J.A.G.Jess.

SUMMARY

An application of Simulated Annealing, a combinatorial optimization method, to gate array placement is described. Simulated annealing is also known as statistical cooling.

Gate arrays are almost fully processed, with the exception of the interconnection masks chips, consisting of a matrix of basic cells. A to be into silicon implemented set of functions has to be mapped on the basic cells, obtaining a realization satisfying the required performance. The placement involves the rearranging the basic cells such that the interconnection is to be made within the required area.

The combinatorial optimization algorithm involving simulated annealing (also referred as statistical cooling) is based on a detailed analogue between the optimization of very large and complex systems and statistical mechanics (for example the growing of a single crystal from a melt by careful annealing).

The most difficult part is the controlling of the parameters, such like the cooling step size, and the definition of cost function parameters to reach finally a "good" placement. The only way now to determine these parameters is by examine some experiments, because this is the missing link in the theory of statistical cooling. Therefore, to receive finally an increasing insight into the best values of these parameters and the best way of controlling them, a flexible implementation of the optimization algorithm has been made.

A minor number of results are obtained by examining some experiments (placing some networks on gate arrays). The overall conclusion given by the experiments is that the final placement is thought to be near to optimal placement reached within an acceptable amount of the CPU-time consumption. The best and fastest cooling scheme, here found, is piece wise linear, controlled by the acceptance ratio. A suitable objective function contains a parameter including the sum of the perimeters of the surrounding box of all nets and also a quantified factor expressing the density of the surrounding boxes at grid positions.

PREFACE

The thesis for the master degree in Electronics I decided to perform within the research group Automatic System Design, due to the existence of a close relationship between software and hardware. This research group is currently working at Computer Aided Design (CAD) for Very Large Scale Integration (VLSI). Thus, my thesis subject is closely related to that particular domain of the VLSI design. It is involved with automatic layout generation of circuit layouts. The original description of the contents of my thesis was said to be :

"AUTOMATIC GATE ARRAY PLACEMENT AND ROUTING"

This is a rough approximation of the contents of the task. It can lead to a enormous amount of work consisting of constructing and developing a large number of algorithms and programs. Therefore, in order to limit the amount of work the decision of choosing two algorithms out of a very large number of algorithms; one placement and one routing algorithm. For the placement algorithm the algorithm of simulated annealing was chosen and for the routing task the "Greedy router" was to be the most obvious one.

One of the most important requirement, the to be developed system has to meet to, is its technology independence. It is very hard to confirm to this requirement, and so far as I concern nobody has succeeded in fulfilling this requirement yet. The most obvious start of the task was doing some literature review and trying to define a general storage structure for defining all kinds of gate arrays of all different kind of silicon foundries.

Within the time claimed for finishing my task, I was only be able in designing the placement algorithm and implementing it one several machines. The implementation of the algorithm was necessary for getting more experience of the course of the placement during the run. The increase of practical experience was inevitable since this placement algorithm cannot be theoretically proven.

With the gathered experience of the implementation and the runs of some placements it was possible to come to a detailed specification (Software Specification) of the algorithm. Thus, for the integration of a placement program in a complete gate array design system, only the implementation phase of the software life-cycle, of this program, must be performed.

[illegible]

Rob Jongen,
Eindhoven,
The Netherlands
Sept 30th, 1984

CONTENTS

CHAPTER 1

INTRODUCTION

The research group Automatic System Design (ES) of the Eindhoven University of Technology department of electrical and electronic engineering keeps working on tools for Computer Aided Design for Very Large Scale Integration (VLSI), like a layout editor and a circuit simulator etc. At the end of 1983 the Dutch government accepted the ICD/NELSIS project. The goal of this project is to design a complete CAD workstation for VLSI circuits. The research group ES is one of the members of the ICD project and it has to take care of the circuit simulator (PWL) task 4, and of a full automatic gate array design station.

In the mid 70's some IC manufacturers feel the need to have a design approach with a typical short turn around time (a few weeks) and low costs intended for circuits count of lower than 10,000 chips a year. One typical design approach that meets these requirements is the gate array design approach.

The design technique, based on gate arrays, is a rapid design approach where the cell structures are more primitive than for example in the standard cell approach, and exist either as groups of 4 to 6 transistors or simple combinational logic gates in a regular two dimensional array. The wiring procedures involved with gate arrays are clearly more complicated than for example standard cell approaches because the functional cells themselves have to be constructed from a number of array (basic) cells, and the interconnection channels are of a fixed width.

The customers that design a personalization of a gate array have to take pencil and paper to manually place the gates, interconnect them, and submit this layout, along with a circuit schematic list, to the gate array manufacturer for digitizing and wafer processing the chips. Most of the manufacturers will also provide some tools such as stencils, etc. for the common used functions and recently they provide also some tools like an interactive layout editor and some verification tools.

Since the production of the first gate array the circuits intend to be integrated are growing and growing in complexity (from LSI to VLSI), and so do the master slice grow from a few hundred cells to several thousands. With this expansion it becomes more and more difficult to design an integrated circuit and the turn around time will also become larger corresponding with increasing costs. Therefore, in order to achieve error free design and a further reduction of design costs and turn around times, automated procedures must be used for placement and routing.

One of the most time consuming processes in the manual gate array design sequence is the layout verification, i.e., insuring that the metal pattern implements the desired schematic that often takes weeks for arrays larger than 1000 gates. Due to the fact that automatic layout generation programs construct by correctness, no layout verification is necessary. Thus, the introduction of automatic generated layout can save a lot of time. Of course this is only true in fully automatic systems, because any manual interaction makes verification necessary.

Up to now the automatic gate array design stations have not succeeded in discovering the market and the manual design will still not be superseded by the automatic one, as a result of the fact that nearly all automatic design stations are not generally applicable for different images. Most engineering houses that are involved in gate array design, have more than one master chip supplier. Therefore, they need a system which can handle different images of different foundries.

Considering the wishes and requirements of an automatic gate array design system, the ICD/NELSIS project has formed a minimum list their system have to fit to. These requirements are:

- the system has to function as soon as possible.
- the system has to be flexible to the different master chips of the suppliers and foundries.
- the system has to use the existing programs as much as possible.
- the system should be fully automatic.

Concerning the layout stage of the design system, fig. 1.1 gives a nice impression of how such a system should look like.

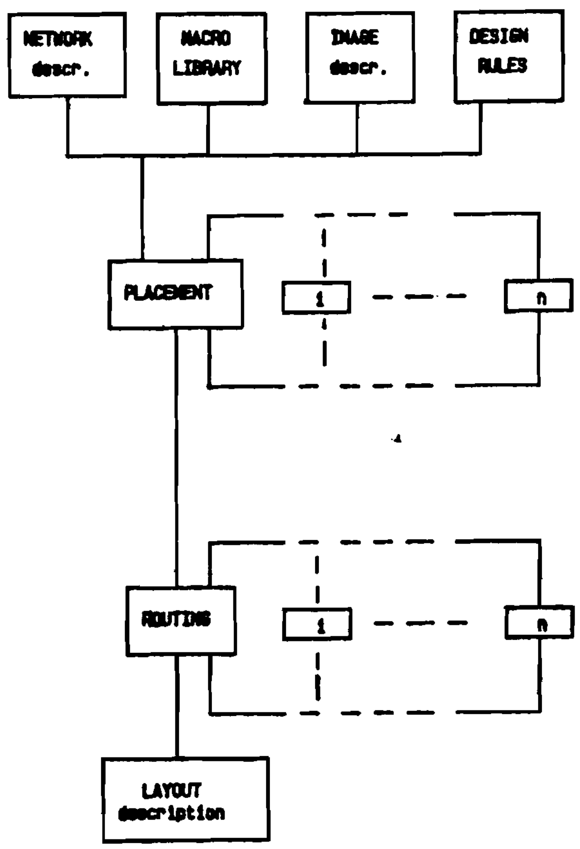

Fig. 1.1.: Gate array layout design system proposed by ICD/NELSIS.

The first implemented placement algorithm is based on simulated annealing. In this report the usage of simulated annealing as a placement method for gate arrays is explained, and it will show how this placement algorithm works in detail.

The qualities of the placement program were evaluated with the help of a few gate arrays of different suppliers. There were some restrictions in reviewing the qualities especially the routability quality because on the moment I finished my research work on this subject there was no suitable gate array router available in our research group. The routability quality of the tested gate arrays were reviewed on some graphics.

In the first two chapters the concepts gate arrays and placement are explained. Afterwards the principle of the simulated annealing algorithm is described. Inherent to a annealing optimization method are the experimental results, which are discussed in the last two chapters. Rest us to inform you about the usage of this annealing application for macro placement.

CHAPTER 2
GATE ARRAYS

The semi-custom silicon technology was introduced to answer the needs of producing chips faster with lower costs. These needs became stronger with the tendency of producing smaller volumes of chips than the traditionally full custom technology required.

There exist two main semi-custom approaches to complement the full custom design route; the gate array and the standard cell approach. The gate array, master slice or uncommitted logic array approach to VLSI design is based on the concept of using an integrated circuit consisting of a matrix of identical components or functional elements that has passed through all stages in the chip fabrication process except the final interconnection (metal) stages. The array chip is a standard component and can be held as a stock item. The specific interconnect pattern derived from the users or customers system logic specification is then applied to the uncommitted chip via one or more metallization masks to complete the chip. This leads to the personalization of the master slice chip. The identical components of the gate array, often called basic cells, consist of one or more transistors and/or gate cells. Clearly, trade-offs exist in the design complexity of the basic cell since a more complex cell will minimize interconnection requirements but will ultimately result in less efficient utilization of the elements within the cell.

2.1. THE ORIGINS

Through the ages there was always a need to produce electronic circuits which are faster, cheaper, smaller and more reliable. The electronic engineers tried to supply this demand by developing new technologies, new design methodologies and providing new design supporting tools especially CAD tools. The decrease of effort necessary for designing a chip has lead to integration of more and more kinds of electronic circuits in silicon. Not only this decrease causes a diverting of integration limits, but also reliability and cost aspects play a relevant rule in this subject. Although, the decreasing tendency in cost of integrated circuits, the total design and production cost are still too high for a small and medium volume of chips of a particular circuit.

The high design costs are due to the relative long time (3 to 6 months) necessary for passing through all stages of the design cycle to reach finally a complete silicon implementation of an electronic circuit. This time-to-market is called the throughput time of a design. The total cost of a single chip can be limited by an increasing volume of processed chips, because the design costs need only to be made once per design.

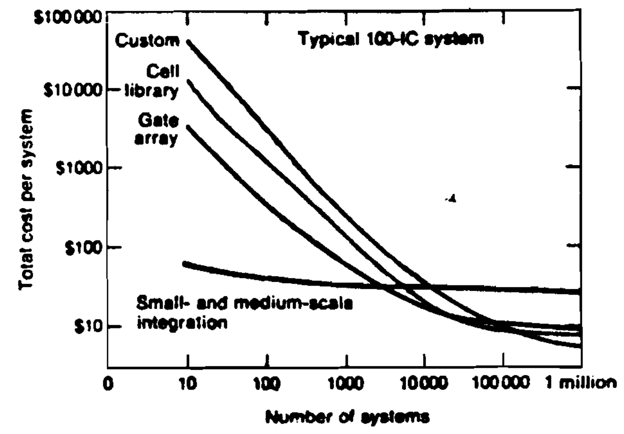

Fig. 2.1.: Cost function of gate arrays.

Of course an obvious method for decreasing these design costs is to reduce the total throughput time for each design. This can be achieved by introducing for example computer aided design tools. But the throughput time of a design is not only dependent of the development time it is also dependent of the process time. To achieve a very large reduction of the throughput time and the involving costs a total new silicon technology and design methodology is needed. The semi-custom silicon technology was introduced involving with standard cell and gate array designs to answer these needs.

The development of semi-custom integrated circuits is the most important recent event in electronic silicon technology since the microprocessor. Unlike the microprocessor revolution of early to mid-70s which was fueled by silicon technology, this revolution is fueled by system design engineers. These engineers are driven by the need for high-performance, low-cost and compact system designs. Also from this point of view, semi-custom design technology answer these needs the best way, due to the almost fully processed semi-custom I.C.'s. Of course the single customization layer, will carry from this point of view the best results, the smallest process time and thus the lowest process costs.

The process time has become small compared with the development time, by the introduction of the semi-custom silicon technology which, of course, still results in high costs. To come to a further reduction of costs the development time need to be shortened radically. Therefore, it is nearly required to have sophisticated CAD workstations available especially from the point of speeding up the development time. Workstations provide productivity enhancement for the design project, satisfying the basic need for accelerated time-to-market. Reaching this goal, today's workstations provides a completely integrated semi-custom design environment. Including schematic entry, logic simulation, design verification, and automatic layout, the workstation addresses many concerns and risks a system designer has with semi-custom design.

Additionally, the system designer often needs to use a variety of technology options given specific application requirements. CMOS, ECL, TTL, linear semi-custom technologies; all have their respective application areas that, until now, required a different design methodology and corresponding set of risk factors for each. The workstation ensures technology independence based on semi-custom libraries ported to the system with the identical design methodology for each technology. This one feature alone encourages the system designer to use the technology and silicon vendor that best meet his or her requirements.

2.2. DESIGN METHODOLOGY

The ultimate goal of chip design is to have a silicon implementation of an electronic circuit. To answer this demand a designer has to pass through different stage in a way determined by the design methodology and the chosen silicon implementation technique. In general these stages are involved with functional specification, logic design, design verification and physical layout design.

2.2.1. GENERAL VLSI DESIGN

Starting at the top level of the design path, functional specification was mostly done in natural language, sometimes accompanied by constraints on in- and output, and sometimes even a prescription on the kind of algorithm to be used. But nowadays the accent of importance is moving more and more from an unstructured approach to the use of system architectures and other structured methods. The functional specification is then handed to the logic designer who will carry out the implementation phase.

The logic designer will then built a logic description of the circuit using technology dependent symbols. The way the logic description arises from the functional specification is strongly determined by the chosen silicon implementing technology. For example a logic design of a CMOS circuit will be totally different from the design in TTL. This design will be verified with a logic and a circuit simulator. The logic design phase has an interactive character: design-verification-redesign-verification and so on. After the logic design is finished the description is passed to the layout design phase.

The ultimate task of a layout design is to produce a layout, a set of data that completely specifies the geometry of the circuit in a unique way. Usually this data is an encoding of patterns, which exist in separate planes. The term mask will be used for each separate plane with a pattern.

Until recently, after the logic diagram for the integrated circuit had been checked, the designer would then commence to layout the circuit, transistor by transistor, in a relatively random manner. Each device (transistor) is defined in the fabrication sequence by masks, which selectively define diffusions, pn-junctions, etc. The designer, therefore, has either to draw his layout in coloured pencil on squared paper, in the most primitive form, or define shapes on a coloured graphical terminal. However, it has become evidently clear that the potential complexity of today's VLSI circuits cannot be handled by this primitive approach. The alternative is using structured layout approaches, such as PLA's, were defined for an easy and fast automatic synthesis of a layout structure. Symbolic approaches which endeavour to divorce the designer from the technology and the process of laying-out primitive shapes to detailed design rules were also introduced. Symbolic approaches, induces the use of leaf-cell libraries and placement and routing programs, to interconnect the chip.

In the fall of 1979 the first structured VLSI design philosophy was published by Carver Mead and Lynn Conway in their book "An Introduction to VLSI Systems" ([Mead 79]). The essential new concept that can be distinguished in the Mead and Conway design style is the divide and conquer principle. The divide and conquer principle is believed to be a solution for the management of complexity during the design of very large scale integrated circuits. A circuit is divide into manageable pieces (cells), each piece is evaluated, and when necessary it is again subdivide into new pieces (cells) with lower complexity. This process is repeated until a collection of pieces has been obtained, that can be easily implemented. This divide and conquer design style also known as hierarchical design style, has lead to a general acceptance by system designers to manage complex designs. Hierarchical design involves the successive decomposition of a complex design into a series of lower complexity problems which can be solved independently. The problem of managing the complexity in design became a major task when VLSI technology permits realization of circuits which contain at least 50.000 devices per chip. In the management of complexity a well structured design will be satisfying in itself. The hierarchical design process is controlled by the use of different partitioning techniques.

2.2.2. GATE ARRAY DESIGN

The main difference in design methodology, that distinguishes in gate array design, is the fact that the gate array approach will not support the hierarchical concept so easily.

The logic design stage in gate array design distinguishes from general circuit design in the importance of low level decisions at higher levels. This is due to the significant difference in implementing basic functions of almost the same kind. For example in a certain technology a three-input NAND gate and a two-input NAND gate will equal in size. But a four-input NAND gate on the other hand will be more difficult to implement because it will need at least two basic cells and therefore it will be two times larger (on the chip) than the one with fewer inputs. Therefore, most of the time a set of real basic functions (basic building blocks) i.e. functions that can be integrated easily (small area) will be determined for the representation of complex functions. It is notable, that layout parameters are reflected in the logic design phase.

In the layout stage of gate array design, the concept of "repetition" will generally fit better than hierarchy, due to the regular structure of a gate array image. This fact that gate arrays not easily support the concept of hierarchy, will lead to a compulsive demand of having automatic placement facilities available. The complexity will be one reason and the number of basic building blocks to be placed is another reason for using placement routines.

2.2.3. WORKSTATION SUPPORTS DESIGN

Designers seeking tangible support for gate array implementations find instead that manufacturers offer minimal design aids during the critical logic and schematic phases. The first hurdle creating the design on paper rather than in a computer-based format. Then the designer must hand-code the network-list description of his circuit into the particular machine language that the manufacturer uses to run automated design tools.

Simulation and analysis, layout and routing are performed by the manufacturer. These are driven by a textual description of the design, and the user's task of generating this syntax-dependent is difficult error-prone. When this process is coupled with the lack of experience of a typical gate array designer, a significant bottle-neck crops up in the development cycle. The usual result is poor use of the resources because of the large number of reruns necessary to produce a correctly simulated design.

Another problem arises early in the design process when key architectural decisions must be made. Often, there is little information available to the designer concerning the amount of logic that will actually fit into a given array. As the design progresses, there is a critical need to be able to jump ahead to determine whether a proposed design approach will be too large or too small for the array. What's more a designer needs to be able to extract information from the physical layout and return it to schematic or logic diagram for further analysis. This is for example important for making timing analyses of the interconnection path.

The key to successful designs of large arrays is aside from the ability to design hierarchicaly, the availability of suitable CAD workstation providing a complete set of hierarchical design tools to handle all aspects of an array design: from logic design to physical layout design. Hierarchical design means subdividing the problem into manageable blocks and designing each block until it functions correctly.

A stand-alone, portable universal development system will fill the void by providing a complete solution to all phases of the gate array design task. Moreover, the "universality" could be used to design arrays produced by a number of semiconductor manufacturers. This is nearly a demand for small and medium sized companies who are involved in gate array design. The "universality" stands in stark contrast to some mainframe-based gate array development systems, which support just one type of array or only arrays produced by one manufacturer. Such a universal development system also allows design development from the functional description level right through to a finished layout, using a common database. That database contains the information required by the specific gate array, describing properties of the bare array before the layers of metallization interconnections are added. These properties present component, layout, and design-rule information of a particular gate array image.

Although, it seems that gate array workstations brings array design down to system designers level, it has to be remarked that due to the possible inability to easily place and route large gate arrays, the freedom, flexibility and time saved with these workstations are lost. This is sometimes called the Achilles heel of workstations for semi-custom I.C. design.

Recently, semi-custom silicon vendors encourage engineers to assume more of the design responsibilities on workstations, allowing the vendors to concentrate on their strength: silicon fabrication. This has lead to a transferring of semi-custom I.C. design responsibility from silicon manufacturers to system designers, involving with a fundamental change in design methodology. This makes the designer no longer dependent of one technology supplied by one particular silicon foundry. Thus, the designer is now able to apply the most suitable silicon technology for the implementation of his circuit.

2.3. IMAGE DESCRIPTION

The gate array image description distinguishes a number of generally applicable terms:

- o core cell
- o interconnection "channels"
- o chip image
- o design rules

The structure of interconnection "channels" (spaces) are very technology dependent, but mostly they can be described as a fixed space with a certain number of design rules defined on its use. In the CMOS single layer gate array technology, there is a further restriction on the use of polysilicon as second interconnection layer, which is of course already preprogrammed. In this technology a channel is constructed of polysilicon cross-unders i.e. small strips of polysilicon with on its both ends a via. These poly cross-unders are used for the connection direction orthogonal at the cell rows. The personalization now includes the determination of the positions of small metal strips to connect poly cross-unders for the connection in one direction and the determination of long metal tracks across the poly cross-unders for the connection in the other direction. By using this trick one has created a single layer personalization gate array that gives the feasibility to interconnect in two directions.

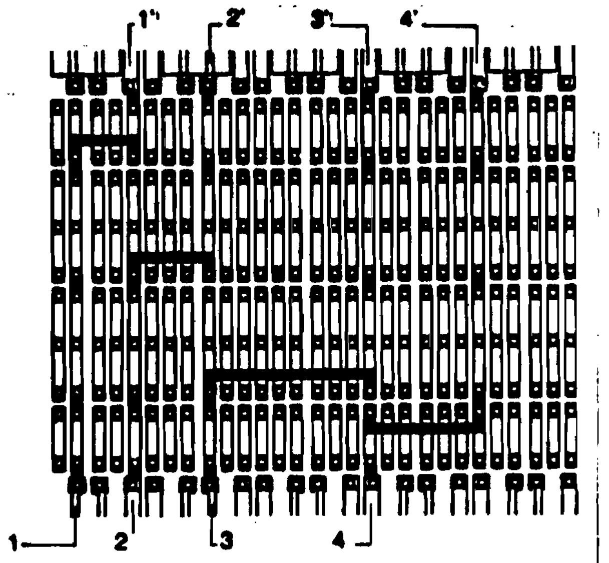

Fig. 2.2.: Routed CMOS channel with poly cross-unders.

In the chip floorplan we can distinguish two main types:

- o cell rows
- o cell blocks

In an array with cells positioned in rows the interconnection between two interconnection channels is somewhat difficult, certainly it is in the middle of a channel, if the use of feed-throughs is restricted. Therefore, in some images the cells are arranged in blocks, mostly consisting of two rows, to make the interconnection easier.

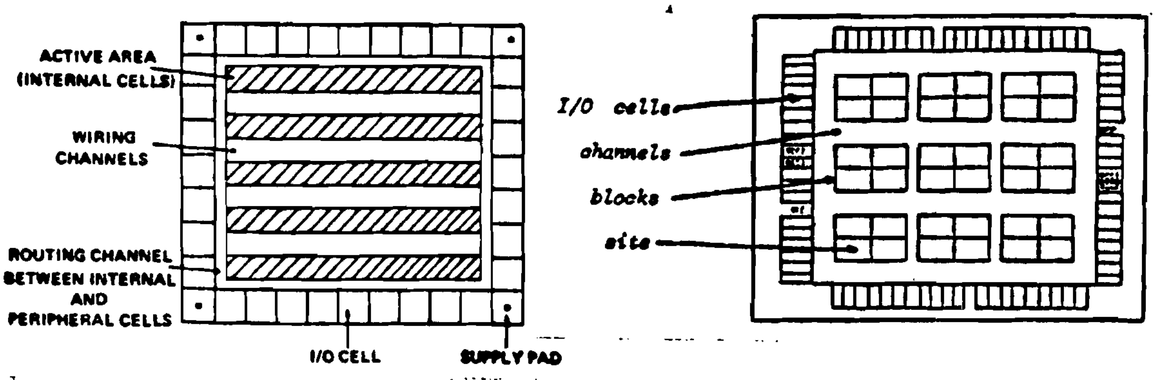
ACTIVE AREA (INTERNAL CELLS)
WIRING CHANNELS
ROUTING CHANNEL BETWEEN INTERNAL AND PERIPHERAL CELLS
I/O CELL
SUPPLY PAD
I/O cells
channels
blocks
site

Fig. 2.3.: Two main types of chip images.

To save time, and to make designing easier, the most appearing basic functions will be already layouted and collected in a library. There may exist more than one different layout pattern (stamp) representing the same function in that library. Common pre-layouted library functions are: 2 input NOR or NAND, 3 input NOR or NAND, inverter, D-flip-flop, and sometimes there exist also some more complex functions such as counters and so on.

There are several types of gate array chips, each offers different performance features. As might be expected, digital master slices are available in virtually all digital technologies: NMOS, CMOS, I2L, ISL, TTL, ECL.

2.4. DIFFERENT GATE ARRAY IMAGES

The cell configurations must be chosen for ease of both internal interconnection to form the required gate functions and external connection of these functions into higher levels of functional capability.

This leads to very different cell configurations for the different technologies. Most cells are made up of components which can easily be configured to form simple logic functions within each cell, except for ECL. ECL cells are generally larger, including up to 20 or more transistors, and are suitable for the implementation of more complex functions within the cell.

2.4.1. "I2L" AND "STL"

All designs using integrated injection logic (I2L) and its Schottky variants employ a single transistor gate, allowing the implementation of the nand function through the use of several inputs. A very dense layout is possible with I2L, although this is not true with STL.

The cells are made of single transistors, each configured as a single input, multiple output inverter.

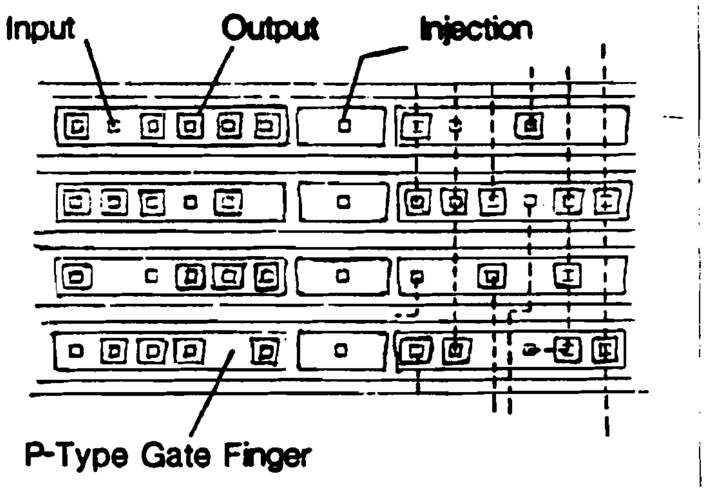

Fig. 2.4.: I2L gate array image.

The I2L cell configuration used is shown in Fig. 2.4., and includes 8 single input, 4 output gates. The contact and the underpass levels are customized as well as the metallization, so simplifying the problems of routing through the cell and providing many of the benefits of two-level metal with only a single layer.

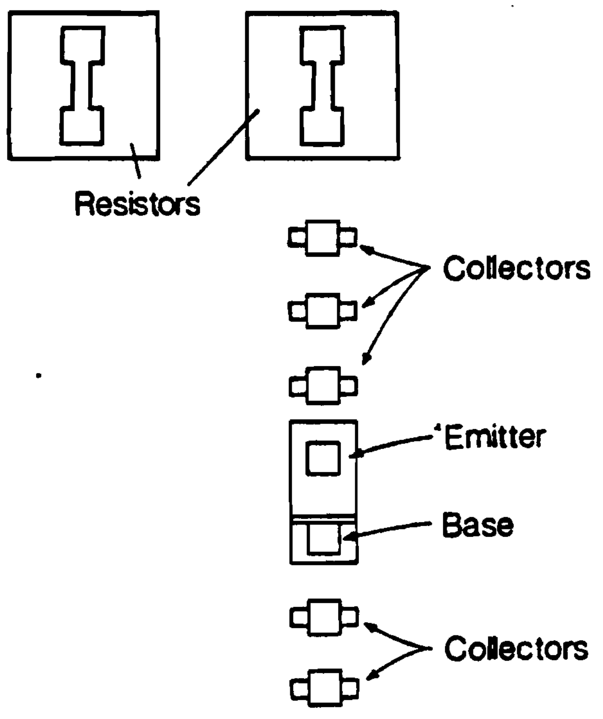

Fig. 2.5.: STL gate array image.

Schottky transistor logic (STL) has a similar layout to that of I2L above with a "stick" structure for ease of interconnection, see FIg. 2. 5. The cell, however, becomes larger due to the addition of two resistors. The design no longer includes customized underpasses and contacts, but instead uses double metal.

2.4.2. "ECL"

ECL designs tend to use larger cells containing greater than 20 transistors in each. This leads to an improvement in the operating speed of complex logic functions, flip-flops for example , but may lead to under-utilization of the cells if many simple gates, such like single NOR and NAND gates, are required.

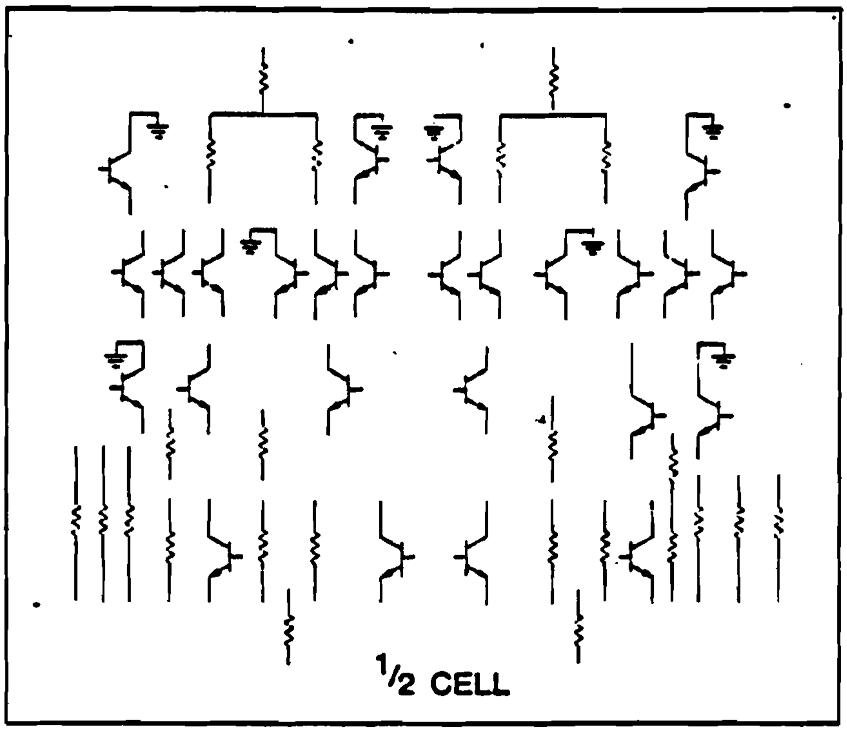

Fig. 2.6.: ECL gate array image.

2.4.3. "CMOS"

All CMOS gate array cells are made up of transistors pairs with common gates which can easily be configured into logic functions. There is a variety of cell types, containing from two to six transistor pairs, each manufacturer claiming his design to be the best, although this depends on the type of logic function to be implemented.

The two transistor pair cells, shown in Fig. 2.7., are the best for simple functions such as two-input gates, but the larger cells are the better for more complex functions, since these can then be implemented within a smaller number of cells.

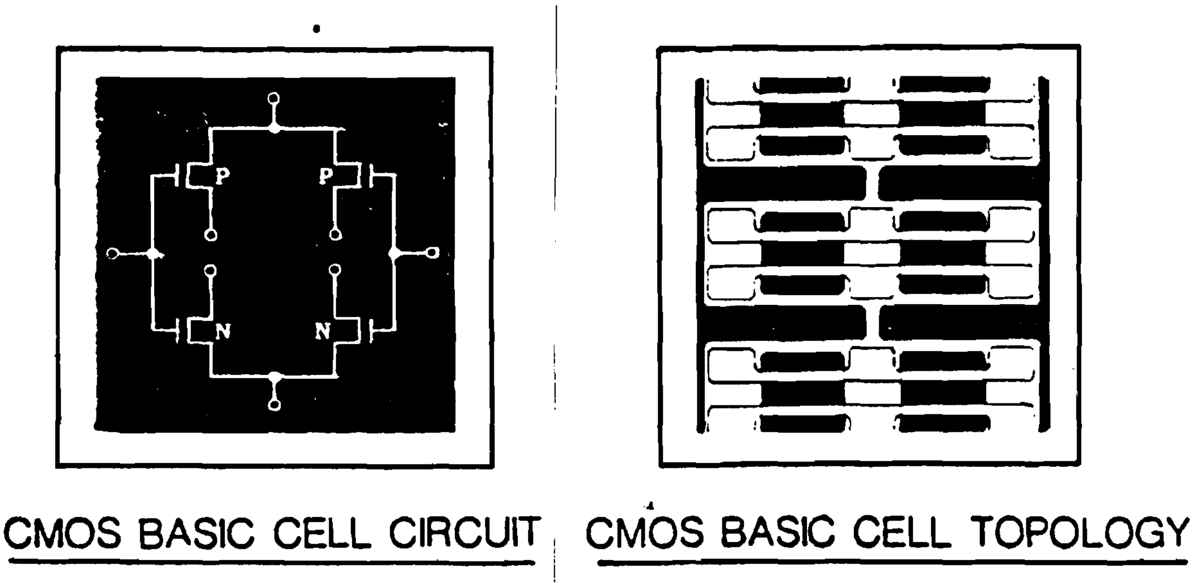

Fig. 2.7.: Fujitsu two transistor pair CMOS cell.

The three/two pair cells have the advantage of a polysilicon cross-under through the middle of the cell for ease of routing. The 'AMI' cell shown in Fig. 2.8. is particularly unusual in having a cross-over gate in the second pair section of the cell, allowing the use of single transistors where this would be useful. This allows economical implementation of transmission gates which require a N and P channel transistor pair with separate gate connections and can simplify the interconnection of other logic functions.

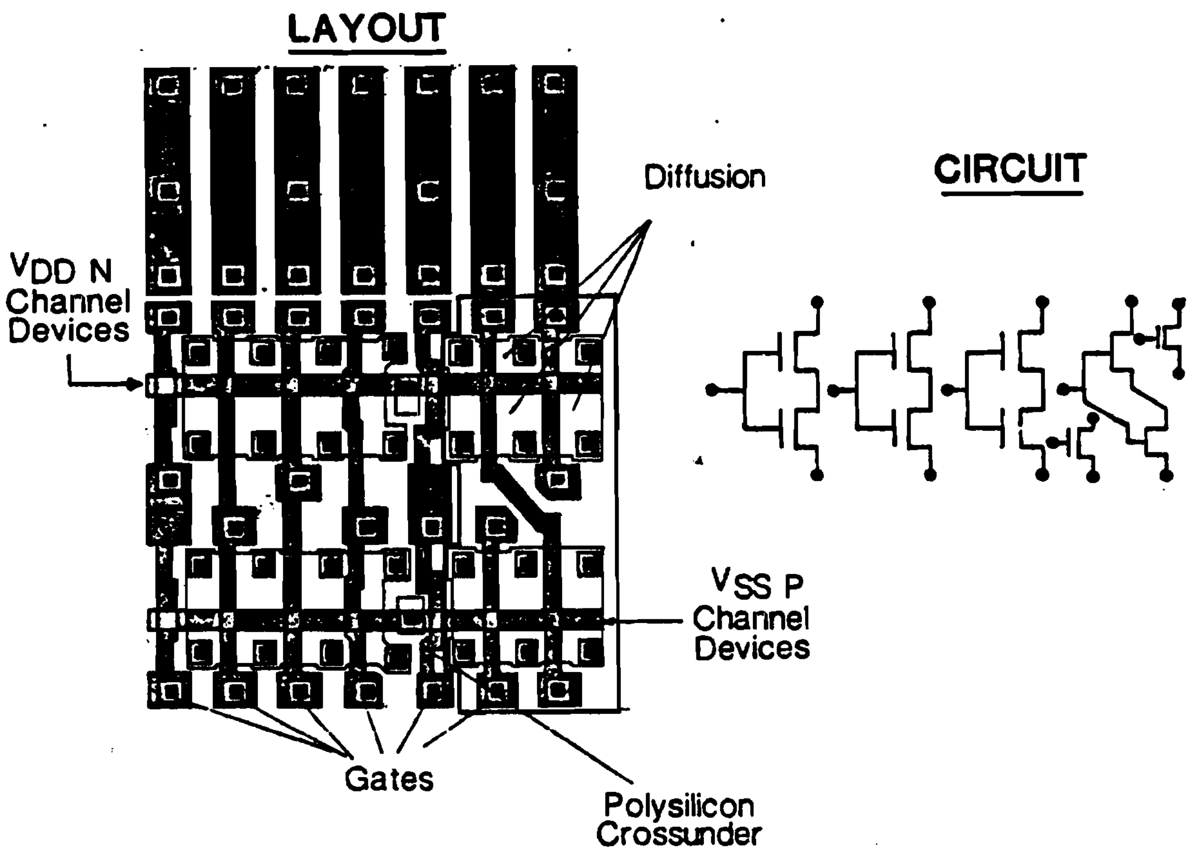

Fig. 2.8.: AMI 3/2 input CMOS cell, with cross-over gate.

The three and four transistor pair cells are sometimes configured as two separate two input gates. There differences from other cell types are commonly expressed in little changes of the routing area.

The silicon foundry 'Harris' uses a cell containing 6 transistors pairs, 4 having a common gate for each pair but the other having separate gates. The independent transistors available in this design allow increased design flexibility.

Cell configurations are also important in the routing phase of the gate array. Particular aspects are for example the possibility of the internal cell routing and the access to contacts from the routing channels.

Most of the cells are surrounded by polysilicon cross-unders to improve the routing in single metal systems.

An important question is how many cells are required for the implementation of logic functions. Table 2.1. presents typical values of required number of cells for some logical functions.

FUNCTION	NUMBER OF CELLS				
	2	2 / 3	3	4	TRANS.PAIRS
inverter	1/2	1/2	1/3	1/4	
2-input NOR	1	1	2/3	1/2	
3-input NOR	3/2	1	1	3/4	
D-flip-flop	5	2	4	2	

Table. 2.1.: Comparision of some gate array images.

2.5. GATE ARRAY VERSUS STANDARD CELL

The other main custom approach is the standard cell approach and differs from the gate array approach in relaying on a not fully processed chip. The standard cell approach involves the use of fully characterized standard circuit cell from a cell library and offers the advantage of proven circuit elements which require simply to be arranged and interconnected on silicon, in the same manner as laying-out a printed circuit board. One limitation of the standard cell approach is that the cells have a fixed positioning of the input and output pins. An advantage of the standard cell to the gate array approach is that the size of the interconnection channels between the rows of cells are not fixed in advance but they are flexible in width and so it is much easier to interconnect the chip. The standard cell approach and the gate array approach offer comparable costs and time-scale features. However, each has specific in that the gate offers very inexpensive commitment whilst standard cell designs the possibility of custom design space on the chip for specific hand-crafted functions.

The dividing lines between full custom, gate array and standard cell approaches tend to blur, since trends are towards standard interconnect arrangements to realize standard functional .cells for use on gate arrays.

o Only interconnect masks (metal) need to be designed, so that prototype time-scales and costs are reduced.

o Since only interconnect masks have to be designed and produced, there is a higher probability of obtaining a correct design first time.

- o The integrated circuit processing stage of metallization is less involved than any of the other processing stages, consisting, in the main, of etching away unwanted aluminium to leave the designed interconnect pattern. Thus a system house can have an effective full custom capability without the need for a fully processing capability.

Simplification of layout, with the possible use of standardized interconnection design, permits automatic routing software to be used for chip commitment. Use of standard functional cell types and standard cell arrangements for those functions together with a standard interconnection grid greatly simplifies the task of designing the required interconnect pattern on the gate array. Firstly all the available track paths are clearly defined by the standard interconnection grid and, secondly, the interconnection pattern is suited to autorouting. The design expertise required to customize the gate array is almost wholly confined to the initial stages of logic and circuit definition with the interconnect pattern being similar to the design of a printed circuit board, and no detailed knowledge of the integrated circuit being required.

The gate array can be regarded as a mask programmable component, or functional array, offering the fully capability of dedicated chip design in terms of incorporation on a single chip of combinational logic functions, sequential circuits and possibly analogue functions.

CHAPTER 3

PLACEMENT

Placement of gates on a gate array corresponds with finding a map of logical functions on a two dimensional array of basic cells such that after routing the interconnections, the realized functions will behave exactly the same as the functions that had to be realized. Therefore the most important requirement of a placement algorithm is that the chip, after placement, guarantees the routability otherwise the to be implemented functions cannot be realized in silicon. Thus, the silicon implementation, obtained by personalizing a gate array (placement and routing), has to reflect the functional requirements.

The placement of the gates must be optimized so that the routability and some other performance factors are guaranteed. This optimizing corresponds with minimizing an objective function (cost function) which contains a certain number of terms, providing a more or less fulfilling of following requirements:

- routability
- minimum total wire length
- minimum number vias
- minimum number feedthroughs
- minimum used area or maximal gate utilization

Automatically optimizing the placement of a large number of gates by using a combinatorial optimization can be a computational nightmare. For example, the performing what's called a "pairwise interchange" (interchanging two cells to reduce interconnect) on 1000 cells can result in 500,000 swaps and an analysis of routability must be made for each swap! So it needs no evident to simplify the optimizing objective function.

Another simplification of the placement optimization can be used, by clustering some cells together in order to reduce the required number of swaps. Therefore, gates are often clustered into highly interconnected groups (IBM calls these "supernodes"), and then these "supernodes" are optimally placed, again using pairwise interchanges and wirability analysis. "Macros" or "macrocells" are commonly used terms to describe clusters that form an identifiable function. Clearly, much care has to be taken about the generated optimum by using pairwise interchanges of macros, because it is not necessary that it is also an optimum of the placement if there were no macros used!

Macros are also used for manual placement to make the manual placement a little more comfortable. Often a designer want to use macros, also when he has automatic placement and routing programs available, to finish the wires the automatic programs couldn't route. In that case sometimes the designer replaces some macros (manually) and tries to complete the routing then.

So far we have assert that a logical function can only be mapped on a whole number of basic cells. This is simplification of the reality. The images of basic cells, consist mostly of a set of transistors connected together in some way. So it is possible that one basic cell contains more than one logic function. Thus, usually the above mentioned assertion will not be satisfied.

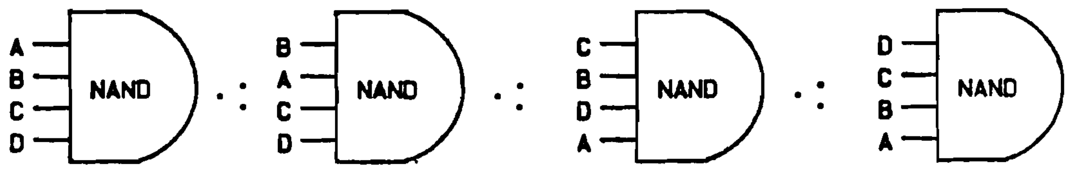

Fig. 3.1.: Example of logic equivalent pins.

Because the netlist (input data) is described in terms of logic functions, there must be an assignment step of logic functions to cells when a pairwise interchanging placement program is wanted to be use. Thus, there is a need of having a tool for this automatic assignment of logic functions to cells. This is a very difficult tool and step because of its technology dependence. Also this assignment step is not unique so it might lead to a sub-optimum of the objective function. It is very easy to show that this assignment step of the placement is not unique. Consider for example a 4 input NAND function, the order of the input pins is not determined or fixed, because of non existing geometric constraints in the logic function. So there are in this example 16 ways of mapping the logic function containing no geometric information to a basic cell.

Now we are able to give an overview of the placement functions:

- technology independent handling of gate array masters
- assignment of logic functions to physical cells
- exchangeability of logic functions within a cell
- exchangeability of logical equivalent pins within a logic function
- special handling of macros or "giant-cells"
- automatic placement of the input- and output cells (pads)
- restricted area concerning cells
- weighting of nets in order to cluster logic functions or cells

3.1. DIFFERENT PLACEMENT ALGORITHMS

There are two main placement methods: the constructive placement and the iterative placement. A constructive placement method consists of a heuristic algorithm which is often based on graph theoretical methods. The module placement algorithm involving with resistive network optimization [Cheng 84] is an example of a graph theoretically constructive placement.

The most famous and well known constructive placement algorithm is the min-cut algorithm first published by Lauther [Lauther 78]. The main strategy of this algorithm is that a module with the highest number of interconnections to an already placed set of modules (cluster) is placed as the next and it is placed as close as possible to the cluster, such that the interconnection length is minimized. This will be repeated until all modules were placed. Thus, this algorithm minimizes the interconnection length of one module and its environment but it does not give any guarantee of creating a final placement of all modules which has a minimum interconnection length. This disadvantage is often seen as the most important one.

Another large disadvantage of the min-cut algorithm is that the interconnections are clustered to the center of the chip which can lead to a significant area consumption necessary for the interconnections. Sometimes the area consumption can be decreased a little by designing very complex and very optimal channel routing algorithms.

Inherent to gate array design is the inflexibility towards all parameters and so the dimensions of the interconnection channels will be fixed. Therefore, the min-cut algorithm will be in general not applicable for gate array placement, due to the non uniformal distribution of the nets (interconnections) on the chip surface after the modules were placed.

The most well known optimization method in the class of combinatorial optimizations is certainly the 'Monte Carlo' optimization. Many derivates of this method were applied to placement optimization problems, although these applications had one main disadvantage. They cannot guarantee finding a global optimum. This optimization method consumes a very large amount of computation time. Therefore it is almost not applicable in iterative processes like chip placement. Another not less important aspect of a iterative placement optimization methods is that the quality of the resulting placement is very dependent of the initial placement.

In reference iterative placement is sometimes in literature represented as a Markov process. A new state will be generated by an exchange the positions of two modules on the chip with a certain perturbation probability. The transition probability from state i to state j will be determined by a perturbation probability for generating state j from state i and by the probability to accept state j if the system is in state i. In practice one obtains a Markov chain by repeatedly generating a new configuration the acceptance criteria. Optimization is performed by starting this chain-generation from an initial state (initial placement).

Placement and routing go hand in hand: The most careful placing is the surest path to creating a gate array that can be routed automatically. In other words, a chip whose elements are optimally placed stands the greatest chance of being optimally routed. Successful automatic placement is the basic ingredient for minimizing wire length, avoiding congestion, and making the fullest use of all available gates.

Gate array placement by simulated annealing

~~CHAPTER 4~~

~~SIMULATED ANNEALING.~~

R.J. Jonges
~~J.A.G. Jess~~

Introduction

Simulated Annealing is a stochastic optimization method. This method is based on the existence of a connection between combinatorial optimization and statistical mechanics. The framework for this optimization is provided by a detailed analogy with annealing in solids. ~~Especially when a system of which the cost function has to be optimized, has the properties of very large and complex systems,~~ Simulated annealing is particularly useful as an optimization method. An algorithm is used for appropriate numerical simulation of the behaviour of a many-body system at a finite temperature. This algorithm provides a natural tool for bringing the techniques of statistical mechanics to bear on optimization.

The model in the statistical mechanics, which is, used as an, analogue for optimization, consists of a mixture of a number of inhomogeneous liquids. This mixture has to be cooled (annealed) until it becomes a mono crystalline structure, ~~with the property being in the~~ associated with a ground state of matter. With other words a single crystal is growing from a melt by lowering the temperature. To lower the temperature is not a sufficient condition for finding the ground states of matter. The procedure which on the other hand guarantees the sufficient condition consists of careful annealing, first melting the substances, then lowering the temperature slowly and spending a long time at temperatures in the vicinity of the freezing point. If this procedure is not exactly followed, for example by lowering the temperature not slowly enough, the substance ~~could be allowed to~~ may get out of equilibrium. The resulting crystal will have many defects, or the substance may form a glass with no crystalline order and only metastable, locally optimal structures.

~~4.1~~. THE ANALOGUE.

In the section above we already talked about the physical model consisting of a mixture of inhomogeneous liquids, which will be used as an example for showing the annealing process in ~~the~~ statistical mechanics, with a On the other hand we have the gate array system ~~which~~ placement cost function that has to be optimized by an ~~combinatorial~~ optimization method (simulated annealing). In the following it will be shown how the physical model can be used as an analogue for the placement of gate arrays.

In the physical model we know the following terms: atoms, movement and collision of atoms, temperature, energy of the system, cooling and heating, single crystal and glass. These terms can be compared with the terms of the gate array placement model~~, whereby the physical model acts as an analogue~~.

Fig. 4.1.: Comparision of annealing in solids and simulated annealing.

There is ~~only one big problem left, a~~ a certain disagreement between the placement and the mixture of liquids. The movement and collision is a typical parallel process i.e. at the same time many (almost all) atoms or molecules are moving through the system. The interchanges of ~~the~~ modules ~~can never be a parallel process~~. ~~Therefore~~ Mostly however, the parallel motion of the atoms ~~must be~~ is simulated by ~~a typical sequential~~ interchanging ~~of~~ two random modules. This simulation ~~accomplishes with comparing~~ associates one time step in the physical model with a fixed number of interchanges. The consequence is that the control function which is time dependent has to be adapted to this change of the time variable. All this will have the effect that the time variable in both cases is equal to a certain number of movements or interchanges. The movement of an atom and the collision of two atoms is simulated by an interchange of two, randomly chosen modules of the gate array.

are generally done sequentially. Only recently parallel organization schemes have been developed for simulated annealing.

~~4.2.~~ METROPOLIS

Iterative improvement, commonly applied to ~~combinatorial~~ optimization problems, is much like the microscopic rearrangement processes modeled by statistical mechanics, with the cost function playing the role of energy. However, only accepting rearrangements which lower the cost function of the system is like extremely rapid quenching from high temperatures to the temperature zero. Thus ~~so~~ it should not be surprising that resulting solutions are usually metastable. The Metropolis procedure derived from statistical mechanics provides a generalization of iterative improvement in which controlled uphill steps can also be incorporated in the search for a better solution.

Metropolis in the earliest days of scientific computing introduced a simple algorithm which can be used to provide an efficient simulation of a collection of atoms in equilibrium at a given temperature. In each step of this algorithm, an atom is given a small random displacement, and the resulting change, ΔE (delta energy), in the energy of the system is computed. If ΔE <= 0, the displacement is accepted, and the configuration with the displaced atom is used as the starting point of the next step. The case ΔE > 0 is treated probabilistically: the probability that the configuration is accepted is

$$P(\Delta E) = \exp(-\Delta E / kT).$$

Random numbers uniformly distributed in the interval (0,1) are a convenient means of implementing the random part of the algorithm. One such number is selected and compared with P(ΔE). If it is less than p(ΔE), the new configuration is retained, if not, the original configuration is used to start the next step. ~~This choice of P(ΔE) has the consequence that the system evolves into a Boltzmann distributio~~n.

~~4.3.~~ INITIALIZATION

The annealing algorithm ~~needs to be initia~~lized. The initialization phase assigns ~~gives to~~ the ~~temperature~~ an initial value and prepares, the initial placement. ~~which~~ This is read from the description the user has defined ~~to the placement~~ ~~that~~ and ~~is called the maximal necessary disorder placement~~.

Before the annealing phase can start one has to be sure that all substances are in a dissolvable condition in the mixture of the analogue annealing model. This means that the placement has to grow worse, i.e. the cells have to be mixed so that the quality of the placement will decrease. The decreasing of the quality of the placement is called the heating phase because the temperature of the system increases during this phase. At one point of the heating phase the increase of the temperature is stopped and the placement created corresponds with the maximal necessary disorder placement. The maximal necessary disorder placement corresponds with the

is "maximal" your keyword!

→ maximally

mixture containing all substances in a dissolvable condition. There is only one problem: we don't know when the point the heating phase ends is reached exactly. In practice heating is stopped when more than 99.5 percent of the moves are accepted. The end of the heating process defines the starting temperature.
~~Although we have introduced the heating phase to get a correct starting condition for the annealing algorithm, we still need a starting temperature. The determination of the starting temperature has become less important since the heating phase was introduced for this phase involves the adjustment of the temperature if it was chosen totally out of range.~~

The determination of the starting temperature ~~and~~ us the heating phase is important to guarantee the algorithm ends with an optimal placement ~~as fast as possible~~. If the reached placement after the heating phase is ~~worse~~ better than the placement corresponding with the maximal necessary disorder, we have to anneal until this placement is reached. This means a lot of computation time wasted. On the other hand, if we don't heat enough, the annealing algorithm cannot find the optimal placement.

The starting temperature is determined by the mean absolute value of the delta-energies of a certain number of moves. This corresponds with the temperature being calculated from the fact that the exponent of the Boltzmann equation is set to one. With this method we can come fast and with a little effort to the staring point of the annealing phase.

4.4. THE CONTROL FUNCTION. (controlling the temperature)

The ~~control~~ function is almost the most important part of this optimization approach. When the temperature lowers too fast the system will end in a glass, and when on the other hand the temperature lowers very slow the system will become a single crystal but the computation time to run the optimization process will be immensely huge ~~and wasted~~. One thing we know is that when we cool infinitely slow the probability of reaching the global minimum of the energy function is equal to one!

Our aim is to find a control function in such a way that we know with a certain confidence that the global minimum of the cost function will be reached and the costs of this optimization process in terms of computation time will be minimal. Thus, the problem of finding a suitable control function is transformed in the problem of finding the fastest temperature lowering scheme such that reaching the global optimum is guaranteed.

In the gate array placement model there is no time variable. Thus ~~so~~ when we want to use a time dependent temperature scheme we have two ~~not~~ un determined variables. We can change the number of interchanges or moves per temperature step and we can change the temperature

At the time we performed our experiments no theoretical results about the temperature lowering schemes were available. Some results are available now [Otten 85]. We tried to find an appropriate scheme experimentally.

step size.

~~It is not possible yet to find the suitable temperature lowering scheme in theory. Therefore we must try to find it by doing some experiments. For one personalization of a gate array this is not very hard to do, but it is very difficult or even impossible to find a lowering scheme this way that will be suitable for every personalization.~~ For this task ~~I~~ we designed a test example that emulates the behavior of a gate array. The fact that the optimal placement is known is what makes this test example special.

The lowering temperature lowering scheme ~~I~~ we found is explained next.

When the number of cells of a gate array is equal to n then the maximum number of moves at one temperature will be 100n and the maximum number of accepted moves (interchanges) will be 10n. These two figures are to be considered as stop conditions for one temperature step i.e. if one of these conditions is reached a new temperature is calculated.

Fig. 4.2: The course of the acceptance level in function of the temperature.

For each temperature the ratio of the number of accepted moves and the total number of moves is calculated, ~~and I~~ We call~~ed~~ this ratio the "acceptance" we compare

as a function of "acceptance ratio" or level". When t~~he course of~~ the acceptation ratio and the energy of the system ~~in~~ temperature a~~re viewed~~ we see globally the same ~~course~~ except ~~but~~ that the acceptance ratio ~~one is~~ turns much smoother than the ~~one of the~~ energy. This means that we can use the acceptance ratio for controlling the temperature of the system.

shape

The controlling of the temperature ~~occurs~~ we choose to be piece wise linear ~~in~~. The temperature lowering step changes at certain points of the acceptance ratio curve, the break points. The determination of the breakpoints depends on the ~~course~~ values of the acceptance level. For ~~the~~ our scheme ~~I found,~~ ~~I~~ we used only two breakpoints ~~(I thought that it was not relevant to choose more breakpoints),~~ one at the acceptance level of 90% and the other one at 10%. These two breakpoints are chosen ~~that~~ this way because the ~~course~~ shape of the acceptance ratio curve suggest ~~is globally to be divided in~~ three different regions. The ~~middle one of these three~~ central region is the most important one because the energy decreases ~~in~~ there very ~~much~~ strongly. Therefore ~~I~~ we chose the ~~following~~ temperature steps in the three regions to be 0.7 in the upper region, 0.9 in the ~~mid~~ central region and again 0.7 in the low~~est~~ ~~acceptance~~ temperature region.

The last thing to fix are the ~~There is only one thing left to say: that is the choice of the~~ stop conditions of the annealing algorithm. ~~which also belongs to the controlling scheme~~. One stop condition is reached if the acceptance level becomes lower than a certain value. The value of the lowest level ~~I~~ we chose is equal to 0.5%. Necessary is also another stop condition. Consider the case that a system has a close clustered configuration which may lead to the fact that the previous stop condition is never reached. In that case we ~~want to~~ need introduce a~~nother~~ stop condition to guarantee the annealing algorithm will always ~~converge~~ finish. This stop condition is given by ~~consists~~ of the maximum number of times the maximum number of moves is exceeded; it is called the freeze condition.

With this scheme ~~I~~ we found the optimal placement of the test example rather fast. ~~I~~ we also used it in some other examples, and it seems to be quite useful in the future as an overall scheme.

the ?

~~CHAPTER 5~~

THE CHOICE OF THE MODEL

simplify the

~~In chapter 3, we have already mentioned that~~ The placement will be a computational nightmare if we do not ~~make some simplifications on~~ modeli~~ng~~. ~~This causes the introduction of a simplified model of the placement which behaviour has to be similar to the original placement model.~~

~~The simplified model is necessary for reducing the quantity of calculations on every update of the placement. The saving of the number of calculations, needed for qualifying the placement program, is one of the most important features of the simplified model. The other function of the simplified model necessary to guarantee afterwards of having the optimal placement, with a minimum effort of computation time, are:~~

In principle the simplified model must meet the following conditions:

- o an update of the placement must be very easy and cost just a few calculations.

- o The minimum of the cost function of the simplified model has to coincide with the optimal placement of the gate array.

The gate array will be considered as a two dimensional array of cells. All cells are equal in dimensions and complete~~e~~ly interchangeable, so they will be considered as points. This is the first simplification. Normally, the cells contains two rows of pins to be connected, one on the top-side and one at the bottom-side, ~~wherein~~ Each pin is connected to one or no net. ~~but~~ When the cells become points the pins will be collapse together to one pin (in the center of the point) connected to a certain number of nets. According to this simplification we ~~lost~~ lose ? the information of pin order and to which pin each net is connected.

~~Remark~~ Note that it is self-evident that ~~the~~ in our model the total "move-space" has to be accessible. For our application this means that every interchange of every two cells must be possible.

The placement of a gate array involves with ~~an already mentioned~~ a cost function (also called object function) which ~~have~~ has to be minimized to a global minimum corresponding to the optimal placement of the persionalized gate array. When making no simplifications the cost function consists of the total wire length of ~~each~~ all net~~s~~s. It is very easy to see that the calculation of the total wire length after every interchange of two cells will be very tedious ~~turn the solution of the placement problem into a computational nightmare~~. Therefore, we try to make-up a cost function which is very easy to update after an interchange of cells.

~~First~~ ~~I~~ we will summarize ~~again~~ the terms contained in the object function:

- routability (uniform distribution of nets)
- total wire length
- number vias
- number feedthroughs

Now we show

~~Then~~ the structure of the simplified cost function ~~is shown~~. The cost function ~~consists of~~ the total sum of the perimeters of the surrounding boxes of all nets. The surrounding box of a net is introduced to ~~get very~~ to obtain in a simple way ~~simply~~ an idea of the length of a particular net. To get to know the length of a net is a great problem because it is not routed yet. ~~Therefore we don't know how this net is situating exactly, because it depends on the routing~~. When we introduce the surrounding box of a net we know that the net must be ~~in there~~ enclosed independent of the routing of the net. If the surrounding box of a net is cut by a cross-line we know for sure that the net is cut at least once ~~time~~ by that ~~cross~~-line. Thus, the surrounding box is a ~~degree~~ measure for the total wire length. ~~if the net is routed. true~~ Certainly it is a lower bound for it.

The gate array is constructed of a number of columns and rows, ~~calling it~~ the grid of the gate array. The smallest grid unit is ~~exactly~~ equal to the dimensions of the smallest possible basic cell. All dimensions and measures in this report will be expressed in terms of ~~these~~ grid units. For example, a 10x12 gate array ~~proposed~~ ~~as~~ is a matrix of 12 rows containing 10 cells each.

To estimate the total wire length, ~~necessary for the cost function of the simplified model,~~ the total chip is cut at each grid coordinate in both the x- and y-direction. At every cut the number of the surrounding-boxes that is hit or cut by the cut-line is determined. If we ~~would~~ add these numbers ~~of hidden cell~~s for all cut-lines we have ~~created~~ an estimat~~ion~~ of the total wire length. This total sum is equal to ~~the~~ half of the sum of the perimeters of all surrounding boxes. It is also possible to express a preference of a certain kind ~~of~~ for the form of the surrounding boxes in the cost function. This is useful if the

Wirelength is however not a measure sufficient to evaluate the quality of a gate array design.

routing of a particular net is more difficult in one direction ~~as~~ than in the other direction. For example it is more favourable to route a net in the "channels" than across the rows of cells, which is only allowed at feedthrough spacings. ~~when there is only one metallization mask used.~~ ~~To give a certain preference of the nets in one particular direction and to have a factor for the uniform distribution of the cut surrounding boxes, received for every cut-line, but~~ We ~~will~~ display the cut-count ~~them~~ in ~~to~~ histograms ~~each~~ for one each direction. ?

Now the cost function can be easily constructed ~~with the help of~~ using these two histograms. The area of such a histogram gives an ~~expression~~ estimate of the total wire length ~~that will be used~~ in one direction. ~~Normally, this should be enough~~ precise ~~for constructing the objective function, provided that the optimization algorithm is not used for gate arrays. Because of the fixed and preprogrammed form of gate arrays it is not sufficient to have only one term, which expressed the total wire length, in the objective function.~~ There is also a need for providing an expression ~~of the uniformity~~ accounting for the inhomogeneity of the distribution of the nets ~~at chip surface~~ ~~in the objective function~~. This expression will be given by the maximum column height of the two histograms. The next example will show this more in detail. Consider a personalized gate array with the following histogram:

Fig. 5.1.: Badly placed chip.

The associated total area of that histogram is rather small, although the ~~belonging~~ placement is very bad, because of the high density of the nets at one place of the chip. This is for gate arrays very unconvenient since the routability can ~~be~~ lost with the consequence that the placement becomes worthless. ~~However~~ Therefore, the cost function we will use to optimize the placement by simulated annealing is defined by:

$$f = w_{hs} \sum_{Col} H_h + w_{hm}\, MAX(H_h) + w_{vs} \sum_{Row} H_v + w_{vm}\, MAX(H_v)$$

where

whs :	Horizontal sum weight factor.	w \ dh \ us
whm :	Horizontal max weight factor.	w \ dh \ mm
wvs :	Vertical sum weight factor.	w \ dv \ us
wvm :	Vertical max weight factor.	w \ dv / mm

The weight factors are meant ~~for~~ to control~~ling~~e ~~and observing~~ the impact of the terms of the cost function on the performance of the ~~total~~ placement. ~~Not the individual values of the weight factors is important but~~ their mutual ratio of the weight factors is ~~are important~~. ~~The ratios of the weight factors whs/whm and wvs/wvm express the ratio of the influence of the two terms of cost function: the total wire length and the uniformity of the nets.~~ The ratios ~~of~~ whm/wvm and whs/wvs gives a ~~expression~~ measure of the preference for the vertical or horizontal direction.

rep wely appearing

Note that the magnitudes of the 4 terms ~~used~~ in the cost function are totally different. This difference can be explained from the fact that the "sum" terms are formed by the sum of all columns of the histograms compared with the "max" terms that are constructed by the size of only one column. One has to correct this by adapting the weight factors in a way that the magnitudes of the terms become ~~of~~ about the same size.

~~CHAPTER 6~~

MACRO PLACEMENT

In the previous chapter we have discussed the simplified model necessary for the annealing algorithm. This model only handles basic cells, which are all the same.

Sometimes the design houses want to have a macro placement program, because until now they placed their gate arrays manually, with the aid of macro's. The advantage of handling with macros in contrast with basic cells is that the number of modules to place is much smaller. Therefore the interconnection is much simpler. For example when we want to use a flip-flop, it has to be designed only once, and it can be handled as one module with the internal cell (inside the module) interconnections separated from the external cell (outside the module) space.

The problem of using the annealing algorithm for macros is that they have not the same dimensions, which make the interchange of two macros very complicated. Before I shall explain the procedure of macro interchanging two assertions have to be made. One assertion is that the macro cells are only one dimensional and all macros lie at the same direction (horizontal or vertical). The second assertion consists of the fact that all macro dimensions are equal to a natural number of grid units. Where the grid unit is equal to the dimension of a single basic cell.

6.1. MACRO INTERCHANGING PROCEDURE

Since we still use the same grid as for the basic cell placement, we determine two cells randomly on the chip. They are both member of different macro cells. Note that a single basic cell is considered also as a macro cell containing only one cell. The largest macro will be the dominant macro. The other macro has to be extended on a way that can be interchanged with dominant macro. To make this interchanging possible, the extended macro (cluster) has to satisfy the condition that the length of it is equal or larger than the dominant's macro length. Even if the cluster is larger then another extra condition must be satisfied: the macros member of the cluster that do not fit in the space of the dominant macro, have to fit in the free spaces distributed over the chip. When it is not possible to satisfy this last condition, the macro interchanging procedure stops this attempt of interchanging and starts from the beginning with two new randomly determined cells.

The smallest macro has to be extended to form a cluster of macros. First look wether there exists a macro on the lefthand-side of the already determined cluster (initially they cluster contains only the non-dominant macro). If it does, add that macro to the cluster. The action done for the lefthand-side of the cluster can also be repeated, if necessary, for the righthand-side. Of course, these actions need only be repeated if the length of the cluster is still not big enough.

Until now we have got two modules (one contains the dominant one) intended to be interchanged. The dominant macro will be replaced at the most left position of the cluster on the chip. The space, of the cluster, left will be added to the freecell list, containing all free spaces of the chip. The macros contained in the cluster have to be placed in the space that became free by replacing the dominant macro to the cluster position. We start with the biggest macro and put it in the most left position of the dominant macro free space, if it fits. Thereafter the biggest but one is replaced and so on. When one macro of the cluster doesn't fit in the dominant macro free space we will try to put it in another free space elsewhere on the chip, if it exists.

It might possible that some macros won't fit in any allowed position at the chip surface. In this case this attempt of interchanging two modules is interrupted and a whole new attempt of two other modules is started.

Summarizing we can say that the macro placement algorithm only deals with one dimensional macros on a two dimensional grid. I think that it is not possible to design an efficient macro interchanging procedure, which has no restrictions to the dimensions or to the degrees of freedom. Therefore I have invented another approach to place macro cells. This approach introduces some extra (strong) forces between the member of cells of a macro and it considers the macros consisting of a whole number of independent basic cells. These intra macro forces have to be controlled during the annealing, i.e., with the decrease of the temperature the forces become more and more dominant. The advantage of this approach is that there are not any restrictions to the "macro" interchanging procedure but the big disadvantage is that the number of independencies is equal to the number of gates on the gate array in contrast with the number of macros, which is of course much smaller. This has an impact on the computation time of the annealing algorithm.

~~CHAPTER 7~~

IMPLEMENTATION ASPECTS

Conform the number of annealing atomic actions necessary for one run, the total run time will be the bottle-neck in the implementation of this optimization algorithm. It will be clear that the run time optimization of the atomic action has got the greatest attention during the implementation phase of the annealing algorithm.

PASCAL was used for the implementation, because on the moment the annealing research on ~~the application~~ of gate array placement was started the language C was not available.(C was the language chosen ~~by~~ for the NELSIS project.)

~~The program was in first ~~instance~~ place designed for development purposes with portability in mind. Therefore, stepwise refinement -- of course -- was used to manage the complexity during implementation [Wirth 71].~~

~~7.1~~ DATA STRUCTURE

Good runtimes are achieved by carefully designed ~~One of the most important run-time improvement has its base in the~~ data structures. ~~Therefore the design of the data structure has to be well considered. Hence, a data structure design will determine the main efficiency of an implementation of an algorithm, accomplishing with the amount of computation time necessary for an optimization job.~~ In our case the most obvious data structure design would ~~consist~~ be by ~~of~~ linked lists: one per net, containing the connected modules (cells) and one per cell, containing the ~~to itself~~ connected nets. The advantage of this approach is the dynamic memory ~~necessary~~ for the storage of all relevant information of the nets and modules, but its big disadvantage is the limited accessibility which results in a time consuming process when records are not directly accessible. ~~It is therefore very easy to see that there exists a trade-off between memory and speed when you design a data structure (this trade-off exists not in this particularly design only, it exists always in every design).~~

Insofar as we consider storage time tradeoffs for the annealing ~~Sofar a data structure design with less memory is shown, but for the annealing it is very useful to use a speed optimized data structure design in spite of an increasing memory use. Fortunately the to be~~ we rather trade time for memory than the other way around.

Fortunately memory is not a problem in the implementation of annealing, even though ~~occupying memory space will not lead to one of the bottle necks in the annealing algorithm. Although~~ the number of cells, nets and modules grow up to several of thousands if VLSI circuits will be used.

Consider the

~~For~~ example of a 5000 gates ~~gate~~ array with 3000 nets and ~~the~~ maximally 20 ~~maximum number of~~ cells connected to one net ~~is 20~~, and each cell has 8 terminals. Then the data structure needs 5000*8+3000*20=100,000 records. So the whole data structure will claim about one million bytes supposing that each record is 10 bytes long. The memory required by using such structure is still ~~quiete~~ quite acceptable.

All these considerations will point to the rather obvious use of ~~static~~ arrays (PASCAL) for the storage of the data of the data structure. ~~The more as we intend to use the PASCAL language for implementing this annealing algorithm.~~ An array structure has the important characteristic that all elements are directly and separately accessible.

There exists also which must not be time consuming the problem of inserting and deleting one element ~~without being a very time consuming process~~. The solution to this problem, for the annealing algorithm ~~has its base in~~ was the fact that the number of elements in the "lists" ~~are~~ is constant. This means that if an element has to be deleted another ~~or~~ the same element will be inserted ~~then. Of course in our case the two elements are the same, determined by the interconnection lists (netlists), only the position data will be different. Thus, deleting or inserting elements in the arrays correspond with shifting some elements one position to the "right" or to the "left". These actions are necessary when we assume that the lists are sorted on the position data of the elements.~~

~~The data structure has store some data store twice for creating a very access of the necessary information~~. The net records contain all cells connected by the net and the cell records contain all nets connecting to the terminals of that cell. ~~This structure is derived from the fact~~ That is because ~~one has~~ to know ~~s~~ the nets which are connected to a ~~the~~ moving cell. Surrounding rectangles must be determined for each of the connecting nets individually. For the determination of that surrounding rectangle the position of all terminals (cells) has to be known. When all terminal positions are known the surrounding rectangle can be determined by finding the minimum and maximum coordinates of the terminals. ~~Of course~~ This is a time consuming process. To improve the performance of the ~~annealing~~ implementation ~~(time consumption)~~, one ~~have~~ has to find a solution for a very fast calculation of a surrounding rectangle of a net.

The solution was found by using sorted arrays for the x-coordinates and for the y-coordinates of the terminals of each net. Now the minimum and the maximum coordinates are resp. the first and last element of the sorted arrays. Note that for this operation we claim that all these arrays must be sorted. Sorting arrays is a ~~very~~ difficult action to do ~~it efficient~~, but in our case it can be very easy. Because of the number of elements being constant and of the fact that only one element is changing position ~~in that array~~ at the same time, sorting is accomplished ~~with~~ by shifting some elements to the right or left.

The introduction of arrays for the implementation of the netlists has caused a speeding-up of a factor 2 or 3. The algorithm is now about 2 or 3 times faster than it would be if it had used linked lists. The speeding-up factor is of course dependent of the size of the array and the complexity of the netlists. For large arrays the speeding-up will be even larger than 2 or 3. ? on
Fig. 1 displays the data structures in PASCAL notation.

```
TYPE
     gate_range          = 0 .. max_num_of_gates;
     column_range        = 0 .. max_num_of_columns;
     row_range           = 0 .. max_num_of_rows;
     cell_range          = 0 .. max_num_of_cells;
     net_range           = 0 .. max_num_of_nets;
     cell_per_net_range  = 0 .. max_cell_per_nets;    nr 1 ?

     netlist_point  = ^netlist_record;
     netlist_record =
       RECORD
         netnum : net_range;
         next   : netlist_point;
       END;

     { "MODREC "}
     cell_record =
       RECORD
         net_list  : netlist_point; { pointer to head of netlist       }
         colnr     : column_range;  { column index of cell             }
         rownr     : row_range;     { row index of cell                }
         length    : integer;       { length of macro, containing cell}
         leftmost  : column_range;  { leftmost column index of macro   }
         rightmost : column_range;  { rightmost column index of macro  }
       END;
     cell_array = ARRAY[ cell_range ] OF cell_record;

     sort_array = ARRAY[ cell_per_net_range ] OF cell_number;

     { "NETREC" }
     net_record =
       RECORD
         xmod     : sort_array;          { cell list on x-coord. order  }
         ymod     : sort_array;          { cell list on y-coord. order  }
         lastmod  : cell_per_net_range;  { number of cells in cell list }
         x_weight : integer;             { weight factor in x direction }
         y_weight : integer;             { weight factor in y direction }
       END;
     net_array       = ARRAY[ net_range ] OF net_record;
```

Fig. ~~AA~~ 1.: Data structure in PASCAL.

~~7~~ PROGRAM

A distintion has to be made between a move and an interchange of cells. The change of energy -- delta energy -- is necessary for the determination of the acceptance of such an interchange. When an interchange is not accepted, the available energy and its corresponding interchange must be undone. This results in a placement ~~system~~ which is exactly equal to the previous placement ~~old system~~ ~~(previous one)~~. To avoid~~ing~~ the undoing of interchanges the moves were introduced. A move exchanges two cells virtually so as to be able to calculate the change of energy implied ~~if they will be~~ by the interchange~~d~~. Now when a move is not accepted nothing has to be done to get the old system configuration back. ~~because the data structure has not changed.~~ This leads to a reduction of the cpu-time with a factor of almost 2. Therefore introduction of moves is very useful the more because the total number of not accepted moves is at least two times larger than the number of accepted moves (interchanges). These are results from experiments. Therefore the benefit of decreasing the computation time ~~consumption~~ of a move is rather large ~~huge~~. This benefit is also very profitable since the time ~~consumption~~ necessary for accepted moves is totally waisted because it contributes nothing to the improvement of the system configuration, ~~only accepted ones do.~~

~~With introduceing the moves a new problem occurs. On the chip an interchange of two cells is a parallel action, cell A goes to the position of cell B and concurrently cell B goes to the position of cell A. Since this interchanging action is emulated by a computer (sequential machine) this parallel action is divided in a number of sequential actions in a certain order, to ensure the same result. The problem now is that the moves are sequential actions, and they will not change the data structure. So when one net connects to cell A as well as to cell B then the calculation of the surrounding box if the cells were interchanged will go wrong when it is calculated like the other nets.~~

The data structure design of the simulated annealing program is the most important part of the implementation concerning the speed optimization. Therefore, a lot of effort has been supplied for the database design, which has lead to an efficient implementation of the annealing algorithm. The implementation consists of a rather simple program controlling the "moves", "energy" and "temperature". During the designing phase a great emphasis was put on the "Stepwise refinement" method elaborating the rudely specified statements.

```
PROCEDURE SIMULATED ANNEALING;

{ stop_melt_acc ::= upper bound # accepted moves during melting.       }
{ stop_melt_att ::= upper bound # moves during melting.                }
{ stop_ann_acc ::= upper bound # accepted moves during annealing.      }
{ stop_ann_att ::= upper bound # moves during annealing.               }
{ accept        ::= indication for successful termination of
                    annealing at current temperature.                  }
{ melt          ::= increases temperature according to schedule.       }
{ anneal        ::= decreases temperature according to schedule.       }
{ temp_not_acc  ::= counts the # of times that annealing was stopped
                    by passing the upper bound on the total # of
                    moves per temperature step.                        }
{ limit_count   ::= upper bound of temp_not_acc.                       }
{ limit_level   ::= lower bound on acceptance level.                   }


BEGIN
  initialize surrounding rectangles;
  calculate initial energy;
  determine initial temperature;

  { heating }
  WHILE NOT accept DO
    BEGIN
      melt;
      metropolis( stop_melt_acc, stop_melt_att, accept );
    END;

  { cooling }
  REPEAT
    anneal;
    metropolis( stop_ann_acc, stop_ann_att, accept );
    IF NOT accept THEN temp_not_acc := temp_not_acc + 1;
  UNTIL ( temp_not_acc > limit_count ) OR
        ( accept_level < limit_level );

  print placement;

END; { simulated annealing }
```

Fig. 2.: Pseudo code description of the annealing program.

The routines MELT and ANNEAL, increase~~s~~ and decrease~~s~~ the temperature by a step according to the temperature schedule. They return a new temperature, which will be used for the Metropolis algorithm to do some interchanges. The Metropolis algorithm determines if the delta-energy of a move belongs to an accepted move or to a rejected one. This Metropolis procedure actually implements the Metropolis algorithm, which emulates the behaviour of the Boltzmann equation. The pseudo code description is shown next.

```
PROCEDURE METROPOLIS( stop_acc, stop_att, accept );

{ count_acc ::= counts # accepted moves.              }
{ count_att ::= counts total # moves.                 }
{ stop_acc  ::= upper bound for # accepted moves.     }
{ stop_att  ::= upper bound for total # moves.        }
{ accept    ::= indication for successful termination of
                annealing at current temperature.      }

BEGIN { metropolis }
  REPEAT
    pick two cells intended for interchanging at random;
    calculate delta-energy;

    { test move can be accepted }
    IF delta_energy < 0 THEN
      BEGIN
        energy := energy + delta_energy;
        effectuate interchange of the cells;
        count_acc := count_acc + 1;
      END
    ELSE
      IF RANDOM < EXP( - delta_energy / temperature ) THEN
        BEGIN
          energy := energy + delta_energy;
          effectuate interchange of the cells;
          count_acc := count_acc + 1;
        END;

    count_att := count_att + 1;

  UNTIL ( count_acc > stop_acc ) OR
        ( count_att > stop_att );

  IF ( count_acc > stop_acc ) THEN
    accept := TRUE;

END; { metropolis }
```

geen ; achter END ? →

Fig. 7.3.: Pseudo code of the Metropolis algorithm.

~~CHAPTER 5~~

EXPERIMENTS

~~Experimental results are very important in the annealing strategy, because of, the already mentioned, lack of theoretical foundation. Therefore, some experiments have to be examined to collect the necessary information for adjusting a suitable overall temperature lowering scheme. From the experiments some experience in the course of the validation of the placement can also be extracted.~~

To determine the best temperature lowering scheme, one ~~have~~ has to know what is the best possible placement, and in what time can it be generated. Therefore, the "final energy" and the computation time ~~consumption~~ for putting the system in the state with that energy are ? in first ~~stage~~ place important for ~~qualify~~ evaluate the scheme. Of course this evaluating collected experience is of general importance and not particular for this application of the annealing algorithm. To increase the insight in what is the best lowering scheme for annealing with gate arrays, some additional information is necessary. In that case not only the ~~amount of~~ final energy ~~is~~ must be satisfying but also some ~~marks~~ figures for the routability are required. The routability factor is very difficult to determine without a routing program. In the examined experiments the routability factor was determined by observing a figure which expresses the net density.

The experiments were done on four different gate arrays:

- o The "RANDOM" gate array.
- o The "CHESS-BOARD" gate array.
- o The "BARS" gate array.
- o The "TEXAS" gate array.

8.1 "RANDOM" GATE ARRAY

The "RANDOM" array consists of 12 rows and 18 columns, and contains 180 occupied cells located at 216 possible positions. This corresponds with an array filling of about 84%. The name of the gate array results from the fact that the personalization of the array is based on a random netlist. The netlist, which contains 100 nets, is generated in a random ~~ly~~ manner but not total~~y~~ly arbitrar~~y.~~ily. On the construction of the netlist there were some constraints ~~[illegible]~~. ~~These constraints are~~: They were

o a net may contain only 2 to 10 cells to which it is connected.

o a cell may only be connected to 4-12 nets.

Of course these constraints were defined empiric ally ~~for~~ creating a netlist ~~which will estimate the practice the best.~~ matching practical cases.
In the first ~~instance~~ place this gate array was designed for testing and verifying the implemented annealing algorithm, and secondly it has also been used ~~for~~ to adjusting or determine a suitable temperature lowering scheme. Adjust

Fig. 8.1.: The start (a) and end (b) configuration of the "RANDOM" array.

Due to the random character of the netlist, the annealing does not converge, the acceptance ratio ~~has~~ reached the lower bound which is about 3%. ~~This provides in a detailed study of the netlist.~~ When we
A study of the netlist provides an explanation of this phenomenon.

this phenomenon

look ~~to~~ at the netlist we see one big "sphere" of elements with strong cohesion forces (during the experiments we have called ~~it~~ the "chewing-gum" effect), I.e. every cell in the "sphere" has an equal connectivity (force) to all other cells in that "sphere".

Due to this strong connectivity effect it will be impossible to place the array in a way that the connectivity is uniquely distributed ~~with~~ and at the same time exhibits a minimum interconnection length. In the center of the array there always will be a congestion of ~~the~~ interconnect~~ion~~. This phenomenon ~~is to be seen~~ appears in the histograms of the final placement (Fig. 8.1.b).

8.2. "CHESS-BOARD" GATE ARRAY

These series of experiments were of highest importance since they were used as reference for the determination and adjustment of the temperature cooling scheme. They involve ~~with~~ the placement of the so called "CHESS-BOARD" gate array, which is an array of 8 by 8 cells with a netlist such that the best (minimal) placement is known before hand. Each net of the netlist connects to four neighbour cells ~~together~~. Hence, the "CHESS-BOARD" consists of 64 cells and 49 nets.

Fig. 8.2.: Net connection.

? deteriorations

The annealing algorithm belongs to the class of combinatorial optimization that are independent of the start configuration, since in the initial phase of the annealing almost an unlimited number of deterioations are allowed by an increasing temperature until the point of maximal necessary disorder is reached. Therefore, the optimal placement can be used as initial placement.

? is

The best placement has the following configuration.

[illegible]

Fig. ~~8.3~~: The configuration of the optimal placement of the "CHESS-BOARD" array.

First of all, the system is observed with the temperature decreasing step equal to 0.95 according to the literature [Kirkpatrick 83]. Especially the acceptance ratio as a ~~in~~ function of the temperature is observed. The acceptance ratio shows in its shapes ~~course~~ three distinguished areas as already mentioned in a previous chapter. It is obvious that these areas characterize the entropy function of the system, and when the control function has a piecewise linear character the parameters of that function will be determined by these areas. The observation of some experimental results shows that when the acceptance ratio changes much during one temperature step, the temperature decreasing step must be small, for example 0.9 or 0.95. Does on the other hand the acceptance ratio change not very much then the decreasing temperature step can be larger ~~smaller~~ for example 0.7. defined by
This piecewise linear temperature lowering scheme will give the result that the final configuration of the optimization process ~~be~~comes rather close to the optimum configuration within a reasonable computation time, ~~consumption comparing with a constant temperature schedule~~.

Fig. 8.4.: The acceptance ratio and energy in function of the temperature.

Furthermore, some other faster schemes were tried out, but they would ~~resulting~~ ~~in~~ not reach~~ing~~ the optimum. The system will be frozen in a local optimum of the objective function, with a local optimum is ~~characterized by a~~ temperature ~~which is~~ too low to get out of the equilibrium, ~~comparing to the temperature necessary~~ The probability ~~for~~ to accept~~ing~~ deterioations of the system to get out of the equilibrium is too low. deteriorations ?

Sofar a "reasonable" temperature lowering scheme is experimentally determined. The final result of the annealing algorithm depends not only on the ~~course~~ shape of the temperature lowering scheme but also on the choice of the weightfactors in the objective function. Weightfactors are, in first ~~instance~~ place, intended for controlling the affect of a particular term of the objective function on the final result. The second role of the weightfactors, which is of course of minor importance, is their influence on the ~~course~~ smoothness of the objective function ~~(the course is sometimes called the "smoothness")~~. ~~This course affection of the weightfactors depends very strong on the inconstancy of each term in the objective function. For example the term which contains the maximum of a histogram is much more constant than a term containing the total sum of that histogram. Thus,~~ If the weightfactor of the "max" term is an order of magnitude larger than the one of the "sum" term the objective function will have a "smoother" characteristic. The ~~[illegible]~~ shape of the objective function is important ~~of~~ for finding an optimal choice of the degrees of freedom of the annealing algorithm, because there exists a very tight

effect ?

relationship between the objective function and the temperature lowering scheme. This was also ~~remarked and shown~~ observed by ~~by~~ Otten [Otten 84].

In case of our "CHESS-BOARD" placement, values for 4 weightfactors have to be chosen. The weightfactors are chosen in a way such that the ~~a~~effects of each term is of equal order in size. This is done ~~for~~ by eliminating any preference of one term in particular. Hence, the values are:

whm	=	8
whs	=	1
wvm	=	8
wvs	=	1

8.3. "BARS" GATE ARRAY

The gate array example circuit -- in, this report called the "BARS" example -- that was intended for ~~qualifying~~ testing the placement algorithm was not really a gate array delivered by a factory as a benchmark. The description of the gate array was extracted from a picture. The picture is shown next.

The shown gate array was used as benchmark for the routing system called BARS (BAth Routing System) by a research group at Bath university. The netlist description was extracted from the shown picture. The nets connected to I/O pads were ignored. Note that in almost all gate array images the basic cells ha~~s~~ve two rows of pins (terminals); one on the top side and one at the bottom side. Each pin in one pin row has its logic equivalent pin on the other pin row. If such logic equivalent pin is used to get one net from one channel to another channel then this connection through the cell is called a feedthrough.

Mostly the definition of feedthroughs is slightly different from the definition here given. The piece of interconnection that goes through a cell row, without having any connection to the underlaying cell is called a feedthrough.

Because the feedthroughs were not indicated in the picture it could be possible that a net was split in two or more nets caused by ignoring the possible existence of feedthroughs. This incompatibility of the netlist with the original netlist might have an effect on the placement result.

After the netlist was extracted and typed in, the annealing job was started. The initial placement was also taken from the picture. In this example the starting energy is of great importance, because this is the energy of the placement they used in BARS to route.

The size of the array is 10x12 cells, whereby 110 cells were occupied by a certain set of functions ~~intended to implement~~. So it can be considered as a relative small example, the more when compared to the goal of our placement program: the feasibility of placing a gate array consisting of at least 1000 cells.

~~The importance of this example was having the ability to compare our placement with the placement of an already routed chip. The beauty of this feasibility was a little undone by the fact that the circuit was extracted from a picture instead of a network description. So there was no control of the use of feedthroughs and pin equivalence.~~

The filling of the array is very dense (about 92%) and the number of wires (nets) compare to the number of cells is very high (175 to 120). compared
Thus we have ~~These will provide~~ a difficult example for the placement and routing programs. Nevertheless, the annealing program placed these 120 cells in about 45 min. CPU-time (HP9000) with a total energy (cost function) about 150% smaller than the energy corresponding to the original placement used in the BARS system. The starting energy was about 3000 units and the energy at the freezing point was about 2000.

~~Fig.16~~.: The acceptance ratio and energy in function of the temperature.

Here again the same course of the acceptance ratio is observed, which ~~will~~ points to the usefulness of our temperature schedule.

8.4. "TEXAS" GATE ARRAY

The experiments involving the "TEXAS" chip, were examined in order to evaluate the performance of the placement program on larger arrays. So far only small to medium sized arrays for ~~adjusting~~ testing the annealing schedule were used. The experiments with the large array can lead to the justification of some parameters of the annealing program, and it is also used for ~~estimation~~ estimating ~~of~~ the time complexity. The second point which makes these experiments so important is expressed in the practical value of the examined placements. ~~i.e., the measure which expresses the degree of usefulness of the placed functions.~~ For example the "RANDOM" chip reflects a bad imitation of a personalized gate array; therefore its practical value ~~will be very low~~ is reduced. ~~This was due to the appearance of the clustered netlist; almost every module or cell is connected to almost every other module or cell.~~

The "TEXAS" array ~~is called so because it was a gate array from~~ is based on an image designed by TEXAS INSTRUMENTS. The to be implemented set of functions ~~were~~
1 5 6 7 2 3 4

derived from a ~~accomplished with an~~ universal I/O controller. The ~~involving~~ circuit was designed by Stevens during ~~on a~~ VLSI design~~ing~~ course. Stevens has tried to implement this circuit ~~in~~ (by) a STL gate array from TEXAS INSTRUMENTS.

Although, the chip was not realized, it is -- ~~in our case~~ -- the most realistic ~~realizable~~ example of a design ~~to be implemented circuit~~. ~~The circuit was never (before) implemented because the claimed number of cells was too large for the on that moment -- existing gate array masters of that particular silicon vendor (TEXAS INSTRUMENTS).~~

~~When one wants to know if a circuit is to implemented in only~~ one gate array one have to calculate the minimum number of cells the suitable gate array must contain at least. Suppose, the personalized gate array has 80% of its possible cell locations occupied and the circuit contains 618 cells, the minimum array size should be in that case 873 square units (>873 cells). The two available gate array masters have a size of resp. 400 and 600 cells. Thus, it is quite obvious that this circuit cannot be implemented in only one array in this silicon technology, connected ~~with the TEXAS INSTRUMENTS silicon vendor.~~

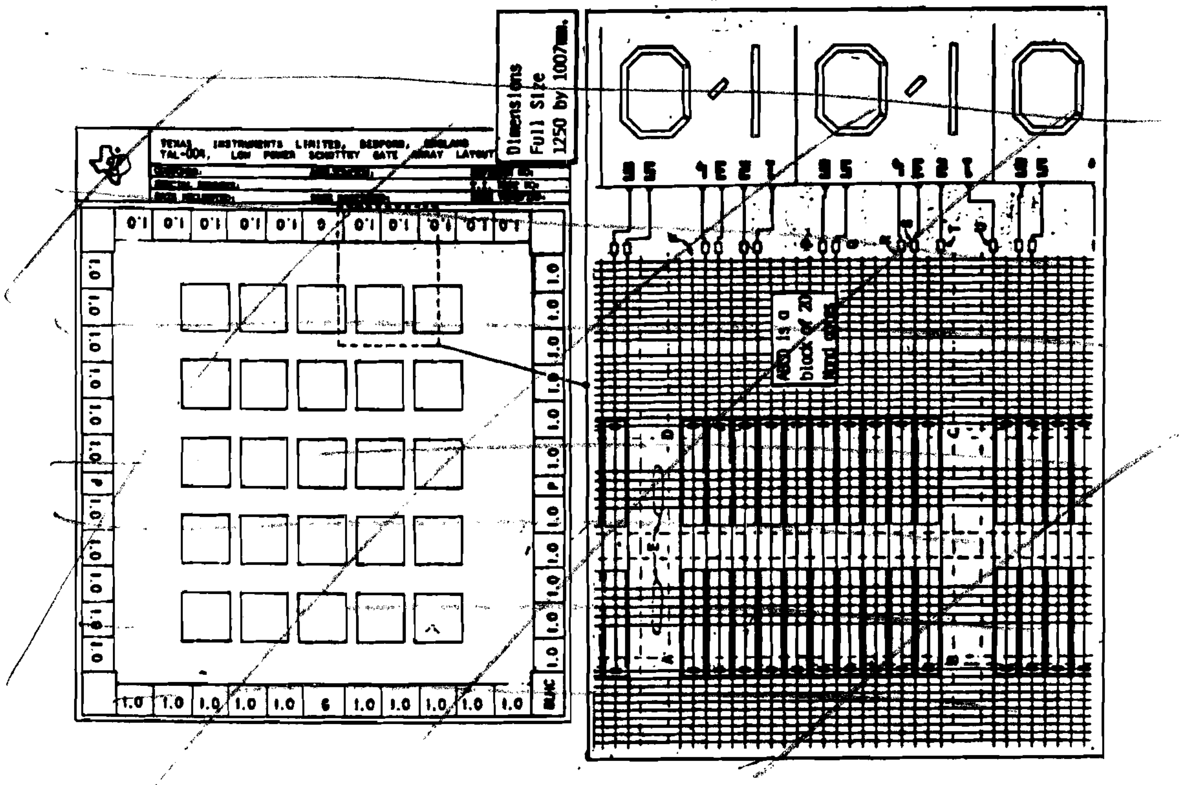
Dimensions
Full Size
1250 by 1007mm
TEXAS INSTRUMENTS LIMITED, BEDFORD, ENGLAND
TAL-004, LOW POWER SCHOTTKY GATE ARRAY LAYOUT

Fig. 6.7.: ~~The "TEXAS" gate array image.~~

~~For~~ automatic wiring systems the average ~~cell occupying~~ density ~~that ever~~ can be ~~reached~~, is und~~oubtly~~ higher than the above mentioned 80%, which is the ~~required~~ number of cells available in a gate array master to ~~make~~ the implementation useful. Therefore the real size of the array will be somewhere between ~~618 and 873 cells necessary for~~

~~the implementation of our the circuit.~~

T~~he silicon technology, determined~~ by the system designers, ~~chosen for~~ the implementation will be STL from TEXAS INSTRUMENTS silicon vendor. Accomplishing with this technology, the gate array image look like the one in Fig. 8.7 The array distinguishes the existence of "blocks" with large wiring spaces between them. Each block contains two rows of 10 cells, with an almost square geometrical shape. Therefore, to provide an approximately square chip (aspect ratio = 1) the number of horizontal blocks must be almost equal to the number of vertical blocks. Résume the last propositions, the suitable chip should contain in our case 720 or 840 cells, corresponding with array sizes of resp 6x6 and 7x6 blocks. The first choice belongs to the smallest one, for maximizing the cell occupying density. The array, 12 x 60 cells, will be used in our case as the master for the implementation of a 618 cells ~~containing circuit accomplishing with a cell density of almost 86%~~

(To be able to evaluate the result)

This array was first placed ~~without any preference of direction. Due to the fact there was no comparision of the placement possible the qualifying was very difficult and not clear~~. ~~Therefore~~ It, a tool was developed that compiled all net surrounding boxes and sorted them on size. ~~With this data,~~ these ~~the placement could be qualified somewhat better, and it can be remarked that~~ The surrounding boxes ~~are~~ turned out to be quiet big, and certainly the boxes are not oblong enough. This effect is Caused ~~brought on~~ by the fact that the cells are about 5 times larger in one direction than the other one, but in our simplified model they are assumed to be square (or points). So ~~when we want to put a geometric interconnection length in the objective function, instead a symbolic one~~ we decided, the weights of the terms in the objective function for one direction ~~need to be enlarged~~ to enlarge. After the weigths were adapted the annealing job was run, with ~~many~~ much better results.

observation

Fig. 8.~~6~~: The annealing result of the "TEXAS" array.

The effect of ~~not equally~~ unequal weight~~ed~~ for both direction~~s~~ results in a second ~~remark~~ observation. In the network there were some macro's ~~visible~~, ~~in this example only consisting of~~ mostly flip-flops. These flip-flops contain 8 cells. Since the wirability will be better in one direction, it will be clear that the existing flip-flops will ~~lay in that~~ be oriented towards that direction to give the best results, because the intra macro wiring is more critical than the inter macro wiring. Here again the results were better when ~~the~~ one direction ~~preference~~ is weigthed stronger in the objective function.

~~CHAPTER 9~~

RESULTS

At the moment the ~~produced~~ computed placements had to be reviewed there was no gate array router available within the research group. This had the consequence that the routability of the generated placements could only be estimated. T~~he absence of the router provided also the fact that the chip design could NOT be verified, and so it was very difficult to qualify the placement program~~.

~~Another~~ One possibility of ~~qualifying the~~ evaluating a placement is to compare the results from the annealing algorithm with the results from one or some other placement algorithms. But these placement algorithms were not available in the research group on that moment.

~~Finally, for being able after all to say something of the quality of the program,~~ Furthermore some graphical tools were developed. Those tools used to observe ~~tools for observing~~ some parameters ~~continually~~ continuously like d~~isplaying~~ the net density and the mean length of all nets during the run of the placement optimization program. ~~and~~ Also some tools for displaying the placement with its cells and nets to get an idea of the correlation between the energy (value of the cost function) and the routability have been established. The nets were displayed from terminal to terminal just as a straight ~~connection~~ line, ~~because routing of the nets was not possible.~~

Fig. 9.1.: Minimum spanning tree net connection.

With these tools only a very general informal discussion of the quality ~~factors~~ of the placement is possible. ~~and certainly not in detail. Another important consequence of not being able to review the resulting placements is that no "good" values for the stop parameters of the annealing and other degrees of freedom of the objective function are being found.~~

~~Not only the missing of the router was the reason for not being able to values for the degrees of freedom but also the fact that on the moment the annealing program was written for the application to the gate array placement problems there were not any already as gate array realized networks available. The supply of real example networks for applying the placement program was nil, so the program was only tested and adjusted on the placement of some fiction gate arrays and networks.~~

~~Since the placed circuits could not be reviewed well, it is almost impossible to qualify the placement program. Thus, also the performance of this implementation of a placement algorithm cannot be reviewed or compared with other implementations because of~~

- ~~the lack of benchmark placements~~

- ~~the missing of a gate array router~~

– the ~~missing of other placement programs~~

~~The performance evaluation which is done quite often in the literature is the comparing of the total computation time consuming necessary for the placement of a gate array containing N cells. This is a quite ridiculous and a not justified comparison because the incompatibility of the two gate arrays. They may be total different, the only compatibility is the total number of cells (modules). The total used computation time depends behalf the compatibility of the arrays on the implementing computer language and machine. It depends also on the stop conditions of the algorithm and the parameter choices of the model. So it is certainly not justified to compare two placement programs (although both may be based on simulated annealing) only on the total consumed computation time necessary for the placement of N cells consisting gate array. Therefore I will not summarize the used CPU times of the placements.~~

In order ~~Though~~ to give an impression of the CPU-time necessary to ~~for the~~ run ~~of~~ an annealing job, in our case a placement generator, we will not summarize the total time but the time needed for the calculation of an accepted and rejected move of the "chess-board" array on a HP9000-UNIX system in PASCAL.

accepted move :	189	per sec.
rejected move :	352	per sec.

This quantity in time depends of course on the implementing machine but it depends also on the size of the array, and on the number of interconnections. Assume that the array contains N cells, then the array size produces a factor N in time. Mostly the number of interconnections is in some way related to the number of cells. Therefore, the real dependency will be of the second order ~~[illegible]~~ O(N**2).

The total number of moves necessary for one run (split ~~divided~~ in accepted and rejected ones) is another concept which contributes to the determination of the total run time. The contribution will be of order N, because the number of moves is set to 10*N and 100*N depending on the phase of the annealing. Together with the previous dependency the total ~~one~~ will be of the third order (worst case).

The routability will be probably, ~~despite of all previous remarks~~, reasonably ~~very~~ good ~~(almost guaranteed)~~. This conclusion is a result from the fact that, in all examples, the uniform distribution of the nets is achieved ~~guaranteed~~ (except for the "random" gate array). This is confirmed by the histograms (Fig. 8.8) of the final placements and also by next figure which shows the number of net surrounding boxes at grid positions.

Fig. [illegible].: Non-uniformal (a) and an annealed ~~uniformal~~ (b) distribution of the nets.

~~In chapter 3 some properties of a good gate array placement were discussed. Now these rules will be discussed in regard to the way the annealing algorithm will fulfil them.~~

The placement program, applied here, is technology independent per definition, due to the definition of the model. The program can handle restricted area constraints, determined by the move calculation routine. Therefore, it is no problem to place ~~eventually the~~ pad cells, by considering them as normal cells present in a restricted area.

~~In this version of the annealing program there was taken no account for the possibility of implementing the logic function mapping on the physical cells.~~

~~The rule of the possibility of using different macro topologies for~~ only one logic function. And using the assertion that some pins are logically equivalent is of course transformable to the previous rule and so it is not satisfied within this implementation of the annealing algorithm. All other rules or properties will be satisfied within this placement program. So the overall conclusion will be that this implementation could be used as a gate array placement program with reasonable qualities for those placements where the constructive placement programs did not generate accurate placements. But a lot of experiments have to be made to understand the controlling and the adjustment of the parameters in a way that they can be automatically adjusted. The invariant in all these experiments must be ~~the "goodness" of the final placements.~~

~~The efficiency of the macro placement is even harder to review than the review of the basic cell placement, because of the lack of~~

practical experience due to the absence of test examples. Additional fact to this is that it is impossible to give an evidence for the "goodness" of this algorithm. Although the "goodness" of the algorithm cannot be guaranteed or be demonstrated the convergence of the program is proved and estimated of computation time will be not so big. The number of degrees of freedom and so the number of modules is significant smaller than with the chip (gate array) without the use of "giant cells" or macros. But the effort necessary for calculating the change of energy (cost function) of a move of two macros will require much more computation time than necessary for the calculation of the energy of a move of two cells. The largest fraction of the required CPU time is needed for the determination of a new move. The implementation of the annealing algorithm presented in this paper is very flexible towards the controlling of the many existence placement parameters. This program implements a technology independent gate array placement algorithm. The gate array image and its silicon implementing technology are more or less parameterized and these parameters will control the final resulting placement. For example there are two vectors involve the pattern sensitivity of the placement. It allocates the channels, regions intended for interconnection, and the regions where interconnection is not allowed except for feedthroughs. It is also possible to allocate a region in where cells may not be moved or only be moved inside that regions itself which accomplishing with not being allowed to cross the boundaries of such regions. That allocating can be managed be passing some parameters to the placement program. To optimize the timing performance of the implemented logic one could use weight factors for the nets. Time critical nets will get high weight factor comparing with non-critical nets. So the total length of these critical nets will be very small in the final placement.

~~CHAPTER 10~~

CONCLUSIONS

In this report we presented an application of the simulated annealing algorithm as formulated by [Kirkpatrick 83] to the placement problem of gate arrays. This technique turns out to be a very powerful tool for optimization of a cost function involved in the gate array placement problems.

The advantages can be formulated as:

- the algorithm is applicable to all type of gate arrays types ?
- the algorithm is very easy to implement
- the result can be as "good" as desired

The disadvantages are:

- automation of the overall cooling scheme is not realizable yet
- the required computation effort (time) may be very large (too large for an interactive process)

~~About the conclusions~~ to be made one thing has to ~~be remarked. The quality of the placement produced by a placement program can~~ only ~~be~~ reviewed well, if the routed ~~(layouted) chip is simulated~~ and if the behaviour of realized functions is criticized. This qualifying of the chip is of course only possible if the chip is routed, which claimed a routable chip first. Thus, the routability of a placement is the most obvious quality factor especially for gate arrays, and therefore a placement has at least to satisfy this condition. The restrictions inherent to gate array design causes the increase of importance of this factor by observing the quality of a placement, compare to full custom design. For the placement which is not routable it is out of sense to talk about the behaviour reviews. But to ensure the routability of a placement a router is almost necessary especially for gate arrays thanks to the restrictions.

~~Although all these facts one thing has to be remarked: this developed placement tool is very useful to apply on gate arrays without using macros (not as fix macros but the using of weighted nets for defining macro does probably lead to an useable application). Concluded from the complexity and the visible quality of the placement. The annealing program will generally be able to produce a placement but not guarantee a placement which is closer to the optimum than any other program will do.~~

The result of the annealing algorithm is highly dependent ~~with~~ of the choice of the cost function. If the global ~~course~~ shape of the cost function is too smooth then the annealing will not work well. The reason is that the metropolis algorithm does not work because the cost function ~~and so the energy~~ will change only ~~in~~ with a ~~limited~~ small ~~number~~ set of interchanges. When the cost function has ~~in contradiction~~ on the contrary i.e. a very capricious character with not one clearly obvious global minimum then the annealing program ~~will~~ spend a ~~lot~~ many of computation time in the region ~~of~~ around the ~~minimum~~ minima without ~~having a~~ making clear progress. The ~~in this research derived~~ we used temperature lowering schedule seems to be ~~considerably~~ generally applicable and is still very simple. ~~This conclusion can be made, due to the acceptable functioning of the model.~~ ~~This~~ Our model is ~~also suitable~~ appropriate for the description of all kinds of gate array images and therefore we may assume that our placement program is fairly technology independent.

REFERENCES

[Beutler 82] R.R.Beutler and D.C.Larson, "A Versatile HMOS Gate Array for Prototyping and Production", IEEE Proc. of the 1982 Custom Integrated Circuits Conference, Rochester May 17-19, 1982, pp. 23-26.

[Breuer 72] M.A.Breuer, ed., "Design automation of digital systems", Englewood Cliffs, New Jersey, Prentice Hall 1972.

[Breuer 77] M.A.Breuer, "A class of Min-cut Placement Algorithms", Proc. ACM IEEE 14th Design Automation Conference, New Orleans, 1977, pp. 284.

[Cheng 84] C.Cheng and E.S.Kuh, "Module Placement Based on Resistive Network Optimization", IEEE Trans. on Computer-Aided Design, Vol. CAD-3, No. 3, July 1984, pp. 218-225.

[Garey 79] R.M.Garey and J.D.Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", W.H.Freeman and Company, San Fransisco, 1979.

[Hilberg 82] W.Hilberg, "Grundprobleme der Mikroelektronik", R. Oldenbourg Verlag GmbH, Muenchen 1982.

[Jepsen 83] D.W.Jepsen and C.D.Gelatt jr., "Macro Placement by Monte-Carlo Annealing", Proc. IEEE Int. Conf. on Computer Design, Port Chester, New York 1983, pp. 495-

[Jess 84] J.A.G.Jess, R.J.Jongen, P.A.C.M.Nuijten and J.C.Bu, "A Gate Array Design System Adaptive to many Technologies", Proc. of International Conference on Computer Aided Design (ICCAD), Santa Clara, 1984, pp. 338-343.

[Kirkpatrick 83] S.Kirkpatrick, C.D.Gelatt Jr. and M.P.Vecchi, "Optimization by Simulated Annealing", Science, vol. 220, pp. 671-680, 1983.

[Lauther 79] U.Lauther, "A Min-Cut Placement Algorithm for General Cell Assemblies Based on a Graph Representation", Proc. 16th Design Automation Conference, 1979, pp. 1-10.

[Mavor 83] J.Mavor, M.A.Jack and P.B.Denyer, "Introduction to MOS LSI Design", Addison-Wesley Publishing Company 1983.

[Mead 80] C.A.Mead and L.A.Conway, "Introduction to VLSI Systems", Addison Wesley, Reading Mass.

[Moullion 84] M.Moullion, "Automatic Customization for Single-Layer Gate Arrays", VLSI Design, Nov. 1984, pp. 86-94.

[Opitz 83] F.Opitz and A.Voelker, "AULIS - Automatic Placement and Routing of Gate Arrays", Proc. of the Sixth European Conference on Circuit Theory and Design (ECCTD'83), Stuttgart 1983, pp. 331-337.

[Otten 84] R.H.J.M.Otten and L.P.P.P.van Ginneken, "Floorplan Design using Simulated Annealing", Proc. of International Conference on Computer Aided Design (ICCAD), Santa Clara, 1984, pp. 96-98.

[Wirth 71] N.Wirth, "Program Development by Step-wise Refinement", Communications of the ACM, Vol. 14, 1971, pp. 221-227.