HIGH SPEED SIMULATED ANNEALING APPROACH FOR NON-SLICING VLSI FLOORPLANNING

A dissertation submitted in partial fulfillment of the requirements for the award of the degree of

Master of Technology

In

VLSI Design

Submitted by

SHAH RAJ SANJIVKUMAR

Roll No: 601461026

Under the guidance of

MRS. MANU BANSAL

ASSISTANT PROFESSOR, ECED

T.U. Patiala

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

(Established under the section 3 of UGC Act, 1956)

PATIALA – 147004 (PUNJAB)

DECLARATION & CERTIFICATE

I hereby declare that the work which is being presented in the dissertation titled **"HIGH SPEED SIMULATED ANNEALING APPROACH FOR NON-SLICING VLSI FLOORPLANNING"** in the partial fulfillment of the requirement for the award of the degree of Master of Technology in VLSI Design submitted in Electronics and Communication Engineering Department of Thapar University, Patiala is an authentic record of my study carried out as under guidance of **Mrs. Manu Bansal** (Assistant Professor, ECED) during 2014-2016.

Date: 05/07/16

SHAH RAJ SANJIVKUMAR

Roll No: 601461026

It is certified that the above statement made by the student is correct to the best of my knowledge and belief.

(MANU BANSAL)

Assistant Professor,

ECED

Thapar University,

Patiala-147004

Countersigned by:

Head

ECED, Thapar University,

Patiala-147004

Dean of Academic Affairs

Thapar University,

Patiala-147004

ACKNOWLEDGEMENT

First of all, I would like to express my gratitude to **Mrs. Manu Bansal**, Assistant Professor, Electronics and Communication Engineering Department, Thapar University, Patiala for her patient guidance and support throughout this report. I am truly very fortunate to have the opportunity to work with her. I found this guidance to be extremely valuable.

I am also thankful to our **Head of the Department, Dr. Sanjay Sharma** for providing us adequate environment in carrying the work.

I am also thankful to **Mr. Ajay Kumar, Mr. Gagandeep Singh Dhingra and Surbhi Jindal** from ECED, Thapar University, Patiala for giving their valuable time and sharing ideas with me. Finally, I would like to extend my gratitude to all those persons who directly or indirectly helped me in the process and contributed towards this work.

ABSTRACT

In Very Large Scale Integration (VLSI) flow, Physical Design Automation plays a very important role. VLSI Floorplanning is a very important stage in the Physical Design of VLSI. Area, power and speed are the major parameters that affect the design and performance of any VLSI chip. The major focus of research in this VLSI is always on optimisation of any parameter or parameters listed above. In VLSI floorplanning the research is focused on the area of the chip, interconnection module wirelength and dead space between the modules. In VLSI floorplanning it is also taken into consideration that the modules which are going to be placed for the optimisation cannot overlap with each other.

In this research work, High speed simulated annealing algorithm (HSSA) is implemented for the solving of VLSI non-slicing floorplanning problem as it is NP-hard problem and is nearly impossible to solve if manually. The SA uses a new acquisitive method to construct an initial B*-tree and a new process to explore the search space. B*-tree is used and implemented as the input for the high speed simulated algorithm. In this dissertation, not only optimisation of area of modules, interconnection wirelength of modules and dead space between modules is done but all optimisations are done with very high speed. Nowadays all floorplans are coming from the non-slicing category so in this research work, non-slicing VLSI floorplans are taken into consideration.

HSSA is implemented along with the B*-tree in the C++ language. The HSSA has been implemented and experimental result with optimisation of area, dead space, wirelength and best timing constraints have been tested on Microelectronic Center of North Carolina (MCNC) benchmarks and also compared with the results mentioned in other research papers. In this, Ubuntu 14.04 OS and GPP compiler is used to compile the C++ code generated for the objectives. Maximum 11.17% reduction in area and 21.73% reduction in wirelength have been found with compared to other research work while in case of computation time, maximum 99.69% reduction has been observed.

TABLE OF CONTENTS

LIST OF ACRONYMS

ACRONYM	MEANING
VLSI	Very Large Scale Integration
MCNC	Microelectronics Centre of North Carolina
RTL	Register Transfer Level
HDL	Hardware Description Language
DRC	Design Rule Check
PDA	Physical Design Automation
CTC	Clock Tree Synthesis
HSSA	High Speed Simulated Annealing
MA	Memetic Algorithm
DDM	Deferred Decision Making
PSO	Particle Swarm Optimization
GA	Genetic Algorithm
SA	Simulated Annealing
ASIC	Application Specific Integration Circuit
B*-tree	Binary Search Tree
O-tree	Ordered Tree
GCC	GNU Compiler Collection

LIST OF FIGURES

LIST OF TABLES

CHAPTER-1 INRODUCTION

1.1VLSI Design Cycle Overview

From the formal specification of the system to the packed chip, there are many steps between the start and end of the cycle of VLSI design involving. A flow chart of such VLSI Design cycle is shown in Figure 1. Each step has its own importance and there is necessity to go through all the steps in the VLSI Design. Earlier, the design was emphasised mainly while there were less important to the verification. Verification at that time was done but not to all stages so the failure of the chip or damage might be high at that time. There were many serious incidents happened in the past due to lake of verifications in the chip manufacturing which led to the growing importance of verification in the chip design and manufacture.

Nowadays, In VLSI design flow, after each stages there is one verification process is there to assure that design is meeting the required specification. So the rate of damage or failure or bug is reduced rapidly. But as the sizing of the transistors is becoming more and more, complexity of chip increases and that leads increase in complexity of verification. There are many tools are available to do verification at each stage.

As shown in Figure 1, every designing process must be followed by verification process so that if any bug or mistake in design can be early determined. If any bug or failure find in any verification process then the flow of design is developed such a way that it will go to the particular design process again to remove the bug.

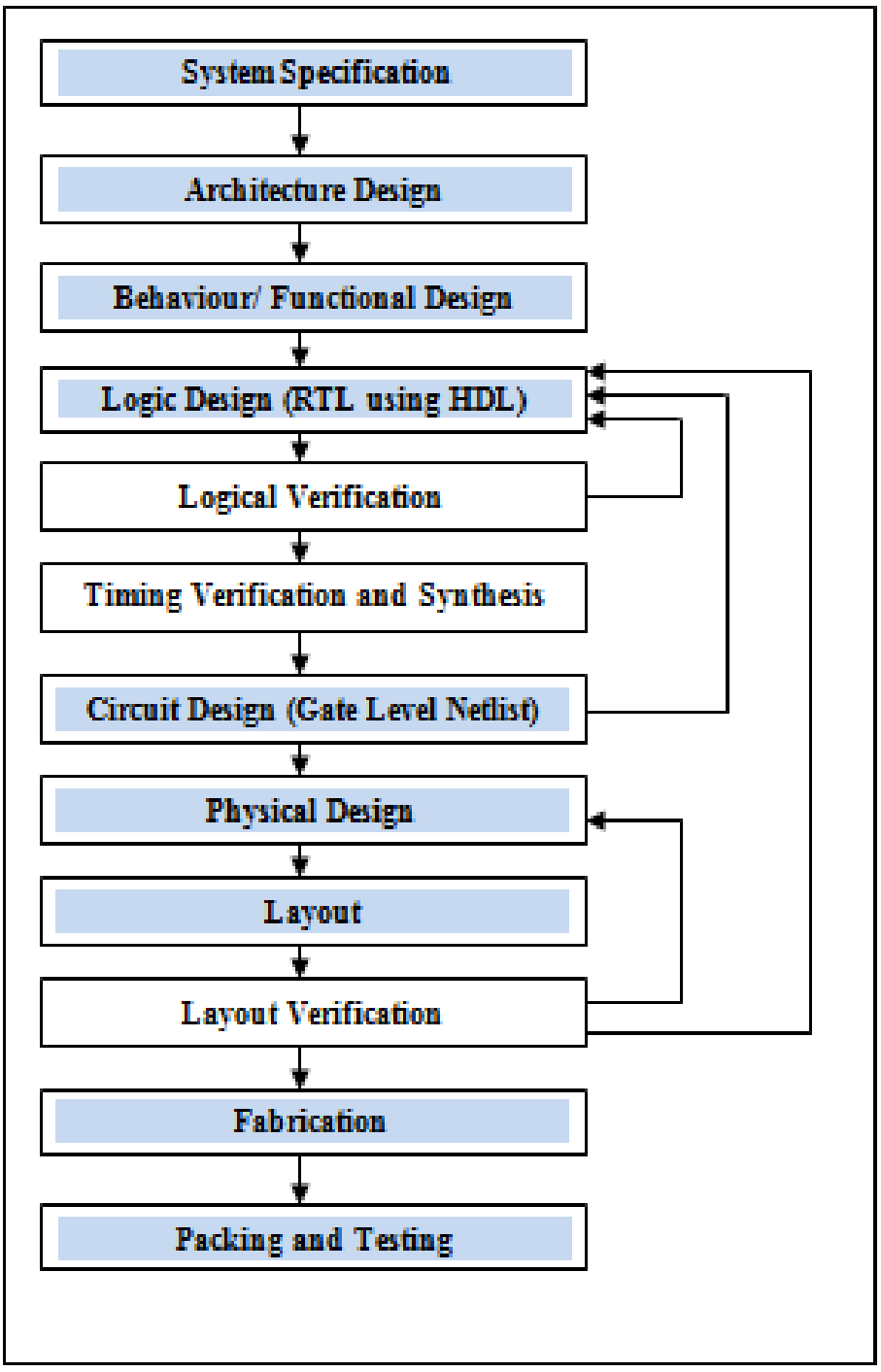


Figure 1 VLSI Design Cycle [1]

The steps involving in the design cycle of VLSI are outlined briefly below.

1.1.1 System Specification

This is the first step of the VLSI Design Cycle. System specification represents the requirements at high level. Performance (in terms of speed and power) of the chip, its functionality and size (in terms of the area) of the chip are the factors that must be considered in whole process. Other than these, time to market, requirements of market, current cutting edge technology and economic instability are also the factors in which system specification should compromise.

1.1.2 Architecture Design

In this step, architectural level design is built by the designers. The decision of architecture such as Reduced Instruction Set Computer or Complex Instruction Set Computer is going to be used in the design is taken in this step. The other components of architecture such as Arithmetic Logic Unit, Floating Point Unit, cache memory size and the number of pipeline or parallel connections with their structures are decided in this step.

1.1.3 Behaviour/ Functional Design

In this step, the behaviour of the design in terms of functional units like interconnection between the units, specification internal structure like adders and multipliers and estimation of area of the chip, power consumption and speed. For example it specifies that adder is required but in which mode the adder is executed that is not specified as may be there may be variety of adders in terms of speed, size and power.

1.1.4 Logical Design

In this step Register Transfer Level (RTL) description is done. All arithmetic operations, logical operations, width of the allocated word, flow of control, list of allocation of register is described in RTL. There are Hardware Description Languages such as VHSIC Hardware Description Language and Verilog which are used for the

RTL description. There are many tools such as Xilinx ISE, Modelsim and Questasim from the Mentor Graphics, Cadence tools etc. are used for the compiling and simulating of the HDLs. Verification of design plays a very important role in the design cycle as it verify that the design is met the specification for which it is made. In this, once the RTL description is expressed by using the HDL, it must pass to the logic verification stage to verify its functionality that it meets all the requirements of the design. Once it functionality is verified then it will pass thorough the timing verification. In timing verification, the timing constraint of the chip is verified with reference to the specified timing constraints. In synthesis, the RTL description which described using HDL accomplish that the RTL design can be implemented using hardware blocks as well. Synthesis fails when some code of HDL cannot be converted to its equivalent hardware. So it is very necessary that the design is synthesisable.

1.1.5 Circuit Design

In this step, the design of the circuit is made based on the logic design. In this gate level circuit is made from the Boolean expressions. Speed and power are also taken into account while designing a circuit. Simulation of circuit is required to check the working of circuit. Netlist specifies the inputs, outputs and interconnections between them. Synthesis tools are nowadays widely used to make the Netlist from the logic design.

1.1.6 Physical Design

In this step, the geometric representation of the design is made from the circuit design. In circuit design there are macros, transistors and gates are used to represent the design and in Physical design that all components are converted to a particular layout or geometry. Different component have their geometric pattern in this stage and multiple layer also can be there. The physical design is subdivided into many stages like partitioning, floorplanning, place and route and timing synthesis as it is very complex step. Depend upon the time to market, physical design can be done manually if more time is there or automatically using tool if time is less.

1.1.7 Layout

In this step, all the schematic symbols are transformed into their physical representation. To make a layout of the chip, the overall architecture which contains the large blocks must be analysed. Nowadays from the physical design, each block in the chip is arranged in such a way that area of total chip and power consumption are minimised and that will be the input to the layout. For the large chip, it is very necessary to understand the layout of the simple gates and arranging the layout of sub circuits as per its floorplan. Verification after layout plays a very important role. Design Rule Check (DRC) process verifies the all geometry pattern that must meet to the design rule given by fabrication. These rules are separation between poly to metal, poly to poly, metal to metal, length of wire etc. There may be millions of wires in the chip and DRC must check for all of them. After that the Circuit Extraction technique is used for the functionality checking of layout. Then Layout Versus Schematics (LVS) verification is done in which the extracted description and circuit description is compared. The performance verification contains the timing analysis of each component and interconnects. The reliability verification is done for checking reliability of layout.

1.1.8 Fabrication

Design is ready for fabrication only after layout and its verification process completed. A tape that contains the data of layout and first step of fabrication is to release the data. This process is called as Tape Out. There are many steps in VLSI fabrication which are silicon manufacturing, wafer processing, lithography, oxide growth and removal, diffusion and implantation, annealing, silicon deposition, metallisation, testing and assembly and packaging. All steps have further sub processes. Photo-lithographic mask are used for layout representation. There is one mast used for each step of fabrication.

Packing and Testing: After the fabrication, the wafer is diced into individual chips. This is the final step in which packing of fabricated wafer is done and then tested

against its specification. Packaging is done is Dual In-line Package (DIP), Quad Flat Package (QFP) Ball Grid Array (BGA) and Pin Grid Array (PGA).

1.2 VLSI Physical Design Flow

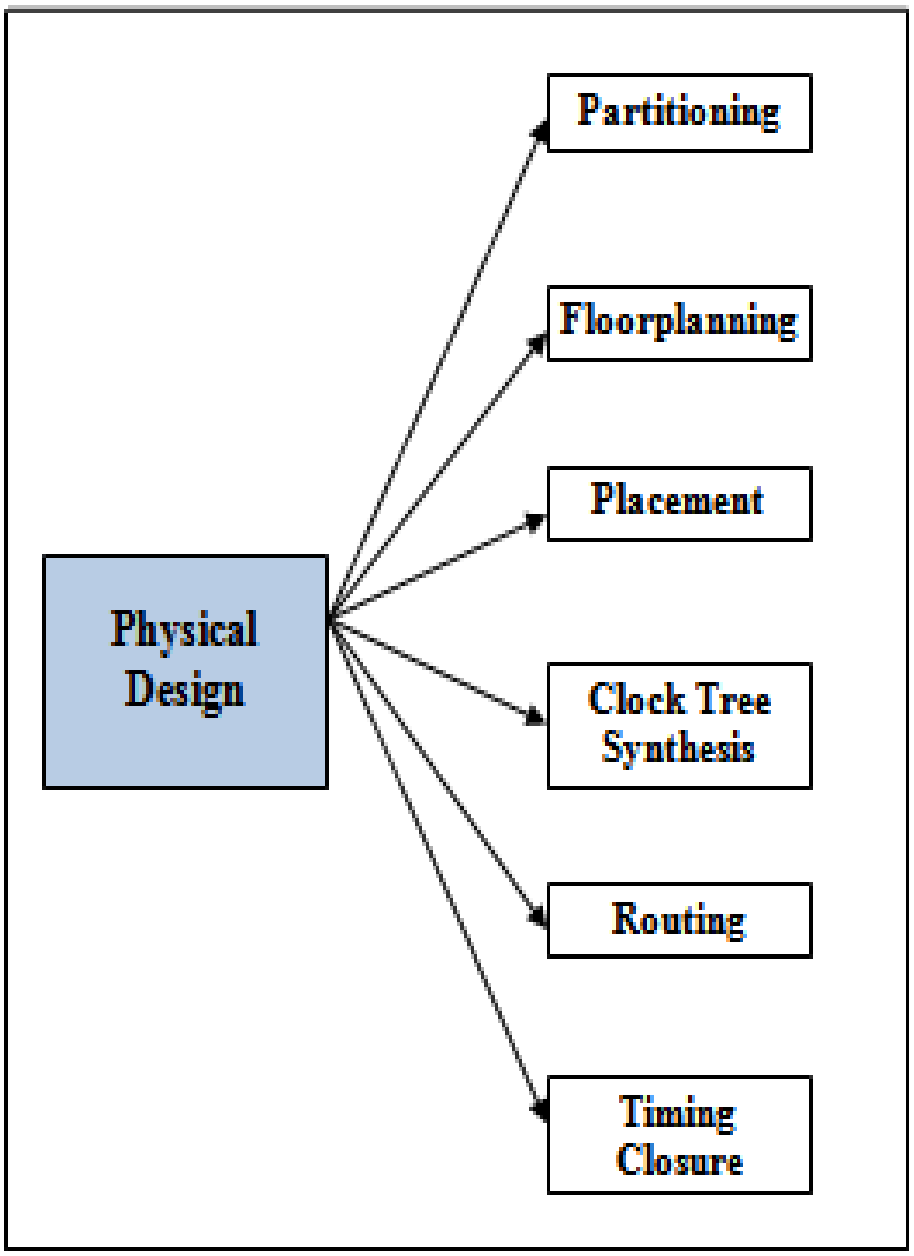


Figure 2 VLSI Physical Design Flow

VLSI Design Cycle contains physical design step which is very necessary after the circuit design. VLSI Physical design is very complex process and so that it is divided into the number of steps to make it conceptually easy [2]. On the basis of the design that is whether it is full custom or semi-custom, the steps inside the physical design can be varied. Different stages of the VLSI physical cycle are shown in the Figure 2 such as partitioning, floorplanning, placement, routing and etc. Each stage is described below in details. Physical Designs take the input from the output of the circuit design and give the output to the layout.

1.2.1 Partitioning

Before many years ago, there were very few components or modules in the chip so layout the entire chip was not too difficult. According to Moore's law the number of transistors doubled after a year. But the technology is scaled very rapidly and size of the transistor is decreasing at very high rate so the complexity of the chip is increasing day by day. Nowadays a chip contains million and billion numbers of transistors. That is the main reason that chip is partitioned into its sub parts. The hierarchical partitioning contains the sub modules which are further partitioned from modules. This sub parts is called as module or block and it may have many transistors too. Partition process is done by considering many parameters such as module size, number of modules, interconnection between modules etc.

So after partitioning, a chip can be converted to set of modules and its interconnection.

1.2.2 Floorplanning

This is one of the most important steps in physical design. This steps deals with the estimation of the total area from the area of the each module. After partitioning, area of each module is estimated based on the number and type of the module. Interconnect area is also calculated in addition to estimate the total area. Earlier when

there were less amount of modules were there, floorplanning was easy but nowadays with too many modules in the chip, floorplanning is going to be very tough. Some CAD tools also can be used for the floorplanning. But if the components in this chip need to be placed according to signal flow, manual floorplanning is very necessary at that place. Manual floorplanning help when location of some modules of components is specific on the chip. Dead space is the space which is left or unused by the modules. Floorplan must be as such that maximum dead space can be minimised. Interconnect wirelength is the length between modules. If the interconnection wirelength is less then total area will also be less.

1.2.3 Placement

After the floorplanning, net step in physical design is the placement. As the name suggest, the modules are placed according to their floorplan. Goal is to place modules such a way that to find the arrangements with minimum area while the performance constrains also taken into consideration. The initial placement is made without considering any constraints but after that initial placement is analysed and improved for the area and performance constraints. Placement process has to be considering that after the process the design is routable as sometimes placement process led to the design which may be unroutable. In such case, there must require repeating the placement process. To prevent this, early estimation of the routing space is required. Good placement always useful for good routing and performance. Late changes led to the increase the size of the chip and also degrade the performance of the design.

1.2.4 Clock Tree Synthesis (CTS)

Sequential components of the chip required clock signal as control signal. Clock tree synthesis process is used for assuring that clock is distributed to each sequential component. Skew and latency are two major issues in the chip clocking. So to minimise it this process is done. CTS take the placement data and clock tree constraint as the input. There are many constraints in CTS and that are clock skew,

clock latency, maximum transition time, maximum capacitance and fan-out, number of buffers and inverters etc.

Delay and latency are maintained by the clock tree buffers while duty cycle is maintained by the clock tree inverters. If there are more inverters and buffers then the area and power will be more. There are some structures used in clock tree synthesis which are used for the distribution of clock and that are H-Tree, X-Tree, multi-level clock tree, Fish bone etc. After the CTS are completed, timing is checked again. Design Exchange Format (DEF), netlist and Standard Parasitic Exchange format (SPEF) and the outputs of the CTS.

1.2.5 Routing

In this step, netlist is provided to make the interconnection between the modules. If the partition process didn't add the space for the routing which called routing space then in routing that space is first partitioned. Channels and Switchboxes are the terms used for that. By using the channel and switch boxes, the routing process has to complete all the connections between the modules in such a way that total wirelength should be minimised. There are two kind of routing process are there in which one is Global Routing and other is the detailed routing. In global routing process, without regarding of exact details of wire and pin geometry the connections are made. There is a list of channel and switchbox for each wire in the global routing. Detail routing on other side includes the routing of each channel and switchbox. It is very complex process so some evolutionary algorithms are used for the routing process.

1.2.6 Timing Closure

Before this step, geometric constraints, area and power constraints are taken into consideration but in this step timing constraints are going to emphasis. There are mainly two timing constraints that are setup constraints and hold constrains in the timing constraints. Set up timing is the amount of time an input data signal should hold its value before the clock edge detected. Hold time is the amount of time an input

data signal should hold its value after the clock edge detected. Designer must pass through these timing constraints. To minimise the signal delays the placement process must be timing driven. To decrease the delay, transistor sizing is done by increasing width or aspect ratio of the transistor. To decrease the propagation delay, buffers must be inserted.

1.3 Objectives of the VLSI Floorplanning

- It has to determine the shape and location of the modules coming from the partitioning process. Shape can be rectangular, L shaped or any shape as fits to the requirements.
- Shaping of the modules has to be done in such a way that it must followed aspect ratio constraints.
- Module must not be overlapped with other module.
- Floorplanning is typically limited to problems with a few hundred or a few thousand modules earlier but maybe there is more number of modules as the complexity increases. Modules also called as blocks sometimes.
- Output of the floorplanning includes the shape and location of each module with satisfying all the constraints specified by requirements.

.

1.4 Organisation of Dissertation

Chapter 2 contains the Literature Review of the VLSI floorplanning problem and its solution by using various algorithms. Various research work have been analysed and research papers have been studied for the survey of research work.

Chapter 3 involves with the details analysis of the VLSI floorplanning, its roles in physical design, parameters to be considered, parameters to be optimised and its representation.

Chapter 4 deals with the problem statement of the VLSI floorplanning and its data structure representation

Chapter 5 illustrates the High Speed Simulated Annealing (HSSA) algorithm for the VLSI non-slicing floorplanning and its methodology

Chapter 6 describes the experimental result of implemented HSSA algorithm using MCNC benchmarks circuit and comparing the algorithm for the best result with the other research works.

Finally chapter 7 concludes the research work and also states some ideas for the future work.

CHAPTER-2 LITERATURE REVIEW

Very Large Scale Integration (VLSI) is the process of designing and manufacturing an integrated circuit by placing millions of transistors into a single chip. In 1970s, when communication technologies and complex semiconductor were in demand, VLSI had begun. Nowadays SoC allows VLSI manufactures to implement lots of components on single chip. Floorplanning in VLSI deals with the functionality, timing, area, shape for different constraints including space allocation, routing area, and distribution of clock.

2.1 Survey from Various Literatures

2.1.1 Floorplanning technique for VLSI hierarchical building blocks layout [3]

In this research work, optimisation of the overall wirelength and the area of the floorplanning problem having rectangular blocks with their size and connection constraints have been done by the authors. The authors have been developed two phases for their approach of getting solution. The combined result of minimising wirelength and minimising area as per the aspect ratio constraints has been succeeded by using the topology of placing the relative building blocks in particular plane. The other phase has been done by the author was spacing, in which the overlap produced by the moving and reshaping of the blocks in the floorplan have been removed. Bounding penalty function has been used to satisfy the shape constraints of the floorplan. The authors observed that to for more efficient area utilisation, large blocks must be connected indirectly. Author has been developed floorplanning algorithm that satisfied various constraints.

2.1.2 Adaptive Genetic Algorithm using Sequence Pair Representation [4]

In this paper, to solve the VLSI floorplanning problem an adaptive genetic algorithm approach has been adopted by the author. Sequence pair representation has been used as to represent the chromosome of the adaptive genetic algorithm. To explore the search space efficiency, new genetic operators have been introduced for the solution of problem. Depend on the individual stage; there was a dynamic strategy that selects the operator of the adaptive genetic algorithm during the execution. Result of the adaptive genetic algorithm has been compared with the simulated Programming language C was used to implement adaptive genetic algorithm using GENESIS. MCNC benchmark circuits have been used to get the results.

*2.1.3 Fast Simulated annealing using B-*tree for Fixed Outline Floorplanning* [5]

In this paper, fixed outline floorplanning problem has been considered by the authors as classical floorplanning techniques only optimise the silicon area by packing the blocks while modern floorplanning has to pack block with the fixed outline constraint and with the other constraints like interconnecting length and position of blocks. Authors have introduced bus floorplanning to face the challenging modern floorplanning constraints like interconnection between blocks and its position parallel. Authors have discussed The Fast- SA has been introduced by using the data structure B*-tree as representation of floorplan based on the three stage simulated annealing. Adaptive Fast – SA has been developed for the fixed outline floorplanning to change the cost function dynamically for the wirelength minimisation under the fixed outline constraints. Authors have been analysed 100% success rate for the fixed outline floorplanning by obtaining their experimental results. B*-tree with the bus constraint has been explored by its conditions for the bus driven floorplanning. Bus driven floorplanning algorithm has been developed based on the Fast- SA and the constraints. Authors also observed maximum reduction in the dead space for the hard and soft micro blocks and obtained experimental results for the same.

2.1.4 A Memetic algorithm using hybridised genetic algorithm for VLSI non-slicing Floorplanning [6]

For non-slicing and hard-module VLSI floorplanning problem, Memetic algorithm (MA) has been developed by the authors in their research work. This MA has been developed by hybridising genetic algorithm. This MA has been developed with the effective genetic search method so that it explores the search space. It has been developed to exploit details in search region by efficiently using local search technique. The MA has been using a static threshold bias search strategy. This strategy has been used such a way that the value of the threshold constant and never changes during the evolution. It has been observed by the authors that if threshold value dynamically changes its value in the light of the status of the population of the MA then more improvement in the performance of the MA.

2.1.5 Deferred Decision Making (DeFer) algorithm for non-slicing VLSI floorplanning [7]

The author has developed an algorithm named DeFergenerates to solve the VLSI non-slicing floorplanning problem. This algorithm which is used for the non-slicing VLSI floorplanning has been also abbreviated as DeFer and has high speed, non-hypothetical. Non-slicing floorplan has been developed by compact the slicing floorplan. DeFer has been developed in such a way that it only consider one slicing tree compare to the searching into various slicing tree using heuristic simulated annealing algorithm. Deferred decision making (DDM) procedure has been used to elaborate the idea of slicing tree. DeFer has not specified the structure, ordering of slicing blocks, line direction of slice when more than one subfloorplans have been joined at each node of the slicing tree. DeFer algorithm has been implemented using both hard and soft block of then VLSI floorplan and also achieved successfully vesr rate, optimised interconnection wirelength, minimised computational time.

2.1.6Two stage simulated annealing approach for floorplanning in VLSI [8]

In this paper, to achieve on-chip thermal management in the overall floorplanning with optimisation in power and scheduling of the temperature has been done by implementing a two stage Simulated Annealing technique based on high-level synthesis. Initially simulated anealing has been inroduced for power solution then another simulated annealing algorithm has been used with this solution obtained to reduce estimated on-chip temperature. By comparing with the traditional average power minimisation techniques, solutions with uniform on-chip temperature distribution along with power optimisation has been achieved with large improvements.

2.1.7 Hybridisation of simulated annealing for VLSI non-slicing problem [9]

In this paper, author has emphasised on the parameters such as performance, size of the chip, yield of the chip, and chip reliability. In this research work of the NP hard problem, hybridised simulated annealing algorithm has been implemented by the authors for solution of VLSI non-slicing floorplanning. In this research work, initially B*-tree data structure has been constructed using new greedy method then operations on that B*-tree have been performed for exploring the search space. For balancing the global exploration and local exploitation, bias strategy has been introduced by the author. Developed HSA has been implemented using MCNC benchmark circuits and experimental results have been successfully achieved.

2.1.8 SKB-Tree representation for the fixed outline and voltage island floorplanning problem [10]

Author has been developed an algorithm with new data structure representation called as SKB-Tree for the VLSI floorplanning problem. This tree representation has been introduced for the both fixed outline floorplanning as well as the voltage island driven floorplanning. All blocks have been allocated space dynamically so that totally all blocks have been placed into specific outline for every solution so that the constraints of the fixed outline floorplanning have been successfully satisfied. It has been developed for the voltage island floorplanning too. To reduce the drop of IR, management of power resources and simplification of the planning of the power,

blocks having the equal voltage must be placed into the same region in the chip. Author has achieved the zero deadspace with this SKB-Tree usage, while the interconnecting wirelength also has been reduced by comparing other techniques. Simulated Annealing has been implemented for only minimisation of the wirelength. Other parameters such as power and area have been optimised too.

2.1.9 Enumeration based technique for the VLSI fixed outline floorplanning [11]

In this research paper, an enumerative floorplan technique has been adopted for the VLSI floorplanning using the enumeration. According to the authors, effect of enumerative floorplanning on the interconnecting wirelength, whitespace or deadspace and the computational time has been never identified though enumeration has been used as one of the common techniques adapted in VLSI floorplanning. Trade-off between the constraints like wirelength, area of chip, computational or run time and estimated maximum order of the enumeration on VLSI floorplan have been analysed successfully. In enumerated floorplanning, dynamic programming enumerative clustering (DEC) technique has been introduced by the authors to reduce the computational time and the worst-case time complexity. It has been observed by the authors that the DEC technique also introduces the equal number of module possible permutations without use of dynamic programming while minimising the redundancy generated in enumerative clustering. A direct cost function has been developed to guide DEC to get the best permutation of cluster, and an extremely thorough local clarification has been introduced for compensating effect of the EC on the interconnect wirelength of the floorplan. Authors compared their experimental results with other for comparing their performance.

2.1.10 Simulated annealing approach for the MSV aware VLSI floorplanning problem [12]

In this research paper, author has been deceloped a MSV aware floorplanning for the multiple objectives like temperature optimisation, power reduction, and recuction in

the complexity parallely. Simulated annealing have been imlemented for the reduction in computation time. Experimental results have been achieved and compared with other for the best result.

2.1.11 Hybridisation of PSO and genetic algorithm for the thermal aware non-slicing VLSI floorplanning [13]

In this research work, author emphasised on a smart decision-making hybrid particle swarm optimisation-genetic algorithm. The goal of algorithm has been described as reducing the area, wirelength, and it also used hotspot to distribute the temperature uniquely across the chip. Data structure representation B*-tree has been used to generate the initial floorplan and after that hybridised PSO-GA algorithm has been used to obtain an optimal solution of the problem. In this hotspot has been introduced for temperature-driven floorplanning at the perturbation stage, thereby optimisation in the average and maximum temperature has been done too. The implemented algorithm and its experimental results have been compared with some other evolutionary algorithms. MCNC benchmark circuits and Alpha processor floorplan benchmark circuits have been used to implement the algorithm and to evaluate the results.

2.2 Research Gaps

Table 1 Research Gapes

Parameters / Index	Area Consumption	Power	Speed	Temperature Optimisation
N.R.D Gracia *et al.* [14]	Yes	---	---	Yes
Shingo NAKAYA *et al.* [4]	Yes	---	---	---
P. Sivaranjanil *et al.* [13]	Yes	---	---	Yes
Maolin Tang *et al.* [6]	Yes	---	Yes	---
Tung-Chieh Chen *et al.* [5]	Yes	---	Yes	---

CHAPTER-3 VLSI FLOORPLANNING

3.1 Role of VLSI Floorplanning in Physical Design

There are two methodologies top down and other one is bottom up for the VLSI design layout and often bottom up methodology is preferred over top down. There are plenty of libraries are available which contains large number of cells or if designer doesn't get the desired one then designer can design its own library and cells for composition of layout of the chip by placement and routing process. This lead to be bad implementation as wiring occupies very large portion of the area.

For the high quality design there must be requirement of the well-planned design methodology. Often planned designing methods are used for the software systems such as hierarchical, parallel processing etc. methods. These methods can also be used for the hardware implementation. But there are many differences in the hardware and software systems such as hardware system is tough in designing than software system, hardware systems has to be implemented in two or three dimensions while software systems can have any number of dimensions etc. Difficulties of designing increase with the increase in interconnection between the modules.

That is the reason why floorplanning based methodology is required. It is top down methodology so the all constraints of the layout are considered in all steps of the design.

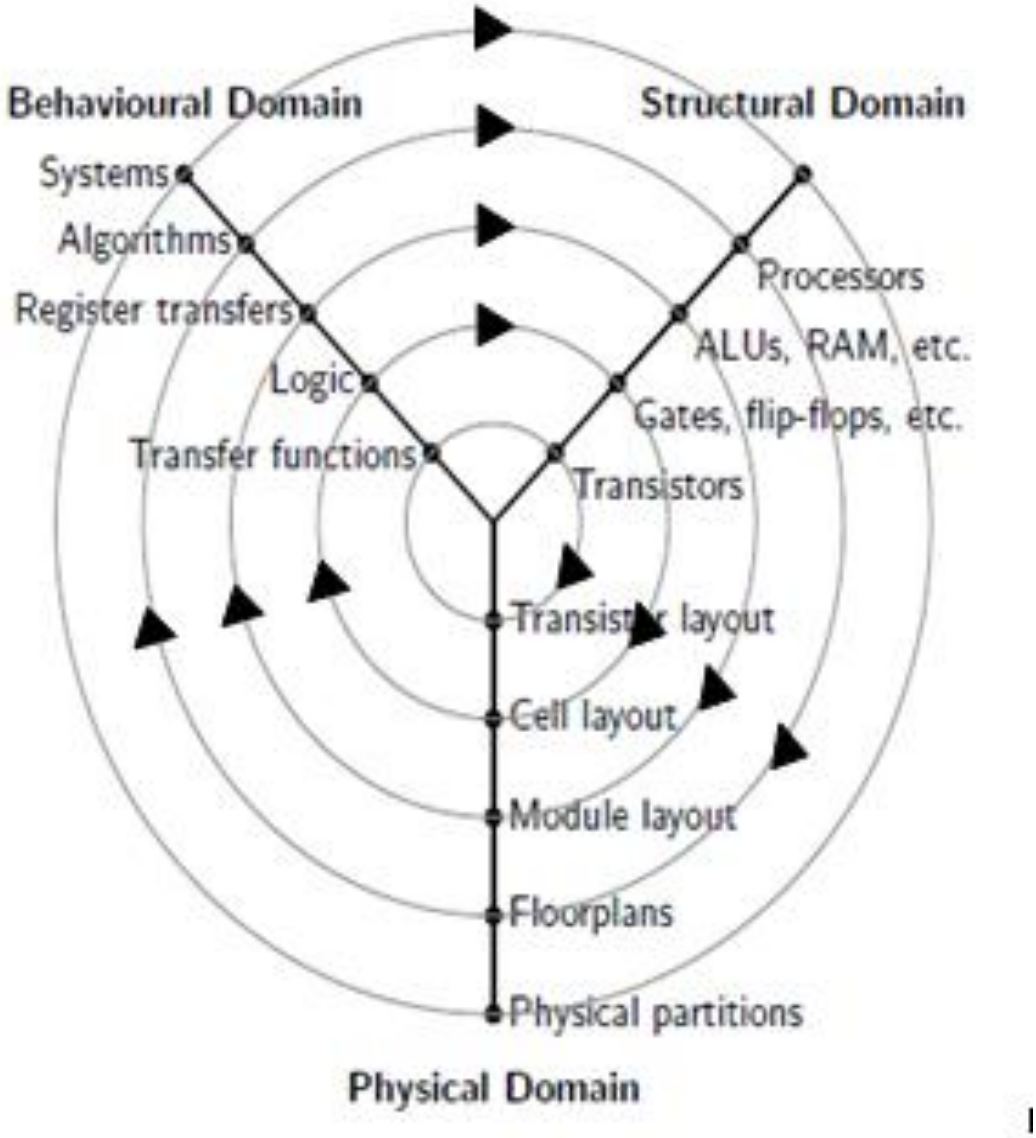


Figure 3 Gajski's Y-chart [1]

Figure 3 shows the Gajski's Y-chart which shows the methodology based on the floorplanning. From Figure 3, there are three domains in the Gajski's Y-chart and they are behaviour, structural and physical domain. A synthesis step must pass to the physical domain to go to the behaviour domain from the structural domain. In this constraints of layout are also taken into account so in structural domain too, some required correctness can also be evaluated for the layout early. Wirelength also can be estimated from the details of layout early so that timing, area and power consumption properties also can be evaluated. Timing and power consumption increase with the increase of wirelength so as increase in the parasitic capacitances between wires.

At the lowest abstraction, if the structural details like number of transistors and their interconnections are known them it is easy at layout stage. As the structural details are not available, to estimate the area occupied by various sub modules and to estimate the pattern of interconnection is done by the process called Floorplanning.

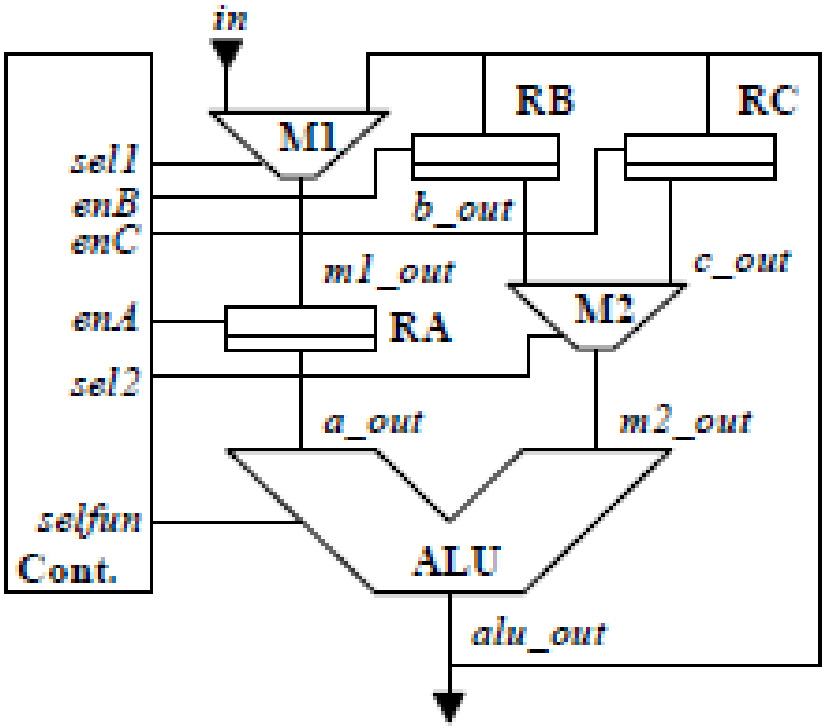


Figure 4 Structure of a Circuit [1]

Figure 4 illustrate the structural description of any circuit at resister transistor level. It contains three registers that are RA, RB, and RC, two multiplexers M1 and M2, a controller section and an arithmetic logic unit (ALU).

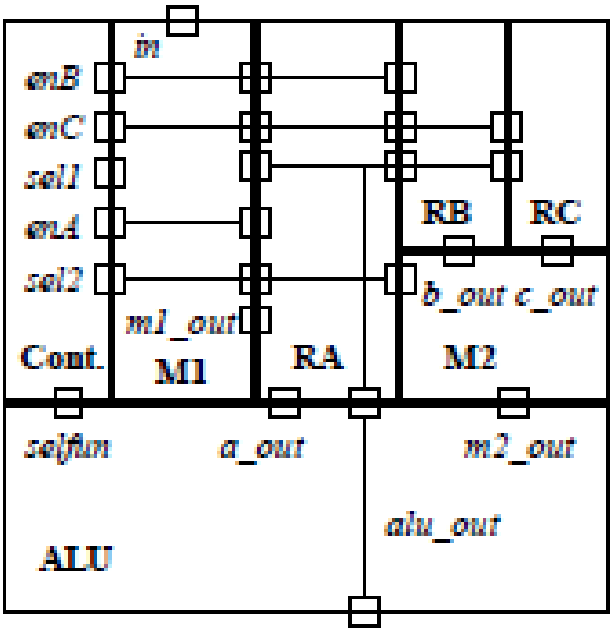


Figure 5 Floorplan of Circuit shown in Figure 4 [1]

Figure 5 illustrate possible floorplan of the circuit shown in Figure 4 with name of nets and cells of the structure. Small squares on the cell boundary indicate the ports. As in reality, there is more information in the floorplan than shown in the Figure 5. It is not necessary to shape the blocks such as adders and multiplexers shown in figure above but one can set any shape that fit best in the floorplan.

3.2 Factors to Be Considered While Making Floorplan

There are some factors that should be considered while doing floorplanning of any chip and they are described below

3.2.1 Shape of Modules

For the simplification of the floorplanning problem, rectangular shape of the modules is assumed but it is not necessary that shape of the block is always rectangular. Aspect Ratio (AR) is the ratio of the module height and module width and is used to

determine the shape of the module. To restrict the dimension of the module, there are upper bound and lower bound in the aspect ratio considered. L shape modules have been considered nowadays but they are expensive computationally.

3.2.2 Routing Area Consideration

Area required for the routing plays a very important role in the floorplanning process so it is necessary to consider it as the part of the process. To complete the routing process between the modules by the routing algorithms, it is necessary that placing of the module in such a way that space for routing must be left. If this is not done proper then placement process must be repeated.

3.2.3 Performance of the Circuit

It is very necessary that all critical nets routed with their timing constraints to get high performance of the circuit. There are some algorithms are used to find the minimum critical path length. Floorplanning process for the high performance is called as performance driven floorplanning.

3.2.4 Packaging Constraints

Heat coming out from the module is the major packaging constraint. As placement algorithm should place the modules such a way that modules that generate uniform heat place in a group. The modules that generate large amount of heat must be placed apart. By doing this, there may be trade off with high performance.

3.2.5 Pre-placed modules

There may be some cases in which some modules have their locations fixed earlier or they have specified their space earlier. In high performance chips, clock buffers must

be located at the centre as to minimise the time of arrival of clock to different modules.

3.3 Parameters for Optimisation in VLSI Floorplanning

In the cutting edge technology, optimisation of various parameters in the VLSI plays a very important role. The list of parameters that should be kept in mind while working on VLSI floorplanning is listed below.

3.3.1 Area

This is the main and very important parameters that must be optimised while making the floorplan. This is also said to be the first quality measures of the VLSI design. Estimation and minimization of the circuit area [15] in the floorplanning stage is very necessary as it can translate to reduction of the cost of silicon needed to implement the chip. There are secondary effects, like packaging area constraints, interconnect length consideration, routing space etc. which should be kept in mind in the context of area estimation.

3.3.2 Dead Space

Dead space is the area between the modules that are unused or say the area which is not occupied by any module of the chip. Dead space must be removed from the chip total area or minimised maximum so that total area of the chip can be reduced as much as possible. Earlier, dead space was not considered as there were very less number of modules in the chip but nowadays it must be considered as chip complexity is increased rapidly.

3.3.3 Interconnecting Wirelength

Congestion and wireability are the major terms play their role in the constraint of interconnecting wirelength. If the congestion at any region in the chip is more than the routing will be longer due to conflicts of various wiring and also wiring constraints.

That will also cause to degrade the performance. Reliability and yield are depended on the wire congestion. Potential of crosstalk and coupling noise are also high at the congested area. So it will cause failure of chip or lower reliability. Bridging effects cause due to high density wiring. Therefore, it is most like to estimate and predict wireability and wire congestion during the early stages in the physical design.

3.3.4 Speed

Speed is a key performance factor in the modern VLSI design practices. Micro-processor and real-time Application-Specific Integrated Circuit (ASIC) designs are the main driving forces for the optimisation of speed. Optimisation in speed is more difficult than optimisation of area because area is an aggregate parameter. Early delay optimization study focused on reduction of delay in a unit delay model, which assumes unit delay for logic gates and zero delay for interconnects. These methods aim at minimisation of the maximum number of logic gates along any inputs to the outputs path, or path between registers. Such a unit delay model is mostly used to optimise delay for technology independent.

3.3.5 Power Consumption

Approaches to power consumption optimisation include system and architectural level methods such as multiple supply voltage designs, bus-encoding, hardware/software partitioning. Some power management and sleep mode exploitation techniques, register transfer level techniques such as power conscious allocation and binding and clock gating are used .Some logic level transformations techniques such as activity-driven gate and logic decomposition and technology mapping ,precomputation-based techniques, layout level techniques such as gate and wire sizing and power conscious placement and partitioning and circuit-level techniques such as dual threshold and low power logic families, adiabatic and charge recovery circuit families, dual edge-triggered flip-flips and circuit techniques for low-power data path elements Most effective way to reduce power consumption in CMOS technology is to lower the supply voltage, which exploits the quadratic dependency of dynamic power on

voltage. Reducing the supply voltage, however, leads to increased delay and decreased clock speed.

3.4 Floorplan Representation

The floorplan can be represented with the hierarchical approach. The cells at the lowest level of hierarchy is called leaf cell and these leaf cells also built the cell for upper hierarchy. Cells that build using lower level cells are called as composite cells. Composite cells can contain both leaf as well as composite cell. So the hierarchy levels can be arbitrary. Subcell of the composite cells are called as children cells and the cells except the cells which are represents the complete chip has parent cell. As the simplicity of the floorplanning, assumption is made that both leaf and composite cells are the rectangular in shape.

Floorplan can be represented as follows.

3.4.1 Slicing Floorplan

Slicing floorplan is the floorplan in which by cutting or bisecting the composite cells horizontally or vertically, the children of the composite cells can be obtained. So in the slicing floorplan by putting the children next to the other children from left to right or on the top, composite cell can be made.

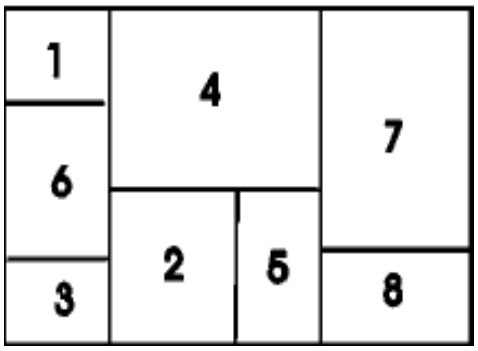


Figure 6 Slicing Floorplan

Figure 6 shows the slicing floorplan of simple circuit. The circuit have total eight modules numbered from 1 to 8. By bisecting the floorplan as vertical or horizontal, each module can be obtained easily.

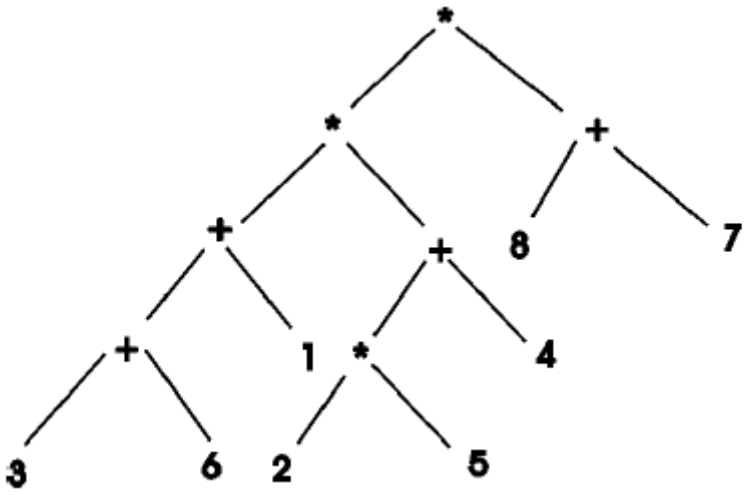


Figure 7 Slicing Tree Representation of the Figure 6 Floorplan

Figure 7 illustrate the representation of the slicing floorplan shown in Figure 6 by using the slicing tree data structure.

A slicing floorplan can be represented in the form of a binary tree, called a slicing tree, in which each internal node of the tree is labelled either * or + , corresponding to a vertical or a horizontal cut respectively. Each leaf represents a basic rectangle and is labelled between 1and n, where n is the total number of basic rectangles

36+25*4+87+*

A slicing tree can be represented, alternatively, using a postfix expression. The postfix expression is derived by carrying out a postorder traversal.

A normalized postfix expression is obtained by traversing a skewed slicing tree in postorder and is characterized by chains of {*, +} operators in which the operators

alternate. For example, the postfix expression 1 2 3 + * 4 * is normalized, but the expression 1 2 3 + +8 is not (because of the two adjacent symbols). A slicing floorplan with cuts will produce basic rectangles. Thus a postfix expression consists of exactly entries.

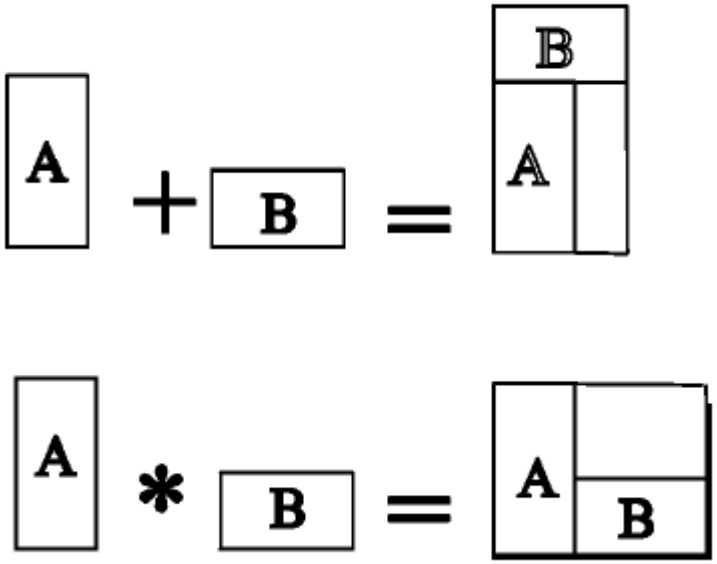


Figure 8 Binary Operations on Slicing Floorplan

Figure 8 shows the actions of the binary operations and on the two rectangles + and *. '+' puts B on top of A and '*' puts B on the right of A. In the example depicted in Fig. 4, the two rectangles and combine under and to form L-shaped modules.

Figure 4 is also one of the examples of the slicing floorplan. The slicing tree for the circuit can be obtained as below.

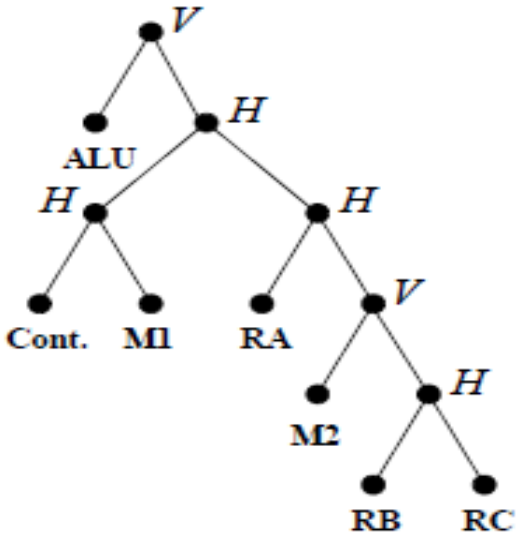


Figure 9 Slicing Tree Representation of the Figure 4 circuit

Leaf cells are the leaves of the tree while nodes are referred as the cells. H in Figure 9 illustrates horizontal composition while V in Figure 9 illustrates vertical composition. Ordering of the tree composition is such as for horizontal composition left to right and for vertical composition bottom to top.

3.4.2 Non-Slicing Floorplan

Nowadays all floorplan are coming in the category of the non-slicing floorplan. Non-slicing floorplan also called as wheel or spiral floorplan. This kind of floorplan cannot obtain its children cells by cutting or bisecting the floorplan vertically or horizontally. For this non-slicing, composing of minimum five cells required for one composite cell. That's why this also called as floorplan of order 5.

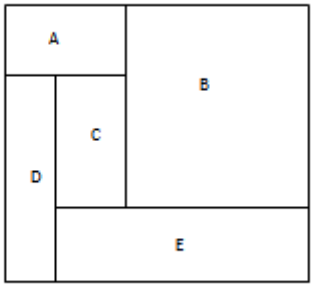


Figure 10 Non-Slicing Floorplan

Figure 10 shows the wheel or spiral or non-slicing floorplanning. It is of order 5 and one of the simple floorplan. It cannot bisect horizontal or vertical. The modules inside the floorplan are labelled as A, B, C, D and E.

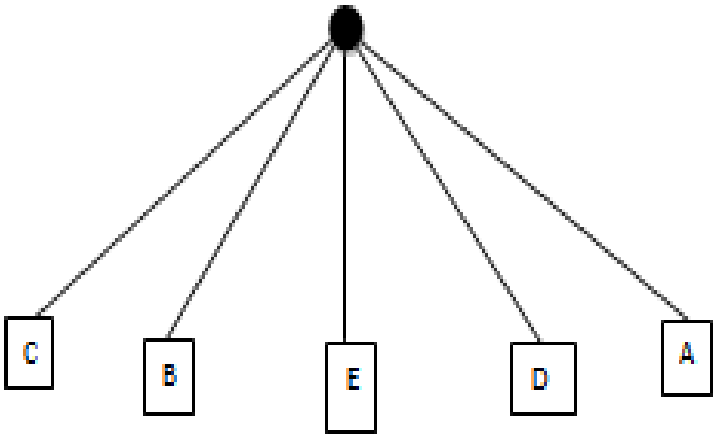


Figure 11 Floorplan Tree of Order 5 Non-Slicing Floorplan

Figure 11 illustrate the floorplan tree of the non-slicing floorplan shown in Figure 10. Floorplan with higher order than 5 can also exist. Hierarchical representation of the floorplan can be used for the floorplan that have more number of orders than 5. Polar graph is the representation mechanism for any floorplan. Polar graph has two directed graph that are vertical and horizontal polar graph.

To connect two cells electrically, their terminal must be correctly ordered. Cells which are connected without use of routing and by placing against each other are called as abut. These kinds of cells are shown in figure below.

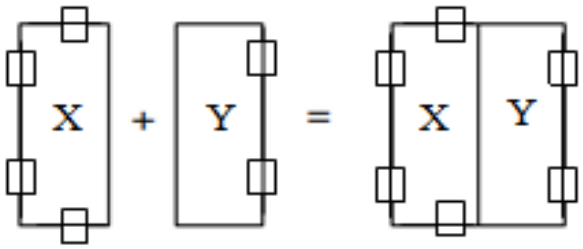


Figure 12 Abut Cells Representation

CHAPTER-4 OVERVIEW OF VLSI FLOORPLANNING PROBLEM AND ITS DATA STRUCTURE REPRESENTATION

4.1VLSI Floorplanning Overview

In VLSI physical design flow, after the process of partitioning in which the complex circuit is subdivided into modules, the very important role of floorplanning comes into the picture. A netlist which specifies the interconnection of modules, to make floorplan of arbitrary sized module in such a way that the modules cannot overlap with each other optimisation of the overall area consumed by all modules, minimisation of dead space that is the space which is unoccupied by modules and reduction of interconnecting wirelength. For many years, floorplanning problem is solved for optimising of chip area and interconnecting wirelength only and all classical techniques are used for the same while dead space plays very important role in overall performance of chip floorplan. In this paper the minimisation of dead space is also emphasised. VLSI floorplanning problem is NP hard problem and to find the optimal solution by globally exploration of search space is impossible as search space is increasing exponentially with the scale of circuit.

Heuristics such as genetic algorithm [16], particle swarm algorithm (PSO) [17], Memetic algorithm, simulated annealing [18-20], and hybridisation of simulated annealing [9], hybridisation of PSO [13] are widely used for the non-slicing VLSI floorplanning. Temperature awareness floorplanning [21] also obtained in. In thermal awareness [22-24] was also taken into consideration while simulated annealing used for optimisation. Geometric programming [19] was also used for mathematical modelling to solve the floorplanning problem. In a heuristic algorithm used with two concepts corner-occupying action and caving degree. An ant colony technique also has been adopted for the multiple objective of VLSI floorplanning [25].

4.2 BINARY SEARCH TREE or B*-Tree overview

The term Binary search tree is a hierarchy model of data which indicates nodes at different level. Moreover, each node has two child nodes: Left child node and Right child node, whereas the left node must be less than their parent node and right node must be greater than their parent node. Basically, each node related to this tree are carried one code either it is number or any key code. It is also very efficient algorithm for computer science for updating or checking in a traversal manner.

The whole structure can be explained by general example as shown in the following figure.

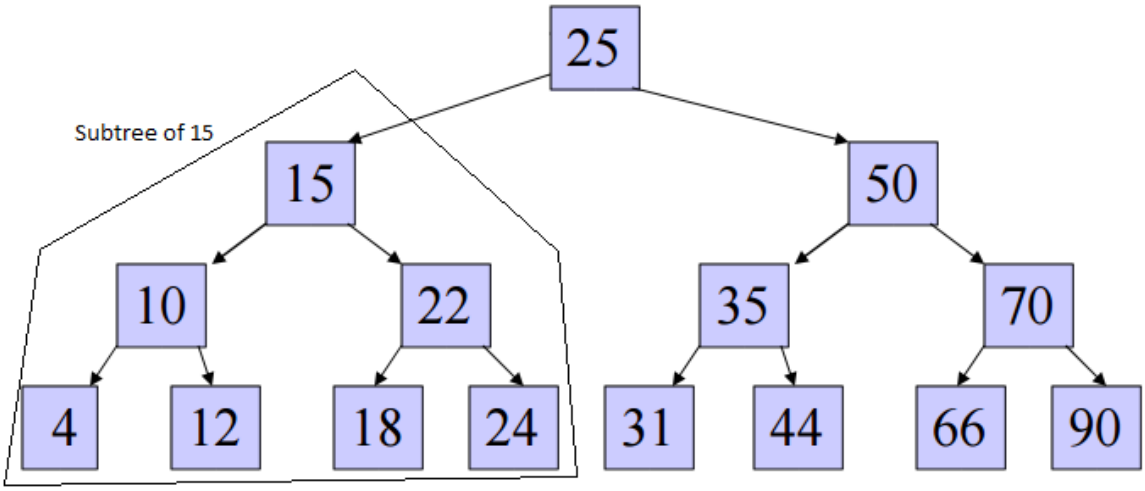


Figure 13 B*-tree Structure

In the B*-TREE (Binary Search tree), the tree algorithm has four basic terminologies which are Root, Parent node, Child node, sibling node, leaf and subtree. These all terms are explained with the help of the above figure. According to above figure the first node which has 25 values is known as a Root, it is always top in the hierarchy. Then after, parent node which has two different nodes known as child nodes. As per the example, when 15 is known as the parent node and it have two child node 10 and 22. In addition for node 22, node 10 is sibling node and vice versa. The next term, leaf node has no child node, for an example node 18. Finally, last term subtree, every node in B*-TREE is a subtree of the B*-TREE rooted at that node. For instance, subtree of 15.

B*-TREE (Binary Search tree) can be defined three different traversal order; Inorder, Preorder and postorder. However, traversal property can be representing by the

ordering three items: Left side node, main node and right side node and in this case left side node always comes before right side node. These all three order are differentiating with their arrangement of three basic nodes of the B*-TREE.

4.2.1 *Inorder*

In this order the first order starts from the left side node than main node and last right side node

4.2.2 Preorder

In this order the first order starts with main node than left side node and last right side node.

4.2.3 Postorder

In this order the first order starts with left side node than right side node and last main node.

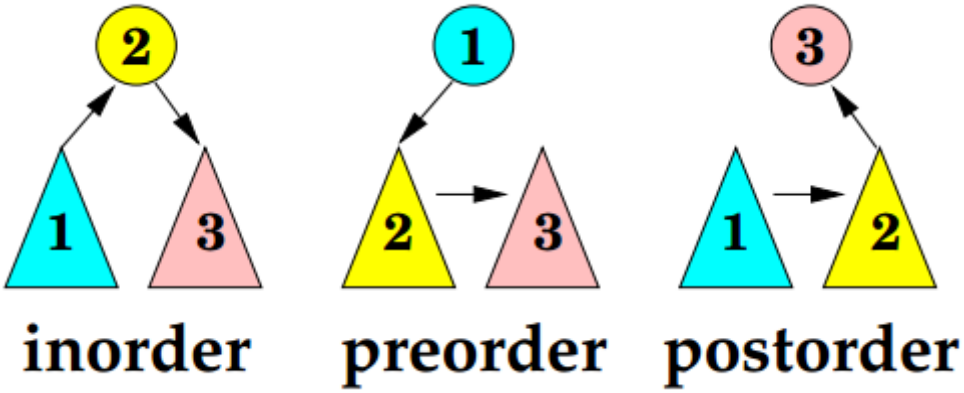


Figure 14 Ordering of B*-tree

4.3 Operations on B*-tree

There are three basic operations occurs in B*-TREE,

4.3.1 Searching component

There are few steps which is helpful to find out a component:

- Starts examine from the root node as current node.

- If the search value matches with the root (or current) node's value than found match, otherwise go to next step.
- If the search value greater than the root (or current) node's value than follow the right child node, if it is match with any node values than the search value exits otherwise it not exists in this tree.
- Similarly, if the search value less than the root (or current) node's value than follow the left child node, if it is match with any node values than the search value exits otherwise it not exists in this tree.

In this case, the process starts with the root node that time root node should be consider as current node, afterwards current value will differentiate at each level of tree. For an example, if user searching for 23 value in above figure then, starting root value is 25 which is greater than search value, so user should follow left side, on this level search value 23 is greater than current value means 15 so on that case user need to go right side which is 22.

4.3.2 Insert the new component

To insert the new component process is also like a searching a new component. In this process if the new component is less than the current node than first look it has a left child node or not if it has no child component than add new one here. Likewise, if the new component is greater than the current node than first look it has a right child node or not if it has no right child component than add new one here.

4.3.3 Delete the component

There are three different cases to delete the components,

- If the deleting node has no child node that simply replace that node to nil components.
- If the deleting node has only one child node, then replace that child node to that place.
- If the current node has two child nodes, then replace it with its left side node. If that left side node is leaf, then there is no issue but if that has child node then follow the upper two cases.

4.4 Problem Statement of Non-Slicing VLSI Floorplanning

A complex chip is partitioned into modules and in VLSI there are two types of modules which are as follows. One is hard module and other is soft module. Hard module is the module in which the area of module and aspect ratio that is width divided by length, are fixed while soft module is the module in which aspect ratio can be varied but area is fixed.

From a set of module s= {m1, m2... m_n} where m_j is specified with height h_j and width w_j, 1< j <n, aspect ratio and area of individual module is determined. A netlist specifies the interconnection of modules. A floorplan f is an arrangement of the module set S such a way that the overlapping of modules is avoided by using bottom left approach. Bottom left approach is nothing but the placing of each module in such a way that the module cannot shift to its left and its bottom in the floorplan f.
A non-overlapping floorplan has a cost function which consists of cost of area and cost of interconnection wirelength. Area cost is the division of total area consumed by all modules including dead space optimisation to the total area of benchmark before optimisation. The wirelength cost is the normalised wirelength that is the wirelength calculated after optimisation divided by initial wirelength. So the cost function is as follows.

$$\textbf{Cost function } (\mathbf{C}) = \mathbf{a} * \textbf{ area cost } + (\mathbf{1} - \mathbf{a})\textbf{wirelength cost} \quad (1)$$

In equation (1), parameter 'a' is the user defined constant. As per our optimisation objective, the value of 'a' is decided between 0 and 1.

4.5 B*-Tree Representation of the Non-Slicing VLSI Floorplanning

The need of B*-tree comes with the requirement of any data structure to represent the non-slicing floorplan. There are many representation techniques in data structure like O-tree [26], slicing tree, sequence pair, cut tree, corner sequence [27], modified corner list and partial 3- trees have been used since. O-tree results in closely packed floorplan [26]. B*-tree has many properties listed as follows which makes it the best for the VLSI non-slicing floorplan over many other conventional representation like

O-tree [28] , slicing tree and other binary decision trees. In this paper, the B*-tree is used as it is ordered binary tree as well as self-balancing search tree. A unique B*-tree can be constructed to arrange each module in the circuit [5]. As stated earlier in this paper, the bottom left approach is used so the root node of the B*-tree is at bottom left corner. Once the module, corresponds the root node is set, the right sided module is going to be arranged using same approach. One VLSI non-slicing circuit having some modules is shown in A part of Figure 13 and in B part of Figure 13, corresponding B*-tree is derived.

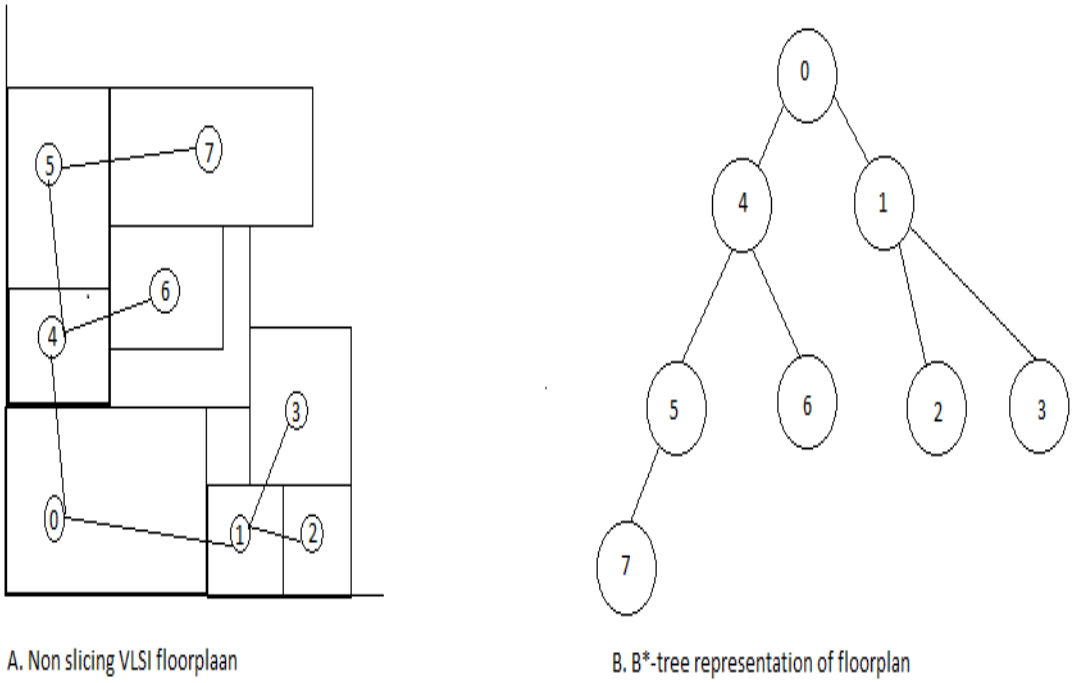


Figure 15 Non-slicing VLSI Floorplan with its B*-tree Representation

From above figure of B*-tree representation of non-slicing VLSI floorplan, module 0 is the root node having the origin coordinates and so at the top of the tree structure. Node 4 and node 1 are the two children of the root node 0 shown in Figure 3. Node 1 has two child node 2 and node 3. Node 4 is the parent of node 6 and node 5. Node 7 can be the child of node 5 and node 6 and according to the optimisation of our floorplan, objectives decides whether it is child of node 5 or node 6, here in this

Figure 13, it is child of node 5.Some space is also there in between some modules, that space are nothing but the dead space which must be minimised.

B*-tree has many operations such as insert node, delete node, swap node, perturb and search which all are used in this research work. B*-tree with these operations are coded in C++ language and tested in Ubuntu 14.04 GNU GCC compiler. B*-tree takes O (log (n)) computation complexity time for all its operations. Details of various data structure tree are shown in Table 2.

Table 2 Comparison of Various Representation of VLSI Floorplan

Representation of Floorplan	**Computation Complexity**	**Search Space**
Ordered Binary Tree (B*-Tree)	$O(n)$	$O[(n!2^{2n-1})/n^{1.5}]$
Ordered tree (O-tree)	$O(n)$	$O[(n!2^{2n-1})/n^{1.5}]$
Sequence pair	$O(n^2)$	$O\ ((n!)^2)$
Transitive Closure Graph-Sequence (TCG-S)	$O(n(\log(n)))$	$O\ ((n!)^2)$

CHAPTER-5 HIGH SPEED SIMULATED ALGORITHM (HSSA) FOR NON-SLICING VLSI FLOORPLANNING

5.1 Simulated Annealing (SA) Approach

Algorithms are very powerful concepts for solution of many complex problems [29]. There are many evolutionary algorithms are used for the non-slicing VLSI floorplanning till now such as genetic algorithm, combine genetic and simulated annealing algorithm, memetic algorithm [30] , particle swarm algorithm and linear programming but the simulated annealing algorithm is one which is simple to understand and can be modified easily for the purpose of application and optimisation. Hierarchical approach for the floorplanning [31-33] has been used if the number of nodes or modules increased. Fixed-outline floorplanning have been used by many as the total area in which all modules should be packed is fixed [34].

From many years simulated annealing is used for various applications and VLSI floorplanning optimisation problem is one of the applications of the simulated annealing algorithm. Use of simulated annealing in VLSI floorplanning is not the novel thing but improvement of the simulated annealing algorithm for better and better optimisation is necessary as the complexity of the chip is increased very rapidly. In this dissertation, conventional simulated annealing is modified to implement new simulated annealing algorithm which not only improved the optimisation of area, dead space and interconnecting wirelength but also increased the speed of the operation of making optimised floorplan.

Annealing is the term described in thermodynamics which is metallurgical process where metals are first heated up and then cooled to reach in low energy state. At that low energy state, metals are very strong in nature. In the simulated annealing, temperature is scaled down slowly and at high temperature to start a random search ultimately becoming real acquisitive drop when temperature reaches zero. More

difficult targets are likely to be in high temperatures. As the algorithm proceeded, an annealing schedule is developed using adjustment of parameters for the well organised reduction of temperature.

Basic algorithm for any optimisation application is as follows. First, a random solution is generated based on the parameters given initially. Then from user defined cost function, its cost is evaluated. After that another solution is generated from random neighbour and its cost is evaluated. Then if the new cost is well optimised from the previous one then cost function updated according to new cost else the process is repeated for other neighbouring solution.

5.2 HSSA Algorithm Using B*-Tree With Its Methodology

The proposed simulated algorithm for non-slicing VLSI floorplan is shown in Figure 4.

Algorithm: High speed simulated annealing using B*-tree

Input: A set of modules and netlist (file formats available are .nets and .blocks).

Output: A floorplan which satisfies the objectives of optimisation of area, interconnecting wirelength, dead space as well as reduction in time.

1 Initialise B*-tree with its parameters like swap rate and rotate rate for the input modules;
2 Run the high speed simulated annealing;
3 Set initial temperature T and initialise time constraints
4 **do**
5 B*-tree perturb;
6 B*-tree packing module;
7 B*-tree get cost;
8 Modification of cost function
9 Update temperature T;
10 **until** converged or cooled enough;
11 **return** best solution and compute time

Figure 16 HSSA Algorithm for VLSI Non-Slicing Floorplanning

As shown in figure the input to the process is set of modules and the netlist. This information of MCNC benchmark circuits is available globally with the files that have the extensions as .blocks, .nets and .pl format. To get the information from these files, the benchmarks circuits must be converted in the format which is readable for example .txt extension.

B*-tree data structure is used in this research work so all modules must be converted to the corresponding graph where one of the node is considered as the root node of the tree graph.

Perturb procedure is also one of biological process in which each gene has to prove its fitness in the present condition to maintain unchanged in the next condition or generation. Let's say S is the current state and M is the group of element. One j belongs to M, if the change is happened from S to with respect to j causes the change in S (j). So the change in S (j) is considered as S' (j) and S' (j) ≠ S (j). Newly generate state S' from a move of S may not be the state as there are many state-constraints so it may be violate it. So there is ordering of the elements which are movable in the M and according to the problem choice of ordering done. Before next iteration started, function must be converted to the state. Ordering is done during each call of perturb.

Reduction in cost due to move of element cause will define in terms of gain.

$$\mathbf{Gain}\ (\mathbf{j}) = \mathbf{Cost}\ (\mathbf{S}) - \mathbf{Cost}\ (\mathbf{S}') \qquad (2)$$

Once each module has undergo to the perturb process, it must be packed into the tree according to their ordering decided by the perturb operation. Fitness function expressed in equation (1) is the cost function used for the ordering of the modules in the B*- tree.

Once module is packed at somewhere in the tree then its cost function is stored to compare with the forthcoming modules fitness function. If the next module has the better fitness function then previous one then cost function must be updated.

After this process, temperature is evaluated again and updated and new iterations will be started for the updated temperature. Status of the annealing is also checked at the end of the process as it is cooled enough or converged. In the last step, the outputs are obtained in form of the best solutions.

For the computer time evaluation clock operator with zero time is started at the beginning and at the end of the process it is evaluated by the ending the clock. So difference between ending time and starting time of the clock will give the computation time.

CHAPTER-6 NON-SLICING VLSI FLOORPLANNING RESULTS USING MCNC BENCHMARK CIRCUITS

High speed simulated annealing using B*-Tree is applied to the MCNC benchmark circuits. MCNC benchmarks are well known benchmarks and used for the verification of the implemented algorithm. Each benchmark circuit contains some modules along with their interconnection details. The benchmarks which are used in this research work are ami33, ami49, apte, hp and Xerox. Table 2 below shows the overall experimental result achieved.

Table 3 Experimental Results using MCNC Benchmark Circuits

Benchmark Circuits	**Number of Modules**	**Aspect ratio**	**Total Area (mm^2)**	**Used Area (mm^2)**	**Wire length (m)**	**Dead Space (%)**	**Time (seconds)**
Ami33	33	1.0383	12.38	1.15	548	6.61%	1.534
Ami49	49	1.1852	41.17	35.44	868	13.91%	2.543
Apte	9	1.20	48.21	46.51	437	3.42%	0.065
Hp	11	1.2666	94.40	8.83	219	6.45%	0.147
Xerox	10	1.3602	20.75	19.35	401	6.75%	0.245

Table 4 Comparison with Other Algorithms in Terms of Area and Wirelength

Benchmark Circuits \ Algorithm		In this paper	PSO based intelligent algorithm	O-tree	Enhanced O-tree	TCG	CS
Apte	**Area (mm^2)**	46.51	47.31	51.9	52	48.5	48.5
	Wire (mm)	437	263	321	321	378	380
Xerox	**Area (mm^2)**	19.35	20.2	20.4	20.4	20.4	20.4
	Wire (m)	401	477	381	381	385	381
Hp	**Area (mm^2)**	8.83	9.5	9.5	9.4	9.5	9.6
	Wire (m)	219	136	153	152	152	149
ami33	**Area (mm^2)**	1.15	1.28	1.28	1.3	1.24	1.25
	Wire (m)	54	69	51	52	50	48.1
ami49	**Area (mm^2)**	35.44	38.8	39.6	39.9	38.2	38.2
	Wire (m)	868	880	689	703	663	690

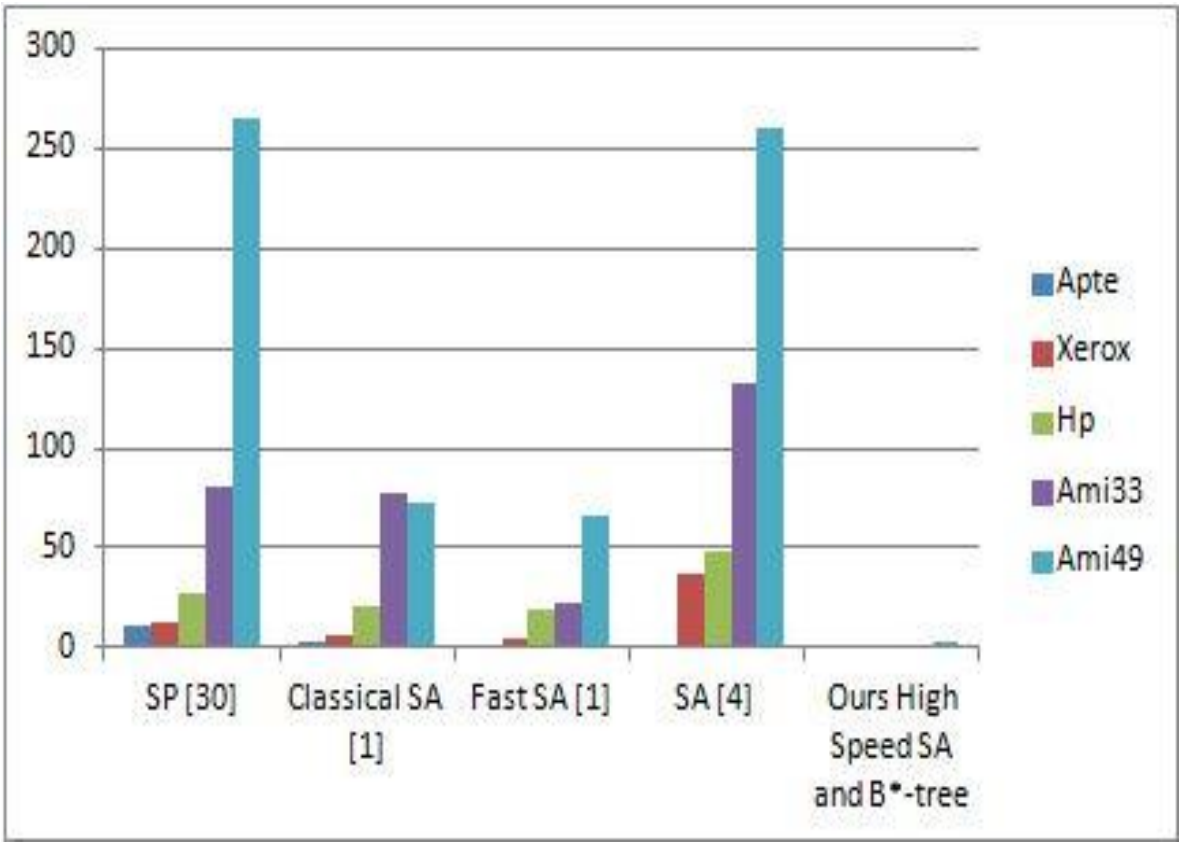


Figure 17 Graphical Representation of Comparison of Computational Time

Table 5 Comparison of Computation Time

Benchmark Circuits	Number of Modules	SP [30] (s)	Classical SA [1] (s)	Fast SA [1] (s)	SA [4] (s)	Ours High Speed SA (s)
Apte	9	11	3	2	0.66	0.065
Xerox	10	12	7	5	36.47	0.245
Hp	11	28	21	20	48.33	0.147
Ami33	33	81	78	22	132.05	1.534
Ami49	49	265	72	66	260.09	2.543

CHAPTER-7 CONCLUDING REMARKS AND FUTURE SCOPE

7.1 Conclusion

Scaling in chip increases day by day and to place components at the right place is becoming very complex. Many years ago when there were less amount of modules available to place on the chip, floorplanning was done manually. Nowadays to make a floorplan manually is impossible as there are millions of components to be placed on the single chip. So there must be any heuristic algorithm required to make floorplan on software and then apply it to hardware. Optimisation of area and wirelength are not enough but the minimization of dead space is also required. In this paper the speed constraints are also taken into consideration. An effective high speed and the best optimised results are achieved on GCC compiler of Ubuntu 14.04 machine using the MCNC benchmark circuits which are also compared with other previous results of many researchers. The B*-tree representation gives the best representation here as compared to other data structure representation techniques.

7.2 Future Scope

Technology is scaling very rapidly day by day so as the inventions and innovations are increasing rapidly in each and every field of VLSI. From the previous work of VLSI floorplanning, new level of optimisation and that too with the high speed is done in this research work.

Some ideas for future work are as bellows.

- In the future work, this work can be carried out by converting 2D floorplan with 3D floorplan as there are going to be innovations done in 3D techniques. Area, power, dead space and interconnecting wirelength can be optimised more using 3D floorplanning.
- In the future work, many other effective evolutionary algorithms or hybridisation of algorithms can be used / to solve this problem

REFERENCES

[1] S.H. Gerez, *Algorithms for VLSI Design Automation*, 1998, John Wileys

[2] N.A.Sherwani, *Algorithms for VLSI Physical Design Automation*, 2002, Kluwer

[3] C. S. Ying and J. S. L. Wong, "An analytical approach to floorplanning for hierarchical building blocks layout [VLSI]," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 4, pp. 403-412, Apr 1989.doi: 10.1109/43.29594

[4] S. Nakaya, T. Koide and S. Wakabayashi, "An adaptive genetic algorithm for VLSI floorplanning based on sequence-pair," *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, Geneva, 2000, pp. 65-68 vol.3.doi: 10.1109/ISCAS.2000.855997

[5] Tung-Chieh Chen and Yao-Wen Chang, "Modern floorplanning based on B*-tree and fast simulated annealing," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 4, pp. 637-650, Apr 2006.
doi: 10.1109/TCAD.2006.870076

[6] M. Tang and X. Yao, "A Memetic Algorithm for VLSI Floorplanning," in *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 1, pp. 62-69, Feb 2007.

[7] J. Z. Yan and C. Chu, "DeFer: Deferred Decision Making Enabled Fixed-Outline Floorplanning Algorithm," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 3, pp. 367-381, Mar 2010.doi: 10.1109/TCAD.2010.2041850

[8] V. Krishnan and S. Katkoori, "TABS: Temperature-Aware Layout-Driven Behavioral Synthesis," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 12, pp. 1649-1659, Dec 2010.doi: 10.1109/TVLSI.2009.2026047

[9] J. Chen, W. Zhu and M. M. Ali, "A Hybrid Simulated Annealing Algorithm for Nonslicing VLSI Floorplanning," in *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 4, pp. 544-553, Jul 2011.doi: 10.1109/TSMCC.2010.2066560

[10] J. M. Lin and Z. X. Hung, "SKB-Tree: A Fixed-Outline Driven Representation for Modern Floorplanning Problems," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 3, pp. 473-484, Mar 2012. doi:10.1109/TVLSI.2011.2104983

[11] C. S. Hoo, J. Kanesan and H. Ramiah, "Enumeration technique in very large-scale integration fixed-outline floorplanning," in *IET Circuits, Devices & Systems*, vol. 8, no. 1, pp. 47-57, Jan 2014.doi: 10.1049/iet-cds.2013.0003

[12] A. Mahabadi, A. Khonsari, B. Khodabandeloo, H. Noori and A. Majidi, "Critical path-aware voltage island partitioning and floorplanning for hard real-time embedded systems," in *Journal Integration, the VLSI Journal*, vol. 48, pp. 21-35, Jan 2015

[13] P. Sivaranjani, A. Senthil Kumar," Thermal-Aware Non-slicing VLSI Floorplanning Using a Smart Decision-Making PSO-GA Based Hybrid Algorithm," in *Circuits, Systems, and Signal Processing*, vol.34, no. 11, pp. 3521–3542, Nov 2015. doi: 10.1007/s00034-015-0020-x

[14] N. R. D. Gracia, S. Rajaram., Nivethitha and A. Sudarsan, "Thermal aware modern VLSI floorplanning," *Devices, Circuits and Systems (ICDCS), 2012 International Conference on*, Coimbatore, 2012, pp. 187-190.doi: 10.1109/ICDCSyst.2012.6188701

[15] Wei-Ming Dai and E. S. Kuh, "Simultaneous Floor Planning and Global Routing for Hierarchical Building-Block Layout," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 6, no. 5, pp. 828-837, Sep 1987.doi: 10.1109/TCAD.1987.1270326

[16] J. P. Cohoon, S. U. Hegde, W. N. Martin and D. S. Richards, "Distributed genetic algorithms for the floorplan design problem," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 4, pp. 483-492, Apr 1991.doi: 10.1109/43.75631

[17] G. Chen, W. Guo and Y. Chen, “A PSO-based intelligent decision algorithm for VLSI floorplanning,” in *Soft Computing*, vol. 14, no. 12, pp. 1329–1337, Sept 2009

[18] Y. Han and I. Koren, “Simulated annealing based temperature aware floorplanning,” in *Journal of Low Power Electronics*, Vol. 3, no. 2, pp. 1–15, Jun 2007.doi:10.1166/jolpe.2007.128

[19] S. Anand, S. Saravanasankar and P. Subbaraj, “Customized simulated annealing based decision algorithms for combinatorial optimization in VLSI floorplanning problem,” in *Computational Optimization and Applications*, vol. 52, no. 3, pp. 667–689, Jul 2012.

[20] Z. Lichen, Y. Runping, C. Meixue, J. Xiaomin, L. Xuanxiang and D. Shimin, "An Efficient Simulated Annealing Based VLSI Floorplanning Algorithm for Slicing Structure," *Computer Science & Service System (CSSS), 2012 International Conference on*, Nanjing, 2012, pp. 326-330.doi: 10.1109/CSSS.2012.89

[21] W. L. Hung, Y. Xie, N. Vijaykrishnan, C. Addo-Quaye, T. Theocharides and M. J. Irwin, "Thermal-aware floorplanning using genetic algorithms," *Sixth international symposium on quality electronic design (isqed'05)*, 2005, pp.634-639.doi: 10.1109/ISQED.2005.122

[22] Y. Li, Y. C. Chen and H. W. Cheng, "Temperature Aware Floorplanning via Geometry Programming," *Computational Science and Engineering Workshops, 2008. CSEWORKSHOPS '08. 11th IEEE International Conference on*, San Paulo, 2008, pp. 295-298.

[23] E. K. Ardestani, A. Ziabari, A. Shakouri and J. Renau, "Enabling power density and thermal-aware floorplanning," *Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM), 2012 28th Annual IEEE*, San Jose, CA, 2012, pp. 302-307. doi: 10.1109/STHERM.2012.6188864

[24] D. Cuesta, J.L. Risco-Martin, J.L. Ayala and J.I. Hidalgo, “3D thermal-aware floorplanner using a MOEA approximation” in *Journal Integration, the VLSI Journal*, vol. 46, no. 1, pp. 10–21, Jan 2013

[25] C.-S. Hoo, K. Jeevan, V. Ganapathy and H. Ramiah, “Variable-order ant system for VLSI multiobjective floorplanning,” in *Journal Applied Soft*

Computing, vol. 13, no. 7, pp. 3285-3297, Jul 2013. Doi: 10.1016/j.asoc.2013.02.011

[26] M. Tang, A. Sebastian, "A genetic algorithm for VLSI floorplanning using O-tree representation," in *Applications of Evolutionary Computing* (Springer, Berlin,2005), pp. 215–224

[27] Jai-Ming Lin, Yao-Wen Chang and Shih-Ping Lin, "Corner sequence - a P-admissible floorplan representation with a worst case linear-time packing scheme," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 4, pp. 679-686, Aug 2003.doi: 10.1109/TVLSI.2003.816137

[28] Pei-Ning Guo, Chung-Kuan Cheng and T. Yoshimura, "An O-tree representation of non-slicing floorplan and its applications," *Design Automation Conference, 1999. Proceedings. 36th*, New Orleans, LA, 1999, pp. 268-73.doi:10.1109/DAC.1999.781324

[29] T. Corman, C. E. Leiserson and R. Rivest, *Introduction to Algorithms*, 1990, McGraw Hill

[30] H. Ishibuchi, T. Yoshida and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," in *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204-223, Apr 2003.doi: 10.1109/TEVC.2003.810752

[31] S. N. Adya and I. L. Markov, "Fixed-outline floorplanning: enabling hierarchical design," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 6, pp. 1120-1135, Dec 2003.doi: 10.1109/TVLSI.2003.817546

[32] J. de San Pedro, J. Cortadella and A. Roca, "A hierarchical approach for generating regular floorplans," *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, San Jose, CA, 2014, pp. 655-662.doi: 10.1109/ICCAD.2014.7001422

[33] T. Lengauer and R. Muller, "Robust and accurate hierarchical floorplanning with integrated global wiring," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 6, pp. 802-809, Jun 1993.doi: 10.1109/43.229754

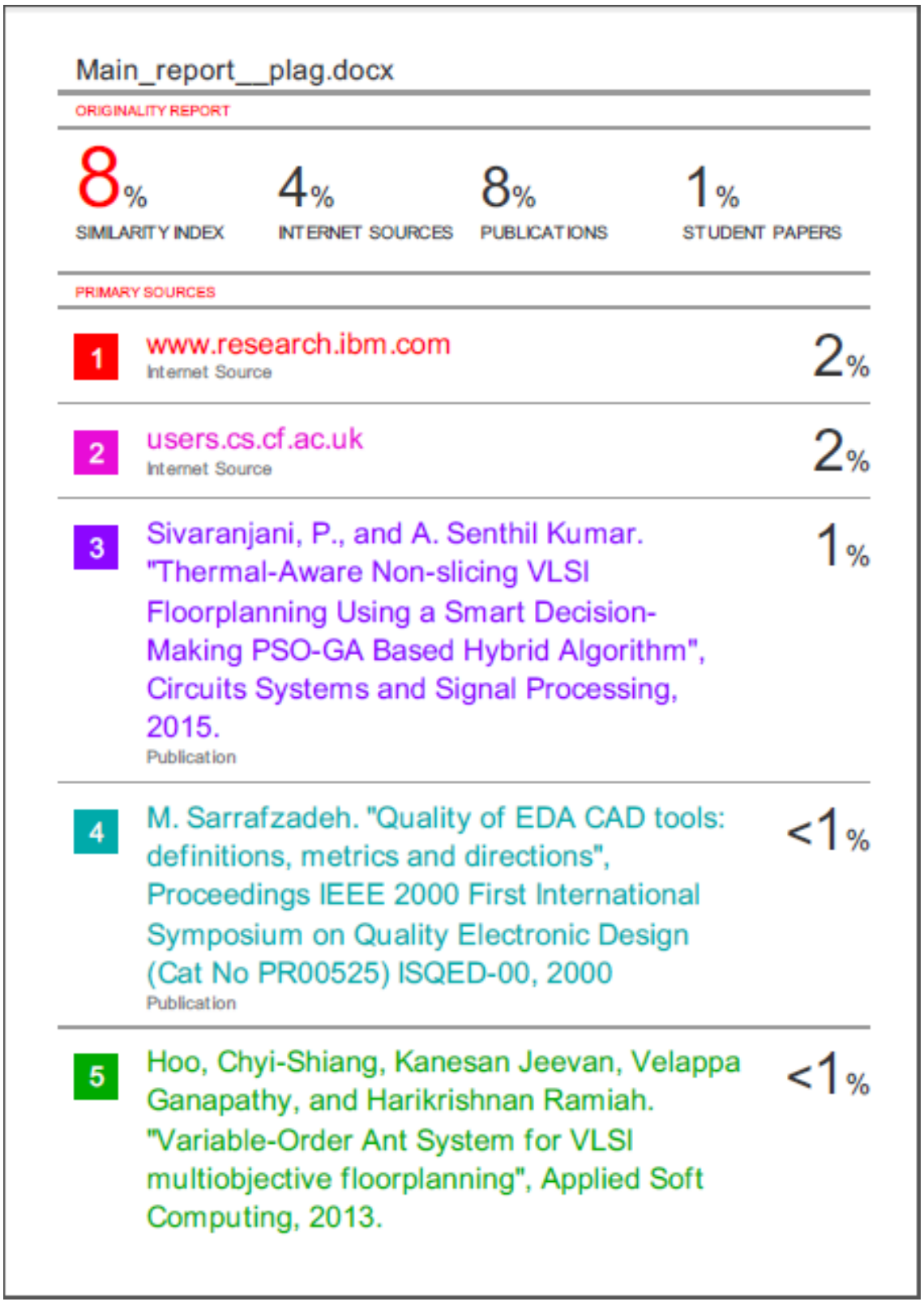
[34] Zhenyi Chen, Gaofeng Wang and Chen Dong, "Hybrid particle swarm optimization algorithm for fixed-outline floorplanning," *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, Harbin, 2011, pp. 1299-1302.doi: 10.1109/ICCSNT.2011.6182198

[35] *The MCNC Benchmark Problems for VLSI Floorplanning*. [Online].Available: http://www.mcnc.org

LIST OF PUBLICATION

[1] R. Shah and M. Bansal," A Novel High Speed Simulated Algorithm for Non-Slicing VLSI Floorplanning using B*-tree Representation", in *Indian Journal of Science and Technology*, 2016. (Communicated)

ORIGINALITY REPORT



Main_report__plag.docx

ORIGINALITY REPORT

8%	4%	8%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.research.ibm.com Internet Source	2%
2	users.cs.cf.ac.uk Internet Source	2%
3	Sivaranjani, P., and A. Senthil Kumar. "Thermal-Aware Non-slicing VLSI Floorplanning Using a Smart Decision-Making PSO-GA Based Hybrid Algorithm", Circuits Systems and Signal Processing, 2015. Publication	1%
4	M. Sarrafzadeh. "Quality of EDA CAD tools: definitions, metrics and directions", Proceedings IEEE 2000 First International Symposium on Quality Electronic Design (Cat No PR00525) ISQED-00, 2000 Publication	<1%
5	Hoo, Chyi-Shiang, Kanesan Jeevan, Velappa Ganapathy, and Harikrishnan Ramiah. "Variable-Order Ant System for VLSI multiobjective floorplanning", Applied Soft Computing, 2013.	<1%

Figure 18 Originality Report

Raj_Shah_601461026

ORIGINALITY REPORT

8%	4%	8%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

MATCH ALL SOURCES (ONLY SELECTED SOURCE PRINTED)

2%

★ M. Sarrafzadeh. "Quality of EDA CAD tools: definitions, metrics and directions", Proceedings IEEE 2000 First International Symposium on Quality Electronic Design (Cat No PR00525) ISQED-00, 2000

Publication

EXCLUDE QUOTES	OFF	EXCLUDE MATCHES	< 8 WORDS
EXCLUDE BIBLIOGRAPHY	OFF		