# РЕФЕРАТ

КОМБИНАЦИОННАЯ СХЕМА, ЛОГИЧЕСКИЙ СИНТЕЗ, ОПТИМИЗАЦИЯ,ПЕРЕПИСЫВАНИЕ, ФУНКЦИОНАЛЬНЫЙ ЭЛЕМЕНТ

Одним из этапов процесса проектирования цифровой аппаратуры является логический синтез. В рамках логического синтеза представление аппаратуры (логическая схема) оптимизируется по некоторым параметрам, таким как количество функциональных элементов в схеме, глубина и т.д. Перспективным подходом к оптимизации логических схем является переписывание (rewriting).

Объектом исследования являются стратегии оптимизации схем, в частности, трансформации путем переписывания.

Предметом исследования является метод функционального переписывания комбинационных схем, предложенный Аланом Мищенко.

Целью проекта является исследование стратегий переписывания комбинационных схем в инструментах логического синтеза. В рамках данного проекта были решены следующие задачи: 1) сделан обзор существующих методов оптимизации комбинационных схем; 2) предложены новые техники оптимизации на основе наиболее эффективных стратегий переписывания 3) предложенные стратегии реализованы в инструменте логического синтеза; 4) проведен экспериментальный анализ реализованных стратегий.

Результаты выполнения НИР: 1) программная реализация метода переписывания комбинационных схем; 2) реализация нескольких стратегий в алгоритме и их анализ на тестовом наборе схем.

Новизна данной работы заключается в экспериментальном анализе различных стратегий переписывания и выборе среди них оптимальной по заданному критерию. Область применения результатов работы — проектирование и оптимизация цифровых схем. Экономическая значимость работы состоит в снижении затрат на производство цифровой аппаратуры за счет повышения уровня автоматизации процесса логического синтеза.

# Содержание

# Термины и определения

В настоящем отчете о НИР используются следующие термины:

Логическая схема — представление электронной аппаратуры в виде элементов и соединений между ними.

Логический синтез — построение схемы по RTL-модели, представленной на языке описания аппаратуры.

Оптимизация в логическом синтезе — процесс улучшения характеристик логической схемы путем изменения количества входящих в нее элементов и структуры соединений между ними.

Функциональный элемент — элемент логической схемы, который реализует некоторую функцию.

Схема из функциональных элементов — ориентированный ациклический граф, истоки и стоки которого помечены переменными, а остальные вершины являются функциональными элементами.

Комбинационные схемы — логические схемы, для которых значения переменных на стоках (выходах) в любой момент времени однозначно определяются значениями переменных на истоках (входах).

Переписывание — трансформация схемы путем добавления или удаления элементов и соединений между ними таким образом, что ее измененное логическое представление будет функционально эквивалентно первоначальному.

Разрез схемы для узла $n$ — набор узлов схемы, называемых *листьями*, таких, что каждый путь от входов до $n$ проходит по меньшей мере через один *лист*.

Тривиальный разрез узла — это разрез, содержащий сам узел и только его.

k-выполнимый разрез — разрез, количество узлов в котором не превышает k.

ДНФ (дизъюнктивная нормальная форма) — форма записи булевой функции, представляющая ее как дизъюнкцию конъюнкций переменных и их отрицаний.

Терм — конъюнкция переменных и их отрицаний (в ДНФ).

# Введение

Логический синтез является одним из ключевых этапов проектирования цифровой микроэлектронной аппаратуры. На этом этапе происходит преобразование описания аппаратуры в логическую схему. Учитывая растущие требования к производительности и энергоэффективности цифровых устройств, необходимо проводить оптимизацию логических схем. Для улучшения характеристик логических схем (таких, как количество узлов, глубина и пр.) применяются методы оптимизации. Одним из эффективных методов оптимизации является переписывание. В данной работе проводится исследование различных стратегий переписывания комбинационных схем с целью оценки их эффективности и повышения уровня оптимизации алгоритма переписывания в логическом синтезе.

С течением времени развивались и совершенствовались различные методы оптимизации логических схем:

1. Метод Куайна-Мак-Класки [1] является одним из первых точных методов оптимизации (опубликован в 1956 г.), основанных на получении совершенной ДНФ, то есть ДНФ, которая содержит минимальное количество термов. Алгоритм основан на построении булева куба – трехмерной таблицы истинности, которая содержит все возможные значения для набора переменных в логической функции. Каждое измерение куба представляет собой значение одной из переменных функции. Каждой комбинации значений переменных соответствует одна ячейка, содержащая значение функции при этих значениях переменных.

В худшем случае, сложность метода оптимизации схемы таким способом равна экспоненциальной от числа переменных в булевой функции, что требует много времени для выполнения.

Кроме того, так как алгоритм является точным, он пытается найти минимальное решение, что также увеличивает время работы.

Еще одним недостатком является то, что решения, полученные с помощью метода получения совершенной ДНФ, не всегда являются оптимальными.

2. Методы MINI [2] и ESPRESSO [3] (опубликованы в 1981 и 1982 гг. соответственно) являются примерами эвристических методов минимизации ДНФ. Они отличаются от метода Куайна-Мак-Класки тем, что не пытаются найти точное минимальное решение, а применяют различные эвристические алгоритмы для получения более быстрого и приближенного решения. Такие методы позволяют обрабатывать графы большего размера, но все еще наследуют недостатки предыдущего подхода – решения могут быть неоптимальными. Однако эвристические методы являются более практичными и широко используются в инструментах проектирования цифровой микроэлектроники. Методы реализованы в инструментах SIS и MVSIS.

3. Оптимизация схем с помощью переписывания:

3.1. Впервые подобный метод был реализован в инструменте SOCRATES [4]. Подход основан на структурном сопоставлении подсхем, где для оптимизации исходной схемы производился поиск конкретных подграфов из заданной технологической библиотеки в оптимизируемой схеме. Библиотека была заранее определена. Для каждой определенной подсхемы в ней существовал набор вариантов замены, которые поочередно подставлялись для замера определенных характеристик в оптимизируемой схеме. Затем выбиралось наиболее оптимальное преобразование, и это преобразование применялось к рассматриваемому участку в схеме.

3.2. Переписывание с функциональным сопоставлением подсхем — это следующий шаг в эволюции методов оптимизации. В отличие от предыдущих методов, здесь используется не только структурное, но и функциональное сопоставление подсхем. Этот метод позволяет искать оптимальные замены подграфов не только на основе их структуры, но и на основе реализуемой конкретным узлом функции. Для каждого узла производится построение таблицы истинности, которую потом используют в

качестве ключа, чтобы в базе найти варианты замены участка, для которого таблица истинности и была построена. В функциональном переписывании могут использоваться следующие подходы:

3.2.1. **Двухуровневое переписывание** [5]. Оно заключается в том, что таблица истинности, по которой потом производится сопоставление, строится, основываясь на наборе узлов, которые удалены максимум на 2 узла по связности от рассматриваемого узла.

3.2.2. **Переписывание, основанное на разрезах** [6]. Для каждой вершины схемы существует разрез — набор вершин, полностью определяющий значение рассматриваемой вершины. Другими словами, любой путь от входов к этой вершине проходит через какую-то вершину из этого разреза. В данном подходе построение таблицы истинности для каждого узла производится именно на основе таких разрезов.

Этот метод расширяет спектр возможностей замены при переписывании и позволяет находить более оптимальные варианты для оптимизации, в отличие от предыдущего подхода.

Еще одним немаловажным принципом при переписывании является контекстная зависимость [6]. Переписывание с учетом данного принципа предполагает собой повторное использование узлов, уже реализующих требуемую функцию в оптимизируемой схеме.

**Актуальность** темы обусловлена тем, что при проектировании современных электронных систем необходимо учитывать жесткие ограничения на быстродействие и энергопотребление, что влечет за собой необходимость оптимизации логических схем, в том числе комбинационных. В настоящее время существуют различные методы оптимизации комбинационных схем, однако

эффективность их работы может быть улучшена за счет разработки новых стратегий переписывания.

# 1. Особенности реализации

## 1. Представление

В статье [6] используется представление AIG (And-Inverter Graph). AIG – граф, состоящий из узлов, представляющих логические операции И и НЕ, а также входов и выходов.

Преимуществом AIG является его простота: каждый узел с логической операцией И в AIG имеет ровно два входа и один выход. В некоторых реализациях AIG-представления используются пометки на дугах, которые выражают логический элемент НЕ. Тогда все узлы AIG-представления реализуют логическую операцию И.

В более поздних статьях [7] появляются варианты использования представления XAG (Xor-And Graph). XAG — граф, узлы которого реализуют логические операции И и исключающее ИЛИ.

В рамках данного проекта алгоритм переписывания будет вестись на графе, реализующем все логические операции. Это позволит эффективно оптимизировать и улучшить работу схемы, а также расширить возможности подхода переписывания.

# 2. Этапы реализации и стратегии

Основные шаги алгоритма переписывания:

1. Поиск разрезов;
2. Перебор узлов и их разрезов;
3. Представление узла таблицей истинности разреза;
4. Взятие необходимого списка подсхем из заранее приготовленной библиотеки;
5. Оценка характеристик при удалении и добавлении полученных подсхем;
6. Замена подсхем в схеме.

## 1. Поиск разрезов

Напомним, разрез схемы для узла $n$ — набор узлов схемы таких, что каждый путь от входов до $n$ проходит через какой-то узел из разреза. Вершина может является разрезом для самой себя и тогда разрез называется *тривиальным*. Таким образом, можно итеративно организовать поиск разрезов для всех вершин. Описание следующего подхода для поиска взято из статьи [6]:

Вычисление разреза начинается с входов логической схемы и происходит в топологическом порядке к выходам. Для входа множество разрезов содержит только тривиальный разрез. Для внутренней вершины $n$ с двумя входами $a$ и $b$, разрезы $\Phi(n)$ вычисляются путем объединения разрезов $a$ и $b$.

$$\Phi(n) = \{\{n\}\} \cup \{u \cup v \mid u \in \Phi(a), u \in \Phi(b), |u \cup v| \leq 4\}.$$

В статье [6] количество узлов в разрезе берется равным четырем. В более поздних статьях этого же автора это количество увеличивается. Количество узлов задается на начальном этапе алгоритма и не меняется.

Одной из стратегий является варьирование количества узлов в разрезе. Предположительно, при увеличении количества узлов будет улучшаться результат переписывания. Однако тогда возникает проблема комбинаторного взрыва числа разрезов при их нахождении, а также хранения большого количества больших разрезов.

Чтобы нахождение разрезов производилось за приемлемое время, можно поставить ограничение на максимальное количество разрезов для узла, то есть после нахождения *n* разрезов их поиск для данного узла прекращается и алгоритм переходит к следующему узлу.

## 2. Перебор узлов и их разрезов

Перебор узлов происходит в топологическом порядке от входов к выходам. Для каждого из узлов перебираются найденные ранее разрезы. Для каждой такой пары строится таблица истинности и получается набор подграфов из технологической базы, после чего происходит их подстановка.

**Стратегии при переборе:**

1. Алгоритм пробует не все разрезы для узла, а только *n* первых, причем возможно задать для них систему приоритетности, что позволит нам тратить время только на более «важные» узлы. В качестве быстрого решения можно просто брать несколько случайных, либо итерироваться по списку разрезов, пока не встретится один подходящий, после чего приступить к анализу другого узла.

2. Количество раз, которое мы проходимся по схеме - каждому узлу и набору его разрезов - также зависит от поставленного нами условия. Например, можно напрямую задавать количество повторов цикла, либо установить порог оптимизации по ключевой характеристике, после достижения которого переписывание завершается.

## 3. Представление узла таблицей истинности разреза

По разрезу производится построение таблицы истинности (ТИ). В данной работе количество узлов в разрезе не превышает 6, поэтому ТИ хранится в числе типа *unit64_t*.

Стоит упомянуть следующие особенности:

- Если количество узлов в разрезе меньше 6, алгоритм все равно строит таблицу для 6 переменных; только добавляет фиктивные узлы, чтобы в сумме получилось 6. “Искусственно” добавленные узлы не влияют на результирующие значение узла, для которого строится ТИ.
- Точно так же и база может выдать подсхему с количеством входов, не совпадающим с количеством узлов в разрезе, для которых была построена ТИ.
    - Если в выданной подсхеме больше входов, алгоритм может либо приписать новые входы существующим входам схемы, либо создать новые.
    - Если в выданной подсхеме входов меньше, алгоритм сопоставляет имеющиеся входы с нужными вершинами в разрезе, а с остальными обрывает связь, т.к. согласно ТИ и базе было выявлено, что те узлы в разрезе на рассматриваемый узел функционально не влияют.

## 4. Взятие необходимого списка подсхем из заранее приготовленной библиотеки

Для получения вариантов замены на переписывание алгоритм обращается к базе, в которой по построенной ТИ получает перечень подсхем, реализующих эту функцию. После чего производится поочередная подстановка всех полученных подсхем.

База — библиотека с готовыми схемами, построенными заранее. Для данной работы было решено взять схемы из библиотеки ABC [8].

## 5. Оценка характеристик при удалении и добавлении полученных подсхем

После получения списка подсхем для рассматриваемого узла и разреза происходит подсчет характеристики оптимизации. В данной работе в качестве целевой характеристики рассматривается количество узлов в AIG-представлении.

**Процесс оценки замены:**

Прежде, чем выполнять замену на оптимизируемой схеме, следует сначала подсчитать, на сколько уменьшится количество узлов после замены.

Для этого, начиная с входов подсхемы-замены, идет сопоставление ее узлов с узлами заменяемого участка в основной схеме. Проходя так в топологическом порядке, начинается учитываться, сколько узлов создается, а сколько узлов используется повторно за счет контекстной зависимости. Обход завершается по достижении выходной вершины при обходе в подсхеме-замене. После чего мы начинаем подсчитывать, сколько узлов удалится из сети, когда старые связи рассматриваемого узла, для которого строилась ТИ, сменятся на новые.

Стоит отметить, что новые связи могут и не появиться, если, например, подсхема-замена полностью совпадает с рассматриваемым участком. Также могут полностью совпасть некоторые пути от входов к выходам в подсхеме-замене и тогда сохранится часть старых и появится часть новых связей.

В процессе анализа имеющихся входящих дуг у рассматриваемого узла в оптимизируемой сети, происходит обработка узлов, с которыми будут удалены связи и если входящих дуг кроме той, которую мы собираемся удалить, у них не нет, то такую вершину мы считаем помечается как удаленная. Входы такой вершины добавляются в очередь, чтобы произвести аналогичную проверку.

После проведения всех подстановок количество узлов в схеме может измениться. Это количество в данной работе является критерием оптимизации, который определяет решение о замене с использованием подсхемы.

Относительно вычисления критерия оптимизации и принятия решения о совершении преобразования можно выработать несколько стратегий:

1. Принятие или непринятие замен нулевой стоимости (англ. zero-cost replacements). Если количество узлов в схеме после замены не изменилось, можно все равно производить замену. Такая стратегия может привести к улучшению последующего переписывания.
2. Выбор одной наилучшей замены из возможных для данного узла на текущей итерации или мгновенная замена для всех оптимальных вариантов.

В работе были реализованы обе стратегии.

**Стратегия 1**

Хранит последнюю лучшую предложенную замену и после окончания итерации по списку подсхем из базы и после завершения итерации по этому списку производит преобразование с наилучшей из них. Здесь также можно расширять период «выжидания» лучшего варианта и, например, производить замену пройдясь сразу по всем разрезам и попробовав для каждого из них все предложенные подсхемы.

В перспективе планируется реализовать многоуровневый подсчет критерия оптимизации, при котором хранятся результаты применения нескольких вариантов подсхем для определенного участка, а далее идет

разветвление: для каждого из сценариев находятся, применяются и сохраняются новые варианты оптимальных для данной ситуации замен. Такой «отход» от жадного алгоритма позволит находить при переписывании более выгодные варианты.

**Стратегия 2**

Замена производится сразу же, если она улучшает критерий количества узлов в схеме. В рамках данной стратегии возможны следующие сценарии:

1. Переход после успешной замены к следующему разрезу текущего узла или следующему узлу.
2. Продолжение поиска новых схем более оптимальных замен из базы для данного разреза на схеме с произведенной заменой.

## 6. Замена подсхем в схеме

Использование контекстной зависимости на этом этапе позволяет избежать создание новых узлов за счёт использования имеющихся.

Процесс замены подсхем производится следующим образом:

1. Проход по узлам схемы в топологическом порядке от входов к выходу;
2. При сопоставлении узлов основной схемы с узлами из базы, происходит либо переиспользование существующего узла, либо создание нового узла.
3. Обновление входов рассматриваемого узла на соответствующие входы в схеме из базы.

4. Проход в обратном топологическом порядке начиная с рассматриваемого узла оптимизируемой схемы с целью удаления узлов, которые остались без выходов.

# 3. Выбранные стратегии и полученные результаты

В итоге проведённого исследования алгоритма были выбраны и реализованы следующие стратегии:

1. Принятие или непринятие замен нулевой стоимости (англ. zero-cost replacements).
2. Моментальное принятие замены и полный перебор в рамках списка подсхем из базы и выбор наилучшей замены на преобразование.
3. Переход к следующему разрезу после произведения замены для данного узла.

Ниже приведены результаты выполнения алгоритма с применения каждой из этих стратегий на тестовом наборе схем:

По горизонтали сверху располагаются названия стратегий. По вертикали слева перечислен тестовый набор схем. В ячейке располагаются значения характеристики оптимизации (количества узлов) для заданной стратегии и для заданной схемы.

| Стратегия zero-cost replacements | Принятие замен нулевой стоимости | | | Непринятие замен нулевой стоимости | | |
|---|---|---|---|---|---|---|
| Стратегия\ Схема | Моментальная замена | Моментальная замена и переход к следующему узлу | Выбор наилучшей | Моментальная замена | Моментальная замена и переход к следующем узлу | Выбор наилучшей |
| Adder | 285 | 269 | 515 | 284 | 176 | 445 |
| Max | 667 | 559 | 1219 | 615 | 421 | 1119 |

| Стратегия zero-cost replacements | Принятие замен нулевой стоимости | | | Непринятие замен нулевой стоимости | | |
|---|---|---|---|---|---|---|
| Стратегия\ Схема | Моментальная замена | Моментальная замена и переход к следующему узлу | Выбор наилучшей | Моментальная замена | Моментальная замена и переход к следующем узлу | Выбор наилучшей |
| Int2float | 88 | 79 | 95 | 88 | 81 | 93 |
| I2c | 299 | 265 | 550 | 289 | 201 | 501 |
| C432 | 1 | 1 | 1 | 1 | 1 | 1 |
| C880 | 21 | 18 | 25 | 22 | 16 | 25 |

Исходя из представленных результатов, можно сделать вывод, что наилучшей стратегией является "Выбор наилучшей с принятием нулевой стоимости", так как она показала наибольшее значение характеристики оптимизации. Варианты сочетания моментальной замены с непринятием замен нулевой стоимости показали менее эффективные результаты, так как все они имеют значения характеристики оптимизации менее из аналогов.

# Заключение

В рамках данной курсовой работы были рассмотрены вопросы оптимизации комбинационных схем при проектировании цифровой аппаратуры. Был проведен обзор существующих методов оптимизации, предложены новые техники на основе переписывания с использованием стратегий оптимизации и реализованы в инструменте логического синтеза.

Экспериментальный анализ показал, что стратегия "Выбор наилучшей замены с принятием нулевой стоимости" является наиболее эффективной. Остальные варианты сочетания с непринятием замен нулевой стоимости имеют менее высокие значения характеристики оптимизации.

Результаты работы могут быть использованы в области проектирования и оптимизации цифровых схем, что приведет к снижению затрат на производство цифровой аппаратуры и повышению уровня автоматизации процесса логического синтеза.

В целом, данная работа является важным шагом в развитии методов оптимизации комбинационных схем и может стать основой для дальнейших исследований в этой области.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] McCluskey E. J. Minimization of Boolean functions // Bell Systems Technology Journal. — Vol. 35, No. 6. — 1956. — P. 1417-1444.

[2] Quinlan R. MINI: A Heuristic Approach for Logic Minimization // IEEE Transactions on Computers. — Vol. C-30, No. 5. —1981. — P. 318-325.

[3] Brayton R., Hachtel G., Hemachandra L., Newton R., Sangiovanni-Vincentelli A. A comparison of logic minimization strategies using ESPRESSO: an APL program package for partitioned logic simulation // Proceedings of the IEEE International Symposium on Circuits and Systems. — 1982. — P. 42–48.

[4] Gregory D., de Geus A., Bartlett K., Hachtel G. SOCRATES: A System for Automatically Synthesizing and Optimizing Combinational Logic // ACM/IEEE Design Automation Conference. — 1986. — P. 79-85.

[5] Bjesse P., Boralv A. DAG-aware circuit compression for formal verification // International Conference on Computer Aided Design. — 2004. — P. 42-49.

[6] Mishchenko A., Chatterjee S., Brayton R. DAG-aware AIG rewriting: a fresh look at combinational logic synthesis // ACM/IEEE Design Automation Conference. —2006. — P. 532-535.

[7] Riener H., Lee S.-Y., Mishchenko A., De Micheli G. Boolean rewriting strikes back: Reconvergence-driven windowing meets resynthesis // ASP-DAC Conference. — 2022.

[8] ABC: A System for Sequential Synthesis and Verification [Электронный ресурс]. URL: https://people.eecs.berkeley.edu/~alanmi/abc (дата обращения: 04.04.2023).