МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет

«Московский институт электронной техники»

Кафедра проектирования и конструирования интегральных микросхем

Скорописов Артем Александрович

Бакалаврская работа по направлению

11.03.04 «Электроника и наноэлектроника»

Создание программы, реализующей алгоритм трассировки методом Соукупа

Студент Скорописов А.А.

Руководитель от кафедры, {уч.степень, уч.звание} Петраков В.И.

Москва 2021

**Содержание**

**Введение......................................................................................................................3**

**1. Теоретическая часть. .....................................………………………………......3**

**1.1. Маршрут проектирования ИС. .......................................…............…...…….3**

**1.2. Этап топологического проектирования.........................…………………...6**

**1.3. Трассировка. ..................................................................................................….8**

**1.4. Форматы топологии. ....................................................................……...…….9**

**1.5. Формат LEF. .................................................................................................…10**

**1.6. Ключевые разделы LEF – файла. …………………………...……...……...12**

**1.7. Программа Klayout. …………………………………………...……….…….13**

**1.8. Формат DEF. ……………………................................................................…14**

**1.9. Программа KLayout. .......................................................…………………...17**

**1.10. Обоснование выбора языка. ............................................................……...18**

**2. Практическая часть. .....................................................................…………….19**

**2.1. Структуры хранения данных. ................................................................…..19**

**2.2. Считывание данных. ..................................................................……………21**

**2.3. Структура программы. ........................................................................……...22**

**2.4. Получение входных данных. ...............................................................…….24**

**2.5. Алгоритм Соукупа. ......................................................................................…26**

**2.6. Улучшение работы программы. ...........................…………………………28**

**2.7. Трассировка реальной схемы. ...............................................................…...29**

**Вывод. …....................................................................................................................32 Список используемой литературы. ...............………………………...………...33**

**Введение.**

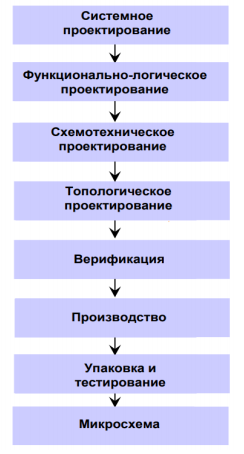
Одной из основных проблем проектирования современных сверхбольших и больших интегральных схем (СБИС, БИС), содержащих огромное количество элементов на одном кристалле является сокращение времени проектирования. Учитывая высокую функциональную сложность интегральных микросхем (ИС), решение данной проблемы возможно лишь при использовании различных методов автоматизации в системах автоматизированного проектирования (САПР), содержащих в себе мощную математическую базу и имеющих грамотную программную реализацию. Особое внимание следует уделить процессу трассировки на этапе топологического проектирования. Он требует применения сложных алгоритмов поиска путей для проведения каждого соединения.

Целью данной выпускной квалификационной работы было создание программы, реализующей алгоритм трассировки методом Соукупа. Для ее достижения были поставлены следующие задачи:

1. Изучение форматов хранения топологии LEF/DEF.
2. Написание программы, реализующей считывание данных и трассировку методом Соукупа, на языке высокого уровня С++.
3. Визуализация полученных результатов.
4. **Теоретическая часть.**

**1.1. Маршрут проектирования ИС.**

В процессе создания каждая интегральная микросхема проходит определенный маршрут проектирования, который представлен на рисунке 1.



*Рис.1. Маршрут проектирования ИС*

Каждый этап в свою очередь разбивается на несколько других этапов и может повторяться для достижения желаемых характеристик ИС.

На этапе системного проектирования создается поведенческая модель разрабатываемой схемы и определяется состав сложно-функциональных блоков (СФ-блоков). Разработка схемы начинается с анализа задач и требований, а также написания системной спецификации. При этом определяются основные эксплуатационно-технические свойства ИС, такие как требуемое быстродействие, допустимые задержки сигналов, допустимая потребляемая мощность и др. Исполняемые спецификации представляются в определенном формате на языках С, С++, Verilog и VHDL. Далее создается алгоритм работы схемы на уровне математического описания на основе разработанной ранее системной спецификации. Производится математическое моделирование разработанных алгоритмов функционирования ИС оценка и требуемых характеристик. Также при необходимости может производиться синтез наборов данных (сигналов), предназначенных для тестирования схемотехнических решений реализуемой ИС. На основе алгоритма работы ИС разрабатывается поведенческая модель системы на уровне СФ-блоков в виде блок-схемы, отражающей принцип взаимодействия СФ-блоков в составе ИС и включающей их основные параметры.

Функционально-логический этап можно условно разделить на функциональный и логические этапы. На этапе функционального проектирования создается RTL-описание (register transfer level - уровень регистровых передач) функциональной модели системы на языке описания аппаратуры (VHDL, Verilog) и осуществляется верификация и функциональное тестирование схемотехнических решений ИС. Функциональная модель ИС реализуется либо на уровне крупных структурных блоков (PMS-level), либо на уровне регистров (RTL-level). Сначала разрабатывают все цифровые и аналоговые СФ-блоки. Для цифровых СФ-блоков создают поведенческую модель (RTL-описания) на языке описания аппаратуры. Аналоговые СФ-блоки проектируют на транзисторном уровне. Задаются необходимые требования и ограничения. На основе всех выработанных требований и ограничений составляют окончательную детальную спецификацию, которую будут использовать при проектировании микросхемы, разработке программы производственного контроля и подготовке рекомендаций по применению изделия. На этапе логического проектирования осуществляется описание схемы на вентильном уровне, представляемое в форме списка соединений (Netlist), который является текстовой формой кодирования схемы. При проведении синтеза формируют список соединений логических библиотечных элементов в базисе выбранного технологического процесса. Выходной формат файла списка соединений — EDIF, Verilog или VHDL. При формировании списка соединений задаются ограничения на временные задержки, размещение элементов и связи. Далее проводится физическая оптимизация путем выполнения синтеза с заданными ограничениями по временным характеристикам, потребляемой мощности, площади и другим параметрам.

На этапе схемотехнического проектирования решаются задачи структурного синтеза принципиальной электрической схемы ИС, статического анализа и оптимизации, производится расчет статических состояний и переходных процессов. Под структурным синтезом понимается получение конфигурации принципиальной электрической схемы, оптимальным образом удовлетворяющей требованиям. Классическим подходом является аппроксимация желаемых характеристик заданным функциям, а затем их реализация с помощью логических элементов (ЛЭ) из выбранного набора. В зависимости от того, какое сочетание значений указанных параметров наиболее полно удовлетворяет поставленные требования, выбирается логика схемы. Это может быть комплементарная логика на полевых транзисторах (КМДП), эмиттерно-связанная логика (ЭСЛ), транзисторно-транзисторная логика (ТТЛ) или интегрально-инжекционная логика (И2Л). Тип выбранной логики во многом определяет принципиальные электрические схемы функциональных устройств и блоков. При анализе статистических характеристик рассчитываются токи и напряжения в любом узле схемы, анализируются вольтамперные характеристики (ВАХ) и исследуется влияние параметров компонентов на них. При анализе динамических характеристик (переходных процессов) ИС обычно исследуется поведение схемы при воздействии импульсных сигналов.

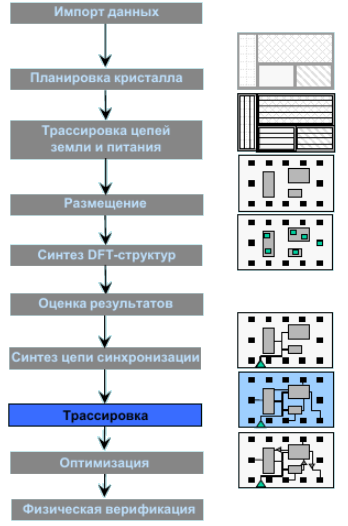
Топологический этап проектирования ИС представляет в данной работе особый интерес, поэтому будет рассмотрен более детально в следующей главе.

Этап верификации и подготовки к производству предназначен для осуществления окончательной проверки ИС на соответствие требованиям спецификаций, осуществления операций для подготовки к производству, разработки конструкторской и сопроводительной документации для передачи изготовителю. Формальную верификацию проводят для того, чтобы проверить проект на соответствие требованиям спецификаций и принять решение о передаче информации для изготовления ИС на производство. Финальное моделирование осуществляют с учетом временных задержек на линиях связи, взаимного влияния элементов, линий связи и других паразитных эффектов. На основе отчетов оценивают соответствие фактических параметров цепей заданным значениям, а также полноту и корректность заданных ограничений. Проводится окончательная проверка топологии ИС на соответствие правилам топологического проектирования (Design Rule Check, DRC) и проверка соответствия электрической схемы и топологии (Layout Versus Schematic, LVS). Подготовка к производству включает в себя добавление в топологию ИС реперных знаков для идентификации контактных площадок при выполнении операции разварки, условных обозначений для идентификации фотошаблонов и изготавливаемых в дальнейшем кремниевых структур, формирование защитного кольца (Seal Ring), дополнительной защитной металлизации на углах (Chip Corner Stress Relief Pattern) и необходимой защиты кремниевых структур от теплового воздействия при выполнении операций резки и корпусирования и другие вспомогательные операции. Финальной стадией является разработка конструкторской документации для изготовления микросхемы как сборочной единицы, включающей корпус, кристалл и элементы соединений. После этого вся разработанная документация передается на фабрику для производства.

**1.2. Этап топологического проектирования.**

Этап топологического проектирования в этом маршруте проектирования интегральной схемы является одним из наиболее трудоемких. Именно этот этап рассматривается в данной работе. На этапе топологического проектирования осуществляется переход от логического уровня проектирования к физической реализации с учетом влияния технологических ограничений и физических факторов. Исходными данными для топологического проектирования являются требования к геометрическим параметрам кристалла, перечисленные выше, а также требования к электрическим параметрам (токи потребления по цепям питания, питающие напряжения), требования по электромагнитной совместимости, ограничения техпроцесса. Планирование геометрии кристалла заключается в разбиении схемы на макроблоки, задании их размеров и форм, а также размещении контактных площадок, планировании размещения СФ-блоков и модулей памяти. Далее производится автоматическое размещение ячеек в подготовленный план кристалла ИС. При этом задаются дополнительные параметры, например плотность расстановки ячеек как на весь кристалл, так и на отдельные его области. Согласно информации о задержках на вентилях и ограничениям на быстродействие, потребляемую мощность и токи утечки проводится оптимизация моделей сигнальных цепей и списка соединений. Для этого определяют наиболее медленные цепи, после чего проводят замену вентилей, входящих в их состав, на более быстродействующие, изменяют их местоположение на кристалле и линии связи между ними. Далее автоматически производится трассировка цепей тактовой синхронизации. Одним из наиболее часто применяемых подходов к проектированию является использование сбалансированных путей (деревьев). После этого проводится повторная оптимизация по временным ограничениям, ограничениям динамического потребления и токов утечки, а также временной анализ. Затем проводится размещение элементов для обеспечения постоянных уровней логической единицы и логического нуля (TieHi, TieLo) для логических элементов, входящих в состав ИС. Автоматическую трассировку сигнальных цепей проводят с учетом технологических требований фабрики, к которым относятся ширина проводника, зазор между полигонами металла и количество слоев металлов. Связи, для которых необходим учет особых требований (ограничения на временные параметры, требования электромагнитной совместимости и т. п.), разводятся отдельно.

Этап топологического проектирования включает в себя несколько этапов, как показано на рисунке 2, конечным результатом которых является топология ИС. **Топология ИС** – геометрический эквивалент электрической схемы, реализуемый в виде геометрических примитивов. Или, другими словами, это зафиксированное на материальном носителе пространственно-геометрическое расположение совокупности элементов ИС и связей между ними.



*Рис.2. Маршрут топологического проектирования.*

**1.3. Трассировка.**

**Трассировка** — одна из задач проектирования топологии, заключающаяся в определении мест расположения проводников на плате с использованием САПР (система автоматизированного проектирования), предназначенной для проектирования интегральных схем. Трассировка соединений является, как правило, заключительным этапом конструкторского проектирования ИС и состоит в определении линий, соединяющих эквипотенциальные контакты элементов, и компонентов, составляющих проектируемое устройство. **Основная задача трассировки** формулируется следующим образом: по заданной схеме соединений проложить необходимые проводники на плоскости (плате, кристалле и т.д.), чтобы реализовать заданные технические соединения с учётом заранее заданных ограничений.

Ограничениями для трассировки являются:

* положения элементов (размещение);
* количество слоев для трассировки;
* допустимые задержки;
* правила проектирования;
* физические/электрические/производственные ограничения (перекрестные помехи, отклонения в тех. процессе, выход годных и литография).

Критериями трассировки ИС обычно являются:

* трассировка 100% цепей;
* удовлетворение правилам проектирования;
* минимизация суммарной длины цепей;
* сокращение задержки в критических путях и т.д.;
* равномерное заполнение области трассировки;
* целостность сигналов;
* Количество использованных коммутационных слоев.

Часто эти критерии и ограничения являются взаимоисключающими, поэтому трассировка всегда является трудоемким и сложным процессом. Исходной информацией для решения задачи трассировки соединений обычно являются список цепей и параметры конструкции, а также данные по размещению элементов. Существует два основных подхода к трассировке.

Последовательная трассировка:

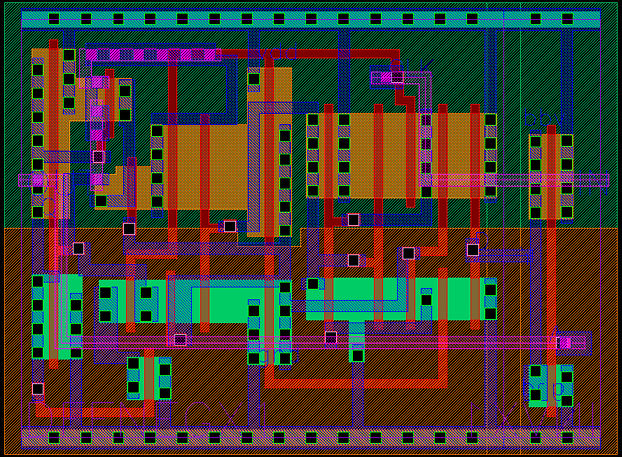
* цепи трассируются по одной за раз;
* порядок зависит от нескольких факторов (критичность, оценка длины цепи, количество терминалов и т.д.);
* когда трассировка больше невозможна из-за блокирующих цепей, то эти цепи передвигаются или разрываются.

Одновременная трассировка:

* все цепи трассируются одновременно (нет никакого порядка);
* формулируется как целочисленное линейное программирование.

**1.4. Форматы топологии.**

Существуют различные способы и форматы создания, хранения и обработки топологии. Наиболее известным и популярным форматом представления топологии является формат GDS II. GDS II (GDS) (Graphic Database System) — формат файлов баз данных для обмена данными по интегральным схемам и их топологиям. Данный формат описывает плоские геометрические формы, текстовые метки и иную информацию в иерархической форме. Данные могут использоваться для обмена между различными САПР или для создания фотошаблонов. Кроме того, популярными форматами считаются OASIS (новый предпочтительный формат), DXF, CIF, Gerber. Пример представления топологии ИС в формате GDS II представлен на рисунке 3.



*Рис.3. Представление топологии ИС в формате GDS II.*

**1.5. Формат LEF.**

При рассмотрении информации для трассировки удобно использовать формат **LEF/DEF** - открытый формат, представляющий собой универсальное и гибкое решение для формирования, хранения и обработки топологии в текстовом виде. Фактически, LEF/DEF состоит из двух частей, соответственно, **LEF** (Library Exchange Format) и **DEF** (Design Exchange Format). Ниже представлено содержание LEF и DEF файлов.

*Фрагмент содержания LEF-файла*

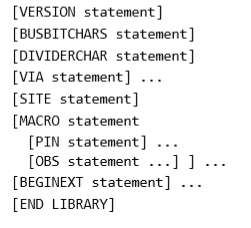
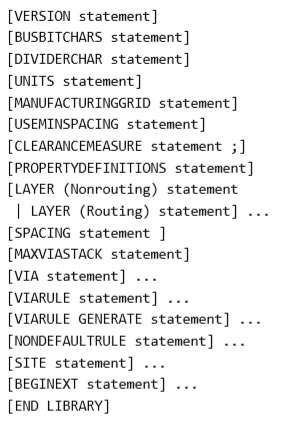
VERSION 5.4 ;  
 NAMESCASESENSITIVE ON ;  
 BUSBITCHARS "[]" ;  
 UNITS  
 DATABASE MICRONS 1000 ;  
 RESISTANCE OHMS 10 ;   
 END UNITS  
 MANUFACTURINGGRID 0.025 ;  
 LAYER M1M  
 TYPE ROUTING ;  
 WIDTH 0.5 ;  
 SPACING 0.45 ;  
 PITCH 1.3 ;  
 DIRECTION HORIZONTAL ;  
 CAPACITANCE CPERSQDIST 0.0000400000 ;  
 RESISTANCE RPERSQ 0.12000000 ;  
 EDGECAPACITANCE 0.0000460000 ;  
 ANTENNAAREARATIO 100 ;  
 ANTENNADIFFAREARATIO 9999999 ;   
END M1M

VIA CLVIA1 DEFAULT  
 RESISTANCE 6 ;  
 FOREIGN CLVIA1 ;  
 LAYER M1M ;  
 RECT -0.450000 -0.450000 0.450000 0.450000 ;  
 LAYER V1M ;  
 RECT -0.250000 -0.250000 0.250000 0.250000 ;  
 LAYER M2M ;  
 RECT -0.400000 -0.400000 0.400000 0.400000 ;  
 END CLVIA1

VIARULE V1M12W GENERATE  
 LAYER M2M ;  
 DIRECTION VERTICAL ;   
WIDTH 10 TO 1000.0 ;   
OVERHANG 0.15 ;  
 METALOVERHANG 0.00 ;  
 ENCLOSURE 1 2;  
 ...  
 RESISTANCE 6 ;  
 END V1M12W

SPACING  
 SAMENET M1M M1M 0.45 STACK ;  
 SAMENET M2M M2M 0.50 STACK ;  
 SAMENET M3M M3M 0.50 STACK ;  
 SAMENET V1M V1M 0.45 STACK ;  
 SAMENET V2M V2M 0.45 STACK ;  
 END SPACING  
 END LIBRARY

**LEF** предусматривает хранение информации о технологическом процессе и о библиотеке компонентов интегральной схемы. В свою очередь LEF, как правило, для удобства использования разделяются на два файла: технологический и библиотечный. Технологический файл содержит в себе информацию, касающуюся техпроцесса. В него входит описание слоев металлизации, межуровневых контактов и ячеек размещения компонентов. Библиотечный файл состоит из списка компонентов в заданном формате. На рисунке 4 представлены спецификации технологического (а) и библиотечного (б) LEF-файлов.



*Рис.4. Спецификации технологического (а) и библиотечного (б) LEF-файлов.*

**1.6. Ключевые разделы LEF-файла.**

**UNITS**

Раздел UNITS задает используемые в файле единицы измерения. Через задание этих параметров регулируется точность используемых значений.

**LAYER**

Секции LAYER описывают параметры слоев металлизации СБИС. Порядок следования слоев в технологическом LEF-файле однозначно устанавливает их порядок расположения в кристалле.

**VIA**

Секция VIA описывает параметры межслойных контактов. И задается в порядке – начальный слой металлизации, межслойный диэлектрик и конечный слой металлизации.

**SITE**

Раздел SITE описывает ячейку для расположения компонентов.

**MACRO**

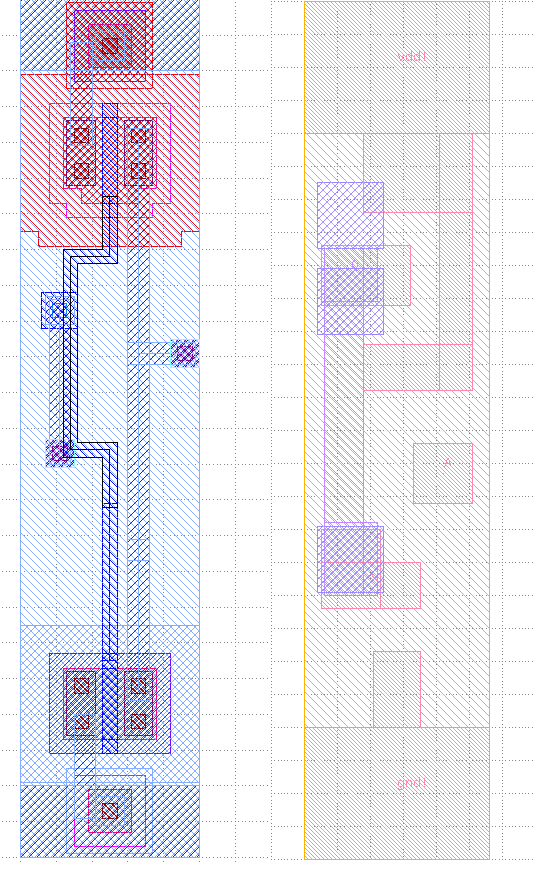
Секция MACRO описывает компоненты СБИС. В описание входят все размеры и расположение контактов каждого компонента.

Основными отличиями форматов LEF и GDS (GDS II) является следующее:

* в LEF файлах информация содержится в текстовом виде, как было показано выше, в файлах формата GDS (GDS II) - в бинарном.
* в формате LEF содержится информация лишь о слоях металла, в то время как GDS (GDS II) содержит полную информацию о топологии ИС.
* LEF файлы используются в основном на этапах трассировки, а GDS (GDS II) файлы на этапах выпуска фотошаблонов.

**1.7. Программа KLayout.**

Для сравнения представления топологии в формате LEF и GDS II удобно воспользоваться программой KLayout. **KLayout** – САПР топологий интегральных микросхем. Данная кроссплатформенная САПР написана на С++ с использованием тулкита Qt, в ней есть встроенная функция просмотра файлов топологий схем. На рисунке 5 показаны файлы GDS (слева) и LEF (справа) представленные через программу KLayout.



*Рис.5. Сравнение представления топологии инвертора в форматах GDS и LEF с помощью программы KLayout.*

Основываясь на последнем отличии, упомянутом выше, в ходе данной работы рассматриваются файлы формата LEF/DEF.

**1.8. Формат DEF.**

Формат **DEF** разработан компанией Cadence, свободно распространяется в качестве средства отображения топологии интегральных схем в формате ASCII и является одним из самых простых способов переноса проекта из среды трассировки и размещения в среду топологического проектирования и верификации. В DEF-файле содержится информация о топологии СБИС. Эти данные состоят из двух частей: логической и физической. Логическая часть может состоять из информации о связности схемы (через список цепей), группировке и физических ограничениях. Физическая часть содержит данные о размещении компонентов и трассировке межсоединений, как показано на рисунке 6.

*Фрагмент содержания DEF-файла*

VERSION 5.8 ;  
 DIVIDERCHAR "/" ;  
 BUSBITCHARS "[]" ;  
 DESIGN testDesign ;  
 UNITS DISTANCE MICRONS 100 ;  
 PROPERTYDEFINITIONS  
 COMPONENTPIN designRuleWidth REAL ;  
 DESIGN FE\_CORE\_BOX\_LL\_X REAL 10.240 ;  
 DESIGN FE\_CORE\_BOX\_UR\_X REAL 259.840 ;  
 DESIGN FE\_CORE\_BOX\_LL\_Y REAL 10.240 ;  
 DESIGN FE\_CORE\_BOX\_UR\_Y REAL 259.840 ;  
 END PROPERTYDEFINITIONS  
 DIEAREA ( 0 0 ) ( 27008 27008 ) ;  
 ROW CORE\_ROW\_0 CORE 1024 1024 FS DO 390 BY 1 STEP 64 0 ;

ROW CORE\_ROW\_1 CORE 1024 1600 N DO 390 BY 1 STEP 64 0 ;

...

TRACKS Y 128 DO 210 STEP 128 LAYER metal6 ;  
 TRACKS X 128 DO 210 STEP 128 LAYER metal6 ;

...

VIAS 5 ;  
 - M6\_M5\_G\_1  
 + VIARULE M6\_M5\_G  
 + CUTSIZE 60 60  
 + LAYERS metal5 v5 metal6  
 + CUTSPACING 36 36  
 + ENCLOSURE 20 20 20 20  
 + ROWCOL 1 1 ;  
 ...

END VIAS

COMPONENTS 1527 ;  
 - bufInput2 buf\_32 + PLACED ( 20416 20608 ) S ;  
 - bufInput3 buf\_32 + PLACED ( 23808 7936 ) FS ;  
 ...

END COMPONENTS

PINS 646 ;  
 - Out0 + NET Out0 + DIRECTION OUTPUT + USE SIGNAL  
 + LAYER metal4 ( -16 0 ) ( 16 63 )  
 + PLACED ( 27008 26944 ) W ;

...

END PINS

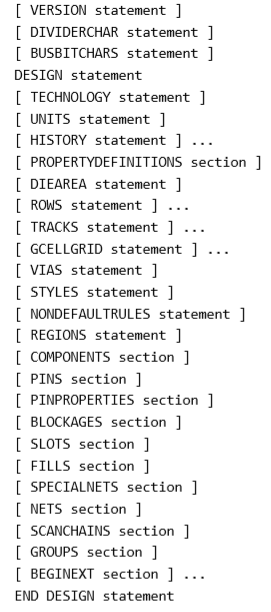
SPECIALNETS 2 ;  
 - VDD ( \* vdd! )  
 + ROUTED metal5 100 + SHAPE RING ( 562 26300 ) ( 26446 \* )  
 NEW metal1 180 + SHAPE FOLLOWPIN ( 1024 25216 ) ( 25984 \* )

...

- net\_bc3  
 ( cell\_sdffpsqb\_4545 sdi ) ( cell\_sdffpsqb\_2545 sdi )

...

END NETS  
 END DESIGN



*Рис.6. Спецификация DEF-файла.*

**1.9. Ключевые разделы DEF-файла**

Структуры **UNITS** и **VIAS** соответствуют структурам **UNITS** и **VIA**, соответственно, в файле LEF.

**DIEAREA**

Раздел DIEAREA описывает размеры и форму кристалла.

**ROWS**

Раздел ROWS описывает строку, состоящую из ячеек для размещения компонентов.

**TRACKS**

TRACKS задает сетку трассировки в слое металлизации.

**COMPONENTS**

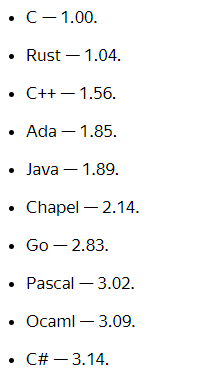
В этой секции описываются компоненты схемы. Описание каждого компонента сводится к заданию его местоположения и указанию значений его атрибутов.

**PINS**

В секции PINS перечислены выводы схемы.

**1.10. Обоснование выбора языка.**

Для создания программы, реализующей алгоритм быстрой трассировки методом Соукупа был выбран язык программирования высокого уровня С++ и среда разработки Microsoft Visual Studio 2019. С++ — компилируемый статически типизированный язык программирования общего назначения. Этот язык наиболее подходит для реализации алгоритма Соукупа и других алгоритмов трассировки, так как имеет много полезных и удобных, встроенных функций и методов, а также высокое быстродействие, что важно на этапе трассировки. C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также высокобюджетных игр (ААА). Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. Например, на платформе x86 это GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder и другие. Синтаксис C++ унаследован от языка C. Одним из принципов разработки было сохранение совместимости с C. Тем не менее, C++ не является в строгом смысле надмножеством C; множество программ, которые могут одинаково успешно транслироваться как компиляторами C, так и компиляторами C++, довольно велико, но не включает все возможные программы на C. Главным преимуществом языка С++ является быстродействие. Программы, написанные на С++ в несколько раз быстрее точно таких же программ, написанных на других языках (кроме ассемблера). Это объясняется тем, что код С++ не должен интерпретироваться во время исполнения, в отличие от большинства других языков. На рисунке 7 показано среднее время выполнения тестов разными языками программирования. (Тесты включали в себя математические вычисления, отрисовку графики, считывание и запись данных в файл и другое).



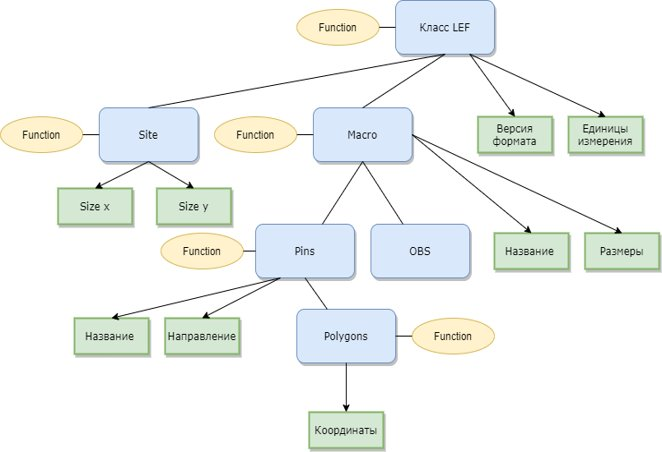
*Рис.7. Сравнение скорости работы языков программирования.*

Для реализации графики в языке С++ существует несколько вариантов. При подключении определенных библиотек достигается возможность использования спецификации OpenGL, что позволяет легко визуализировать полученный результат. OpenGL (Open Graphics Library) — спецификация, определяющая платформонезависимый (независимый от языка программирования) программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику. Она включает более 300 функций для рисования сложных трёхмерных сцен из простых примитивов. Используется при создании компьютерных игр, САПР, виртуальной реальности, визуализации в научных исследованиях.

1. **Практическая часть.**

**2.1 Структуры хранения данных.**

Для получения исходной информации в качестве входных параметров для программы необходимо извлечь ее из файлов хранения топологии LEF и DEF, а также из файла формата Verilog, который содержит в себе информацию о соединениях элементов ИС. Кроме того, для дальнейшего использования полученной информации необходимо создать структуру, в которой эта информация будет храниться. Рассмотрим схематическое представление общей структуры данных на примере структуры для хранения информации из LEF-файла. Она представлена на рисунке 8.



*Рис.8. Структура данных LEF.*

Для данной программы в качестве таких структур выступают классы (class LEFFile, class DEFFile и class VERFile для данных из LEF, DEF и Verilog файлов, соответственно). Структура и содержимое класса LEFFile представлена на рисунке 9.



*Рис.9. Код класса LEFfile.*

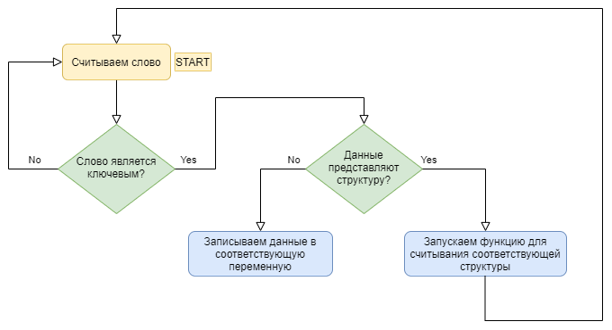
Класс LEFFile содержит собственные переменные, такие как fileName, содержащей в себе имя LEF-файла, version, отвечающей за версию формата LEF, microns, отвечающей за единицы измерения, применяемые в данном файле, и другие переменные. Кроме того, он содержит в себе векторы классов macroes и sites, которые построены аналогичным образом и содержат информацию о ячейках. Также класс LEFFile имеет собственные функции ReadUnits, ReadSite, ReadMacro для чтения соответствующих фрагментов LEF-файла. На рисунке 10 представлена структура класса Macro, содержащая в себе переменные и векторы классов для хранения информации о конкретных ячейках, и собственные функции для чтения определенных данных.



*Рис.10. Код класса Macro.*

**2.2. Считывание данных.**

За считывание данных отвечают соответствующие функции, принцип работы которых показан на рисунке 11.



*Рис.11. Принцип работы функций считывания данных.*

Рассмотрим принцип работы функция на примере получения данных о контактах определенной ячейки LEF-файла, рисунки 12 и 13. Обозначим ключевые слова для поиска нужных данных. Информация о контактах идет после ключевого слова “PIN” в стандартной ячейке. Информация о ячейках начинается после ключевого слова “MACRO” в LEF-файле. Считываем данные из файла пословно функцией Read. Как только встречается ключевое слово “MACRO”, программа понимает, что далее пойдет информация о стандартных ячейках и вызывает функцию чтения ячеек ReadMacro. Функция ReadMacro работает аналогично, она считывает данные пока не встретит ключевое слово “PIN”. После данного слова начинается информация о контактах ячейки, поэтому вызывается функция чтения контактов ReadPin, которая находит и записывает в переменные класса Pin нужные данные (координаты, размеры и направление контактов).



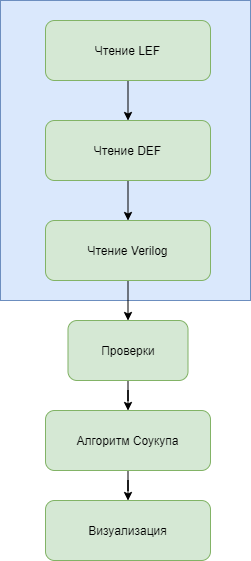
*Рис.12. Фрагмент функции чтения LEF-файла.*



*Рис.13. Фрагмент функции чтения ячейки.*

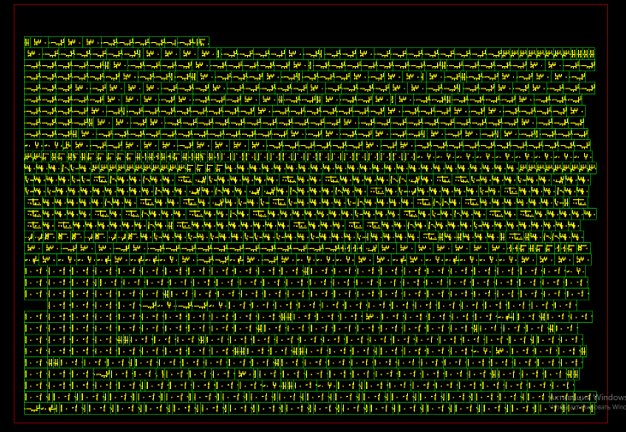
**2.3. Структура программы.**

Общая структура программы показана на рисунке 14. Сначала происходит чтение LEF-файла, затем DEF-файла и Verilog-файла. После этого полученные данные о координатах контактов используются в качестве входных данных в алгоритме трассировки Соукупа. Завершающим этапом является визуализация топологии ИС на основе полученных данных из форматов топологии и визуализация цепей трассировки. Визуализация осуществляется с помощью библиотеки glut64.lib и glut.h спецификации OpenGL.



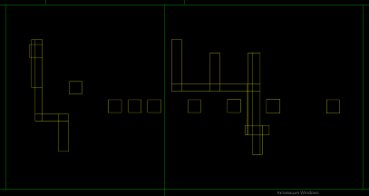
*Рис.14. Структура программы.*

На основе данных, полученных из DEF-файла, формируется форма кристалла ИС, а также определяется положение всех элементов. Элементы формируются в строки. На основе данных, полученных из LEF-файла, формируются стандартные ячейки определенной формы и размера со всеми контактами. При визуализации всех этих данных получается топология ИС со всеми ее ячейками и контактами ячеек, как показано на рисунке 15. Красным цветом обозначены границы кристалла ИС, зеленым - границы стандартных ячеек, а желтым - контакты ячеек.

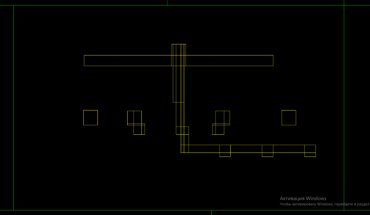


*Рис.15. Топология ИС.*

При увеличении масштаба можно более детально рассмотреть каждую ячейку, как показано на рисунках 16 и 17.



*Рис.16. Топология отдельных стандартных ячеек.*



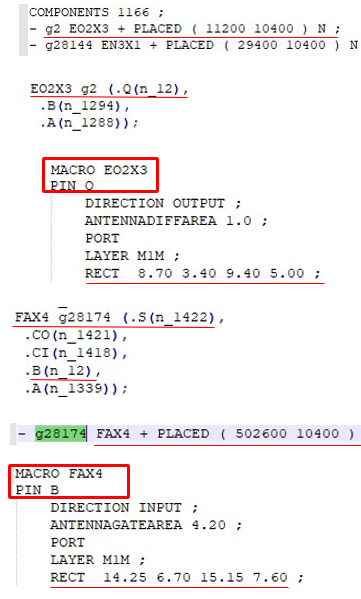
*Рис.17. Топология отдельных стандартных ячеек.*

**2.4. Получение входных данных.**

Входными данными алгоритма трассировки являются координаты контактов элементов и список элементов, которые необходимо соединить проводником. Информация о том, какие контакты каких элементов нужно соединить находится в Verilog-файле. Если 2 и более контакта разных элементов имеют одинаковое название подключенного проводника (wire), значит эти контакты нужно соединить. Для определения таких контактов и их координат используется следующий алгоритм:

* рассмотрим первый элемент из DEF файла и запомним его координаты
* находим этот элемент в Verilog-файле и рассмотрим его первый контакт;
* находим соответствующий элемент в LEF-файле;
* находим соответствующий контакт и определяем его координаты (x1, y1);
* возвращаемся в Verilog-файл и определяем соединение, подключенное к рассматриваемому контакту;
* находим элемент, контакт которого подключен к соединению с таким же названием, запоминаем его координаты;
* находим соответствующий элемент в LEF-файле;
* находим соответствующий контакт и определяем его координаты (x2, y2);
* повторим алгоритм для последующих контактов рассматриваемого элемента, а после для других элементов.

В файлах LEF, DEF и Verilog этот алгоритм выглядит как показано на рисунке 18.



*Рис.18. Порядок определения нужных контактов и их координат.*

**2.5. Алгоритм Соукупа.**

**Алгоритм быстрой трассировки методом Соукупа** – итерационный волновой алгоритм поиска с ортогональным распространением волн и восстановление обратного пути по точкам. Содержит в своей основе алгоритм Ли, но имеет перед ним 2 значительных преимущества:

* позволяет осуществлять трассировку из нескольких точек, что ускоряет процесс;
* волны распространяются по приоритетам, что позволяет располагать трассы, избегая тупиковых ситуаций.

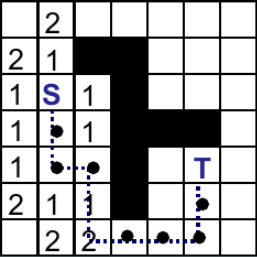
При использовании данного алгоритма необходимо придерживаться определенных правил:

* трассировка происходит по приоритетам, поэтому не должно быть 2 и более цепей с одинаковым приоритетом;
* терминалы цепи не должны затрагиваться;
* построенные трассы неразъединимы.

Принцип работы алгоритма Соукупа для одной цепи:

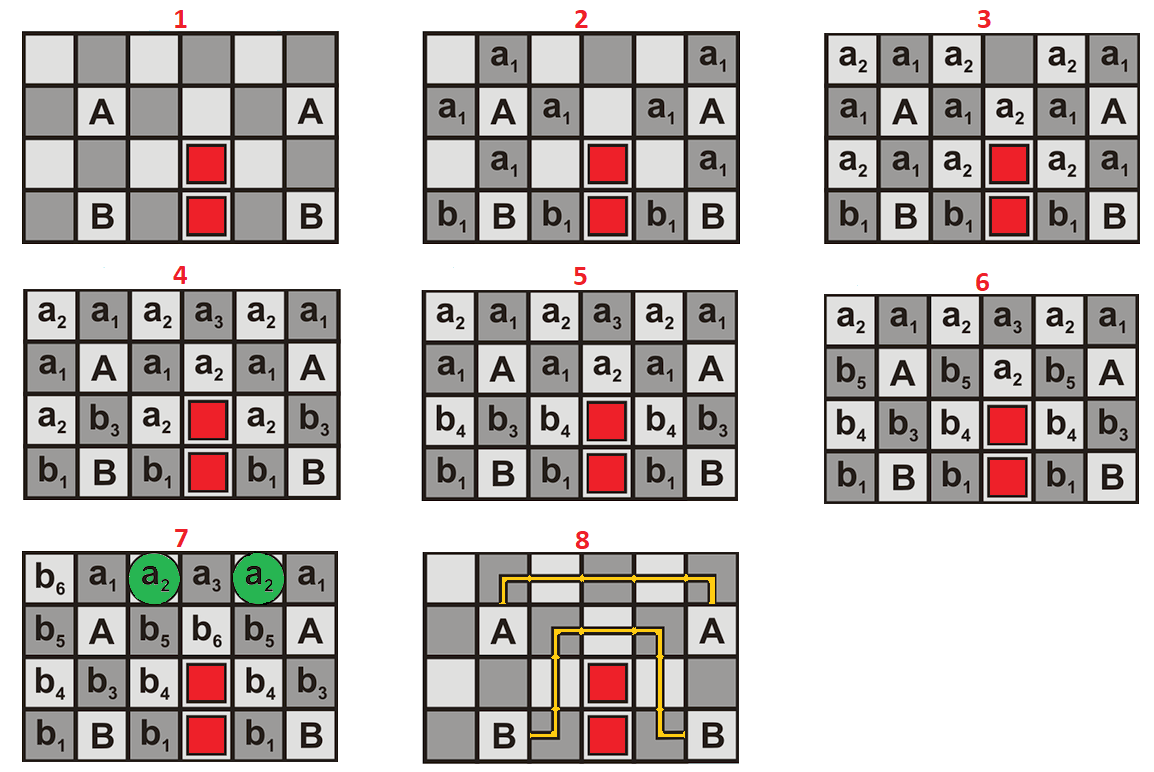
* использует поиск в глубину и ширину;
* распространяется в направлении приемника без смены направления;
* если встречается препятствие, то ищет обход поиском в ширину (алгоритм Ли);
* если препятствие преодолено, то ищет в направлении приемника.

Принцип работы алгоритма для одной цепи показан на рисунке 19.



*Рис.19. Принцип работы алгоритма Соукупа для одной цепи.*

**Принцип работы алгоритма Соукупа для нескольких цепей.** На первом шаге определяются приоритеты цепей. Приоритет незавершенных цепей всегда больше, чем завершенных. При этом приоритет завершенной цепи пропорционален ее размеру, а у незавершенной - обратно пропорционален расстоянию до ближайшего фрагмента. После этого от каждого терминала в соответствии с приоритетом испускаются волны. То есть, волна, испускаемая от терминала с большим приоритетом, может накладываться на волну, испускаемую терминалом с меньшим приоритетом. Сама цепь представляется множеством дискретов. Места, которые представляют потенциальную опасность разрыва цепи называются критическими дискретами. Их нельзя перекрывать другими волнами и трассами, так как в таком случае произойдет трассировка не всех цепей. Волны распространяются таким образом, пока не поменяется приоритет цепей. Это может произойти, если одна из цепей окажется соединена. Эти шаги будут повторяться, пока не будет проведена трассировка ВСЕХ цепей. Принцип трассировки нескольких цепей показан на рисунке 20. Таким образом мы получаем обход препятствий за счет алгоритма Ли и трассируемость всех цепей за счет разницы и смены приоритетов.

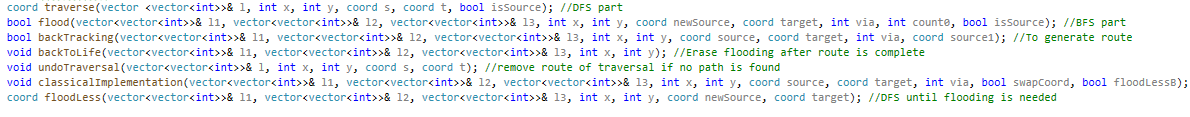


*Рис.20. Принцип работы алгоритма Соукупа для нескольких цепей.*

В созданной программе работа алгоритма Соукупа осуществляется за счет функций:

* input - определяет координаты контактов, которые необходимо соединить, как было описано выше;
* flood - выполняет роль волнового алгоритма (алгоритма ЛИ);
* backToLife – восстанавливает обратный путь;
* priority - определяет приоритет цепи;
* backTracking - строит соединение

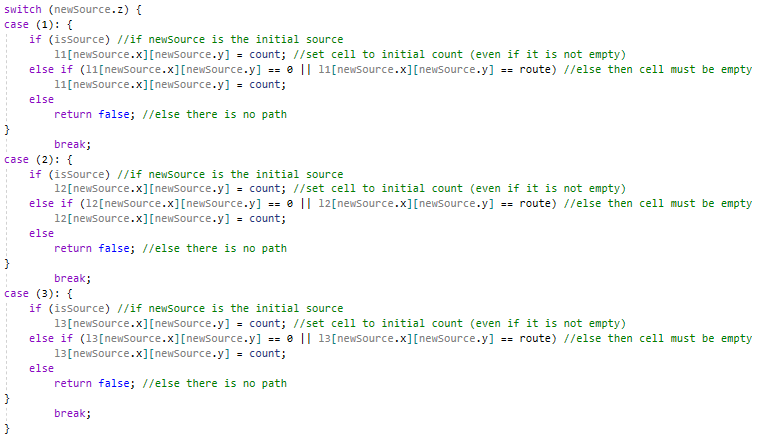
На рисунке 21 представлены объявления данных функций с их аргументами.



*Рис.21. Функции, реализующие алгоритм Соукупа.*

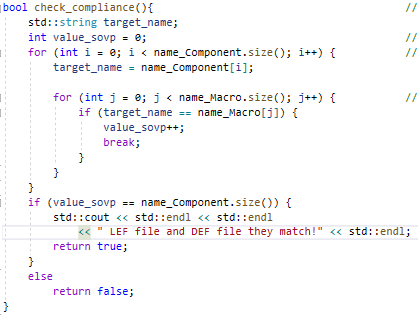
**2.6. Улучшение работы программы.**

Алгоритм Соукупа способен проводить трассировку в одном слое. Но для современных ИС, имеющих миллионы элементов, этого может оказаться недостаточно. Для решения этой проблемы в реализованной программе было добавлено еще два слоя для реализации соединений, как показано на рисунке 22. В первом слое проводятся вертикальные компоненты соединений, а во втором горизонтальные. Третий слой добавлен как резервный. В нем могут находится вертикальные соединения, если потребуется такая необходимость. Каждый слой представляет собой дискретное рабочее поле (ДРП), которое задается в виде двойного массива. Современные серьезные САПР содержат несколько больше слоев металлизации (обычно от 3 до 7). Это обеспечивает 100% трассируемость соединений для схемы с большим количеством элементов, но приводит к увеличению времени выполнения трассировки.



*Рис.22. Трассировка в трех слоях.*

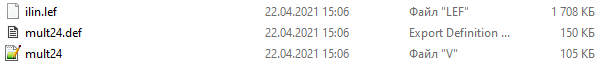
Кроме этого, программа проводит ряд проверок перед началом трассировки. Ключевой из них является проверка на совместимость файлов LEF, DEF и Verilog. Фрагмент проверки на совместимость файлов представлен на рисунке 23. Говоря иначе, все 3 файла должны описывать одну ИС. Проверка происходит путем сравнения количества элементов и их названий во всех файлах. Если один или два файла не пройдут проверку на совместимость, программа сообщит об этом. Если все файлы прошли проверку, значит они описываю одну определенную схему. В таком случае начинается процесс трассировки.



*Рис.23. Проверка файлов на совместимость.*

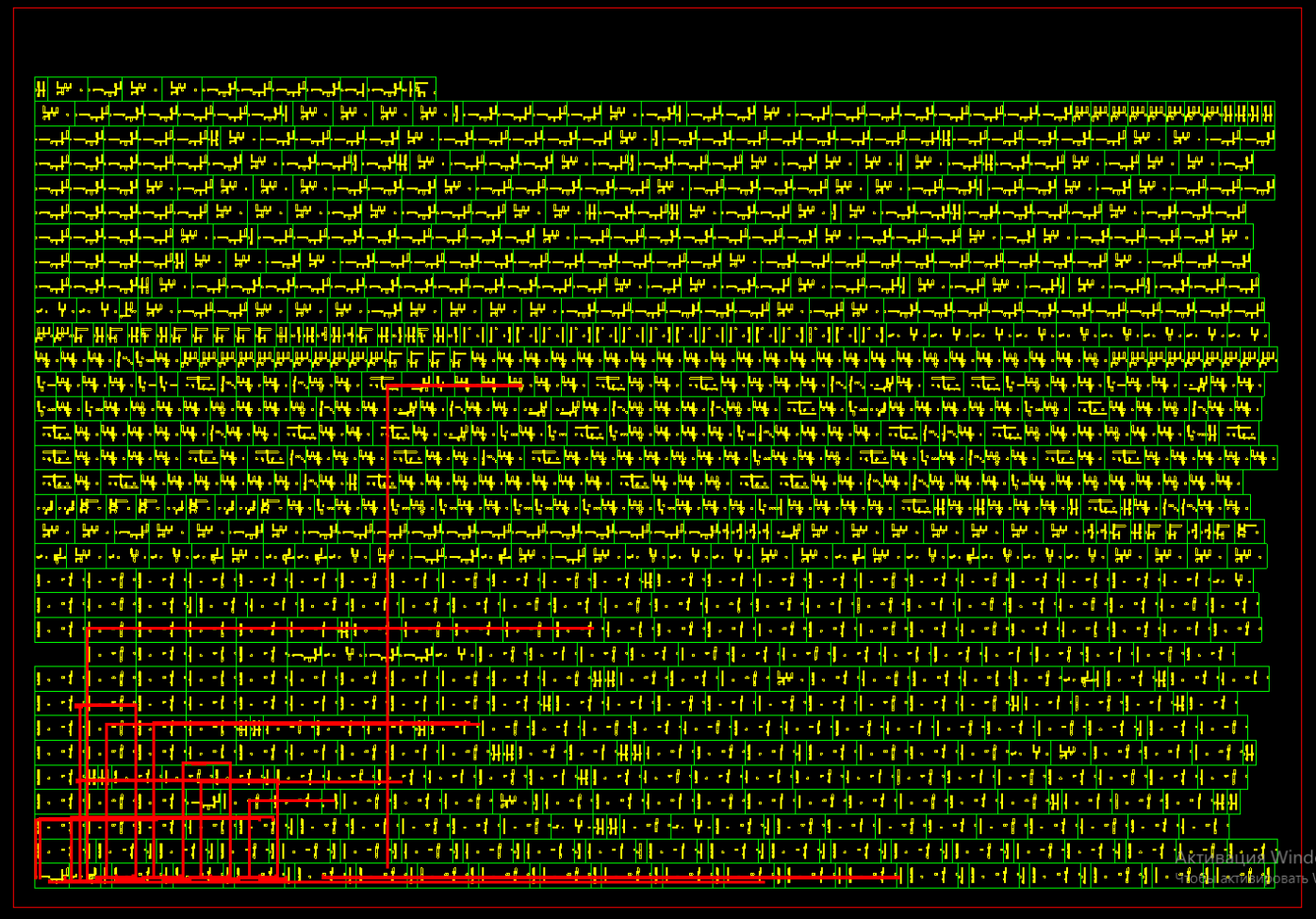
**2.7.** **Трассировка реальной схемы**

Для проверки работоспособности программы была проведена трассировка схемы, рассматриваемой на лабораторных работах по курсу “Топологическое проектирование” и имеющей 1166 элементов. Каждый элемент имеет 1-4 контакта, исключая контакты земли и питания. В качестве входных файлов (форматы LEF, DEF, Verilog) выступают: D\_CELLS\_test.lef, mult24.def, mult24.v. Рисунок 24.



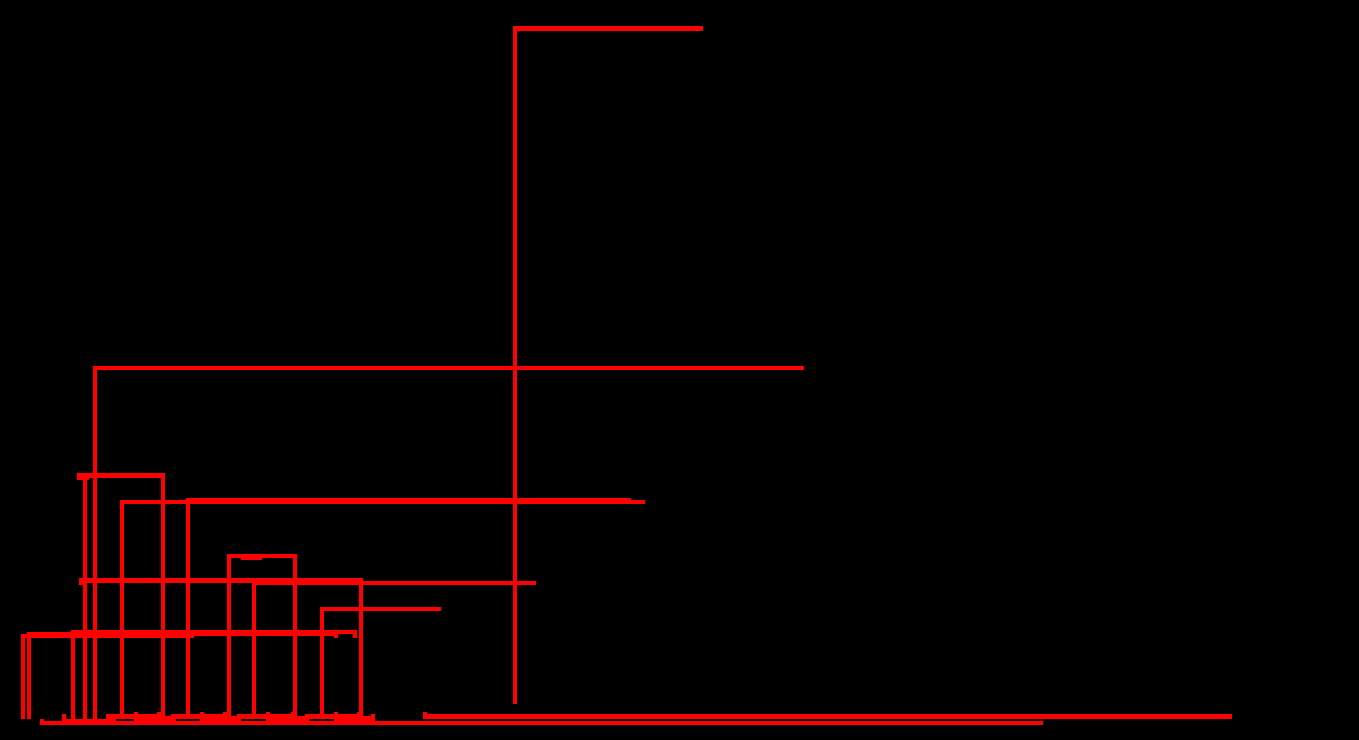
*Рис.24. Входные файлы.*

В виду того, что полная трассировка занимает много времени, а также для наглядности, в качестве результатов приводится трассировка первых 25 цепей ИС. Они показаны на рисунке 25.



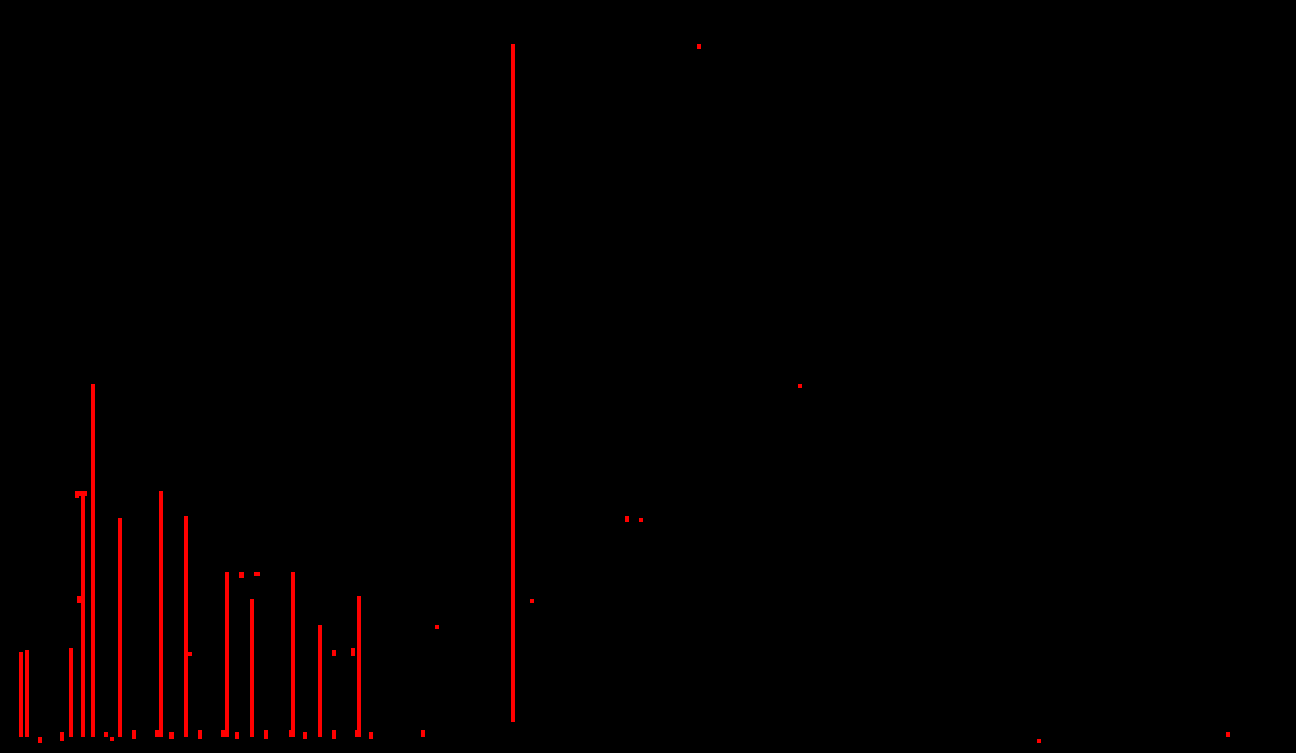
*Рис.25. Демонстрация трассировки.*

Для большей наглядности на рисунке 26 отдельно представлены первые 25 соединений ИС.

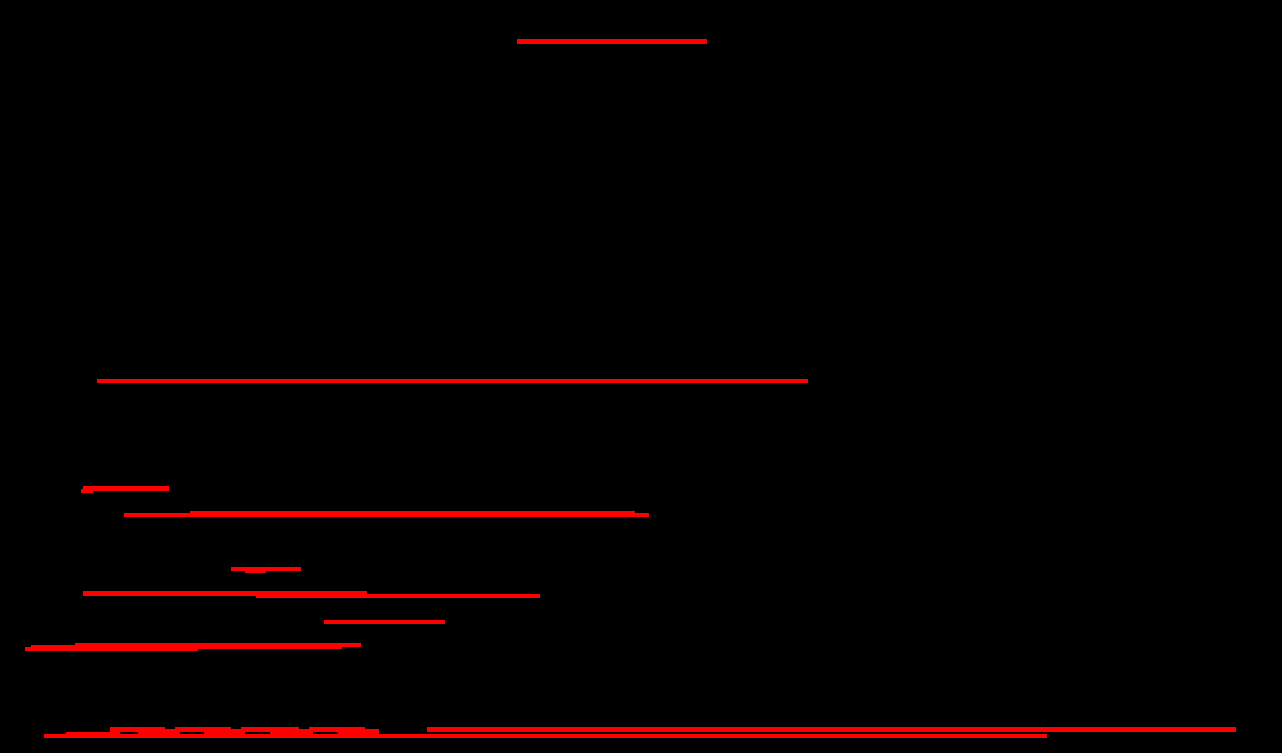


*Рис.26. Первые 25 соединений.*

На рисунке 27 показан первый слой, содержащий вертикальные компоненты соединений, на рисунке 28 - второй слой, содержащий горизонтальные компоненты. В данном демонстрационном примере третий слой не был задействован.



*Рис.27. Слой 1 - вертикальные компоненты соединений.*



*Рис.28. Слой 2 - горизонтальные компоненты соединений.*

Для удобства просмотра топологии и соединений в программу добавлены следующие функции:

* перемещение схемы по вертикали и горизонтали;
* приближение и отдаление;
* отображение/скрытие каждого слоя металлизации.

**Выводы**

В результате выполнения данной выпускной квалификационной работы были получены следующие результаты:

1. Создана программа, реализующая алгоритм быстрой трассировки методом Соукупа в нескольких слоях.
2. Визуализирована топология ИС и построенные соединения.

Также, были углублены знания языка программирования С++, приобретены практические навыки работы с файлами и объектно-ориентированным программированием. Изучены форматы хранения и обработки топологии LEF/DEF. Произведено знакомство с программой визуализации топологии KLayout и библиотекой работы с графикой в высокоуровневых языках программирования - спецификацией OpenGL.

**Списоклитературы**

* Р. Лафоре “Объектно-ориентированное программирование в С++. Классика Computer Science. 4-е изд;
* Sadiq M. Sait и Habib Youssef “VLSI Physical Design Automation: Theory and Practice” 1999 г.;
* Г. Г. Казённов. — М. “Основы проектирования интегральных схем и систем“ 2005 г.;
* Рабаи Ж. М., Чандракасан А., Николич Б. Цифровые интегральные схемы. Методология проектирования. Пер. с англ. М.: ИД «Вильямс», 2007.;
* интернет-ресурс <https://ravesli.com/uroki-cpp/>.
* интернет-ресурс <https://greenlab.di.uminho.pt/wp-content/uploads/2017/09/paperSLE.pdf>;
* интернет-ресурс <https://ru.wikipedia.org/wiki/GDSII>;
* интернет-ресурс <https://www.signoffsemi.com/lef-def-lib/>;
* интернет-ресурс <https://klayout.de/>.
* интернет-ресурс <https://greenlab.di.uminho.pt/wp-content/uploads/2017/09/paperSLE.pdf>;
* интернет-ресурс <https://www.signoffsemi.com/lef-def-lib/>;