Automation and Optimization in Memory Cell Generators for Future Technologies

Submitted By

Apurva P. Mehta

13MCEC09

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2015

Automation and Optimization in Memory Cell Generators for Future Technologies

Major Project

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering of Institute of Technology, Nirma University, Ahmedabad.

Submitted By

Apurva P. Mehta

(13MCEC09)

Guided By

Prof. Gaurang Raval

Nirma University

Mr.Ashu Talwar

ST Microelectronics

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2015

Certificate

This is to certify that the major project entitled **"Automation and Optimization in Memory Cell Generators for Future Technologies"** submitted by **Apurva P. Mehta (13MCEC09)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. K. P. Agrawal
Guide & Associate Professor,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Prof. Vijay Ukani
Associate Professor,
Coordinator M.Tech - CSE
Institute of Technology,
Nirma University, Ahmedabad

Dr. Sanjay Garg
Professor and Head,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr K Kotecha
Director,
Institute of Technology,
Nirma University, Ahmedabad

Statement of Originality

I, **Apurva P. Mehta**, Roll. No. **13MCEC09**, give undertaking that the Major Project entitled "**Automation and Optimization in Memory Cell Generators for Future Technologies**" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Apurva P. Mehta
Date:25th MAY, 2015
Place:Ahmedabad

Endorsed by
Prof. Gaurang Raval

Acknowledgements

I would also like to thank my Internel guide **Mr. Gaurang Raval**, Professor, Computer Science Department, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout this work.

It gives me an immense pleasure to thank **Dr. Sanjay Garg**, Hon'ble Head of Computer Science and Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

I would also like to thank **Dr. K.R.Kotecha**, Director, Institute of Technology, Nirma University, Ahmedabad for providing me an opportunity to get an intern ship at ST Microelectronics, Greater Noida.

I would like to thank my all faculty members for exchanging knowledge during my postgraduate program.

Sincere thanks to **Mr.Ashu Talwar** Manager, ST Microelectroniocs, Greater Noida. I enjoyed his vast knowledge and greate future vision and owe his lots of gratitude for having a profound impact on this report.

A spacial thank to, **Mr. Abhinav Arora** for his valuable guidance. The appreciation and continual support he has imparted has been a great motivation to me in reaching a higher goal. His guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

I want to thank my Mentor, **Mrs. Tanvi Ahuja**, ST Microelectroniocs, More prominent Greater Noida for her important direction. All through the preparation, she has issued me important exhortation on undertaking work. She has issued me all sort of backing to handle complex circumstance.

I would like to thank my Teammate, **Mr. Nirav Patel** for his help. Throughout the training, he has given me valuable advice and support on project work.

- Apurva P. Mehta

13MCEC09

Abstract

Memory is represented by different views as an example Timing views, Model view, Layout views. For generating memory cell different view is useful. Back end compiler is product which is used to generate memory for various technology and various specification.This process is know as cut generation. Two of views, layout view and net list view are mostly useful. All the technological,architectural and other data are encoded in BE Compiler.The GDS record contains the format data. CDL document contains the schematic data.It implies each memory generator arrangements have distinctive Back-End compiler, however all have a ton of usually utilized code. To Provide single interface for various type of file is one aspect of uniBE.It Support multiple technology like 28,14 nm etc.It also Support multiple architecture.User can insert data using single interface.Multiple criteria for compiler are also supported using this product.unibe must be faster enough and accurate.

Abbreviations

CDL	Circuit Description Language
GDS	Graphical Data Stream
BTP	Basic Text Parser
DRC	Design Rule Check
DRM	Design Rule Manual
DTO	Data Transfer Object
REGEX	REGular EXpression
XML	eXetensible Markup Language
XSD	Xml Schema Definition
XSLT	Extensible Stylesheet Language Transformations
JAXB	Java Architecture for Xml Binding
DOM	Document Object Model
SAX	Simple API for Xml
ANN	Artificial Neural Network

Contents

Certificate **iv**

Statement of Originality **v**

Acknowledgements **vi**

Abstract **vii**

Abbreviations **viii**

List of Figures **xi**

List of Tables **1**

1 Introduction **2**
1.1 General . . . 2
1.2 Objective Of Work . . . 2
1.3 Scope Of Work . . . 3
1.4 Activities . . . 3

2 Literature Survey **4**
2.1 Architecture of Memory . . . 4
2.1.1 Basic Architecture . . . 4
2.1.2 Split Core Architecture . . . 5
2.2 Process Designing Memory . . . 6
2.3 circuit Descriptive Language . . . 8
2.3.1 Basic detail . . . 8
2.4 Regular Expression . . . 9
2.4.1 Basic of regular expression . . . 9
2.5 Technique for Parsing . . . 10
2.5.1 Text Parser . . . 10
2.5.2 DOM Parser(Xml parser) . . . 11
2.5.3 SAX Parser(Xml parser) . . . 11
2.5.4 StAX Parser(Xml parser) . . . 11

3 Project Overview **12**
3.1 CDL(netList) and GDS(Layout) . . . 12
3.2 UniBe Process Flow . . . 13
3.3 Object generation flow for Architectural Library . . . 14
3.4 Template Files . . . 15

4 Implementation and Result **17**
4.1 Refine architecture . . . 17
4.2 Reflection API . . . 18
4.3 Design Factory Implementation . . . 18
4.4 Other Optimization . . . 18

5 Standard Interface **22**
5.1 XML Parser . . . 22
5.1.1 DOM Parser . . . 22
5.1.2 SAX Parse . . . 23
5.1.3 Parser Characteristics . . . 23

6 Routing Problem **26**
6.1 Possible approaches . . . 26
6.1.1 Ant olony . . . 26
6.1.2 Genetic Algorithm Flow . . . 27
6.1.3 Artificial Neural Network . . . 28
6.2 Selected Aprroach . . . 29

7 Some Challanging Problems **30**
7.1 Equation Evaluation . . . 30
7.2 Checker . . . 30
7.2.1 Type of Checker . . . 31
7.3 Boundry Function . . . 31

8 Project Execution **33**
8.1 Product Generation . . . 33
8.2 Steps to use and execute product . . . 33

9 Scope of the Project **34**

10 Tools and Technology **35**

11 Conclusion And Enhancement **36**

References **37**

List of Figures

2.1 Architecture of Memory(256x256) 4
2.2 Split Core Architecture of bank 5
2.3 Page Architecture Memory of 64K 6
2.4 Memory Layout 7
2.5 Flow of LVS 8
2.6 Example of netlist(cdl file) 9
2.7 Regex aliases 10

3.1 Schematic view of circuit 13
3.2 Layout of circuit 14
3.3 Process Flow 15
3.4 Flow Diagram of Unibe 16
3.5 Template File 16

4.1 Refined architecture 17
4.2 Reflection Implementation snapshot-1 19
4.3 Reflection Implementation snapshot-2 20
4.4 Design Factory Implementation snapshot-1 20
4.5 Design Factory Implementation snapshot-2 21

5.1 DOM parser 23
5.2 SAX parser 24

6.1 Genetic Algorithm 27
6.2 Genetic Algorithm Behaviour 28
6.3 Artificial Neural Network 29

7.1 Boundry function 32

List of Tables

5.1 XML parser characteristics 25

11.1 Improvements 36

Chapter 1

Introduction

1.1 General

Achievement in the microelectronics business is fundamentally relies on upon its unwavering quality and execution. Not simply should a thing execute as looked for, it ought to work for an extended period of time without fizzle, ordinarily 10 years or more. Dependability related issue torment in the industry are of two sorts: deformity related issues and destroy issues. Imperfection related issues are predominantly created by the assembling imperfections, for example, a missing methodology step, soil, or other unavoidable catastrophes. Indeed the most effective procedure lines experience the ill effects of an incidental deformity related issue. Wear-out is a result of the circuit or the thing just wearing out, without any starting distortions being accessible.

The First venture in memory outlining is the formation of the design utilizing some product. At that point this design is checked with the assistance of DRC to verify that the outline doesn't abuse any tenets indicated in the DRM which is needed for the correct working of the chip. At that point once the format is DRC clean it is contrasted and the detail model of the configuration to guarantee that the format speaks to the same circuit as it is in the detail model.

1.2 Objective Of Work

This project is a system for memory generation which is called memory cut.This product is called UniBe can be used for all technologies and all specification with limited control to user.

- User control
- Support multiple interfaces
- Support different technology
- Support various bank architecture

1.3 Scope Of Work

The principle goal is to give an interface between others product which generates the cuts and lessen generation time.

Unibe item underpins all engineering. That implies Unibe will have the capacity to create view for everything technology,no need to utilize diverse compiler to produce view for distinctive technology.

Other important thing is control of data is given to user using Template files(may be text,xml etc).Single interface is given to all different kind of file.

Reduce time of BTP.BTP take around **45sec** to execute for single cut.

1.4 Activities

- Understanding various processes involved in the Circuit Design flow
- Understanding Technology that can be useful in project infrastructure
- Learning the Reflection API, understanding the usage of Reflection API and analyse risk
- Learning the design factory pattern for infrastructure
- Analyse various parser for XML parsing
- Implement Text Parser
- Implement XML Parser
- Implement boundry function
- Implement checker

Chapter 2

Literature Survey

2.1 Architecture of Memory

2.1.1 Basic Architecture

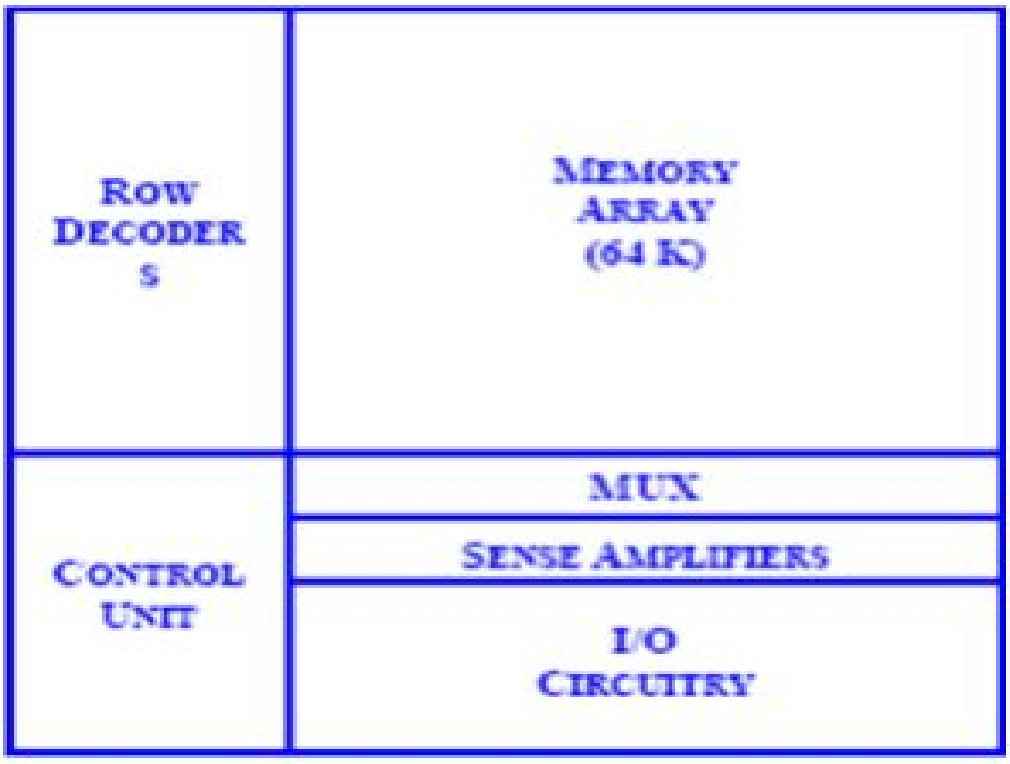

Figure 2.1: Architecture of Memory(256x256)
[1]

All the cells joined with the word line in the column are active,when the chose lines are high hence dispersal increases.In the essential building design, bit access time and the

word line access time are the two main considerations add to the read access .When the span of the Memory expands the heap is lessened on the grounds that the quantity of cells associated with the word line increases.Thus, the word line deferral builds in view of the increment in the word line capacitance.By decreasing the bit line capacitance and the word line capacitance are the two variables that can be enhanced , yet this is attained to strictly when utilizing an alternate construction modeling. [1]
Some architecture are shown in figure 2.2 , 2.1 and 2.3.

2.1.2 Split Core Architecture

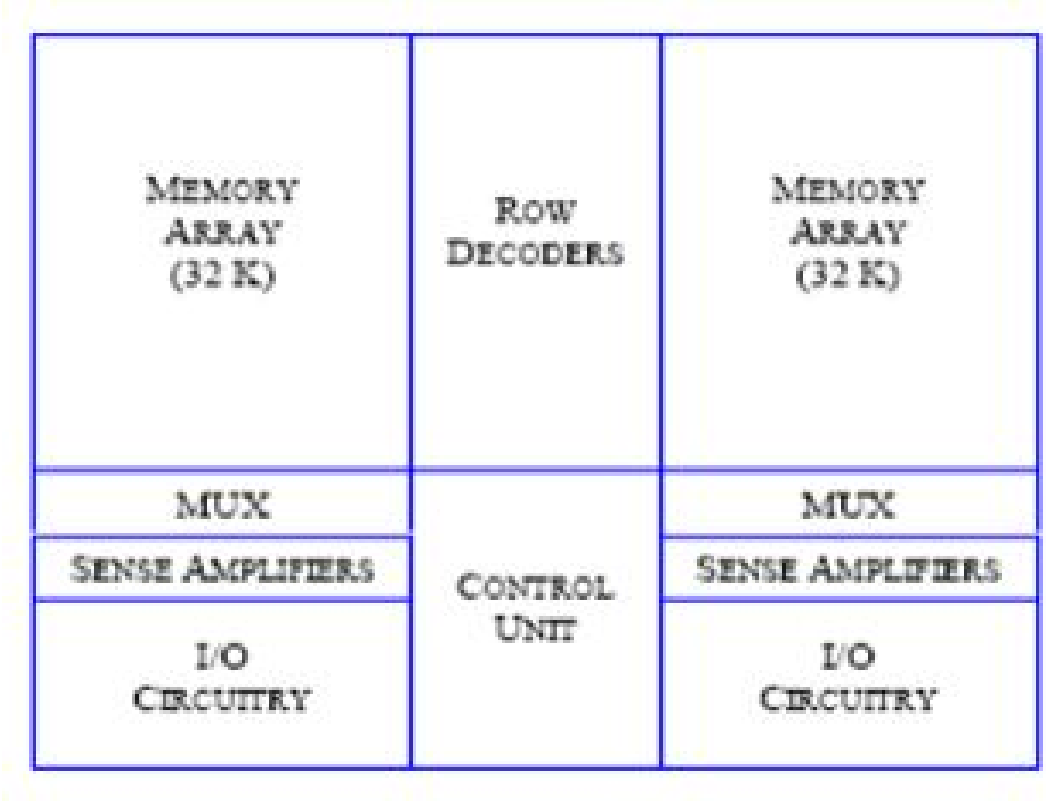

Figure 2.2: Split Core Architecture of bank
[1]

Lessening is performed by part the framework into the littler pieces. Along these lines the ensuing structural planning is called Split-Core architecture. Because of the part bank lessening in the RC deferral is watched, yet here excessively the initiation of a word line actuates the whole cell in both territories. So alternate structural planning is required in which could give some point of interest against the power dispersal. So PAGE structural engineering is presented.[1].Memory Layout is shown in 2.4.

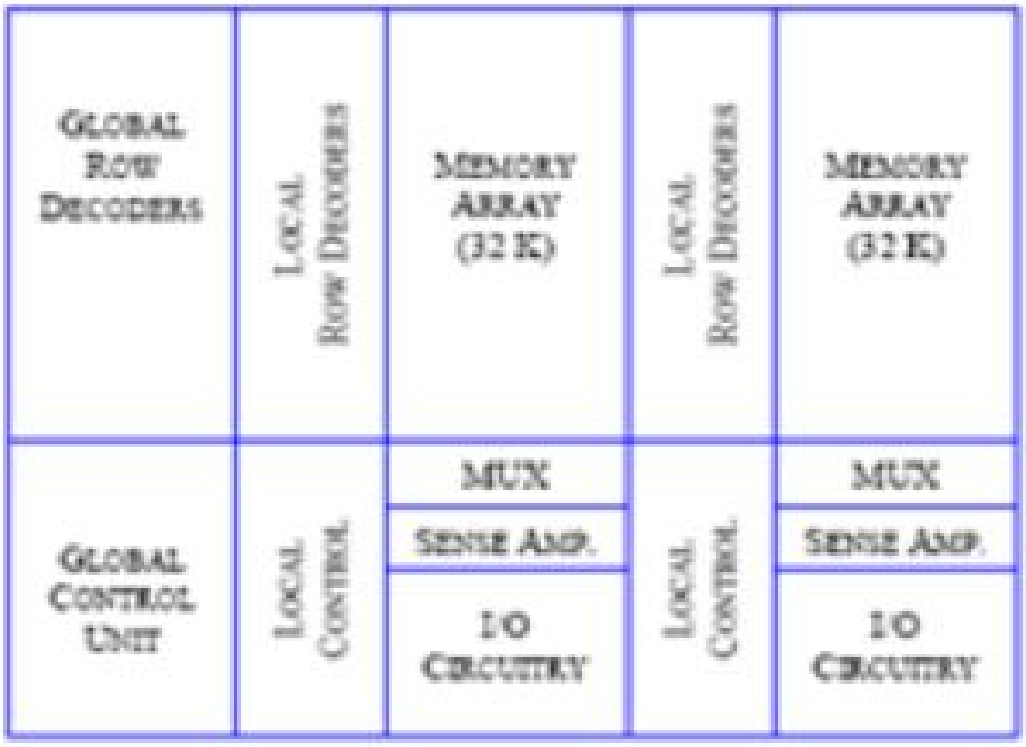

Figure 2.3: Page Architecture Memory of 64K
[1]

2.2 Process Designing Memory

The initial phase in IC planning is the making of the design utilizing some so ware.then this design is checked using DRC to verify that the configuration does not disregard any tenets determined in the DRM which is needed for the best possible working of the chip. At that point once the design is DRC clean it is contrasted and the flavour model of outline to guarantee that the format speaks to the same circuit as it is in the zest model. following are data about the checks :

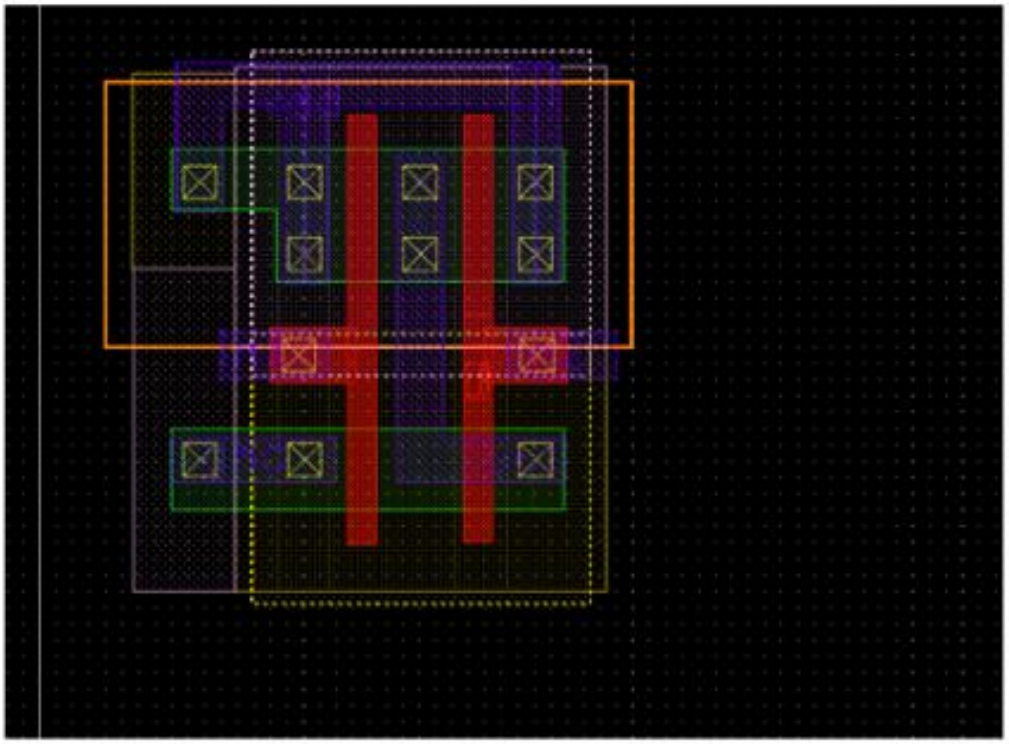

Figure 2.4: Memory Layout
[2]

- An outline guideline set points out certain geometric and network confinements to guarantee sufficient edges to record for variability in semiconductor producing star cesses, in order to guarantee that a large portion of the parts work accurately. While outline tenet checks don't approve that the outline will work accurately, they are developed to check that the structure meets the methodology obligations for a given outline sort furthermore handle engineering. DRC programming generally takes as enter a format in the GDSII standard organization, and produces a report of outline principle infringement that the creator might decide to adjust. Deliberately "extending" or waiving certain configuration standards is regularly used to expand execution and segment thickness at the cost of yields.

- Lvs identify whether the layout and the schematic diagrams are identical or not. Connection presented in both layout and schematic must be same. Inputs for doing LVS may be GDS file or CDL file. The GDS file represent information for circuit. CDL file represent the schematic information required for circuit.

Flow of LVS is shown in figure 2.5.

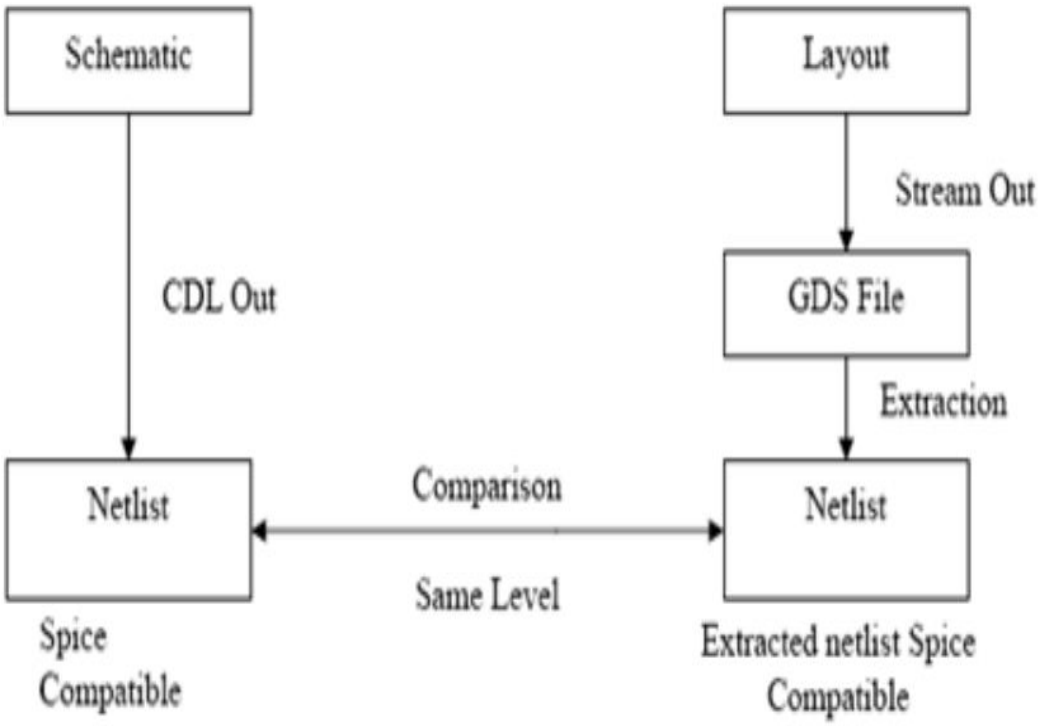

Figure 2.5: Flow of LVS
[2]

2.3 circuit Descriptive Language

2.3.1 Basic detail

The CDL provide us full modularity. Permitted settling of sub circuit and sub circuit can be utilized as a part of better places and distinctive parameters. It can not be gotten to from outside as interior hubs and data in variable is encapsulated inside a sub circuit .Circuit Descriptive dialect itself recording the circuits.

Figure 2.6 shows Example of netlist(cdl file). A library of standard force framework model can undoubtedly be utilized by clients who even do not have the information of inward circuit .CDL is cross-stage and can be utilized as standard yield organization of distinctive GUI circuit editors.[4]

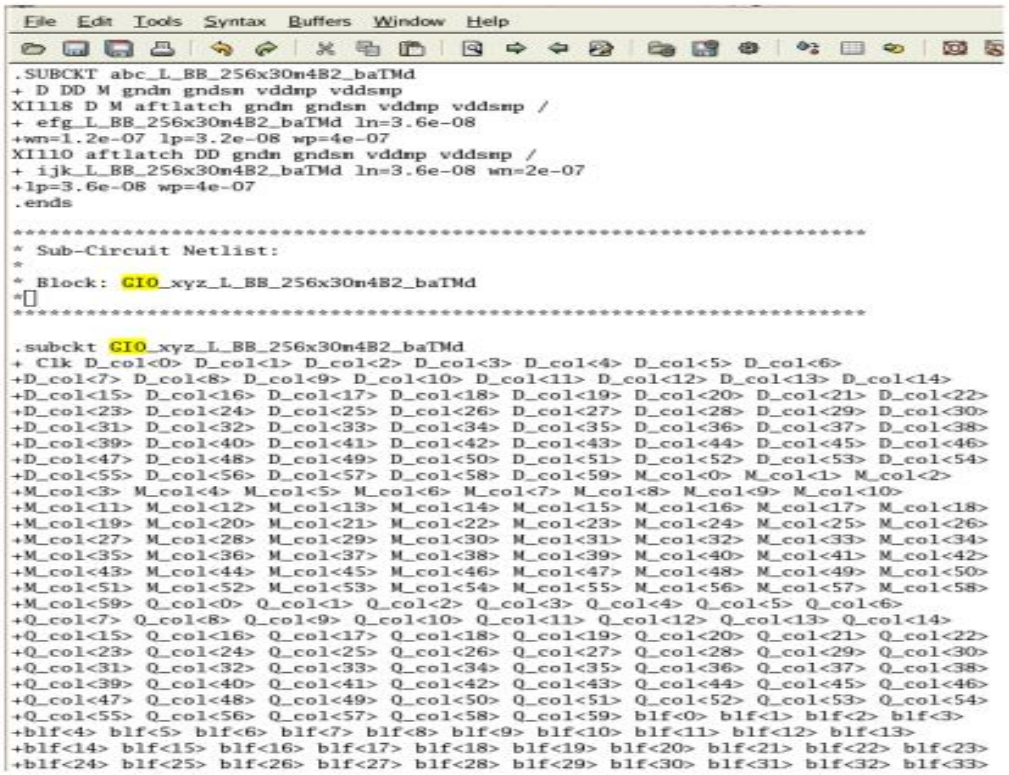

Figure 2.6: Example of netlist(cdl file)
[2]

2.4 Regular Expression

2.4.1 Basic of regular expression

A **Regular Expression** are patterns used to match character combinations in strings. **java.util.regex** package is used in java to match patterns using regex. Regular Expression remain same as in shell Scripting or java or perl. Regular expression literals provide compilation of the regular expression when the classes is loaded. When the regular expression will remain constant, use this for better performance. Regex Package in java has a following classes:

- **Pattern** object is create by compiling string containing regex. It uses static method to create pattern object.We have to pass regular expression argument in static methods.
- **Matcher** is the regular expression engine object that matches the input String pattern with the pattern object created. This class doesnt have any public con-

structor and we get a Matcher object using pattern object matcher method that takes the input String as argument. We then use matches method that returns boolean result based on input String matches the regex pattern or not.

- **PatternSyntaxException** it represent syntax error in a pattern.

Regular Expression	Description
`\d`	Any digit, short for `[0-9]`
`\D`	A non-digit, short for `[^0-9]`
`\s`	A whitespace character, short for `[ \t\n\x0b\r\f]`
`\S`	A non-whitespace character, short for `[^\s]`
`\w`	A word character, short for `[a-zA-Z_0-9]`
`\W`	A non-word character `[^\w]`
`\S+`	Several non-whitespace characters
`\b`	Matches a word boundary where a word character is `[a-zA-Z0-9_]`.

Figure 2.7: Regex aliases
[9]

2.5 Technique for Parsing

2.5.1 Text Parser

Text parser parse the text file using the stream class provided by java. Text parser parse the file data into various part that is to be passed to ParserUtility class.It create object.That means is store data in object and passed to view generator to generate appropriate view.

2.5.2 DOM Parser(Xml parser)

The DOM Parser loads complete XML content into a memory and use tree structure to represent or to access it. An XML-DOM parser reads XML, and converts it into an XML DOM object[5]

2.5.3 SAX Parser(Xml parser)

The SAX Parser does not loaded into the memory like DOM parser. Different tags are used by SAX parses the XML . Various tags are used like opening tag to start parsing, closing tag to finish parsing, character data to define data, comments etc.
It creates data model itself using data.[5]

2.5.4 StAX Parser(Xml parser)

Using StAX parser we can extract the data from the template file in xml format from current position of cursor. SAX parser it was event based parser. we can say that StaX is faster and it also create its own data model.[5]

Chapter 3

Project Overview

Main use of BE Compiler is generation of layout and net list as per data of specification. For Different technology different BE compiler is come into picture. Every Web gen Configuration have different BE Compiler, but all have a lot of commonly used piece of code.Following are the purpose of Unibe:

- Provide single interface for user inputs data
- Memory designer only fill the template as per as their requirements
- Support multiple technology, architecture.

3.1 CDL(netList) and GDS(Layout)

A standard cell is a blend of transistor and it connect internal structures. The least difficult cells comprises of NAND, NOR, and XOR boolean capacity. The boolean rationale capacity of a phone is called its legitimate view.Boolean polynomial maths comparison characterize the practical conduct of cell.
Figure 3.1 shows Schematic view of circuit.

Memory Designers announces the data parameter to re-enact the electronic conduct of the netlist and after-ward figuring the circuit's chance area reaction. The recreations checks whether the netlist actualizes the coveted yield and foresee others parameters, for example, force or utilization .Since the perspectives are valuable for theoretical reproduction, and not for the gadget fabrication.layout perspective is the most reduced level of

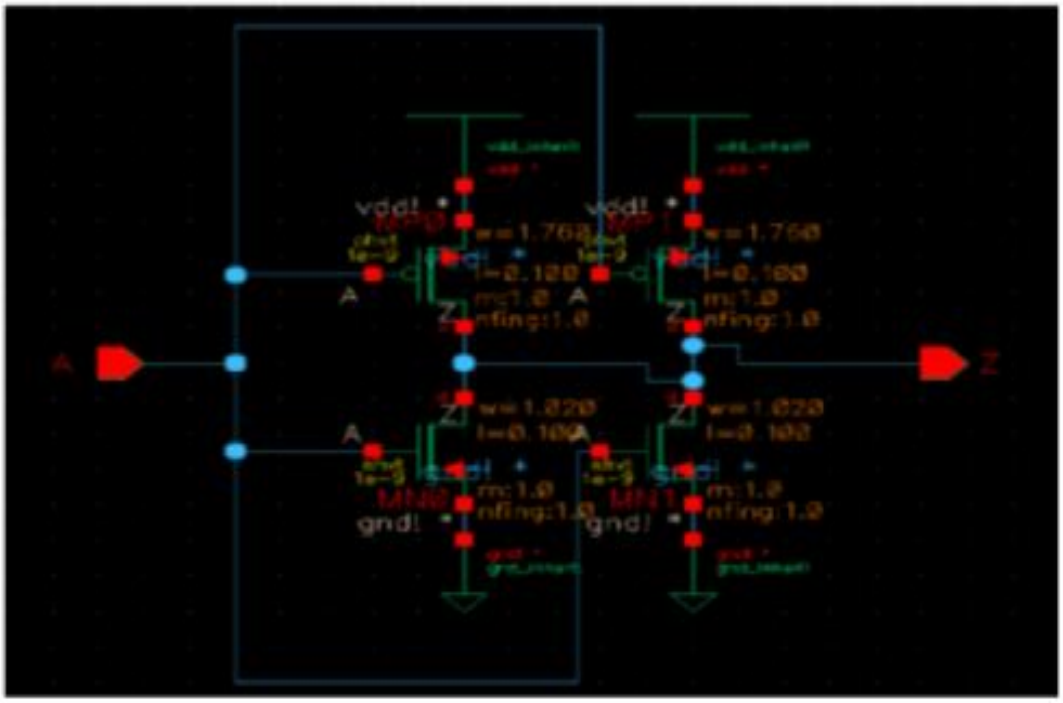

Figure 3.1: Schematic view of circuit

configuration reflection . Format perspective portray the complete data of interconnection between the terminal components.

3.2 UniBe Process Flow

UniBe Process Flow is shown in 3.3 and 3.4.

Process Flow

- To begin with Memory Designer solicitation to produce the memory cut.
- Web gen send request with data to the unibe.
- Unibe Module working flow
  - Take command
  - View Call
  - Unibe request to create object of top structure from the parser.

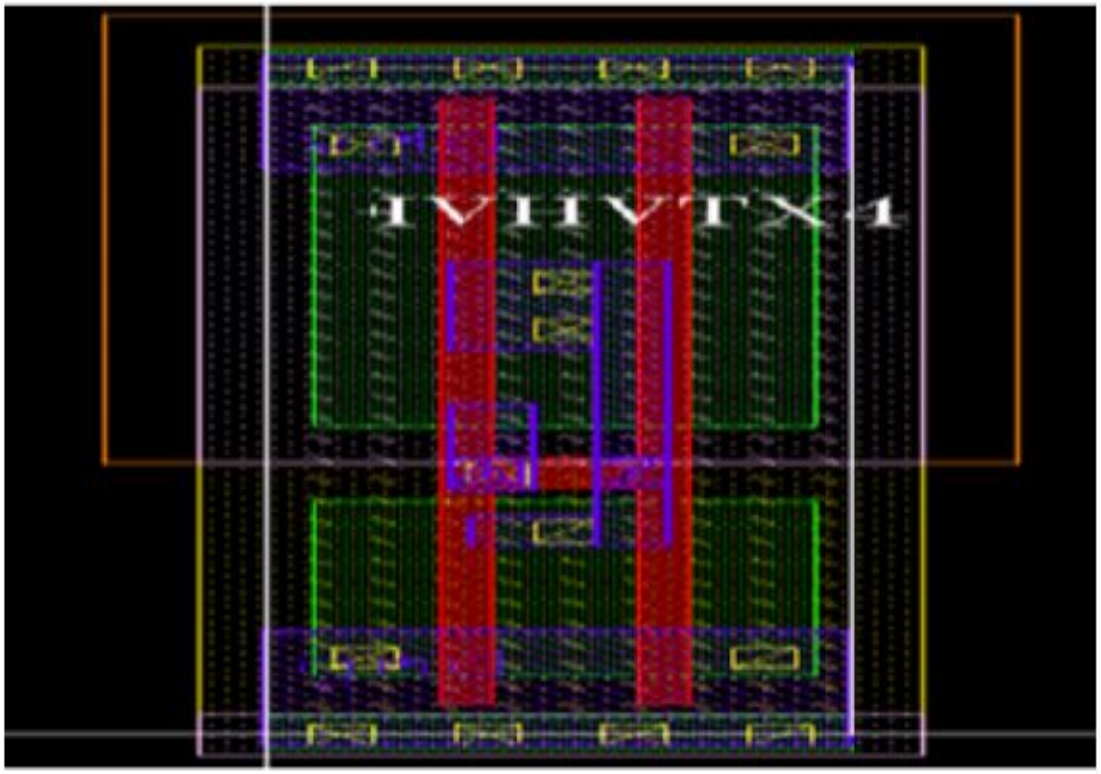

Figure 3.2: Layout of circuit

  - Parser return object of structure
  - This object is converted into the specific object using jar file.

- Pass object to view generator.
- Generate views
- View is returned(cdl or gds)

3.3 Object generation flow for Architectural Library

- Request for building a Structure.
- Request for building Alias by Structure is sent.
- Request to build Block specific block Object.
- Request is sent to generation of Sub Blocks.

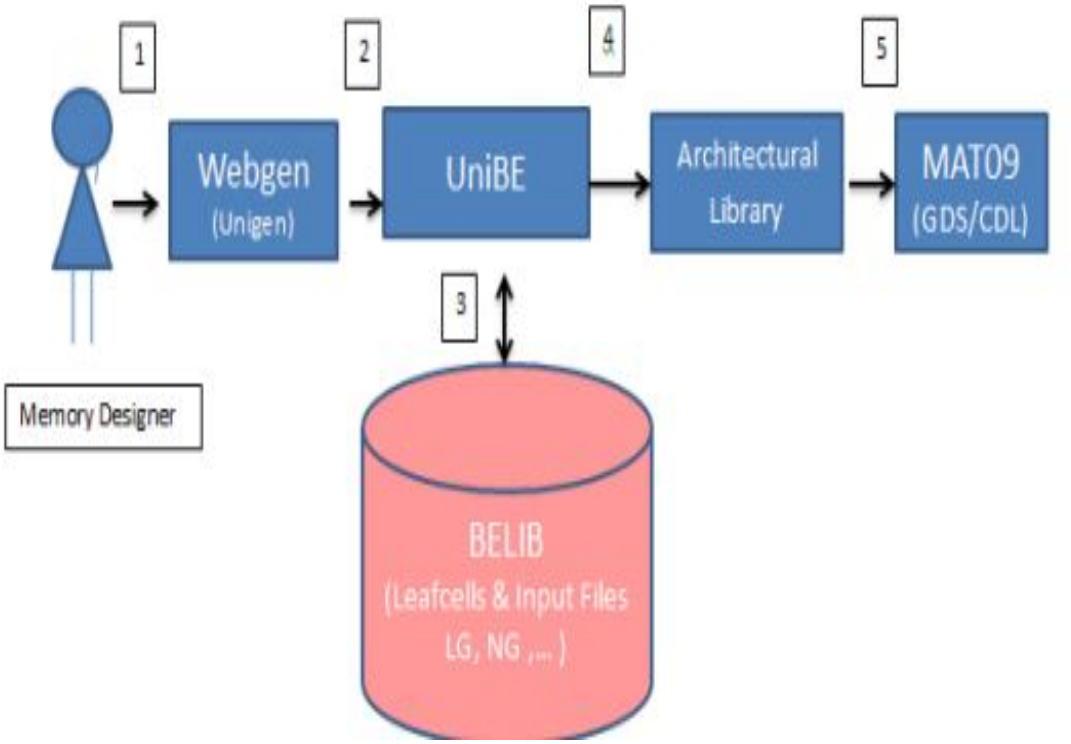

Figure 3.3: Process Flow
[3]

- Sub Block object is returned which is to be placed on top of BLOCK.
- Object of Block is Returned.
- STRUCTURE Object is Return.

3.4 Template Files

These are data file given by end user.User will define detail with specific format. Example of sample template file is shown in figure 3.5.

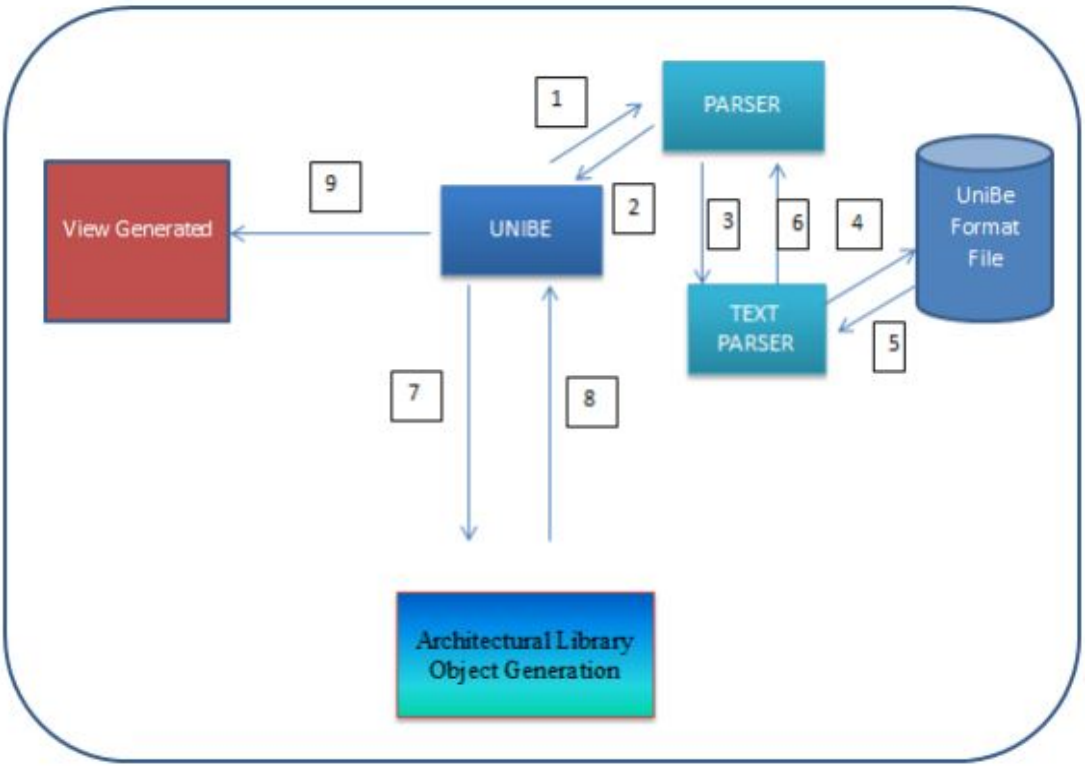

Figure 3.4: Flow Diagram of Unibe
[3]

new 1

```
1
2  BLOCK_MAIN BEGIN
3
4      BLOCK_NAME          BLOCK(SUB_BLOCK1)
5      BLOCK_NAME       BLOCK(SUB_BLOCK2)
6      LOCATOIN         PARALLEL
7
8  BLOCK_MAIN END
9
10 SUB_BLOCK1 BEGIN
11     VLAUE BATTERY
12     LOCATION SERIAL
13     VOLTAGE +5
14 SUB_BLOCK1 END
15
16 SUB_BLOCK2 BEGIN
17     VLAUE RESISTANE
18     OHM 10
19     VOLTAGE +5
20 SUB_BLOCK2 END
21
22
23
```

Figure 3.5: Template File

Chapter 4

Implementation and Result

4.1 Refine architecture

First goal is to make architecture global and stable for future technologies. That means architecture should work for any kind of file type. Best way to do this is add extra layer which handle complexity and provide abstract view.

Refined architecture is shown in figure 4.1.

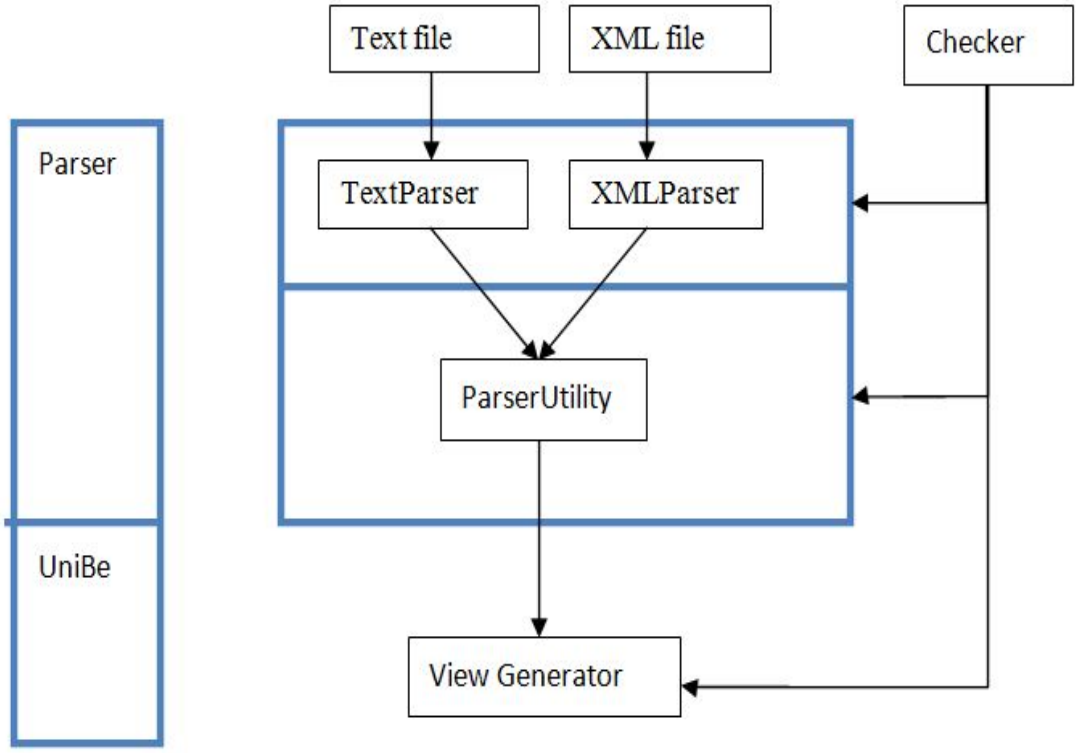

Figure 4.1: Refined architecture

- **parser**

  This module parse any kind of file that is supported by UniBE and convert it to DTO. DTO is specific type of format to store data. We are using Objects to store data. Finally DTO are transferred to object of particular class.

  - File to DTO Parser
  - DTO to Object Parser

- **checker**

  This module deals with check which should be applied to user data in template files.

- **view generator**

  This module generate view for given template file.

4.2 Reflection API

Here problem statement is first we have to check which object is to be created from text.It means we have to check all kind of object and we can decide which kind of object is created.

One of the solution for this is to use reflection API.Using Reflection API we can create run time object using fully qualified class name.

Disadvantage or this is risk,because we may get runtime error.We can not verify these errors during compilation.rapi

Implementation is shown in figure 4.2 and 4.3

4.3 Design Factory Implementation

Pproblem statement with reflection is risk of runtime errors. Alternative solution is to use design factory implementation.

Figure 4.4 and 4.5 shows factory design pattern implementation.

4.4 Other Optimization

Some other optimizations are also applied to reduce time,Line of code and increase accuracy.

Figure 4.2: Reflection Implementation snapshot-1

- Replace if-else with switch case for string(provided in java-7).It save around 50 msec.
- Remove pattern compilation code out of check pattern method which is called multiple time.It save around 15 msec.
- Used HashMap with is stead of ArrayList.It save around 100 msec.
- Patter checking for comment is done before business logic is applied.It save around 15 msec.
- Used StringBuilder instead of String to concatenate multiple string.
- Define lenght variable out side loop.

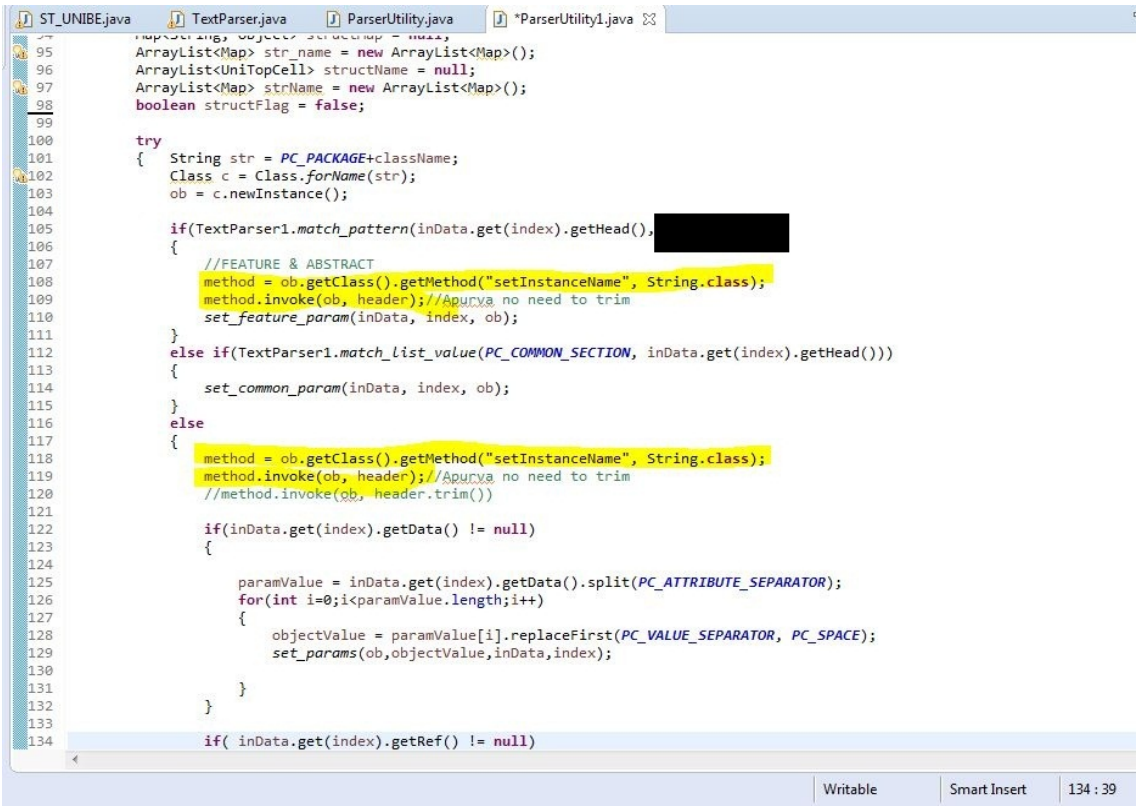

Figure 4.3: Reflection Implementation snapshot-2

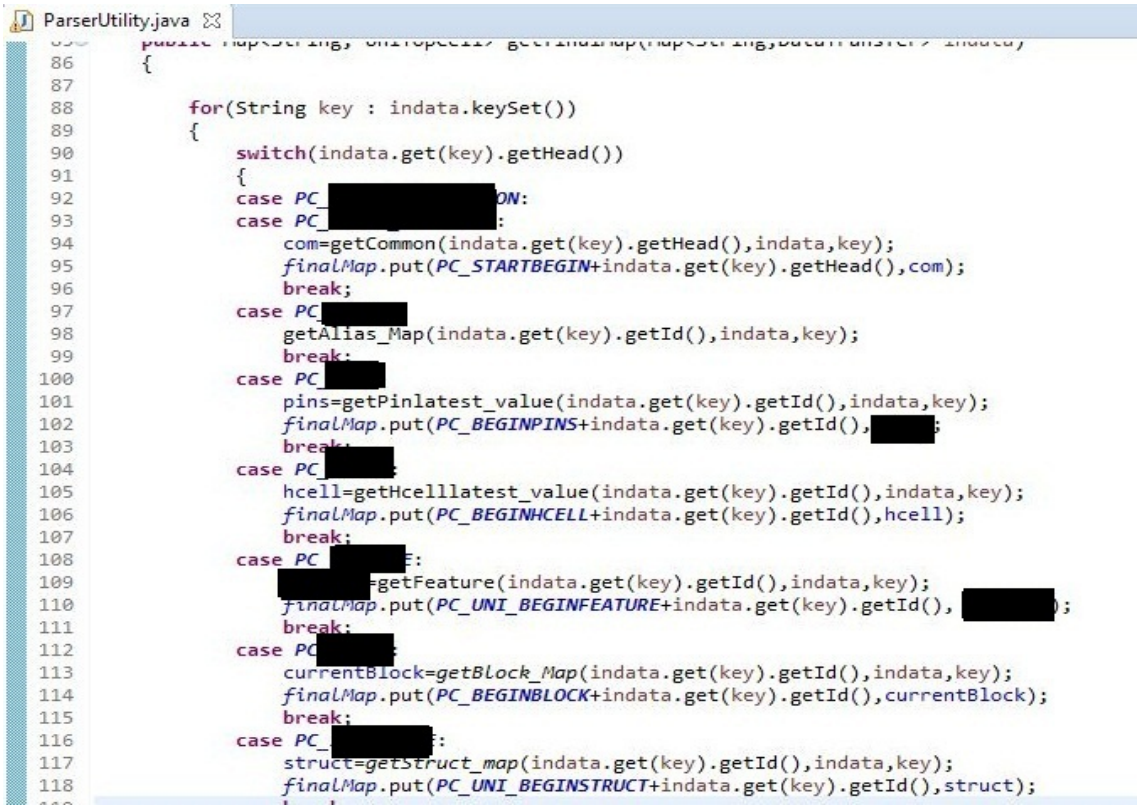

Figure 4.4: Design Factory Implementation snapshot-1

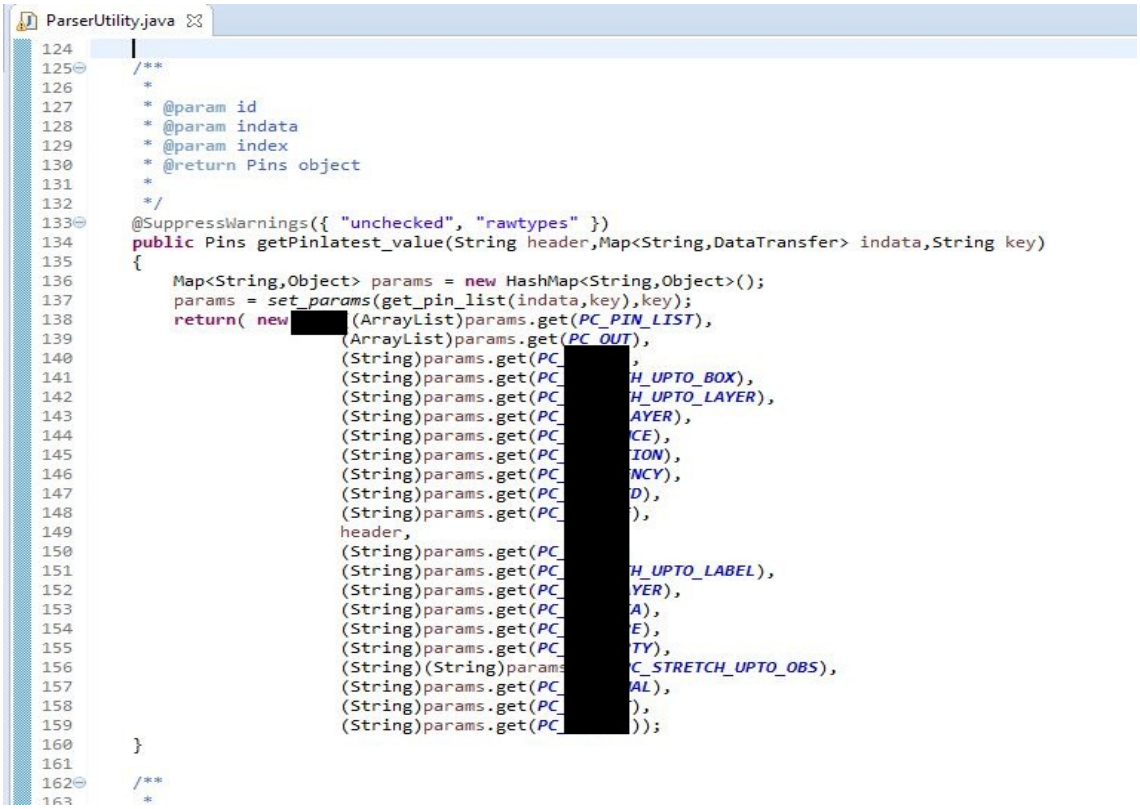

Figure 4.5: Design Factory Implementation snapshot-2

Chapter 5

Standard Interface

For future support it is better to provide standard interface for files which can work with any kind of application like web service,desktop software,web portal etc. XML international standard interface to transfer data with inbuilt check facility. It provide emphasize simple and general view. It support all kind of data to transfer. XSD and XSLT provide support for validation.

5.1 XML Parser

5.1.1 DOM Parser

DOM parser is XML parser which is cross-platform and also language independent. It can be used for HTML, XHTML, and XML documents.
DOM use tree structure to represent data stored in XML. It is called DOM tree.
We have to traverse tree node by node to collect data.
Architecture of DOM parser is shown in figure 5.1.

Following are points we have to consider to select XML parser:

- Methods to traverse XML trees, access, insert, and delete nodes.
- Load whole XML in memory
- Easy and simple to code

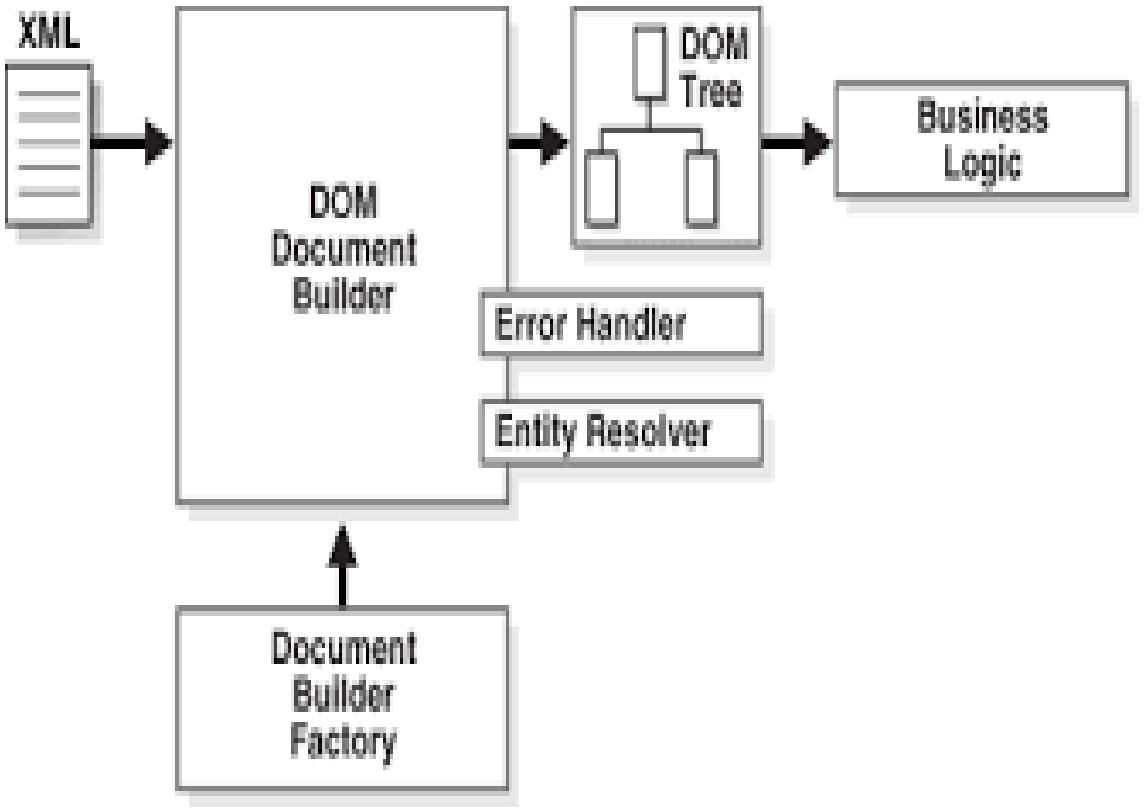

Figure 5.1: DOM parser

5.1.2 SAX Parse

SAX parser is event driven XML parser. It process each state independently.
This parser does not make any tree structure. It does not store opened tags.
It simply prosess xml tag based on event,so we can say tha it require less memory.
Architecture of DOM parser is shown in figure 5.2. Following are points we have to consider to select XML parser:

- Event based parser
- Low level APIs
- Use it's own data model

5.1.3 Parser Characteristics

See some characteristics of some other parsers are shown in table 5.1.

Based on the following charasterictics of parser DOM parser is prefered:

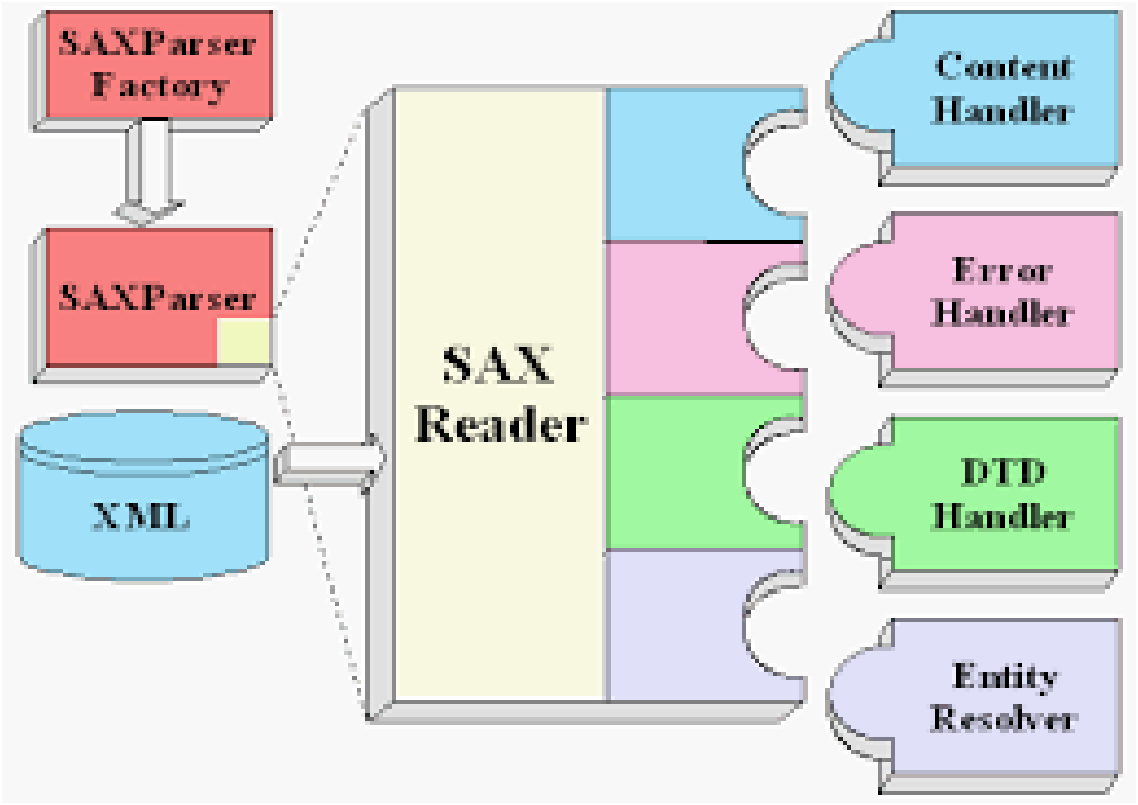

Figure 5.2: SAX parser

- SAX parser is complex as compare to DOM parse
- StAX parser doesn't support Schema validation
- DOM parser is suitable for back and forth traversing

Feature	StAX	SAX	DOM	TrAX
API Type	Pull,Streaming	Pull,Streaming	In Memory Tree	XSLT Rule
Esa of Use	High	Medium	High	Medium
XPath Capability	No	No	Yes	Yes
Cpu and Memory Efficiency	Good	Good	Varies	Varies
Forward Only	Yes	Yes	No	No
Read XML	Yes	Yes	Yes	Yes
Write XML	Yes	No	Yes	Yes
Create,Read,Update,Delet	NO	No	Yes	No

Table 5.1: XML parser characteristics

Chapter 6

Routing Problem

In circuit we need to connect two entity dynamically. First we select shortest way in manner that thay do not intersect. Here problem is selection of dynamic path with time constraint.

6.1 Possible approaches

Following are some of the possible approaches to solve NP-complete problem.

6.1.1 Ant olony

- Select rendom population intially
- Calculate pheromones
- Evaporation of pheromones
- Path of next ants may changes according to pheromones value

Pros and Cons [6]

Pros

- Performance is better

Cons

- Required more memory

  Required memory for collection of all entity.

- Less affected by poor initial solutions

  If intial population is poor ,bad result can be achieved because less chance to select new paths.

Figure 6.1 shows flow of genetaic algorithm.

6.1.2 Genetic Algorithm Flow

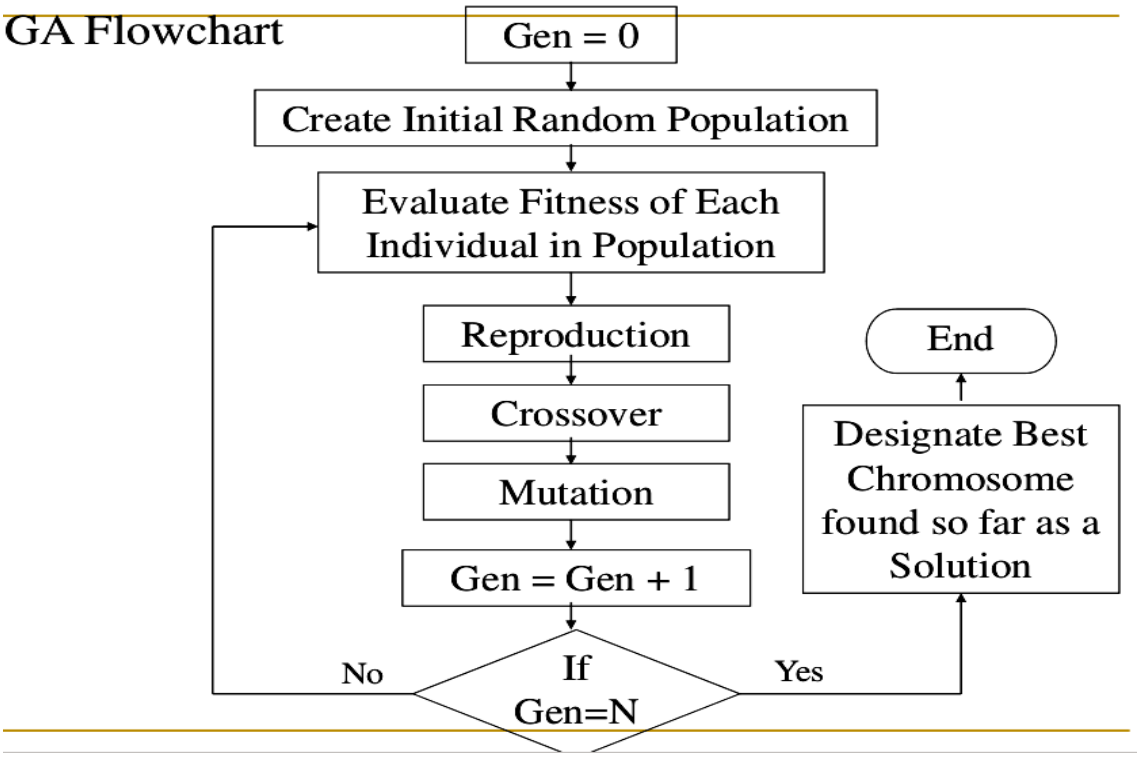

Figure 6.1: Genetic Algorithm

Figure 6.2 shows behaviour of genetic algorithm.

pros and cons[7]

Pros

- Requires less information of problem
- Not require derivatives
- Effective for noisy environments
- Implicit parallelism

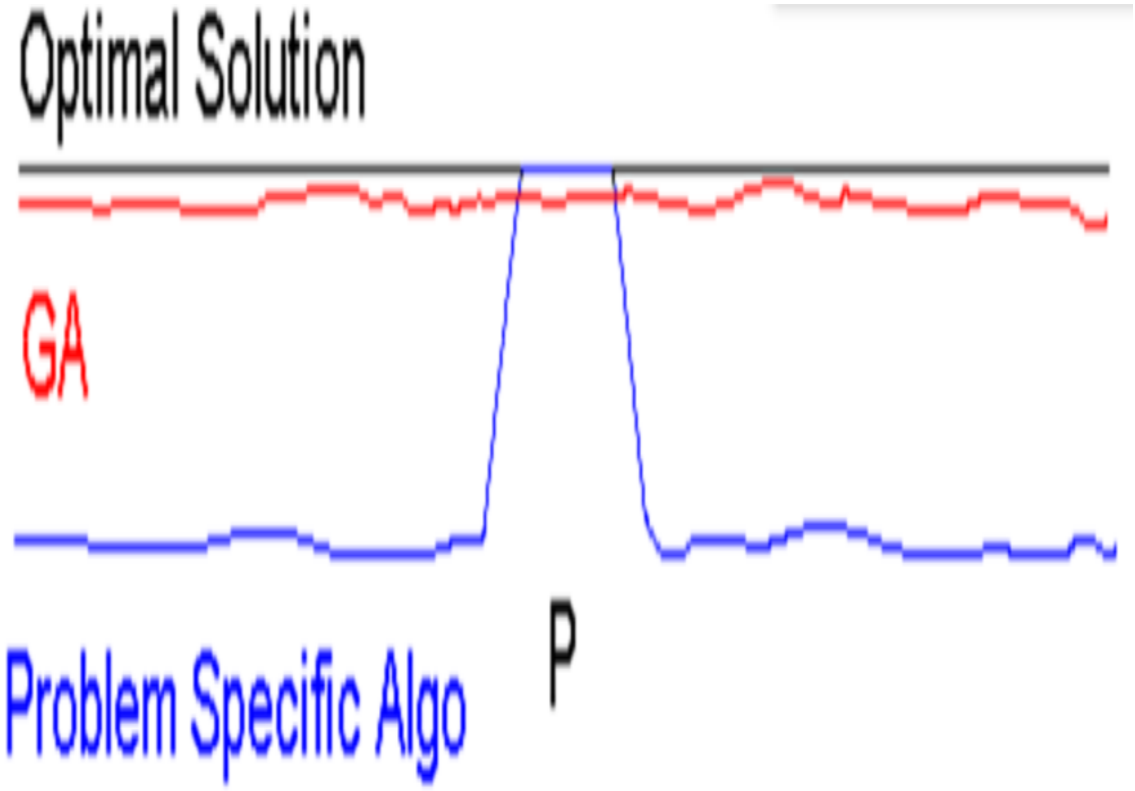

Figure 6.2: Genetic Algorithm Behaviour
[7]

Cons

- There is no guarantee of best solution

  Mostly when the populations have a multiple of subjects.

Figure 6.3 shows flow neural network algorithm.

6.1.3 Artificial Neural Network

pros and cons [8]

Pros

- Fast processing speed
- Flexibility and ease of maintenaince
- Robustness

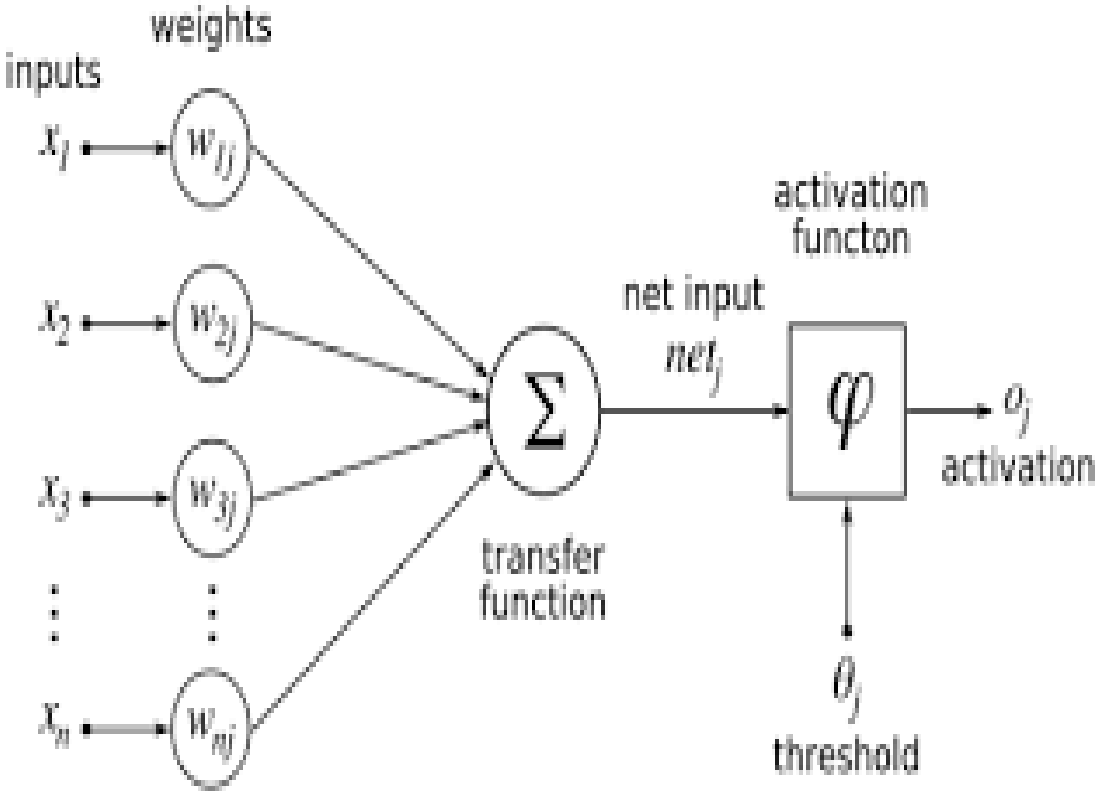

Figure 6.3: Artificial Neural Network

Cons

- Flexibility

  ANNs do not produce an explicit model even though new cases can be fed into it and new results obtained.

6.2 Selected Aprroach

ANN do not produce an explicit model even though new cases can be fed into it and new results obtained,So we can not use that algorithm.

Ant Colony required more memory and is slower as compare to Genetic algorithm.

Genetic Algorithm give us fast processing speed and good result with dynamic behaviour.

Genetic Algorithm is selected to solve problem.

Chapter 7

Some Challanging Problems

7.1 Equation Evaluation

User are allowed to write mathematical condition or ternary condition. Now challange how to evaluate that conditions when there is lot of combinations possible and all these values are depends on specific counter value.
Suppose counter value is **100** and total equations are **100** and two possible values of each equation. We will get **(1000.100)** possible combination.
Followings are possible ways.

- Third party tools
- Use jsEvaluation of java

Based on efficency(time and functionality) we have used jsEvaluation engine with multithreading.
Here all data are stored in jsEvaluation engnine object.

7.2 Checker

Data in files are inserted by end user. Technical data can be incorrect. Checker is functionality to verify this kind of checks.

- Find checks and classify it.
- User must define data in given form.
- Dependency check, Ex- pair

- Data range
- Possible value

7.2.1 Type of Checker

- Static
  - Checks that are directly applied on values of file.
- Dynamic
  - Checks that are not directly applied on values of file.
  - First value or equation is evaluated to get final value.
  - Dynamic checks are applied on final evaluated value.

7.3 Boundry Function

For end user circuit is like black box,i.e they give simply connection from out side. How this connections are connected inside circuit is not known to end users. All these impementation are included in Boundry function.

Here challange is to generate generic boundry,because in different technology different size and shape of circuit are generated. In short we are dealing with different kind of polygon and try to create its boundry.

Following figure 7.1 shows problem defination.

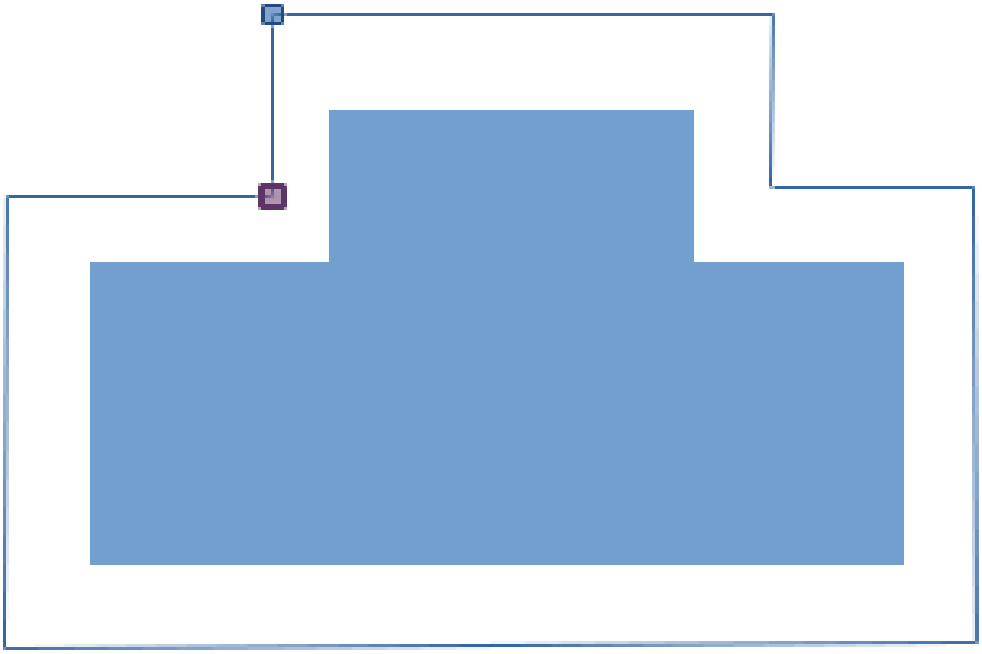

Figure 7.1: Boundry function

Chapter 8

Project Execution

This chapter shows steps to execute project.

8.1 Product Generation

Basically For making product only compilation is not enough. We must provide user proper environment. Set environmnet is one of the important part. User must be able to use all products,so we have to combine our product with other products independetly as much as possible.
We can say in simple word it is like convert code to exe and also set path variable in environment.

8.2 Steps to use and execute product

- Use product on which our product is dependent.
  Simply user has to change path in one file.
- Also use main product
- Check Envirement variable value to verify wether product is correctly sourced or not
- Give file paths and other required detail
  For testing purpose script is used to generate number of cuts.

Chapter 9

Scope of the Project

- Multiple way of input with single interface
- Provides a single platform for all technology i.e 28fdsoi,14fdsoi etc
- Provides for various memory architecture
- Provides uniqe way to generate different file for different technology
- User friendly way to deal with template file
- Support for Any kind of file in future with very less support

Chapter 10

Tools and Technology

- Core Java for Business logic implementation
- Eclipse IDE
- Emmacode coverage plugin
- DOM parser for XML
- Reflection API
- Shell scripting

Chapter 11

Conclusion And Enhancement

Input for this project is Files contain data updated or created by user. These files are parsed and all data is converted in to intermediat form. Ultimately this product can support different type of files like text and xml. This data in intermediate form is processed and data is given to view generator and view generator generate view(which is specified by user). Same product is used to generate view for different technology. User has to change data given in input files. By using all these we have tried to create best architecture for project and reduced time and line of code. Improvement is shown in table 11.1.

Parameter	**Previous Value**	**Current Value**
Time(sec)	45	20
lines	2167	1500
views	2	5
interface	text	text and xml

Table 11.1: Improvements

Only parsing module need to be developed to support new kind of file and provide interface for international standard.

AS a enhancement user may given GUI to give file path of input files and also for updating files and other extra views will be added for verification purpose.

References

[1] ST internal Documents.

[2] ST tools documents .

[3] ST UniBE document

[4] Taku Noda ," Proposal of Circuit Descriptive language",2001,"Research Insitute of Electric Power Endustry,Japan"

[5] W. Jai Singh, S. Nithya Bala, "An Adaptive and Efficient XML Parser Tool for Domain Specifc Languages",2011 International Journal of Scientifc and Engineering Research Volume 2

[6] Bhanu Pratap Singh, Sohan Garg, "A Characteristics Study of Ant Colony Optimization Algorithms for Routing Problems", March 2013,International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 3

[7] Richa Garg,Saurabh mittal, "Optimization by Genetic Algorithm", April 2014,International Journal of Advanced Research in Computer Science and Software Engineering Volume 4, Issue 4

[8] Ms. Sonali. B. Maind,Ms. Priyanka Wankar, "Research Paper on Basic of Artificial Neural Network",International Journal on Recent and Innovation Trends in Computing and Communication Volume: 2 Issue: 1

[9] http://www.vogella.com/tutorials/JavaRegularExpressions/article.html

[10] http://docs.oracle.com/javase/tutorial/reflect