

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное
образовательное учреждение высшего образования
ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

на правах рукописи

Советов Станислав Игоревич

**ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ ПЛИС FPGA, РЕАЛИЗУЮЩИЕ
НЕСКОЛЬКО ФУНКЦИЙ ОДНОВРЕМЕННО**

2.3.2. Вычислительные системы и их элементы

ДИССЕРТАЦИЯ

на соискание ученой степени
кандидата технических наук

Научный руководитель: доктор
технических наук, профессор,
Тюрин С. Ф.

Пермь – 2024

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.....	6
ГЛАВА 1. ИССЛЕДОВАНИЕ СУЩЕСТВУЮЩИХ МОДЕЛЕЙ И МЕТОДОВ РЕАЛИЗАЦИИ ЛОГИЧЕСКИХ ФУНКЦИЙ В ПЛИС. ПОСТАНОВКА НАУЧНОЙ ЗАДАЧИ.....	13
1.1. Анализ объекта исследования – программируемых логических интегральных схем (ПЛИС).....	13
1.2. Анализ предмета исследования – научно–методического аппарата синтеза логических элементов ПЛИС.....	21
1.3. Математическая постановка научной задачи и частных задач исследования.....	26
1.4. Выводы по главе 1	29
ГЛАВА 2. РАЗРАБОТКА МНОГОФУНКЦИОНАЛЬНОГО ЛОГИЧЕСКОГО ЭЛЕМЕНТА n – LUT, РЕАЛИЗУЮЩЕГО ОДНОВРЕМЕННО 2^v ФУНКЦИЙ ОТ ОДНИХ И ТЕХ ЖЕ n ПЕРЕМЕННЫХ	30
2.1. Разработка модели реализации в одном n – LUT одновременно 2^v функций от одних и тех же переменных	30
2.2. Разработка метода реализации в одном n – LUT одновременно 2^v функций от одних и тех же переменных	34
2.3. Разработка схемы электрической функциональной n –LUT, реализующего одновременно 2^v функций от одних и тех же переменных... ..	40
2.4. Алгоритм подключения дополнительных транзисторов для реализации многофункционального элемента n –LUT.....	43
2.5. Пример применения метода для разработки 3–LUT, реализующего четыре функции одновременно	48
2.6. Пример применения метода для разработки 4–LUT, реализующего две функции одновременно	51
2.7. Выводы по главе 2	53
ГЛАВА 3. РАЗРАБОТКА ЛОГИЧЕСКОГО ЭЛЕМЕНТА N – LUT, РЕАЛИЗУЮЩЕГО ЛОГИЧЕСКУЮ ФУНКЦИЮ И ДЕШИФРАЦИЮ НАБОРА ПЕРЕМЕННЫХ.....	55
3.1. Разработка моделей n – LUT, реализующих логическую функцию и дешифрацию набора переменных	55
3.1.1. Существующий элемент LUT с n уровневим деревом транзисторов. ..	55
3.1.2. Разработка модели адаптивного элемента LUT с одноуровневым 2^n деревом (2^n ветвей по n транзисторов).....	56

3.1.3. Разработка модели LUT, выполняющего дешифрацию набора переменных одновременно с использованием неактивной половины дерева транзисторов	61
3.1.4. Разработка модели LUT, вычисляющего значение заданной логической функции и дешифрацию набора переменных одновременно без использования неактивной половины дерева транзисторов	64
3.2. Разработка электрических функциональных схем элементов LUT, вычисляющих значение заданной логической функции и дешифрацию набора переменных	67
3.2.1. Электрическая функциональная схема адаптивного элемента LUT с одноуровневым 2^n деревом	67
3.2.2. Электрическая функциональная схема LUT, вычисляющая значение заданной логической функции и дешифрацию набора переменных одновременно с использованием неактивной половины дерева транзисторов	69
3.2.3. Электрическая функциональная схема LUT вычисляющего значение заданной логической функции и дешифрацию набора переменных одновременно без использования неактивной половины дерева транзисторов	74
3.3. Методы синтеза предложенных моделей	76
3.4. Выводы по главе 3	78
ГЛАВА 4. МОДЕЛИРОВАНИЕ РАЗРАБОТАННЫХ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ ПЛИС	80
4.1. Схемотехническое моделирование элемента n – LUT, реализующего одновременно 2^v функций от одних и тех же переменных.....	80
4.1.1. Статическое моделирование n – LUT, реализующего одновременно 2^v функций от одних и тех же переменных	80
4.1.2. Динамическое моделирование n – LUT, реализующего одновременно 2^v функций от одних и тех же переменных	88
4.1.3. Топологическое моделирование n – LUT, реализующего одновременно 2^v функций от одних и тех же переменных.....	94
4.2. Схемотехническое моделирование элементов LUT, вычисляющих значение заданной логической функции и дешифрацию набора переменных	102
4.2.1. Моделирование адаптивного элемента LUT с одноуровневым 2^n деревом.....	102

4.2.2. Статическое моделирование логического элемента, вычисляющего значение заданной логической функции и дешифрацию набора переменных одновременно с использованием неактивной половины дерева транзисторов	104
4.2.3. Статическое моделирование логического элемента выполняющего дешифрацию набора переменных одновременно без использования неактивной половины дерева транзисторов	106
4.2.4. Динамическое моделирование логического элемента, вычисляющего значение заданной логической функции и дешифрацию набора переменных одновременно с использованием неактивной половины дерева транзисторов	110
4.2.5. Динамическое моделирование логического элемента выполняющего дешифрацию набора переменных одновременно без использования неактивной половины дерева транзисторов	112
4.2.6. Топологическое моделирование элементов LUT, вычисляющих значение заданной логической функции и дешифрацию набора переменных одновременно без использования неактивной половины дерева транзисторов	117
4.3. Выводы по главе 4	121
ГЛАВА 5. СРАВНИТЕЛЬНЫЕ ОЦЕНКИ ЭФФЕКТИВНОСТИ РАЗРАБОТАННЫХ МЕТОДОВ	122
5.1. Оценка эффективности предложенного метода реализации нескольких логических функций одновременно по количеству транзисторов и площади, занимаемой на кристалле	122
5.2. Оценка эффективности предлагаемого метода реализации дешифрации набора переменных с использованием неактивной половины дерева транзисторов элемента LUT	126
5.3. Оценка эффективности предлагаемого метода реализации дешифрации набора переменных одновременно с вычислением логической функции	128
5.4. Оптимизация по Парето блоков логических элементов	131
5.4.1 Оптимизация по Парето блоков логических элементов из известных и предложенных многофункциональных элементов	131
5.4.2 Оптимизация по Парето блоков логических элементов, обеспечивающего одновременное вычисление логической функции и дешифрацию набора переменных без использования неактивной половины дерева транзисторов с известными элементами и предложенным многофункциональным элементом	136
5.5. Выводы по главе 5	144
ЗАКЛЮЧЕНИЕ	145

Список сокращений	147
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ	149
Приложение А	160
Приложение Б	168
Дополнительные результаты топологического моделирования в системе Microwind	168
Приложение В	188
Акты внедрения результатов диссертационного исследования	188

ВВЕДЕНИЕ

Актуальность темы исследования. В настоящее время наблюдается резкий рост использования программируемой логики в цифровой аппаратуре [64,66] и вычислительных системах [4,21,56,60,84,100]. Программируемые логические интегральные схемы (ПЛИС) реализуются по разным технологиям, например существует программируемые логические матрицы (Programmable Logic Array – PLA) [18,53], в которых логические функции вычисляются в дизъюнктивной нормальной форме (ДНФ). Другой тип программируемых интегральных схем, содержащих блоки энергонезависимой памяти являются CPLD (Complex Programmable Logic Device) [40] в которых логические функции также вычисляются в ДНФ. Наиболее востребованными ПЛИС являются FPGA [57,61] (Field-Programmable Gate Array) поскольку в них наилучшим образом реализуется гибкость, высокая скорость и параллельность вычислений. Вычисление логических функций в FPGA реализуется в совершенной дизъюнктивной нормальной форме (СДНФ). В современных ПЛИС для вычисления логических функций используется базовый логический элемент, а именно LUT (Look Up Table) [68]. Количество логических элементов LUT в современных ПЛИС постоянно растет и на данный момент составляет порядка десятка миллионов [14,67,87]. При этом растет и количество используемых переменных в одном LUT, в начале это было три – четыре, теперь шесть – семь, причем наблюдаются тенденции к их дальнейшему росту. Так, в некоторых ПЛИС реализуются пока не все возможные логические функции уже и восьми переменных. Однако, с другой стороны, существующие функциональные возможности используются не в полной мере, один LUT реализует при данной конфигурации только одну логическую функцию.

Анализ показал, что современные методы синтеза позволяют получить новые логические элементы с большим количеством входов, однако, не полностью учитывают функциональные возможности имеющихся логических элементов. Для ряда задач в том числе критического применения [89] (в бортовых цифровых

вычислительных комплексах и в военной технике) имеющихся возможностей все равно недостаточно.

Таким образом, актуальным является проведение исследований по разработке моделей и методов синтеза логических элементов ПЛИС FPGA, обеспечивающих повышение их функциональных возможностей, особенно в связи с необходимостью разработки отечественной электронной элементной базы.

Степень разработанности темы исследования. Вопросы синтеза логических элементов, в том числе универсальных, которые настраиваются на реализацию заданной функции, рассматривались ранее в работах отечественных авторов Евреинова Э.В. [10,11,51], Косарева Ю.Г. [10], Прангишвили И.В. [11], Балашова Е.П. [2,3], Пузанкова Д.В. [2,7], Викентьева Л.Ф. [26], Аляева Ю.А. [26], Шальто А.А. [26], Несмелова В.А. и др. еще до разработки первых LUT FPGA.

Логические элементы LUT FPGA исследовались в работах Строгонова А.В. [1,36–38], Цыбина М.В. [38], однако, вопросы реализации нескольких функций при одной настройке рассмотрены не были. В работах Денисова А.Н. [9] рассматриваются логические элементы базовых матричных кристаллов (БМК), настраиваемых только в заводских условиях.

В работах зарубежных авторов Берски Д. [5], Shubham R. [53], Monther A. [81], Yervant Z. [99], Mehta N. [80,82], Харченко В.С. [27,75], Дрозда А.В. [62,69], Хаханова В.И. [50], Zhong Chen [101], Liang Kong [76], Tomoaki Sato [55], Jason Cong [59] вопросы синтеза многофункциональных элементов LUT не рассматривались. В работах Alireza Kaviani [74], Stephen Brown [74], Chi Wai, Yu [58] предлагалась гибридная ПЛИС, комбинирующая ПЛИМ CPLD и LUT FPGA. Задачи повышения надежности логических элементов решались в работах Городилова А.Ю. [8], Каменских А.Н. [15]. Создание самосинхронных логических элементов ПЛИС исследовались в работах Скорняковой А.Ю. [88]. Однако, задача реализации нескольких функций одновременно в одном LUT в этих работах не рассматривалась. Синтез многофункциональных элементов LUT в работах Прохорова А.С. [24,45] рассмотрен не в полной мере. Реализация дешифрации набора переменных для реализации систем функций (DC LUT – работы Вихорева Р.В.,

Тюрина С.Ф. [6,95]) характеризуется высокой сложностью, поэтому целесообразно ее уменьшить.

Объектом исследования является логический элемент LUT (Look Up Table) ПЛИС FPGA.

Предметом исследования является научно–методический аппарат синтеза логических элементов в ПЛИС.

Цель исследования заключается в решении задачи разработки научно–методического аппарата синтеза логических элементов LUT, в которых одновременно вычисляется несколько логических функций.

Декомпозиция научной задачи позволяет выделить следующие частные **задачи исследования**:

1. Аналитический обзор, анализ, сравнение существующих моделей, методов и алгоритмов синтеза логических элементов ПЛИС.
2. Разработка моделей, реализующих вычисление нескольких функций одновременно и дешифрацию набора переменных вместе с вычислением основной функции.
3. Разработка метода синтеза многофункционального логического элемента, реализующего одновременное вычисление 2^v , $v = 1, 2, 3, \dots, n-1$ логических функций.
4. Разработка метода синтеза логического элемента, реализующего вычисление основной логической функции одновременно с дешифрацией набора переменных.
5. Разработка алгоритмов подключения дополнительных транзисторов в логических элементах LUT, реализующего вычисление нескольких функций одновременно и дешифрацию набора переменных вместе с вычислением основной функции.
6. Получение сравнительных оценок сложности по показателям количества транзисторов, площади, занимаемой на кристалле, потребляемой мощности и временной задержке предлагаемых и известных логических элементов LUT.

7. Апробация и внедрение разработанных моделей, методов и алгоритмов, предлагаемых логических элементов, реализующих вычисление нескольких функций одновременно и дешифрацию набора переменных вместе с вычислением основной функции в ФИЦ ИУ РАН и учебном процессе ФГАОУ ВО «ПНИПУ».

Новые научные результаты и положения, выносимые на защиту:

1. *Модели* логических элементов LUT, отличающиеся тем, что обеспечивается одновременное вычисление несколько функций от одних и тех же переменных, а также вычисление основной логической функции совместно с дешифрацией набора переменных. Это позволяет увеличить количество реализуемых логических функций ПЛИС при одной и той же площади кристалла при уменьшении количества транзисторов (п. 2 «Разработка принципиально новых методов анализа и синтеза вычислительных систем и их элементов, с целью улучшения технических характеристик, включая новые процессорные элементы, сложно–функциональные блоки, системы и сети на кристалле, квантовые компьютеры» паспорта научной специальности 2.3.2).

2. *Метод* синтеза многофункционального логического элемента LUT ПЛИС FPGA, который отличается от существующих тем, что синтезируется логический элемент, в котором одновременно вычисляется 2^v , $v = 1, 2, 3, \dots, n-1$ логических функций, что приводит к снижению аппаратных затрат от 15%. (п. 2 «Разработка принципиально новых методов анализа и синтеза вычислительных систем и их элементов, с целью улучшения технических характеристик, включая новые процессорные элементы, сложно–функциональные блоки, системы и сети на кристалле, квантовые компьютеры» паспорта научной специальности 2.3.2).

3. *Метод* синтеза логического элемента LUT ПЛИС FPGA, который отличается от существующего тем, что синтезируется логический элемент, выполняющий одновременное вычисление логической функции и дешифрацию набора переменных, что приводит к снижению аппаратных затрат от 15%. (п. 2 «Разработка принципиально новых методов анализа и синтеза вычислительных систем и их элементов, с целью улучшения технических характеристик, включая

новые процессорные элементы, сложно–функциональные блоки, системы и сети на кристалле, квантовые компьютеры» паспорта научной специальности 2.3.2).

4. *Алгоритмы* подключения дополнительных транзисторов в многофункциональном логическом элементе LUT, реализующего вычисление нескольких функций одновременно, и подключения дополнительных транзисторов, реализующие дешифрацию входного набора, отличающиеся тем, что позволяет синтезировать требуемый многофункциональный логический элемент и логический элемент с дешифрацией входного набора (п. 2 «Разработка принципиально новых методов анализа и синтеза вычислительных систем и их элементов, с целью улучшения технических характеристик, включая новые процессорные элементы, сложно–функциональные блоки, системы и сети на кристалле, квантовые компьютеры» паспорта научной специальности 2.3.2).

5. *Оценки* сложности многофункционального логического элемента LUT, реализующие вычисление нескольких функций одновременно, и логического элемента LUT, реализующего одновременно вычисление логической функции и дешифрацию набора переменных, которые позволяют осуществить выбор наиболее эффективного варианта реализации логического элемента (п. 2 «Разработка принципиально новых методов анализа и синтеза вычислительных систем и их элементов, с целью улучшения технических характеристик, включая новые процессорные элементы, сложно–функциональные блоки, системы и сети на кристалле, квантовые компьютеры» паспорта научной специальности 2.3.2).

Рекомендуется использование предлагаемых элементов при разработке ПЛИС для аппаратуры критического применения, в том числе бортовых цифровых комплексов военной техники.

Теоретическая значимость диссертационной работы состоит в том, что разработанные модели, методы, алгоритмы синтеза и оценки сложности логических элементов LUT расширяют научно–методический аппарат синтеза элементов программируемой логики, что позволяет синтезировать новые элементы, реализующие несколько функций одновременно.

Практическая значимость исследования заключается в том, что разработаны новые, запатентованные логические элементы, предложены схемы электрические принципиальные, топологии новых логических элементов, которые позволяют снизить аппаратные затраты в количестве транзисторов и площади кристалла более 15 % (акт внедрения ФИЦ ИУ РАН). Это позволяет расширить возможности существующих ПЛИС в том числе САПР для ПЛИС, которые будут учитывать новые возможности логики. Получены свидетельства о регистрации программ для ЭВМ, позволяющие синтезировать новые элементы.

Методология и методы исследования. В диссертационной работе используются методы и средства схемотехнического и топологического моделирования, анализа и синтеза схем, структурное программирование. Применяемые методы и средства основаны на положениях дискретной математики, математической логики, теории булевых функций и автоматов, комбинаторики, теории надежности, принципах МОП–схемотехники.

Область исследования, обозначенная в сформулированных задачах, соответствует п. 2 «Разработка принципиально новых методов анализа и синтеза вычислительных систем и их элементов, с целью улучшения технических характеристик, включая новые процессорные элементы, сложно–функциональные блоки, системы и сети на кристалле, квантовые компьютеры» паспорта научной специальности 2.3.2.

Достоверность и обоснованность результатов, полученных в диссертационной работе, заключается в том, что они не противоречат теоретическим положениям, известным из научных публикаций отечественных и зарубежных исследователей, а также подтверждается результатами, полученными в трех системах моделирования (Multisim, Microwind, Cadence Virtuoso), апробацией и внедрением предложенных в диссертации методов, моделей и алгоритмов подключения.

Апробация работы. Основные теоретические и практические результаты работы докладывались на научно–технических конференциях: «Инновационные технологии: теория, инструменты, практика» (InnoTech–2022), «2023 Seminar on

Microelectronics, Dielectrics and Plasmas», 24th International Conference of Young Professionals in Electron Devices and Materials (EDM) (2023), «Автоматизированные системы управления и информационные технологии» (АСУИТ–2023), «Инновационные технологии: теория, инструменты, практика» (InnoTech–2023) и в других международных и региональных конференциях.

Работы по теме диссертационного исследования выполнялись в рамках договора о научно–техническом сотрудничестве с отделом 52 ФИЦ ИУ РАН.

Публикации. Основные результаты диссертационной работы опубликованы в 16 печатных работах, из них 6 публикаций в ведущих рецензируемых научных изданиях, 2 публикации в изданиях, индексируемых в международных базах цитирования Scopus, 3 патента на изобретение, 2 свидетельства о регистрации программ для ЭВМ.

Объем и структура работы. Диссертация состоит из введения, пяти глав, заключения, списка литературы из 101 наименования и 3 приложений. Полный объем диссертации составляет 189 страниц, из которых 145 страниц занимает основной текст диссертации, включающий 116 рисунков и 13 таблиц.

ГЛАВА 1. ИССЛЕДОВАНИЕ СУЩЕСТВУЮЩИХ МОДЕЛЕЙ И МЕТОДОВ РЕАЛИЗАЦИИ ЛОГИЧЕСКИХ ФУНКЦИЙ В ПЛИС. ПОСТАНОВКА НАУЧНОЙ ЗАДАЧИ

1.1. Анализ объекта исследования – программируемых логических интегральных схем (ПЛИС)

Большие интегральные схемы (БИС), в том числе с программируемой логикой, строятся на МОП–транзисторах (металл–оксид–полупроводник, MOSFET – Metal–Oxide–Semiconductor Field–Effect Transistor), изобретенных в начале 60–х годов XX века [10]. В это же время появляется усовершенствованная технология КМОП (Комплементарная структура Металл – Оксид – Полупроводник, CMOS – Complementary Metal–Oxide–Semiconductor), в которой используются спаренные n–МОП (транзистор n–типа) и p–МОП (транзистор p–типа) так, что один в каждой паре всегда выключен [98]. Это снижает выделение тепла, поскольку во время перехода отсутствует ток покоя. Были разработаны соответствующие логические элементы, используемые в так называемых заказных микросхемах (ASIC – application–specific integrated circuit, «интегральная схема для конкретного применения»). Следующим большим шагом к созданию программируемой логики было изобретение в 70–х годах программируемого постоянного запоминающего устройства (программируемое ПЗУ – ППЗУ) вместо существовавшего постоянного запоминающего устройства (ПЗУ), куда данные устанавливались при производстве [13,16]. ППЗУ может программироваться пользователем на специальном программаторе, но однократно. Затем появляются и электрически перепрограммируемые ПЗУ (ЭППЗУ), и флэш–ПЗУ [12]. ПЗУ может быть использовано для реализации логических функций и систем функций в совершенной дизъюнктивной нормальной форме (СДНФ).

Для реализации логических функций в СДНФ может быть использован и мультиплексор (MUX, MS, MX) путем настройки константами входов каналов.

В 1975 году появляется первая программируемая логическая схема, а именно программируемая логическая матрица (ПЛМ) [18], которая состоит из матрицы И – для программирования конъюнкций и матрицы ИЛИ – для программирования

дизъюнкций на специальном программаторе. В ПЛМ логические функции представлены в дизъюнктивной нормальной форме (ДНФ). Архитектура ПЛМ изображена на рисунке 1.1.

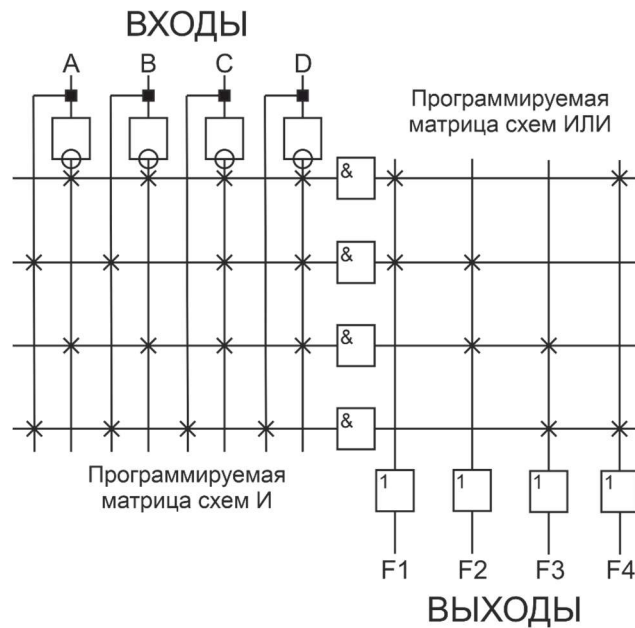


Рисунок 1.1 – Упрощенная архитектура ПЛМ

Для реализации n входных переменных требовалось 2^n вентилях И, а реализация m выходов осуществлялась с помощью m вентилях ИЛИ, каждый из которых имеет программируемые входы от всех вентилях И.

Аналогичная структура программируемой логики появилась тремя годами позднее и называлась – программируемая матрица логики (ПМЛ), отличающаяся тем, что программируется только матрица И (рисунок 1.2).

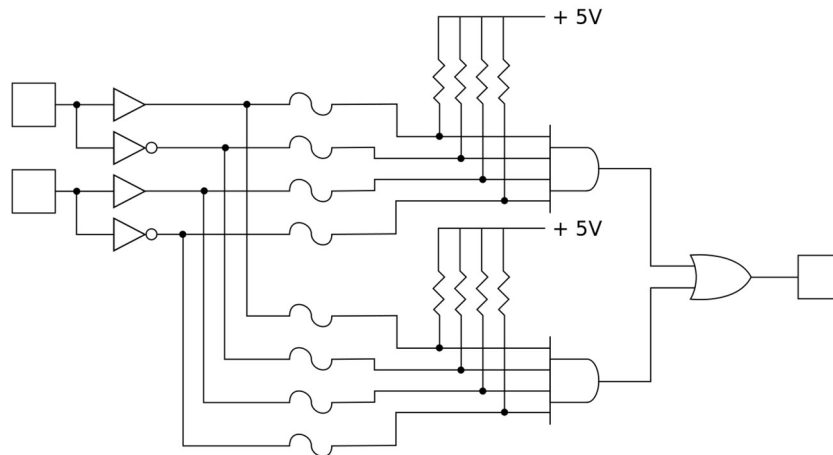


Рисунок 1.2 – Упрощенная архитектура ПМЛ

Синтез универсальных логических элементов в базисе КМОП логики, ПЗУ, ПЛМ, MUX рассматривался в работах Евреинова Э.В. [10,11], Косарева Ю.Г. [10], Прангишвили И.В. [11], Балашова Е.П. [2,3], Пузанкова Д.В. [2,7], Викентьева Л.Ф. [26], Аляева Ю.А. [26], Шалыто А.А. [26], Несмелова В.А.

ПЗУ, ПЛМ, мультиплексоры иногда назывались универсальными логическими модулями, поскольку реализовывалась настройка функций пользователем, а не в условиях производства. Но не хватало настройки связей.

Дальнейшее развитие технологии привело к реализации на одном кристалле нескольких ПМЛ, объединяемых программируемыми соединениями. Подобные архитектуры получили название сложных программируемых логических устройств (ПЛУ, CPLD – Complex Programmable Logic Device) [17,40].

Параллельно с CPLD развивались архитектуры вентильных матриц (GA – Gate Array) и матриц логических ячеек (LCA – Logic Cell Array, ULA – uncommitted logic array), в русскоязычной литературе получившие название базовых матричных кристаллов (БМК) [9,32]. Такие БМК, выпускает ТЦ МИЭТ (г. Зеленоград) [19], в которых используют даже не сами логические элементы, а четырехтранзисторные ячейки.

Первые вентильные матрицы были полузаказными, то есть программировались во время изготовления, что сильно сдерживало их широкое использование.

Однако, в 1985 г. фирма Xilinx выпустила программируемую пользователем вентильную матрицу (ПЛИС – программируемая логическая интегральная схема, FPGA – Field Programmable Gate Array) под названием XC2064 [52]. Эта БИС состояла из 64 программируемых логических блоков (CLB – Configurable Logic Block) и использовала программируемую матрицу для конфигурирования связей между программируемыми логическими блоками. Уже в свое время FPGA на 64 CLB значительно сэкономило место на печатной плате, а доступность реконфигурации добавила возможность обновлять функциональность устройств после изготовления во время эксплуатации, как говорят «in the field» (отсюда и название – Field-Programmable Gate Array).

В общем случае архитектура ПЛИС содержит конфигурируемые логические блоки [16,29,41], которые реализуют требуемую логическую функцию в СДНФ, программируемые электронные связи между конфигурируемыми логическими блоками, программируемые блоки ввода/вывода, которые обеспечивают связь внешнего вывода микросхемы с внутренней логикой (рисунок 1.3). В современных ПЛИС также встречаются дополнительные блоки памяти, блоки цифровой обработки сигналов (DSP – Digital Signal Processing), фазовая автоподстройка частоты (PLL – Phase Locked Loop) и другие.

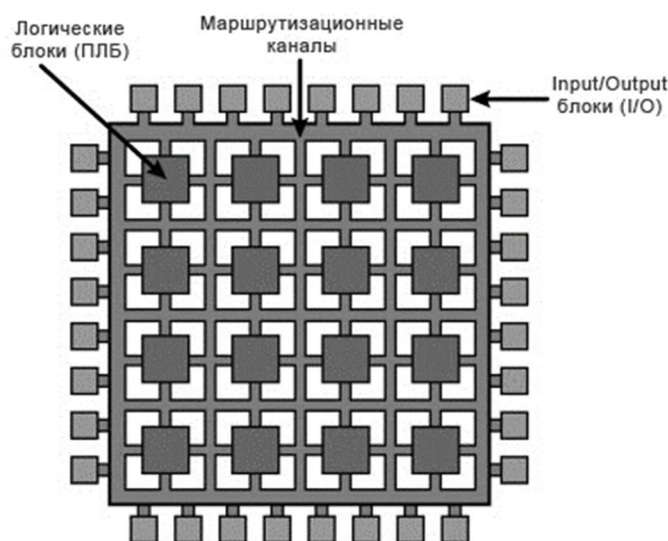


Рисунок 1.3 – Пример островной архитектуры ПЛИС

По типу хранения конфигурации в ПЛИС наиболее популярным является SRAM (Static Random Access Memory) – это ячейки статической памяти, изготовленные по технологии CMOS. Преимущества данной технологии – это возможность многократного перепрограммирования ПЛИС [48,49,98]. К недостаткам можно отнести не самое высокое быстродействие, после включения питания прошивку нужно вновь загружать. На плате должен еще стоять загрузчик, специальная микросхема или микроконтроллер – все это удорожает конечное изделие.

Основным элементом ПЛИС является конфигурируемый логический блок [67,93], в котором может быть выполнена логическая функция или реализовано

хранение результата вычисления в регистрах. Сложность и структура конфигурируемого логического блока определяется производителем. Например, в компании Altera встречается выражение Logic Array Block (LAB) – массив логики [67], а у компании Xilinx – Configurable Logic Block (CLB) [39,93]. Конфигурируемый логический блок может быть как очень простым – отдельный транзистор, так и очень сложным – целый процессор. Это крайние точки реализации. В первом случае потребуется огромное число программируемых связей, чтобы потом из отдельных транзисторов собрать требуемую схему. Во втором случае связей может быть нужно и не так много, но теряется гибкость проектирования пользовательской схемы. Поэтому конфигурируемый блок обычно представляет из себя что-то среднее: он обычно достаточно сложен, чтобы можно было бы зашить туда некоторую функцию, но и довольно мал, чтобы разместить множество таких блоков внутри ПЛИС, и чтобы была возможность связать их в единую схему. Таким образом, выбор структуры конфигурируемого логического блока производителем ПЛИС – это всегда поиск компромисса по площади кристалла, быстродействию и энергопотреблению. В качестве примера на рисунке 1.4 приведена схема базового логического элемента CPLD MAX II [70] компании Altera.

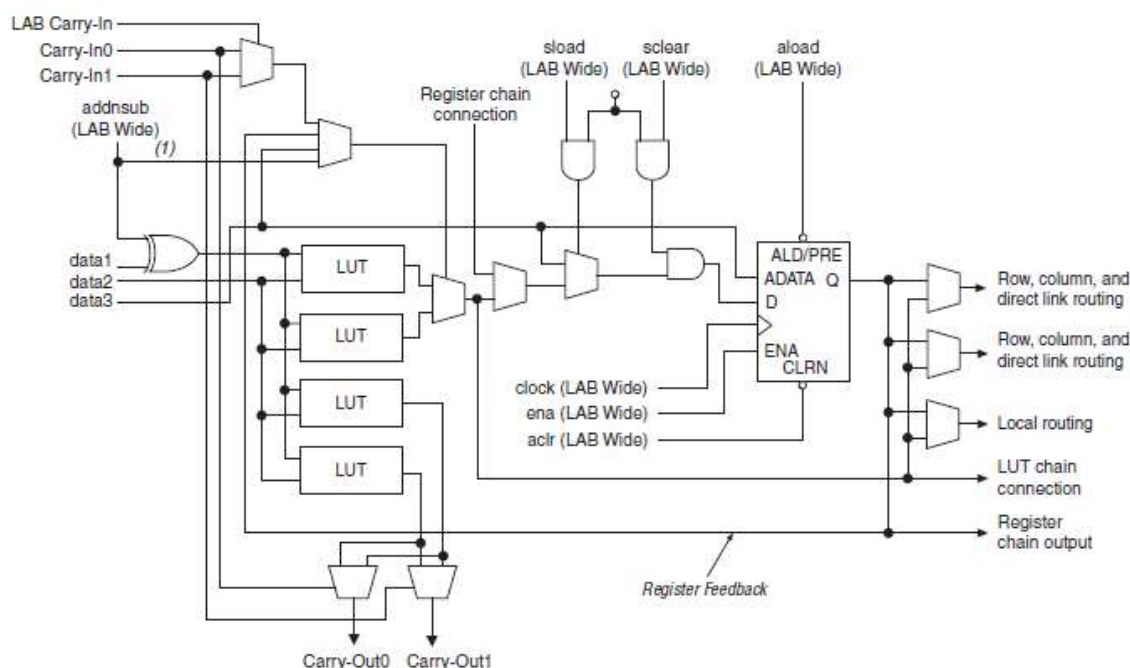


Рисунок 1.4 – Базовый логический элемент CPLD MAX II компании Altera

Работа триггера, мультиплексоров и отдельных логических элементов хорошо известна, поэтому основным элементом для совершенствования конфигурируемых логических блоков является LUT (Look-Up Table) [92]. Look-Up Table или просто Lookup Table [63], что дословно можно перевести как "справочная таблица" или "таблица поиска". LUT – это больше, чем таблица, LUT – это метод реализации функции, в котором непосредственное вычисление заменяется поиском по таблице готовых решений. Применительно к ПЛИС это позволяет реализовать любую логическую функцию в виде памяти SRAM, где адрес – это аргумент, а содержимое ячейки – значение.

Таким образом для того, чтобы описать логическую функцию трех переменных достаточно памяти на 8 ячеек (рисунок 1.5). Линейное представление логической функции LUT трех переменных определяется следующим образом:

$$z(x_3x_2x_1d) = d_0 \cdot \bar{x}_3\bar{x}_2\bar{x}_1 \vee d_1 \cdot \bar{x}_3\bar{x}_2x_1 \vee d_2 \cdot \bar{x}_3x_2\bar{x}_1 \vee d_3 \cdot \bar{x}_3x_2x_1 \vee d_4 \cdot x_3\bar{x}_2\bar{x}_1 \vee d_5 \cdot x_3\bar{x}_2x_1 \vee d_6 \cdot x_3x_2\bar{x}_1 \vee d_7 \cdot x_3x_2x_1, \quad (1.1)$$

где $d_0, d_1, d_2, d_3, d_4, d_5, d_6, d_7$ – данные конфигурации функции трех переменных $z(x_3x_2x_1)$. Комбинируя $d_0, d_1, d_2, d_3, d_4, d_5, d_6, d_7$, мы можем получить 2^8 функций.

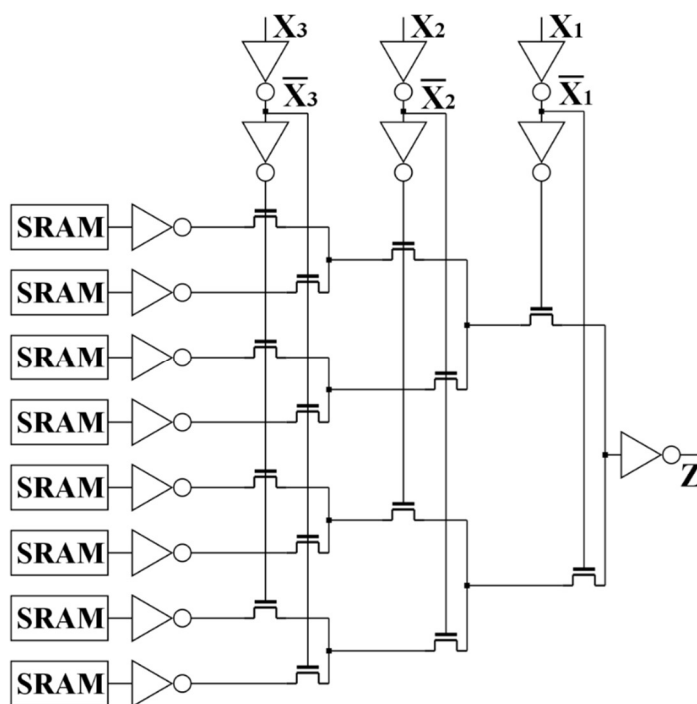


Рисунок 1.5 – Таблица поиска (LUT) на три переменные

В данном LUT трех переменных можно заметить, что реализуется только одна логическая функция в совершенной дизъюнктивной нормальной форме (СДНФ) [96]. Для реализации другой функции от этих же входных переменных необходимо либо загрузить новую настройку в ячейки памяти, либо задействовать дополнительный логический элемент. В результате вычисления логической функции задействуется только часть передающих транзисторов, что свидетельствует о возможностях повышения функциональности данных элементов.

Увеличение количества CLB также достигается с помощью уменьшения технологий реализации передающих транзисторов, которые претерпели значительные изменения за последние десятилетия и продолжают совершенствоваться. До техпроцесса 28 нм включительно транзисторы были планарными (каналы и затворы состояли только из плоских элементов) и просто уменьшались с минимальными изменениями в конструкции. Начиная с техпроцесса 22 нм, пришлось менять конструкцию более существенно с целью более эффективного управления затвором [65,73,91]. Каналы стали делать в виде «плавников», и затвор стал обволакивать их фактически с трёх сторон. Данная технология получила название – finFET (fin Field-Effect Transistor) [81]. Технология производства finFET хорошо масштабируется: от 22 нм до 5 нм. Но эта возможность по отношению к процессу finFET начинает становиться проблемой в узлах с меньшим характерным размером. Следовательно, для дальнейшего масштабирования полупроводниковых устройств потребуются новые технологические решения для производства интегральных микросхем. Следующая более компактная технология стала GAA (Gate All Around), которая дает значительные преимущества в освоении технологии до 1 нм (рисунок 1.6) [72].

Всенаправленный затвор, метод обработки полупроводников нового поколения, предлагает два уникальных преимущества по сравнению с finFET. Во-первых, GAA-транзисторы решают многие проблемы с токами утечки, поскольку каналы в этих устройствах горизонтальные.

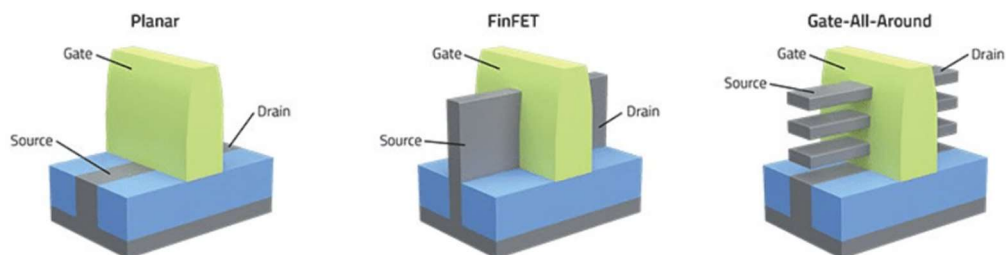


Рисунок 1.6 – Сравнение технологий производства транзисторов

Технология GAA заключается в наложении множества наночастиц или нанопроволок и окружении этих каналов затворными материалами со всех сторон. Это обеспечивает более высокую пропускную способность по току чем finFET, которые требуют размещения вертикальных «ребер» бок о бок, чтобы увеличить максимальное количество электрической энергии, которая может проходить через транзистор. Во-вторых, транзисторы GAA окружены затворами со всех четырех сторон. Это улучшает их структуру, позволяя затвору контактировать со всеми четырьмя сторонами по сравнению с тремя в техпроцессе finFET. В результате конструкция GAA может управлять током более точно, чем finFET. Следует отметить, что архитектура GAA-транзистора на 90% похожа на finFET, а остальные 10% различия связаны с укладкой горизонтальных наночастиц друг на друга. Это приводит к большему контролю над потоком тока, что приводит к увеличению энергоэффективности. В результате электронные устройства, использующие чипы на основе GAA работают быстрее и потребляют меньше энергии, чем изготовленные с использованием техпроцесса FinFET.

ПЛИС FPGA в начале своего изобретения содержали тысячи транзисторов и десятки, сотни логических элементов LUT, в настоящее время у передовых производителей их десятки миллионов [93], а количество транзисторов в БИС FPGA составляет десятки миллиардов. LUT представляют собой мультиплексоры из деревьев передающих транзисторов (Pass Transistors), настраиваемые конфигурационной памятью на реализацию заданной функции, а сами LUT связаны программируемыми матрицами коммутаций глобальных связей – с выходами ПЛИС, локальных связей – друг с другом и другими элементами.

Количество переменных в одном LUT в начале это было три – четыре, теперь шесть – семь, наблюдаются тенденции к их дальнейшему росту [30,31,57,78]. В некоторых ПЛИС реализуются пока не все возможные логические функции восьми переменных.

Но в то же время, существующие функциональные возможности используются не в полной мере, один LUT реализует при данной конфигурации только одну логическую функцию, причём всегда активируется только половина транзисторов в дереве передающих транзисторов (Pass Transistors). То есть, число элементов и количество переменных увеличивается, а используется в каждый момент времени, на данном наборе переменных половина элемента.

1.2. Анализ предмета исследования – научно–методического аппарата синтеза логических элементов ПЛИС

Ключевую роль в применении ПЛИС играет программное обеспечение, которое позволяет изменять конфигурацию системы (САПР). Основной задачей для разработчиков является процесс логического синтеза схем и качество конечного устройства значительно зависит от того, насколько эффективно этот синтез проведен. Это подчеркивает необходимость разработки и применения алгоритмов синтеза, способных максимально учесть архитектуру ПЛИС [28]. Процесс логического синтеза проекта может быть проведен двумя способами: технологически–зависимым, когда исходный код преобразуется в код на основе предопределенной библиотеки логических компонентов, и технологически–независимым, результатом которого становится код, основанный на таблицах поиска (Look–Up Table). Однако, многие библиотеки элементов не могут полностью покрыть все функциональные возможности программируемой логической ячейки в ПЛИС, что делает технологически–независимый синтез более предпочтительным в таких случаях. С другой стороны, независимый синтез не способен учитывать и эффективно использовать особенности архитектуры целевой ПЛИС.

Технологически–зависимый способ реализуется в системах специализированного проектирования таких производителей как: Altera (Intel) (Quartus) [86], Xilinx (Vivado Design Suite) [97], Microsemi (Liberio IDE) [77] с выделением параметров по площади, занимаемой на кристалле, потребляемой мощности и быстродействию. Например, для уменьшения размера проекта в САПР Xilinx имеется функция Area Reduction, а в Quartus II возможно минимизировать память автомата с помощью установки State Machine в Minimal Bits.

В технологически–независимом направлении совершенствования ПЛИС производители оптимизируют архитектуру ПЛИС с помощью преобразования LUT. Используется метод увеличения разрядности одного LUT, либо каскадирования для синтеза LUT на большое число переменных из LUT меньшей размерности.

Например, четыре 4–LUT вместе с мультиплексором 4:1 или еще 2 4–LUT используются для построения 6–LUT, но реализация использует только 6 из 16 доступных входов и создает дополнительные задержки между различными LUT. Для реализации LUT с 6 входами вся архитектура должна быть оптимизирована специально для 6–LUT в качестве базового логического блока. В приведенной оценке (рисунок 1.7) использования площади кристалла в зависимости от количества реализуемых переменных в LUT [54], также демонстрируется неэффективность использования LUT на большое число входов.

В ПЛИС Intel Agilex используется подход, который позволяет более эффективно использовать устройства. Как показано на рисунке 1.8, ПЛИС Agilex [71] имеют новую структуру 6–LUT с 8 входами, называемую адаптивным логическим модулем (ALM – Adaptive Logic Module). Некоторые входы меньших LUT разделены, чтобы обеспечить дополнительную гибкость.

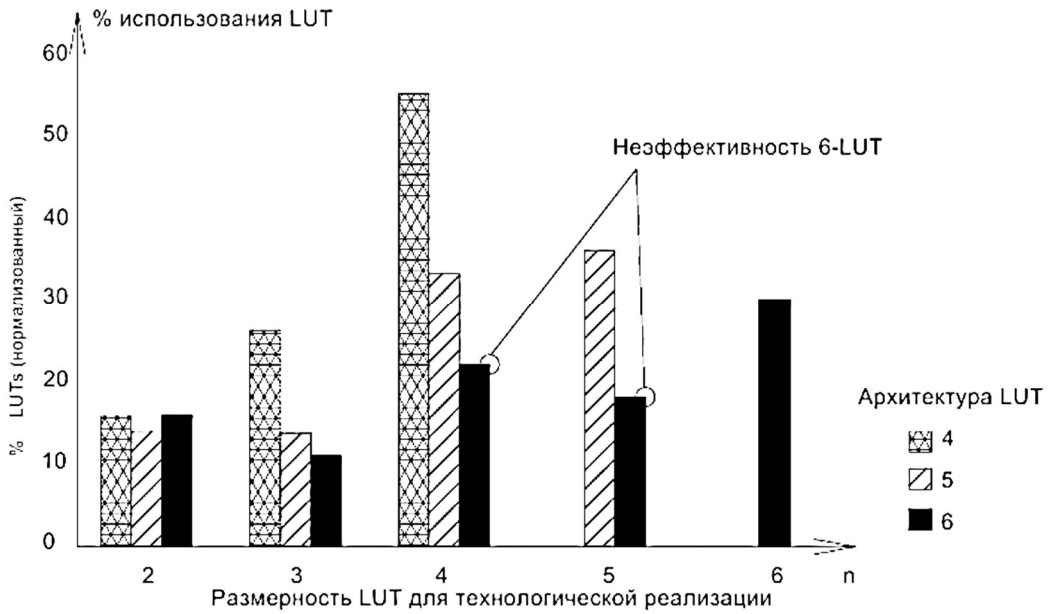


Рисунок 1.7 – Распределение логических функций по площади с использованием различных LUT

С помощью дополнительных входов можно комбинировать более мелкие логические функции, зависящие от разных сигналов.

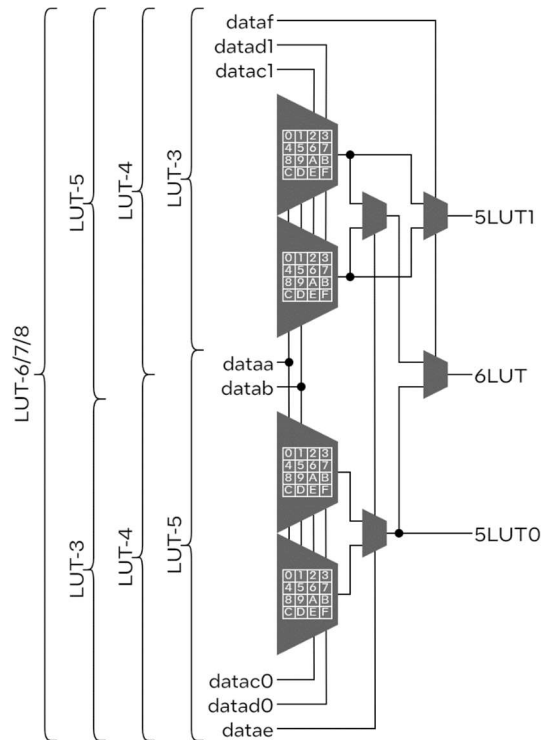


Рисунок 1.8 – 6-LUT с 8 входами фирмы Intel Agilex

ПЛИС AMD Virtex UltraScale+ используют базовую LUT с 6 входами и дополнительный выход LUT с 5 входами [93], которая может действовать либо как двойное асинхронное 32–битное ПЗУ (с 5–битной адресацией), либо реализовывать любые две логические функции с 5 входами с общими входами или реализовать логическую функцию с 6 входами и логическую функцию с 5 входами с общими входами и общими логическими значениями (рисунок 1.9).

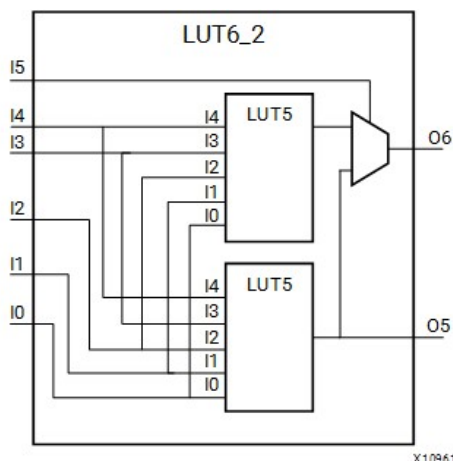


Рисунок 1.9 – 6–LUT с 6 входами и двумя выходами фирмы AMD Virtex UltraScale+

Таким образом, простое увеличение числа переменных в LUT более четырех часто является нецелесообразным по времени задержки и площади кристалла, поэтому основные компании производители применяют более эффективные методы организации архитектуры таблицы поиска для повышения функциональности ПЛИС.

Для ряда важных задач, в том числе критического применения (в бортовых цифровых вычислительных комплексах авиационной, космической и военной техники), необходимо расширение имеющихся возможностей логики ПЛИС для совершенствования соответствующих вычислительных систем.

Повышения эффективности ПЛИС возможно также методом комбинирования подходов CPLD и FPGA [58], однако этот метод требует кардинального изменения технологии и не рассматривается в данной работе.

Один из методов повышения функциональных возможностей FPGA – использование неактивных на данном наборе переменных транзисторов в LUT. Это, например, использовано в работе Городилова А. Ю [8] для двухканального вычисления логической функции с целью повышения достоверности.

Тот же подход может быть использован для вычисления второй функции от тех же аргументов [24,45]. Но при этом остаются и другие неактивные части (не только половина, но и четверть, восьмая часть и т.д.). Именно в этом направлении и возможно дальнейшая модификация LUT.

Однако, подобное дробление возможно только до уровня $n-1$. К тому же, реализация дешифрации набора переменных, необходимая, например, в задачах кодирования, в этом случае затруднена, так как требуется два в степени n функций, содержащих всего одну конъюнкцию каждая. При этом обычных LUT также требуется два в степени n .

Использование обратного дерева передающих транзисторов для реализации систем функций [6] путем реализации дешифратора показана на рисунке 1.10.

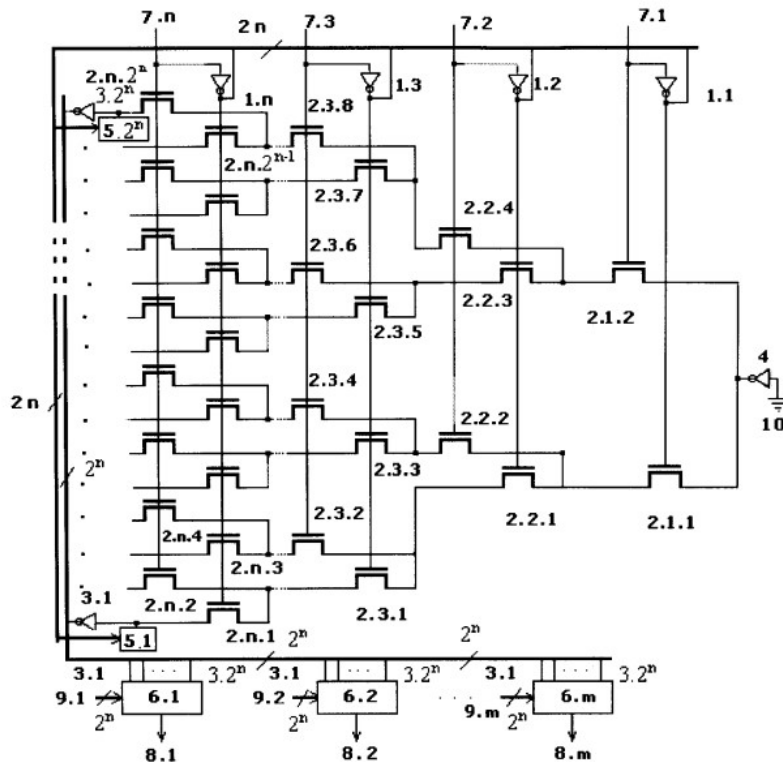


Рисунок 1.10 – LUT-DC, реализующий системы функций, 6.1,6.2,...6.m – блоки дизъюнкций.

Но этот метод ориентирован на реализацию систем функций с помощью блоков дизъюнкций и его применение для дешифрации не эффективно. Поэтому целесообразно реализовать дешифрацию с меньшей сложностью. Кроме того, имеет смысл обеспечить и реализацию основной функции, что не выполняется на рисунке 1.10.

1.3. Математическая постановка научной задачи и частных задач исследования

Анализ объекта исследования позволил установить тенденцию увеличения количества логических элементов LUT ПЛИС FPGA с одновременным увеличением числа переменных реализуемых логических функций, но, в отличие от логических элементов ПЛИС CPLD, в LUT FPGA реализуется только одна логическая функция для данной конфигурации несмотря на то, что переменных достаточно много. В то же время, увеличение количества входов LUT с целью повышения функциональности ПЛИС сталкивается с проблемами эффективного использования площади кристалла и быстродействия.

Совершенствование логики ПЛИС в Российской Федерации особенно актуально в связи известными проблемами импортозамещения электронной компонентной базы.

Анализ предмета исследования показал, что синтез многофункциональных элементов LUT рассмотрен не в полной мере, имеются возможности его дальнейшего совершенствования.

Таким образом, необходимо разработать научно-методический аппарат, позволяющий синтезировать логические элементы LUT, в которых одновременно вычисляется несколько логических функций для данной конфигурации.

Математическая постановка задачи включает в себя разработку моделей и методов синтеза логических элементов LUT, вычисляющих несколько логических функций одновременно.

Дано: Существующие модели логических элементов (LUT), реализующие функции от n входных переменных:

$z_0(d_0 \dots d_{2^n-1} x_n \dots x_1)$ – известная модель LUT;

$z_{1,2}(d_{0,1} \dots d_{2^n-1,1} d_{0,2} \dots d_{2^n-1,2} x_n \dots x_1)$ – модель LUT, вычисляющая две функции;

$z_j = \bigwedge_{i=1}^n x_j^{\delta_{i,j}}, j = 0, 1, 2, \dots, 2^n - 1$ – известная модель логического элемента DC

LUT, вычисляющего функции дешифрации входного набора для реализации систем функций.

Требуется:

1) разработать модель и метод реализации в одном n – LUT нескольких функций z одновременно

$$z_{0 \dots 2^v-1}; 1 < v \leq n-1; \quad (1.1)$$

2) разработать модель и метод реализации в одном n – LUT логической функции и дешифрации набора переменных

$$z(d_0, d_1, d_2 \dots d_{2^n-1}, x_n \dots x_1), z_j; \quad (1.2)$$

3) разработать функциональные электрические схемы, выполнить схемотехническое моделирование и получить оценки сложности по показателям $Q, i = 1 \dots 4$:

– количества транзисторов: $L_{mf}(n)$ – предлагаемого многофункционального элемента $z_{0 \dots 2^v-1}$, $L_{DC}(n)$ – предлагаемого элемента для дешифрации $z(d_0, d_1, d_2 \dots d_{2^n-1}, x_n \dots x_1), z_j$;

– площади, занимаемой на кристалле: $S_{mf}(n)$ – предлагаемого многофункционального элемента, $S_{DC}(n)$ – предлагаемого элемента для дешифрации;

– потребляемой мощности: $W_{mf}(n)$ – предлагаемого многофункционального элемента, $W_{DC}(n)$ – предлагаемого элемента для дешифрации;

– временной задержки: $T_{mf}(n)$ – предлагаемого многофункционального элемента, $T_{DC}(n)$ – предлагаемого элемента для дешифрации;

- 4) выбрать варианты, имеющие показатели

$$L_{mf} < 2^v \cdot L_0, L_{DC} < 2^n \cdot L_0, \quad (1.3)$$

L_0 – количество транзисторов известного логического элемента (z_0),

$$S_{mf} < 2^v \cdot S_0, S_{DC} < 2^n \cdot S_0, \quad (1.4)$$

S_0 – площадь кристалла известного логического элемента (z_0);

5) получить: а) множество Парето–оптимальных вариантов блока логических элементов, реализующего g функций; б) множество Парето блоков элементов, обеспечивающих одновременное вычисление логической функции и дешифрацию набора переменных.

б) выбрать требуемый вариант при заданных ограничениях $W_{\text{допустимое}}$, $T_{\text{допустимое}}$.

Декомпозиция научной задачи позволяет выделить следующие частные задачи исследования:

1. Аналитический обзор, анализ, сравнение существующих моделей, методов и алгоритмов синтеза логических элементов ПЛИС.

2. Разработка моделей, реализующих вычисление нескольких функций одновременно и дешифрацию набора переменных вместе с вычислением основной функции.

3. Разработка метода синтеза многофункционального логического элемента, реализующего одновременное вычисление 2^v , $v = 1, 2, 3, \dots, n-1$ логических функций.

4. Разработка метода синтеза логического элемента, реализующего вычисление основной логической функции одновременно с дешифрацией набора переменных.

5. Разработка алгоритмов подключения дополнительных транзисторов в логических элементах LUT, реализующего вычисление нескольких функций одновременно и дешифрацию набора переменных вместе с вычислением основной функции.

6. Получение сравнительных оценок сложности по показателям количества транзисторов, площади, занимаемой на кристалле, потребляемой мощности и временной задержке предлагаемых и известных логических элементов LUT.

7. Апробация и внедрение разработанных моделей, методов и алгоритмов, предлагаемых логических элементов, реализующих вычисление нескольких функций одновременно и дешифрацию набора переменных вместе с вычислением основной функции в ФИЦ ИУ РАН и учебном процессе ФГАОУ ВО «ПНИПУ».

1.4. Выводы по главе 1

1. Проведенный анализ показал, что в настоящее время в базовых логических блоках используются модифицированные таблицы поиска LUT на 6 и более входов. Дальнейшее увеличение числа переменных с целью повышения быстродействия затруднительно в связи с растущей сложностью.

2. Количество логических элементов ПЛИС и количество реализуемых функций постоянно увеличивается с одной стороны, а с другой стороны существующие функциональные возможности используются не в полной мере. Современные методы синтеза позволяют синтезировать новые логические элементы с большим количеством входов, однако не в полной мере учитывают функциональные возможности имеющихся логических элементов.

3. Известные методы реализации дешифрации входного набора не предполагают реализации основной функции логического элемента, что снижает функциональные возможности известного логического элемента.

4. Рекомендуется разработать новые модели, методы синтеза логических элементов ПЛИС, обеспечивающих одновременную реализацию нескольких логических функций, а также дешифрацию входного набора.

ГЛАВА 2. РАЗРАБОТКА МНОГОФУНКЦИОНАЛЬНОГО ЛОГИЧЕСКОГО ЭЛЕМЕНТА n – LUT, РЕАЛИЗУЮЩЕГО ОДНОВРЕМЕННО 2^v ФУНКЦИЙ ОТ ОДНИХ И ТЕХ ЖЕ n ПЕРЕМЕННЫХ

2.1. Разработка модели реализации в одном n – LUT одновременно 2^v функций от одних и тех же переменных

Реализация логических функций в ПЛИС типа FPGA в LUT основана на вычислении значения одной функции z от n аргументов x_i , заданной в совершенной дизъюнктивной нормальной форме (СДНФ):

$$z(d_0 \dots d_{2^n-1} x_n \dots x_1) = \bigvee_{j=0}^{2^n-1} (\&_{i=1}^n x_i^{\sigma(i,j)} d_j), \quad (2.1)$$

где $\sigma(i, j)$ – показатель инверсирования соответствующей СДНФ, равный $1 - B[i(j)]$, где $B[i(j)]$ – бинарное представление i – го разряда числа j ; $d_j \in \{0, 1\}$ – конфигурационный бит, значение функции в соответствующей строке таблицы истинности функции z , которое записывается в оперативную память (в некоторых вариантах ПЛИС FPGA используется также и флэш-память, как в CPLD).

Более широкая трактовка модели логического элемента включает также задержку, потребляемую мощность (статическую и динамическую) для различных значений напряжения питания и частот входных сигналов, сложность реализации в количестве транзисторов и/или площади, занимаемой устройством на кристалле кремния. Кроме того, может оцениваться еще и надежность в виде вероятности безотказной работы для заданного времени. Эти параметры имеют важное значение для оценки эффективности предложенных технических решений.

Выражение (2.1) – линейное представление логической функции, которое схематически может быть реализовано стандартными логическими элементами в некотором технологическом базисе. Более компактное представление основано на использовании деревьев из передающих МОП транзисторов.

В таком случае (2.1) реализуется либо параллельным соединением 2^n цепочек из n передающих транзисторов (2.2) (одноуровневым 2^n деревом, с 2^n

ветвей по n транзисторов, $(\vee\bullet)$ – корень дерева, реализующий операцию монтажного ИЛИ (wired OR):

$$z(d_0 \dots d_{2^n-1} x_n \dots x_1) = \dots \left. \begin{array}{l} \frac{d_{2^n-1} x_1 \dots x_n}{\phantom{d_{2^n-1} x_1 \dots x_n}} \\ \frac{d_{2^n-2} x_1 \dots \bar{x}_n}{\phantom{d_{2^n-2} x_1 \dots \bar{x}_n}} \\ \frac{d_1 x_1 \dots \bar{x}_n}{\phantom{d_1 x_1 \dots \bar{x}_n}} \\ \frac{d_0 \bar{x}_1 \dots \bar{x}_n}{\phantom{d_0 \bar{x}_1 \dots \bar{x}_n}} \end{array} \right\} (\vee\bullet), \quad (2.2)$$

либо, с целью уменьшения количества транзисторов, n -уровневым бинарным деревом транзисторов (2.3):

$$z(d_0 \dots d_{2^n-1} x_n \dots x_1) = \dots \left. \begin{array}{l} \frac{d_0 \bar{x}_1}{\phantom{d_0 \bar{x}_1}} \\ \frac{d_1 x_1}{} \\ \cdot \quad \frac{\bar{x}_{n-1}}{\phantom{\bar{x}_{n-1}}} \\ \dots \quad \frac{x_{n-1} \quad \bar{x}_n}{\phantom{x_{n-1} \quad \bar{x}_n}} \\ \cdot \quad \frac{\bar{x}_{n-1} \quad x_n}{\phantom{\bar{x}_{n-1} \quad x_n}} \\ \frac{d_{2^n-2} \bar{x}_1 \quad x_{n-1}}{\phantom{d_{2^n-2} \bar{x}_1 \quad x_{n-1}}} \\ \frac{d_{2^n-1} x_1}{\phantom{d_{2^n-1} x_1}} \end{array} \right\} (\vee\bullet). \quad (2.3)$$

Ранее предложена реализация в одном LUT сразу двух функций [24] Z_1, Z_2 за счет использования неактивной половины дерева передающих транзисторов. В этом случае выражение (2.3) преобразуется к виду:

$$\begin{aligned}
& \frac{\overline{d_{0,1}x_n}}{\overline{d_{2^{n-1},2}x_n}} \frac{\overline{x_1}}{\overline{x_1}} \\
& \frac{\overline{d_1x_n}}{\overline{d_{2^{n-1}+1}x_n}} \frac{\overline{x_1}}{\overline{x_1}} \\
& \cdot \frac{\overline{x_{n-1}}}{\overline{x_{n-1}}} \frac{\overline{x_n}}{\overline{x_n}} \bullet \vee(z_1) \\
z_{1,2}(d_{0,1} \dots d_{2^{n-1},1} d_{0,2} \dots d_{2^{n-1},2} x_n \dots x_1) = & \cdot \dots \frac{\overline{x_{n-1}}}{\overline{x_{n-1}}} \frac{\overline{x_n}}{\overline{x_n}} \\
& \cdot \left. \frac{\overline{d_{2^{n-2}x_n}}}{\overline{d_{2^{n-1}-2}x_n}} \frac{\overline{x_1}}{\overline{x_1}} \frac{\overline{x_{n-1}}}{\overline{x_{n-1}}} \frac{\overline{x_n}}{\overline{x_n}} \right\} \bullet \vee(z_2) \\
& \frac{\overline{d_{2^{n-1}x_n}}}{\overline{d_{2^{n-1}-1}x_n}} \frac{\overline{x_1}}{\overline{x_1}}
\end{aligned} \tag{2.4}$$

Предлагается дальнейшая модификация модели (2.3) с целью реализации сразу нескольких функций (2^v , $v=2,3,\dots,n-1$) за счет использования нескольких неактивных поддеревьев:

$$\begin{aligned}
& \frac{\overline{D_{v,0} \overline{X_{v,0} x_1}}}{\overline{\overline{D_{v,1} \overline{X_{v,1} x_1}}}} \\
& \frac{\overline{D_{v,2^{n-1}-2} \overline{X_{v,0} x_1}}}{\overline{\overline{D_{v,2^{n-1}-1} \overline{X_{v,1} x_1}}}} \\
z_{0 \dots 2^v-1}(d_{0,1} \dots d_{2^{n-1},1} \dots d_{0,2^v} \dots d_{2^{n-1},2^v} x_n \dots x_1) = & \cdot \frac{\overline{x_{n-v}}}{\overline{x_{n-v}}} \frac{\overline{X_{v,0}(\bullet z_0)}}{\overline{X_{v,1}(\bullet z_1)}} \\
& \cdot \frac{\overline{x_{n-v}}}{\overline{x_{n-v}}} \frac{\overline{\overline{X_{v,2^v-2}(\bullet z_{2^v-2})}}}{\overline{\overline{X_{v,2^v-1}(\bullet z_{2^v-1})}}} \\
& \frac{\overline{D_{v,2^{n-1}-2} \overline{X_{v,0} x_1}}}{\overline{\overline{D_{v,2^{n-1}-1} \overline{X_{v,1} x_1}}}} \frac{\overline{x_{n-v}}}{\overline{x_{n-v}}} \frac{\overline{\overline{X_{v,2^v-2}(\bullet z_{2^v-2})}}}{\overline{\overline{X_{v,2^v-1}(\bullet z_{2^v-1})}}}
\end{aligned} \tag{2.5}$$

где $D_{v,i}, i=1 \dots 2^v-1$ – дополнительные настройки по вектору переменных $\overline{X_{vi}}$ и соответствующими поддеревьями [34,90].

Таким образом, в общем случае используется не одна старшая переменная, а вектор $\overline{X_{v,j}}, j=0 \dots 2^v-1$ части v старших переменных.

Пример реализации в n -LUT двух функций одновременно

В этом случае имеются параметры $n, \nu=2$, $S = \{n\}$. Активное поддерево n -LUT определяется старшей переменной x_n . Тогда при $x_n = 0$:

$$\bar{z}_{out.x_n^0} = \bar{z}_{out.\bar{x}_n} = \bigvee_{i=1}^{2^{n-1}} \left(\bigwedge_{j=1}^{n-1} x_j^{\sigma(i-1,j)} \wedge \bar{d}_i \right), \quad (2.6)$$

при $x_n = 1$:

$$\bar{z}_{out.x_n^1} = \bar{z}_{out.x_n} = \bigvee_{i=2^{n-1}}^{2^n} \left(\bigwedge_{j=1}^{n-1} x_j^{\sigma(i-1,j)} \wedge \bar{d}_i \right). \quad (2.7)$$

В таком случае, неактивное поддерево можно использовать для вычисления второй логической функции или для контроля, с этой целью необходимо дублировать старшую переменную, ввести переменную x_n^* и дублировать настройки, задавая второй набор настроечных данных d :

$$\begin{aligned} \bar{z}_{1.out.x_n^\sigma} &= \bigvee_{i=1}^{2^{n-1}} \left(\bigwedge_{j=1}^{n-1} x_j^{\sigma(i-1,j)} \wedge \bar{d}_{1,i} \right) \bigvee_{i=2^{n-1}}^{2^n} \left(\bigwedge_{j=1}^{n-1} x_j^{\sigma(i-1,j)} \wedge \bar{d}_{1,i} \right), \\ \bar{z}_{2.out.(x^*)^\sigma} &= \bigvee_{i=1}^{2^{n-1}} \left(\bigwedge_{j=1}^{n-1} x_j^{\sigma(i-1,j)} \right) \bigwedge_{i=2^{n-1}}^1 \bar{d}_{2,i} \bigvee_{i=2^{n-1}}^{2^n} \left(\bigwedge_{j=1}^{n-1} x_j^{\sigma(i-1,j)} \right) \bigwedge_{i=2^n}^{2^{n-1}} \bar{d}_{2,i}. \end{aligned} \quad (2.8)$$

То есть таблица истинности $\bar{d}_{2,i}$ записывается наоборот – старшими разрядами вперёд, чтобы использовать неактивную часть дерева. Управление подключением настройки каждой половины дерева осуществляется переменной $t(x_n)$:

$$\begin{aligned} \bar{z}_{1.out.x_n^\sigma} &= \bigvee_{i=1}^{2^{n-1}} \left(\bigwedge_{j=1}^{n-1} x_j^{\sigma(i-1,j)} \wedge \bar{d}_{1,i} \right) \wedge t(x_n) \bigvee_{i=2^{n-1}}^{2^n} \left(\bigwedge_{j=1}^{n-1} x_j^{\sigma(i-1,j)} \wedge \bar{d}_{1,i} \right) \wedge \bar{t}(x_n), \\ \bar{z}_{2.out.(x^*)^\sigma} &= \bigvee_{i=1}^{2^{n-1}} \left(\bigwedge_{j=1}^{n-1} x_j^{\sigma(i-1,j)} \right) \bigwedge_{i=2^{n-1}}^1 \bar{d}_{2,i} \wedge \bar{t}(x_n) \bigvee_{i=2^{n-1}}^{2^n} \left(\bigwedge_{j=1}^{n-1} x_j^{\sigma(i-1,j)} \right) \bigwedge_{i=2^n}^{2^{n-1}} \bar{d}_{2,i} \wedge t(x_n). \end{aligned} \quad (2.9)$$

2.2. Разработка метода реализации в одном n – LUT одновременно 2^v функций от одних и тех же переменных

Метод предполагает задание структуры исходного (базового) логического элемента, параметров n, v и соответствующую процедуру синтеза n – LUT, реализующего одновременно 2^v функций от одних и тех же n переменных.

При $v = 0$ имеется исходный элемент, реализующий одну функцию, поскольку число дополнительно реализуемых функций равно $2^v - 1$.

При $v = 1$ получаем одну дополнительную функцию, то есть всего две функции 2^1 .

При $v = 2$ получаем три дополнительные функции – всего четыре функции 2^2 .

Таким образом, вводится множество Φ реализуемых функций с элементами $\{0, 1, 2, \dots, 2^v - 1\}$, где нулевой номер имеет исходная функция.

В отличие от существующего метода дополнительные логические элементы LUT могут подключаться не только ко входам и выходам других LUT последовательно, но и к внутренним узлам схемы параллельно.

Рекомендуются следующие наборы параметров: $n = 3, v = 1$; $n = 4, v = 1, 2$; $n = 5, v = 1, 2, 3$; $n = 6, v = 1, 2, 3, 4$; $n = 7, v = 1, 2, 3, 4, 5$.

Предлагаются следующие варианты реализации исходного логического элемента:

1. Идеальный элемент n – LUT в виде одного дерева передающих транзисторов (S0).
2. Составной элемент n – LUT из элементов на одну переменную 1–LUT (S1).
3. Составной элемент n – LUT из набора S разных элементов i –LUT, $i = 1, 2, \dots, n - 1$. Например, $S = \{1, 2, 3\}$.

Процедура синтеза включает построение исходного n – LUT в зависимости от выбранного варианта и использует существующий метод, производящий

суперпозицию заданных LUT, но при этом определяются внутренние узлы схемы для дополнительных LUT.

Введем понятие «поддерево». Поддеревами будем называть части деревьев, которые активизируются старшей переменной или наборами (векторами) ν старших переменных.

Для $n = 1$ поддереьев нет (=0), поскольку нет старших переменных.

$$z(d_0 d_1 x) = \left. \begin{array}{l} \overline{x} \\ x \end{array} \right\} (\vee \bullet). \quad (2.10)$$

Для $n = 2$ их два, поскольку старшая переменная одна. Выход одного поддерева – по входу $\overline{x_3}$, второго – по входу x_3 .

$$z(d_0 d_1 d_2 d_3 x_2 x_1) = \left. \begin{array}{l} \overline{x_2} - \\ \overline{x_3} \\ \overline{x_3} \\ \overline{x_2} \\ \overline{x_2} \end{array} \right\} (\vee \bullet). \quad (2.11)$$

Для $n = 3$ их четыре, поскольку есть две старших переменных. Выход одного поддерева – по входу $\overline{x_2}$, $\overline{x_3}$, второго – по входу $\overline{x_2}$, x_3 , третьего по входу x_2 , $\overline{x_3}$, четвертого по входу x_2 x_3 .

$$z(d_0 d_1 d_2 d_3 d_4 d_5 d_6 d_7 x_3 x_2 x_1) = \left. \begin{array}{l} \overline{d_0 x_1} \\ \overline{d_1 x_1} \\ \overline{d_2 x_1} \\ \overline{d_3 x_1} \\ \overline{d_3 x_1} - \\ \overline{x_2} - \\ \overline{x_3} \\ \overline{x_3} \\ \overline{d_4 x_1} \\ \overline{x_2} \\ \overline{d_5 x_1} \\ \overline{x_2} \\ \overline{d_6 x_1} \\ \overline{d_7 x_1} \end{array} \right\} (\vee \bullet). \quad (2.12)$$

Рассмотренные деревья являются базовыми. Теперь рассмотрим составные деревья, поскольку в силу ограничений Мида–Конвей [79] необходимо восстановление уровня сигнала после каждых трех последовательно соединенных передающих транзисторов.

Для $n = 4$:

$$z(d_0 \dots d_{15} \overline{x_4} \overline{x_3} \overline{x_2} \overline{x_1}) : \left. \begin{array}{l} \overline{d_0 x_1} \\ \overline{d_1 x_1} \\ \overline{d_2 x_1} \\ \overline{d_3 x_1} \\ \overline{d_4 x_1} \\ \overline{d_5 x_1} \\ \overline{d_6 x_1} \\ \overline{d_7 x_1} \end{array} \right\} (\vee \bullet) \left. \begin{array}{l} \overline{d_8 x_1} \\ \overline{d_9 x_1} \\ \overline{d_{10} x_1} \\ \overline{d_{11} x_1} \\ \overline{d_{12} x_1} \\ \overline{d_{13} x_1} \\ \overline{d_{14} x_1} \\ \overline{d_{15} x_1} \end{array} \right\} (\vee \bullet) \left. \begin{array}{l} \overline{x_2} \\ \overline{x_3} \\ \overline{x_4} \end{array} \right\} (\vee \bullet) \left. \begin{array}{l} \overline{x_2} \\ \overline{x_3} \\ \overline{x_4} \end{array} \right\} (\vee \bullet). \quad (2.13)$$

Здесь получаем сумму исходных деревьев, то есть восемь. Выход первого поддерева – по входу $\overline{x_2}$, $\overline{x_3}$, $\overline{x_4}$, ... восьмого – по входу $\overline{x_2}$, $\overline{x_3}$, $\overline{x_4}$.

Будем нумеровать корни этих поддеревьев i (или входы основного и дополнительных выходных LUT) исходя из параметра $\nu: i = 2^\nu - 1, \nu = 0, 1, 2, \dots, 2^{n-1} - 1$. Аналогичной будет нумерация входов дополнительных входных LUT.

Таким же образом строятся составные деревья для $n=5, 6, 7, 8$.

Предлагаются правила подключения конфигурационных входов к дополнительным входным LUT и подключения поддеревьев к дополнительным выходным $(2^\nu - 1)$ -LUT [33].

Правило суммирования по модулю 2^n для подключения конфигурационных входов к дополнительным входным LUT.

Определение номера конфигурационного бита для подключения ко входам дополнительных входных LUT необходимо выполнить суммирование исходного номера с числом $2^{n-\nu}$ по модулю 2^n .

Пусть $n = 4, \nu = 1$, то есть реализуется одна дополнительная функция в 4-LUT, всего две функции от переменных a, b, c, d . В этом случае к каждому входу настройки основного элемента требуется подключить дополнительные 1-LUT, всего их будет 16.

Получим номер входа настройки дополнительной функции с учетом того, что инвертируется старшая переменная a . Видим, что в этом случае две половины вектора конфигурационных бит меняются местами. То есть, основная половина дерева реализует первую функцию, а вторая половина реализует вторую, дополнительную функцию. Например, на наборе переменных 0000 подключается конфигурационный бит основной функции – по активной половине дерева со старшей переменной $a=0$, а по неактивной со старшей переменной $a=1$ – конфигурационный бит дополнительной функции.

Таблица 2.1 – Доказательство правила сложения по модулю 2^n для подключения конфигурационных входов к дополнительным входным LUT

a	b	c	d	№ входа Исх. функции	№ входа Доп. функции	№ входа Исх. функции + 8 по модулю 16
0	0	0	0	0	8	$(0+8) \bmod 16 = 8$
0	0	0	1	1	9	$(1+8) \bmod 16 = 9$
0	0	1	0	2	10	$(2+8) \bmod 16 = 10$
0	0	1	1	3	11	$(3+8) \bmod 16 = 11$
0	1	0	0	4	12	$(4+8) \bmod 16 = 12$
0	1	0	1	5	13	$(5+8) \bmod 16 = 13$
0	1	1	0	6	14	$(6+8) \bmod 16 = 14$
0	1	1	1	7	15	$(7+8) \bmod 16 = 15$
1	0	0	0	8	0	$(8+8) \bmod 16 = 0$
1	0	0	1	9	1	$(9+8) \bmod 16 = 1$
1	0	1	0	10	2	$(10+8) \bmod 16 = 2$
1	0	1	1	11	3	$(11+8) \bmod 16 = 3$

1	1	0	0	12	4	$(12+8) \bmod 16 = 4$
1	1	0	1	13	5	$(13+8) \bmod 16 = 5$
1	1	1	0	14	6	$(14+8) \bmod 16 = 6$
1	1	1	1	15	7	$(15+8) \bmod 16 = 7$

Очевидно, что номер входа дополнительной функции определяется суммой номера исходной с числом 8 по модулю 16, например, для №15 имеем $(15+8) \bmod 16 = 7$. Для №0 имеем $(0+8) \bmod 16 = 8$.

В случае $\nu = 2$, необходимо выполнять суммирование исходного номера дополнительной функции с номером 1, 2, 3 соответствующими числами $2^{n-1}, 2^{n-2}, 2^{n-1} + 2^{n-2}$ по модулю 2^n .

В общем случае для $2^\nu - 1$ дополнительной функции выполняется суммирование по модулю 2^n с соответствующими числами $B = \{\{2^{n-1}\}, \{2^{n-2}\}, \dots, \{2^{n-\nu}\}, \dots, \{2^{n-1} + 2^{n-2} + \dots + 2^{n-\nu}\}\}$, где B – Boolean номеров без пустого множества, соответствующего исходной функции:

$$N_{rez} = (N_{in} + N_i) \bmod 2^n, i \in B; \quad (2.14)$$

$$B = \{\{2^{n-1}\}, \{2^{n-2}\}, \dots, \{2^{n-\nu}\}, \dots, \{2^{n-1} + 2^{n-2} + \dots + 2^{n-\nu}\}\}.$$

Правило циклического сдвига для подключения поддеревьев к дополнительным выходным $(2^\nu - 1) - LUT$

Для подключения поддеревьев к дополнительным выходным $(2^\nu - 1) - LUT$ необходимо выполнить циклический сдвиг номеров поддеревьев.

Пусть $\nu = 1$, тогда имеются номера 0 (основная функция – выход первой половины поддерева) и 1 (дополнительная функция – выход второй половины поддерева). Таким образом последний 1-LUT с двумя входами поддеревьев 0,1, поэтому дополнительный 1-LUT имеет входы поддеревьев 1,0.

Пусть $\nu = 2$, тогда имеются номера 0 (основная функция – выход первой половины поддерева) и 1, 2, 3 (дополнительные функции – выходы 1, 2, 3 четвертой поддерева).

Тогда подключение дополнительных 2–LUT может быть описано в таблице 2.2.

Таблица 2.2 – Подключение дополнительных 2–LUT

0	1	2	3	Исходный 2–LUT, №0
1	2	3	0	Дополнительный 2– LUT №1
2	3	0	1	Дополнительный 2– LUT №2
3	0	1	2	Дополнительный 2– LUT №3

Таким образом, номер подключаемого входа равен номеру входа исходного LUT + по модулю 3 номер дополнительного LUT.

То есть, в общем случае получаем: номер подключаемого входа равен номеру входа исходного LUT + по модулю $2^v - 1$ номер дополнительного v –LUT:

$$N_{rez.LUT} = (N_{in.LUT} + N_{add.v-LUT}) \bmod (2^v - 1). \quad (2.15)$$

Можно сделать вывод, что и первое правило интерпретируется циклическим сдвигом, но не одного бита, а групп из 2^{n-v} бит. Например, для таблицы 2.1 получаем таблицу 2.3:

Таблица 2.3 – Циклический сдвиг групп из 8 бит настройки для таблицы 2.1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7

Для $v=2$ получим:

Таблица 2.4 – Циклический сдвиг групп из 4 бит

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----

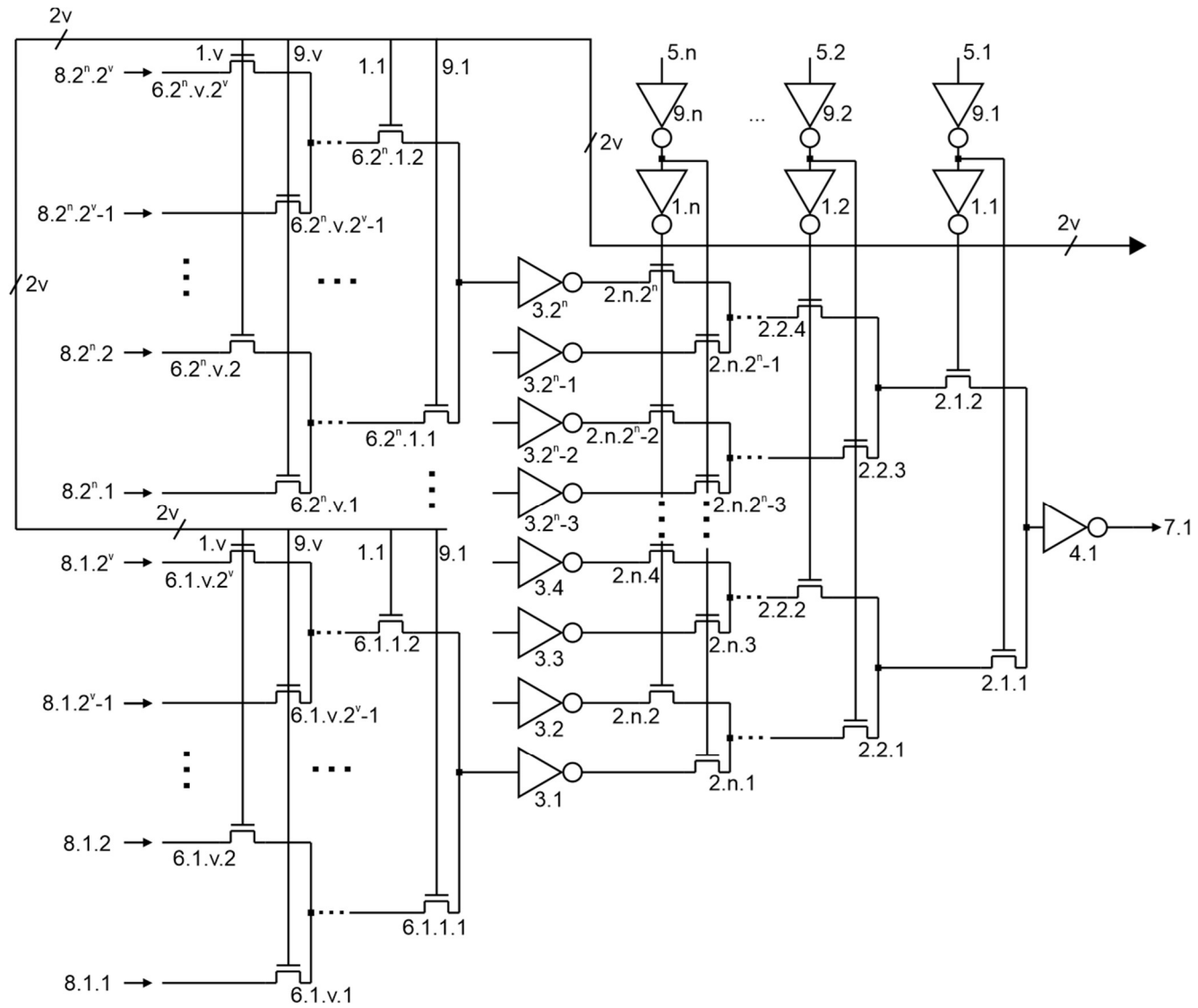
4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3
8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11

Таким образом, предлагаемый метод реализации в одном n -LUT нескольких функций одновременно содержит следующие действия.

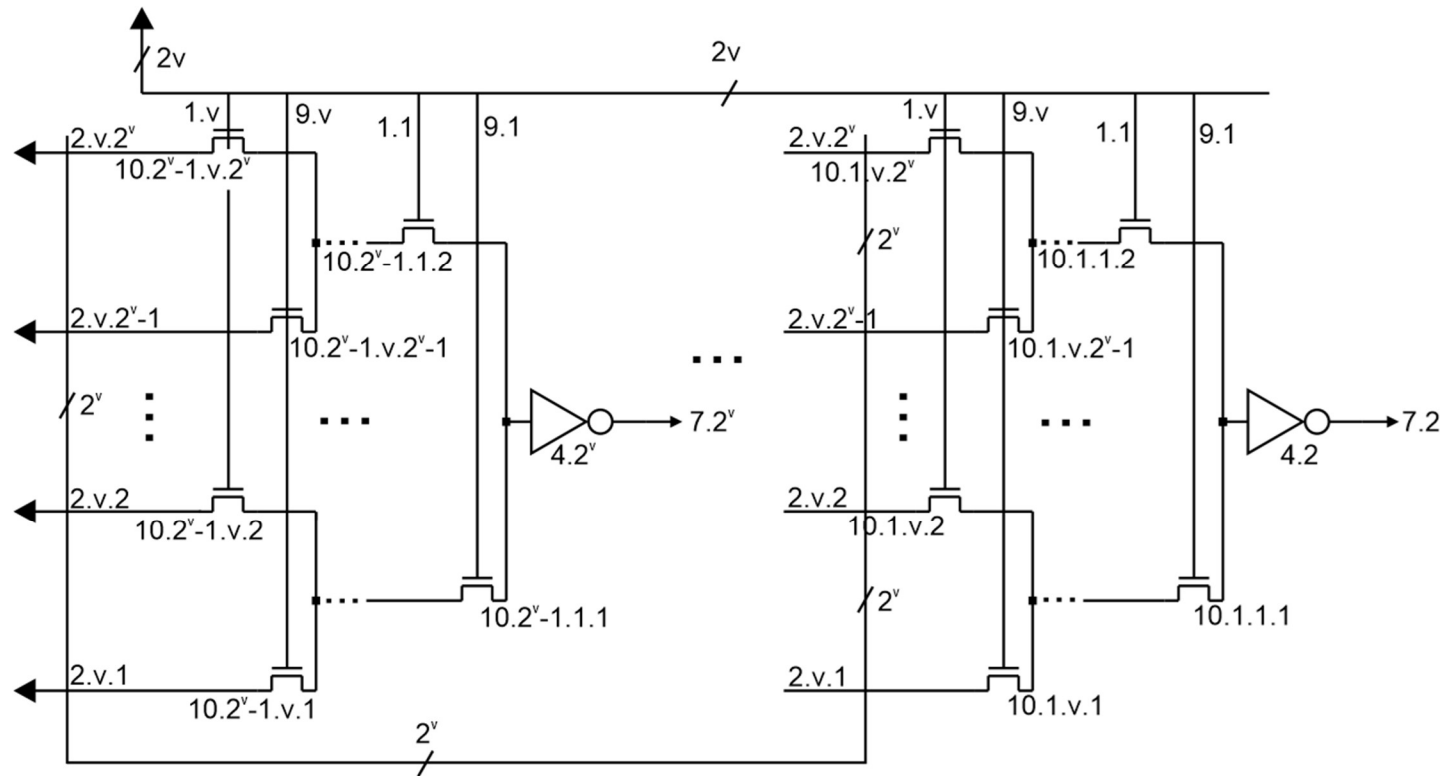
1. Задание параметров и варианта набора базовых блоков.
2. Синтез n -LUT из базовых блоков с определением корней поддеревьев.
3. Определение номеров дополнительных конфигурационных входов (правило 1).
4. Определение номеров корней поддеревьев для подключения дополнительных LUT (правило 2).
5. Проверка корректности путей одновременной реализации функций.

2.3. Разработка схемы электрической функциональной n -LUT, реализующего одновременно 2^v функций от одних и тех же переменных

В соответствии с разработанным методом реализуется схема электрическая функциональная (рисунок 2.1) n -LUT с вычислением 2^v функций ($v=1..n-1$). В общем случае для каждого n -LUT дополнительно подключаются $2^v - 1$ выходов устройства, состоящие из v каскадов передающих транзисторов и $2^v - 1$ инверторов. Подключение осуществляется к v каскаду основного LUT. Для управления данными памяти SRAM дополнительно подключаются v каскадов передающих транзисторов настройки, которые управляются v старшими переменными согласно циклическому сдвигу группы. Дополнительные v каскадов передающих транзисторов управляются сигналами переменных таким образом, чтобы использовать неактивные части основного дерева передающих транзисторов [23,94].



a)



б)

Рисунок 2.1 – Схема электрическая функциональная LUT, реализующего одновременно 2^v функций от одних и тех же n переменных а) основное дерево LUT с транзисторами настройки; б) дополнительные деревья LUT, реализующие многофункциональность

Схема электрическая функциональная содержит:

- выход базовой функции 7.1, выходы дополнительных функций 7.2...7.2^v;
- выход базового инвертора 4.1, выходы инверторов 4.2...4.2^v;
- первую и вторую группы инвертеров переменных 1.1...1.*n*, 9.1...9.*n*;
- *n* каскадов транзисторов базового LUT 2.1...2.*n* по 2^{*i*} транзисторов в *i*-м каскаде;
- инверторы настройки 3.1...3.2^{*n*};
- входные сигналы *n* переменных 5.1...5.*n*;
- передающие транзисторы входов настройки 6.1...6.2^{*n*};
- входные сигналы настройки 8.1...8.2^{*n*};
- дополнительные группы передающих транзисторов 10.1...10.(2^{*v*}–1);

2.4. Алгоритм подключения дополнительных транзисторов для реализации многофункционального элемента *n*–LUT

Реализованный алгоритм обеспечивает синтез логического элемента LUT для определенного количества переменных, вычисляющий базовую логическую функцию и подключение дополнительных транзисторов для одновременного вычисления добавочных функций [33,46].

Входными параметрами алгоритма являются: количество переменных *n* и количество логических функций *m*, реализуемых в одном LUT одновременно. Количество реализуемых функций зависит от *n* как 2^{*v*} (*v*=1...*n*–1).

Общая схема алгоритма приведена на рисунке 2.2.

При корректности введенных данных синтезируется базовое дерево LUT на *n* переменных из базового элемента 1–LUT с обозначением индексов транзисторов, соответствующих их каскадному расположению.

Следующим этапом происходит синтез дополнительных LUT для реализации добавочных функций, индексы транзисторов в которых обозначают подключение

истоков данных транзисторов к истокам транзисторов базового LUT с соответствующим индексом.

Синтез каскадов транзисторов настройки осуществляется следующим этапом и зависит как от количества используемых входных переменных, так и от количества реализуемых функций, подключение транзисторов которых осуществляется к входным инверторам.

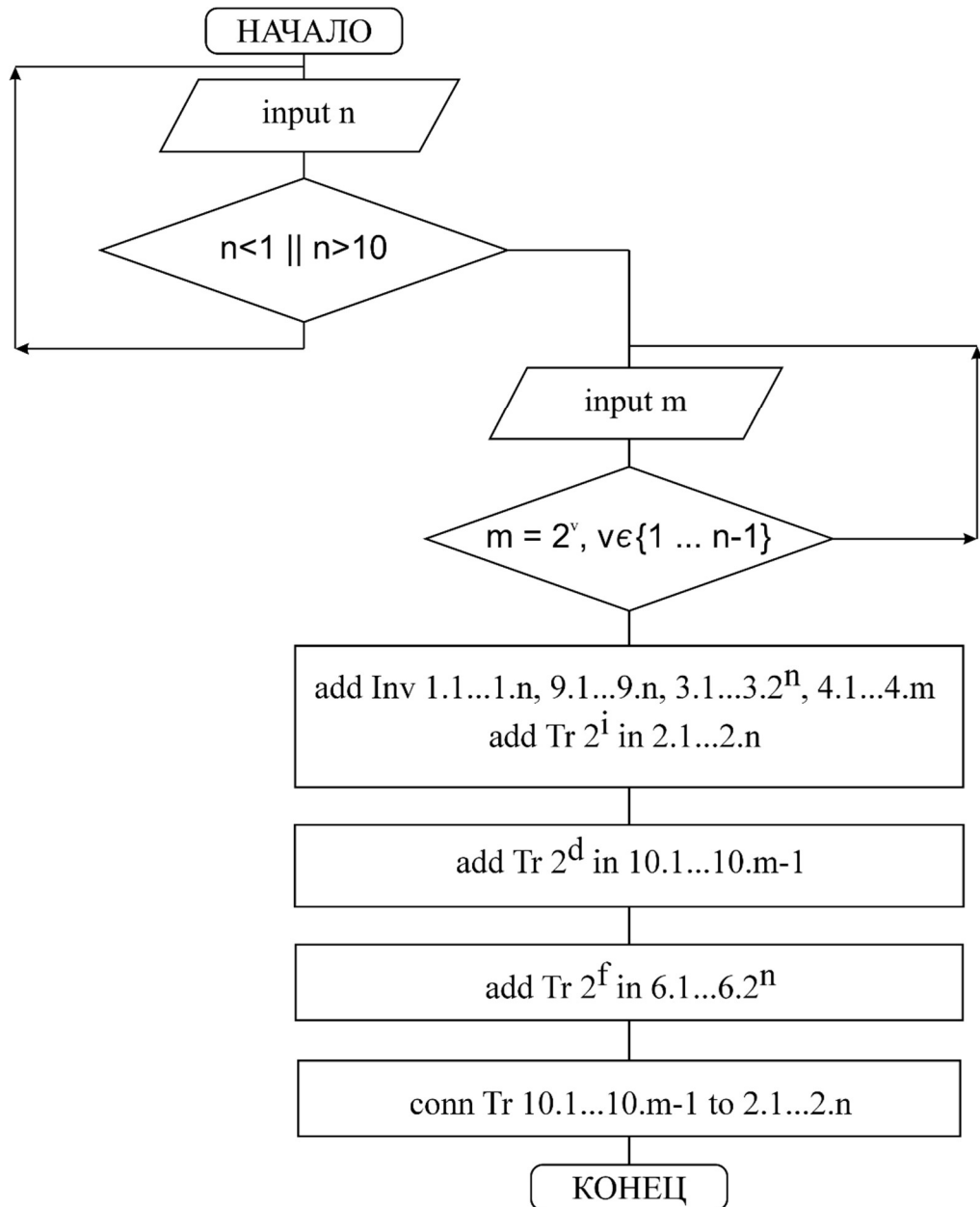


Рисунок 2.2 – Алгоритм подключения дополнительных транзисторов

Подключение к каждому стандартному LUT к v каскаду 2^{v-1} деревьев LUT v переменных, в которых подключение затворов передающих транзисторов к

управляющей переменной в каскаде v инвертировано в каждом четном LUT v переменных. И в каждом следующем $v-1$ каскаде (от v до 1) подключение затворов передающих транзисторов осуществляется к $v-1$ переменной инверсно каждой четной паре групп транзисторов, пара групп транзисторов понимается как объединение нечетной и четной группы транзисторов в v каскаде.

Подключение дерева LUT v переменных к каждому из 2^n инверторов настройки, в которых имеется v каскадов. Затворы транзисторов v каскада подключены к v переменной, причем затвор каждого нечетного транзистора в каскаде подключен к инвертированному значению переменной в четных $1/2^v$ частях каскада, а в нечетных $1/2^v$ частях каскада к не инвертированному значению переменной. Затворы транзисторов $v-1$ каскада подключаются к $v-1$ переменной, причем затвор каждого четного транзистора в каскаде подключен к не инвертированному значению переменной в четных $1/2^v$ частях каскада, а в нечетных $1/2^v$ частях каскада к инвертированному значению переменной.

Пример применения метода для разработки 3-LUT, реализующего две функции одновременно

Базовым блоком является 1-LUT состоящий из двух транзисторов, двух входов переменной, двух сигналов настройки и одним выходом. Для синтеза LUT на три переменные к каждому каскаду транзисторов добавляется 2^n базовых блоков выходы которых подключаются к входам предыдущего каскада. Каскады нумеруются от старшей переменной к более младшей или справа на лево. Дополнительные конфигурационные входы подключаются к последнему каскаду через инверторы настройки также в форме каскадов в зависимости от количества реализуемых функций. Например, для реализации двух функций ($v = 1$) требуется v каскадов дополнительных транзисторов настройки. Определение поддеревьев подключения также определяется количеством реализуемых функций, для реализации двух функций подключение осуществляется к первому каскаду ($v = 1$). Для проверки корректности путей одновременной реализации функций необходимо

провести сигналы по включенным и отключенным транзисторам базового дерева для отслеживания отсутствия пересечений и корректности работы двух половин базового дерева LUT.

Пусть $f_j, j=0...2^n-1$ – настройка логической функции по одному из 2^n входов настройки, осуществляемая путём настройки ПЛИС записью файла в конфигурационную оперативную память SRAM (рисунок 2.2).

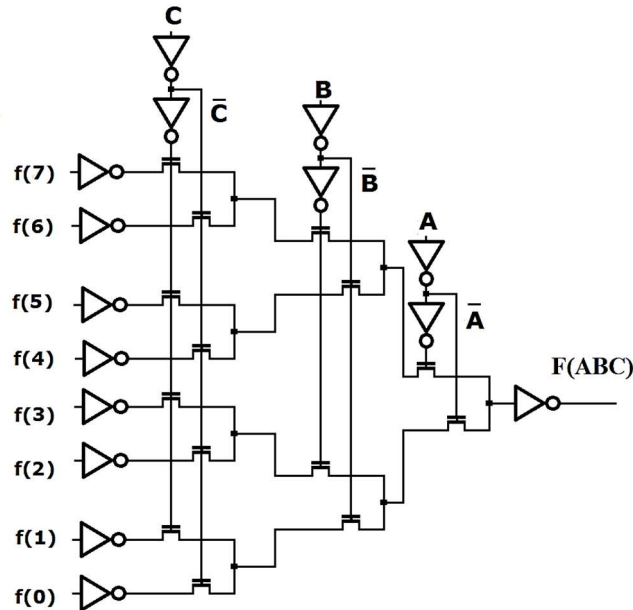


Рисунок 2.2 – 3–LUT, реализующий одну функцию трех переменных

Выражение для 3–LUT определяется следующим образом:

$$\begin{aligned}
 F(ABCf) = & f_{1.0} \cdot \overline{ABC} \vee f_{1.1} \cdot \overline{A}BC \vee f_{1.2} \cdot \overline{A}\overline{B}C \vee \\
 & f_{1.3} \cdot \overline{A}B\overline{C} \vee f_{1.4} \cdot \overline{A}B\overline{C} \vee f_{1.5} \cdot \overline{A}BC \vee f_{1.6} \cdot \overline{A}BC \vee \\
 & f_{1.7} \cdot ABC.
 \end{aligned} \tag{2.16}$$

При реализации одной функции в LUT вторая половина дерева передающих транзисторов в количестве 2^{n-1} неактивны, таким образом для любого из восьми значений на входах SRAM ($f(0)$ – $f(7)$) включается только одна цепочка передающих транзисторов, а на оставшихся передающих транзисторах, как минимум, одна цепочка полностью неактивна. На каждой неактивной цепочке может быть активированная другая логическая функция тех же аргументов, например функции

суммирования или переноса. Комбинируя эти цепочки по ИЛИ, мы можем получить логический элемент с несколькими выходами.

В предлагаемом 3-LUT с одновременным вычислением двух функций используется то же дерево передающих транзисторов с дополнительными сигналами настройки, которые используются по второй, неактивной в данном такте, половине дерева по значению старшей переменной:

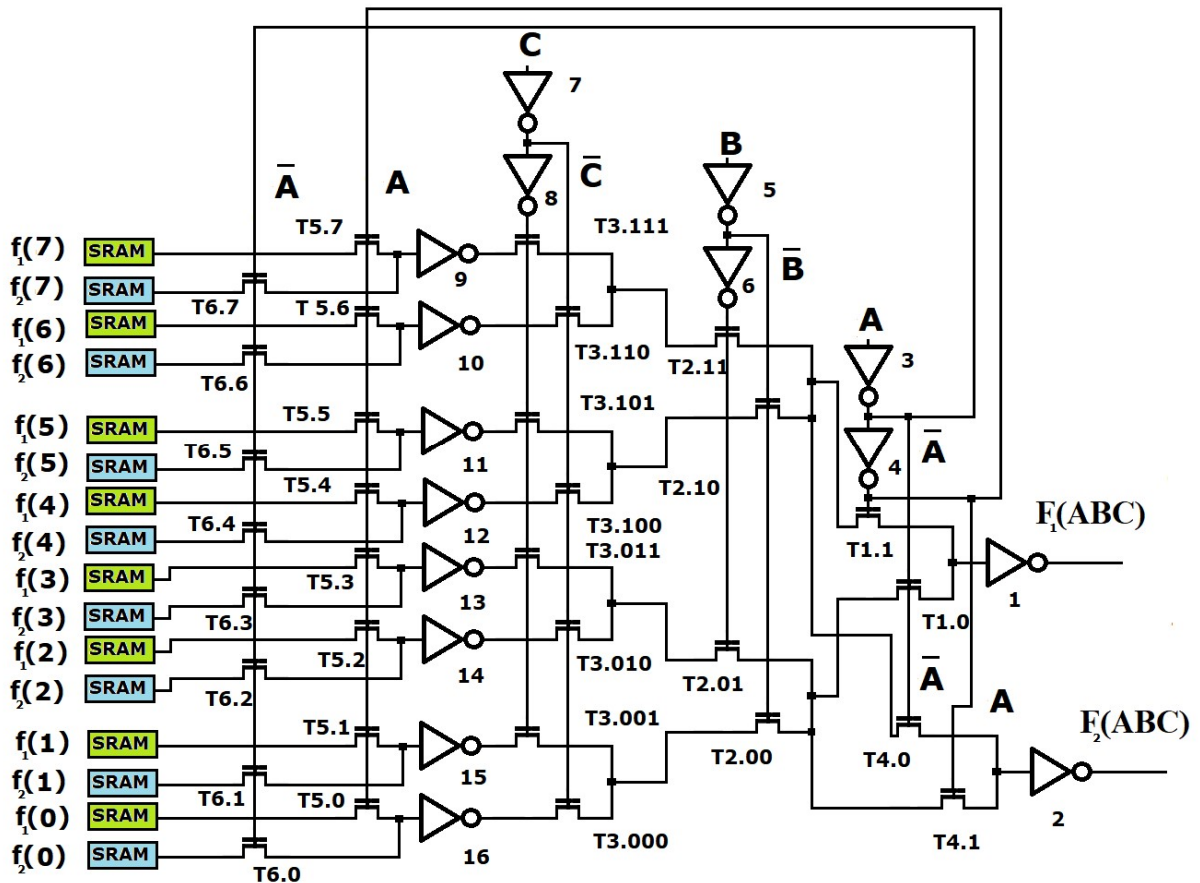


Рисунок 2.3 – 3-LUT, реализующий две функции трех переменных

Выход $F_2(ABC)$ с помощью двух дополнительных $T_{4.0}, T_{4.1}$ транзисторов и инвертора №2, реализующего дополнительный 1-LUT, подключают неактивную $n-1$ половину дерева передающих транзисторов на выход, при этом к этой половине подключается настройка второй функции. Выход $F_1(ABC)$ в это же время работает на второй половине дерева передающих транзисторов. Выражение для 3-LUT, реализующего две функции одновременно, определяется следующим образом:

$$\begin{aligned}
 F_1(ABCf) &= \bar{A}[f_{1.0} \cdot \overline{BC} \vee f_{1.1} \cdot \overline{BC} \vee f_{1.2} \cdot \overline{BC} \vee \\
 & f_{1.3} \cdot BC] \vee A[f_{1.4} \cdot \overline{BC} \vee f_{1.5} \cdot \overline{BC} \vee f_{1.6} \cdot \overline{BC} \vee \\
 & f_{1.7} \cdot BC]; \\
 F_2(ABCf) &= A[f_{2.4} \cdot \overline{BC} \vee f_{2.5} \cdot \overline{BC} \vee f_{2.6} \cdot \overline{BC} \vee \\
 & f_{2.7} \cdot BC] \vee \bar{A}[f_{2.0} \cdot \overline{BC} \vee f_{2.1} \cdot \overline{BC} \vee f_{2.2} \cdot \overline{BC} \vee \\
 & f_{2.3} \cdot BC].
 \end{aligned} \tag{2.17}$$

В данном случае увеличивается количество используемой статической памяти, поскольку реализация дополнительных двух функций требует в два раза больше статической памяти. При этом для корректного использования вычисления логических функций данные статической памяти сдвигаются при работе в параллельном режиме двух половин базового дерева.

2.5. Пример применения метода для разработки 3–LUT, реализующего четыре функции одновременно

Рассмотрим предлагаемое совмещение четырех функций в 3–LUT от одних и тех же переменных. Согласно схеме, изображенной на рисунке 2.3 при реализации двух функций по переменным В и С, также остаются четверти неактивных передающих транзисторов. Поэтому возможна реализация еще двух дополнительных функций, всего $2^v, v=2$ одновременно реализуемых функций. Для этого дополнительно подключается три дерева 2–LUT к переменной В, в которых транзисторы первого каскада управляются переменной В, а транзисторы второго каскада управляются переменной А. Также необходимо подключение дополнительных транзисторов настройки под управлением переменных В и А. В результате общая схема 3–LUT, реализующего четыре функции одновременно определяется следующим образом:

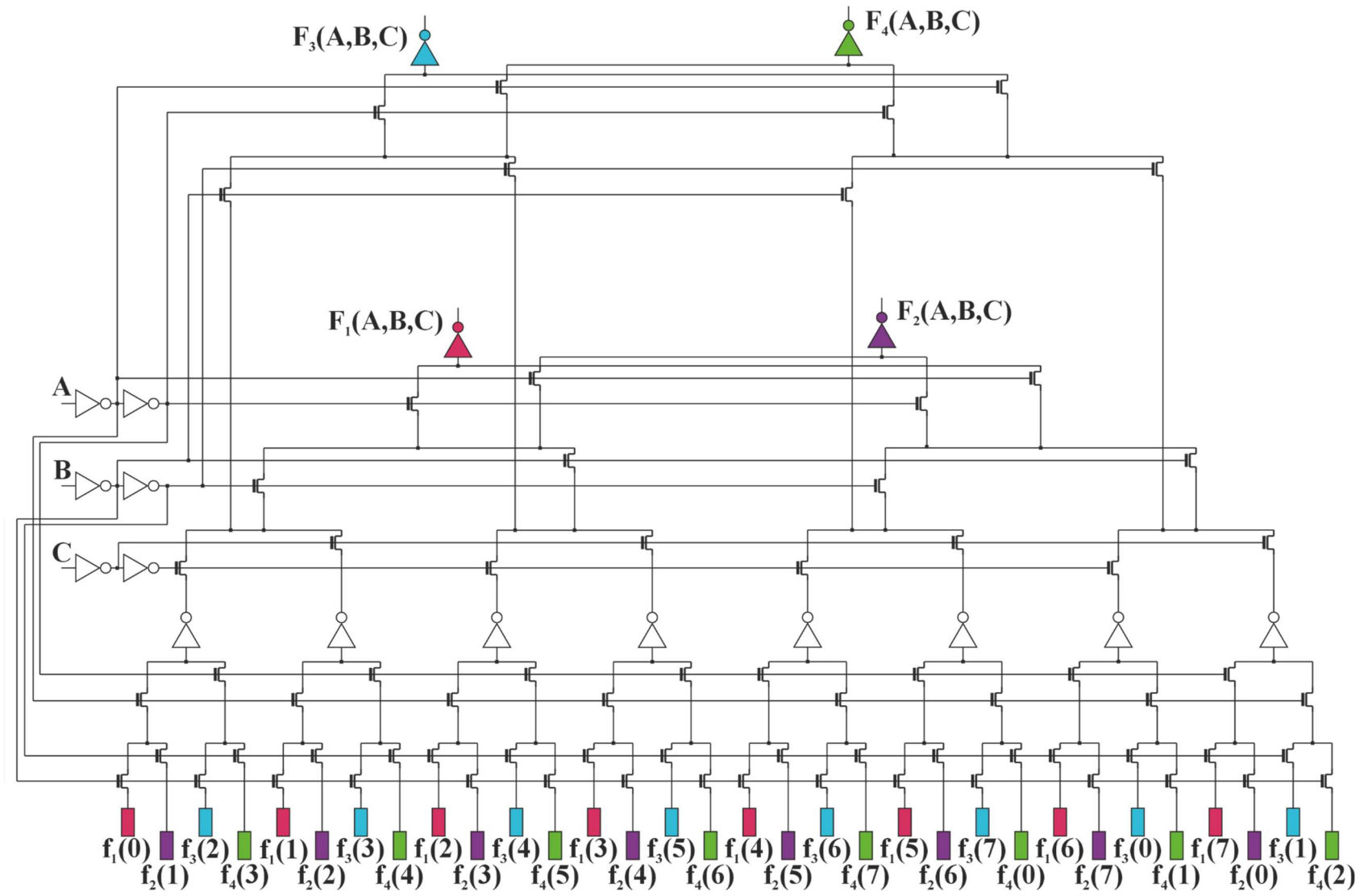


Рисунок 2.4 – 3-LUT, реализующего четыре функции трех переменных

Выражения для 3–LUT, реализующего четыре функции одновременно, определяется следующим образом:

$$\begin{aligned}
 F_1(ABCf) &= \overline{AB}[f_{1,0} \cdot \overline{C} \vee f_{1,1} \cdot C] \vee \\
 &\vee \overline{AB}[f_{1,2} \cdot \overline{C} \vee f_{1,3} \cdot C] \vee \\
 &\vee \overline{AB}[f_{1,4} \cdot \overline{C} \vee f_{1,5} \cdot C] \vee \\
 &\vee AB[f_{1,6} \cdot \overline{C} \vee f_{1,7} \cdot C],
 \end{aligned} \tag{2.18}$$

$$\begin{aligned}
 F_2(ABCf) &= \overline{AB}[f_{2,2(0)} \cdot \overline{C} \vee f_{2,3(1)} \cdot C] \vee \\
 &\vee \overline{AB}[f_{2,4(2)} \cdot \overline{C} \vee f_{2,5(3)} \cdot C] \vee \\
 &\vee AB[f_{2,6(4)} \cdot \overline{C} \vee f_{2,7(5)} \cdot C] \vee \\
 &\vee \overline{AB}[f_{2,0(6)} \cdot \overline{C} \vee f_{2,1(7)} \cdot C],
 \end{aligned} \tag{2.19}$$

$$\begin{aligned}
 F_3(ABCf) &= \overline{AB}[f_{3,4(0)} \cdot \overline{C} \vee f_{3,5(1)} \cdot C] \vee \\
 &\vee AB[f_{3,6(2)} \cdot \overline{C} \vee f_{3,7(3)} \cdot C] \vee \\
 &\vee \overline{AB}[f_{3,0(4)} \cdot \overline{C} \vee f_{3,1(5)} \cdot C] \vee \\
 &\vee \overline{AB}[f_{3,2(6)} \cdot \overline{C} \vee f_{3,3(7)} \cdot C],
 \end{aligned} \tag{2.20}$$

$$\begin{aligned}
 F_4(ABCf) &= AB[f_{4,6(0)} \cdot \overline{C} \vee f_{4,7(1)} \cdot C] \vee \\
 &\vee \overline{AB}[f_{4,0(2)} \cdot \overline{C} \vee f_{4,1(3)} \cdot C] \vee \\
 &\vee \overline{AB}[f_{4,2(4)} \cdot \overline{C} \vee f_{4,3(5)} \cdot C] \vee \\
 &\vee \overline{AB}[f_{4,4(6)} \cdot \overline{C} \vee f_{4,5(7)} \cdot C].
 \end{aligned} \tag{2.21}$$

Таким образом, каждому набору переменных A, B соответствует своя настройка. В выражениях (2.18) – (2.21) номер настройки по отношению к первой функции определяется как номер набора старших переменных A, B (0, 1, 2 или 3) плюс номер группы (1, 2, 3: $2^v - 1; v=2$) по модулю 2^v . Далее выполняется конкатенация с младшей переменной C и получается номер подключения (указан в скобках). Поэтому для реализации четырех функций устанавливаются сигналы из комбинированных частей таблиц истинности дополнительных функций.

2.6. Пример применения метода для разработки 4-LUT, реализующего две функции одновременно

Для реализации в LUT четырех переменных двух функций, аналогично рисунку 2.3 вводится 1-LUT по старшей переменной с дополнительными транзисторами настройки, управление которых осуществляется также старшей переменной. Предлагаемая схема 4-LUT, реализующего две функции одновременно представлена на рисунке 2.5:

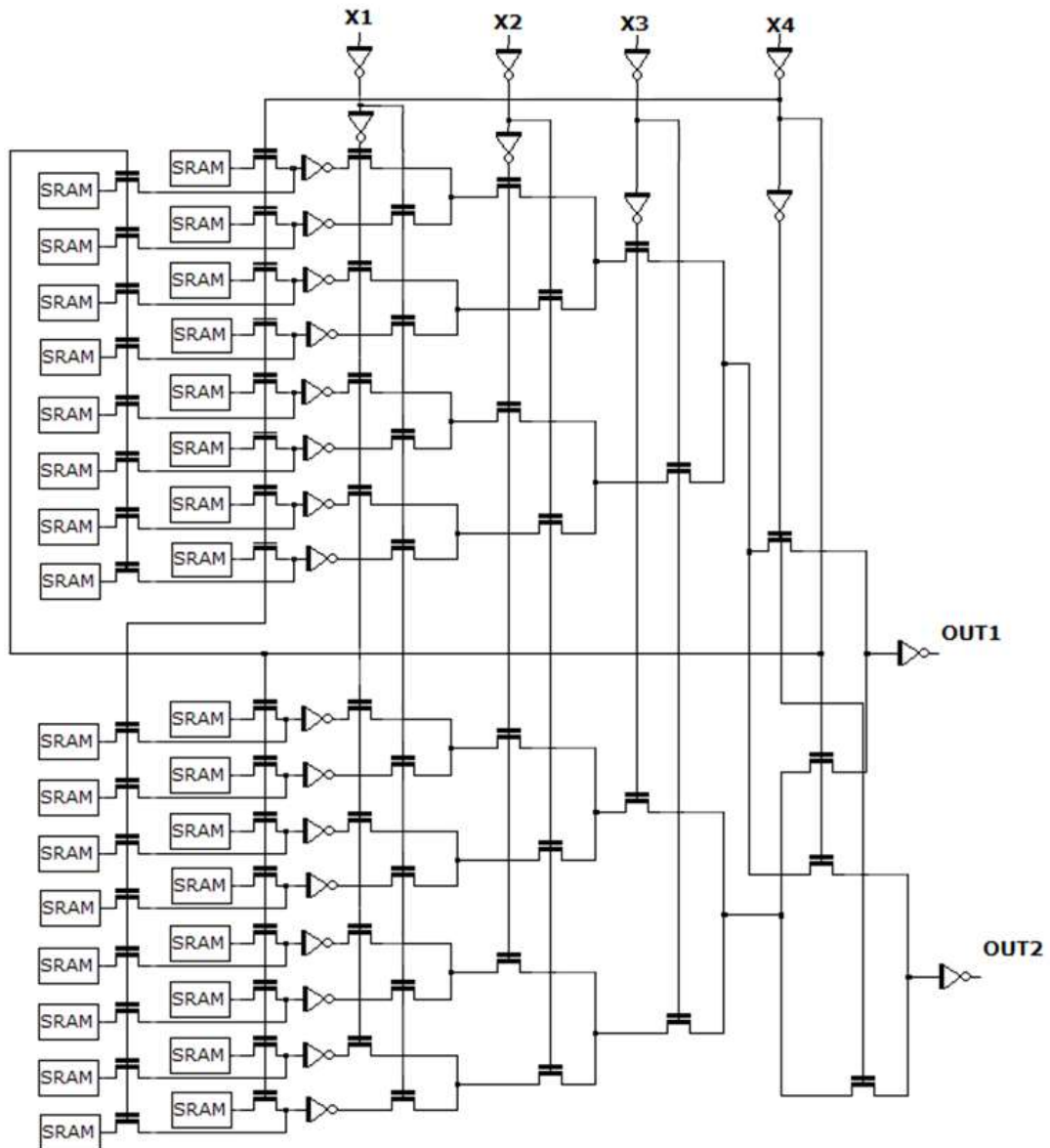


Рисунок 2.5 – 4-LUT, реализующего две функции одновременно

Предложенный подход можно дополнительно использовать для контроля вычислений, например, в случае работы в условиях воздействия тяжелых заряженных частиц (ТЗЧ), тогда:

$$\begin{aligned} \bar{z}_{1.out.x^n} &= \bigvee_{i=1}^{2^{n-1}} \left(\bigwedge_{j=1}^{n-1} x_j^{\sigma(i-1,j)} \wedge \bar{d}_{1,i} \right) \wedge t(x_n) \vee \bigvee_{i=2^{n-1}}^{2^n} \left(\bigwedge_{j=1}^{n-1} x_j^{\sigma(i-1,j)} \wedge \bar{d}_{1,i} \right) \wedge \bar{t}(x_n), \\ \bar{z}_{2.out.(x^*)^n} &= \bigvee_{i=1}^{2^{n-1}} \left(\bigwedge_{j=1}^{n-1} x_j^{\sigma(i-1,j)} \right) \wedge \bar{d}_{2,i} \wedge \bar{t}(x_n) \vee \bigvee_{i=2^{n-1}}^{2^n} \left(\bigwedge_{j=1}^{n-1} x_j^{\sigma(i-1,j)} \right) \wedge \bar{d}_{2,i} \wedge t(x_n), \\ f &= \bar{z}_{1.out.x^n} \oplus \bar{z}_{2.out.(x^*)^n}, \end{aligned} \quad (2.22)$$

где f – сигнал ошибки. Реализованная схема вычисления логической функции с контролем представлена на рисунке 2.6:

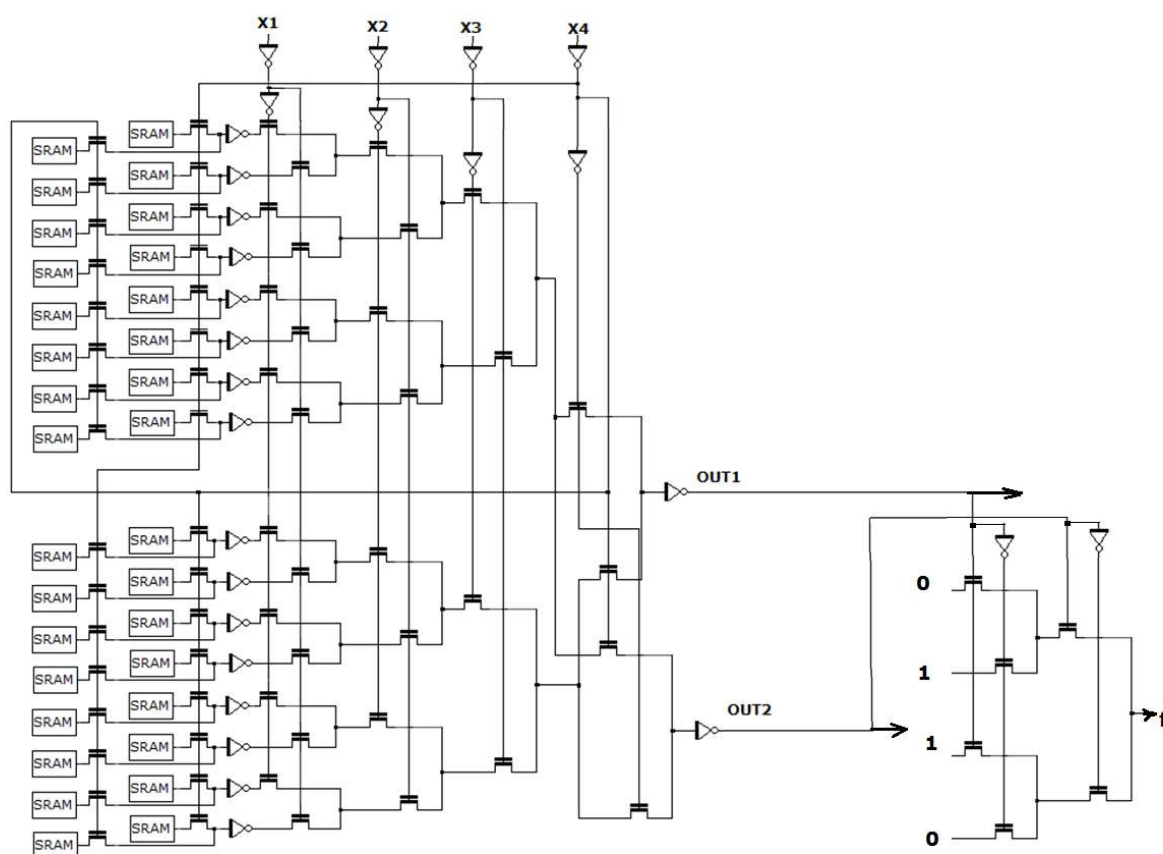


Рисунок 2.6 – 4-LUT, реализующего две функции одновременно с возможностью контроля корректности вычислений

На рисунке 2.6 схема сложения по модулю два (XOR) построена как LUT на две переменные – OUT1, OUT2 (LUT2) с заданием таблицы истинности 0110.

Контроль может быть использован в процессе предварительного тестирования ПЛИС. В дальнейшем предложенная схема может вычислять две функции одновременно. В высоконадёжных приложениях контроль может быть использован в процессе вычислений. Отказоустойчивая реализация с учетом имеет вид:

$$\begin{aligned}
\overline{Z_{1.out.x}^{\sigma_n}} &= \bigvee_{i=1}^{2^{n-1}} \left(\bigwedge_{j=1}^{n-1} \left[\bigvee_{\chi=1}^{r+1} (\& x_{i,\chi}) \right]_j^{\sigma(i-1,j)} \wedge \overline{\bigvee_{\chi=1}^{r+1} \& d(x_n)_{i,\chi}}_{1,i} \right) \wedge \overline{\bigvee_{\chi=1}^{r+1} (\& t(x_n)_{i,\chi})} \\
\bigvee_{i=2^{n-1}}^{2^n} \left(\bigwedge_{j=1}^{n-1} \left[\bigvee_{\chi=1}^{r+1} (\& x_{i,\chi}) \right]_j^{\sigma(i-1,j)} \wedge \overline{\bigvee_{\chi=1}^{r+1} \& d(x_n)_{i,\chi}}_{1,i} \right) &\overline{\bigvee_{\chi=1}^{r+1} (\& t(x_n)_{i,\chi})}, \\
\overline{Z_{2.out.(x^*)}^{\sigma_n}} &= \bigvee_{i=1}^{2^{n-1}} \left(\bigwedge_{j=1}^{n-1} \left[\bigvee_{\chi=1}^{r+1} (\& x_{i,\chi}) \right]_j^{\sigma(i-1,j)} \right) \overline{\bigwedge_{i=2^{n-1}}^{2^n} \left[\bigvee_{\chi=1}^{r+1} \& d(x_n)_{i,\chi} \right]_{2,i}} \wedge \overline{\bigvee_{\chi=1}^{r+1} (\& t(x_n)_{i,\chi})} \vee \\
\bigvee_{i=2^{n-1}}^{2^n} \left(\bigwedge_{j=1}^{n-1} \left[\bigvee_{\chi=1}^{r+1} (\& x_{i,\chi}) \right]_j^{\sigma(i-1,j)} \right) &\overline{\bigwedge_{i=2^n}^{2^{n-1}} \left[\bigvee_{\chi=1}^{r+1} \& d(x_n)_{i,\chi} \right]_{2,i}} \wedge \overline{\bigvee_{\chi=1}^{r+1} (\& t(x_n)_{i,\chi})}, \\
f &= \overline{\bigvee_{\chi=1}^{r+1} (\& z_{\cdot,\chi})}_{1.out.x^{\sigma_n}} \oplus \overline{\bigvee_{\chi=1}^{r+1} (\& z_{\cdot,\chi})}_{2.out.(x^*)^{\sigma_n}}.
\end{aligned} \tag{2.23}$$

В случае использования (2.23), получаем дублированную схему с транзисторным резервированием в каждом канале и в схеме сравнения. В качестве варианта предлагается транзисторное резервирование только в схеме сравнения.

2.7. Выводы по главе 2

1. Разработанная модель и метод синтеза логического элемента LUT реализующего несколько функций при одной заданной конфигурации, реализует одновременно $2^v, v=2,3,\dots,n-1$ логических функций, что существенно повышает функциональные возможности известного элемента.

2. Полученные правила подключения транзисторов настройки для реализации многофункционального LUT и соответствующий алгоритм позволяют получать логический элемент на заданное количество дополнительных функций.

3. Многофункциональный элемент может быть включен в блоки элементов как дополнительный элемент либо он создается в процессе конфигурирования проекта пользователя в составе адаптивного логического модуля.

4. Возможности многофункциональных элементов необходимо учесть при модификации систем автоматизированного проектирования (САПР) для ПЛИС использующие эти новые элементы.

5. Для дальнейшего исследования работоспособности функциональных схем необходимо провести схемотехническое и топологическое моделирование.

ГЛАВА 3. РАЗРАБОТКА ЛОГИЧЕСКОГО ЭЛЕМЕНТА $N - LUT$, РЕАЛИЗУЮЩЕГО ЛОГИЧЕСКУЮ ФУНКЦИЮ И ДЕШИФРАЦИЮ НАБОРА ПЕРЕМЕННЫХ

3.1. Разработка моделей $n - LUT$, реализующих логическую функцию и дешифрацию набора переменных

3.1.1. Существующий элемент LUT с n уровневым деревом транзисторов

В LUT , вычисляющем значение заданной логической функции z от n аргументов x_i в совершенной дизъюнктивной нормальной форме (СДНФ) реализовано n -уровневое бинарное дерево передающих (ключевых) транзисторов.

В (3.1) всегда активируется только один путь от конфигурационного бита до выхода

$$z(d_0 \dots d_{2^n-1} x_n \dots x_1) = \left. \begin{array}{l} \overline{d_0 x_1} \\ \underline{d_1 x_1} \\ \cdot \quad \quad \quad \overline{x_{n-1}} \\ \cdot \quad \quad \quad \dots \quad \underline{x_{n-1}} \quad \overline{x_n} \\ \cdot \quad \quad \quad \overline{x_{n-1}} \quad \underline{x_n} \\ \underline{d_{2^n-2} x_1} \quad \underline{x_{n-1}} \\ \underline{d_{2^n-1} x_1} \end{array} \right\} (\vee \bullet). \quad (3.1)$$

Существующий реверс дерева позволяет выполнить дешифрацию входного набора:

$$\begin{array}{l} \overline{x_1} \Rightarrow \overline{x_n x_{n-1} \dots x_1} \\ \underline{x_1} \Rightarrow \underline{x_n x_{n-1} \dots x_1} \\ \overline{x_n} \quad \overline{x_{n-1}} \quad \cdot \\ (Ground) \quad \underline{x_{n-1}} \quad \cdot \\ \underline{x_n} \quad \overline{x_{n-1}} \quad \cdot \\ \underline{x_{n-1}} \quad \overline{x_1} \Rightarrow \overline{x_n x_{n-1} \dots x_1} \\ \underline{x_1} \Rightarrow \underline{x_n x_{n-1} \dots x_1} \end{array} \quad (3.2)$$

Корень является входом константы нуля, а выходы – это реализуемые конъюнкции входных переменных. Однако, для данного набора переменных активируется только одна цепочка, а остальные имеют разрыв, что приводит к неопределенному состоянию выходного инвертора, не указанного в выражении (3.3). Неопределенное состояние не допустимо, поэтому необходимо использование подтягивающего резистора на входе инвертора NOT_i ($i=1..2^n$), либо блоков конститuent нуля:

$$\begin{array}{c}
 \overline{x_1} \Rightarrow \frac{\overline{x_n \overline{x_{n-1} \dots \overline{x_1}}}}{\overline{(Ground)(x_n \overline{x_{n-1} \dots \overline{x_1}})}} (\vee \bullet) NOT_1 \\
 \overline{x_1} \Rightarrow \frac{\overline{x_n \overline{x_{n-1} \dots \overline{x_1}}}}{\overline{(Ground)(x_n \overline{x_{n-1} \dots \overline{x_1}})}} (\vee \bullet) NOT_2 \\
 \begin{array}{c}
 \overline{x_n} \overline{x_{n-1}} \quad \cdot \\
 (V_{cc}) \quad \overline{x_{n-1}} \quad \cdot \\
 \overline{x_n} \overline{x_{n-1}} \quad \cdot
 \end{array} \\
 \overline{x_{n-1}} \overline{x_1} \Rightarrow \frac{\overline{x_n \overline{x_{n-1} \dots \overline{x_1}}}}{\overline{(Ground)(x_n \overline{x_{n-1} \dots \overline{x_1}})}} (\vee \bullet) NOT_{2^{n-1}} \\
 \overline{x_1} \Rightarrow \frac{\overline{x_n \overline{x_{n-1} \dots \overline{x_1}}}}{\overline{(Ground)(x_n \overline{x_{n-1} \dots \overline{x_1}})}} (\vee \bullet) NOT_{2^n}
 \end{array} \quad (3.3)$$

В выражении (3.3) использована константа единицы (V_{cc}) на входе дерева, чтобы на выходе дешифратора были активные нули. Однако, в этом случае, реализация одной логической функции невозможна без дополнительных блоков дизъюнкции конститuent единицы, используемых для вычисления системы функций от одних и тех же переменных.

3.1.2. Разработка модели адаптивного элемента LUT с одноуровневым 2^n деревом (2^n ветвей по n транзисторов)

Для обеспечения возможности реализации одной заданной логической функции без дополнительных блоков дизъюнкции конститuent единицы предлагается адаптивный элемент. При этом, логическая функция реализуется параллельным соединением 2^n цепочек из n передающих транзисторов,

($\vee\bullet$) – корень дерева, реализующий операцию монтажного ИЛИ (wired OR):

$$z(d_0 \dots d_{2^n-1} x_1 \dots x_n) = \dots \left. \begin{array}{l} \frac{d_{2^n-1} x_1 \dots x_n}{\dots} \\ \frac{d_{2^n-2} x_1 \dots x_n}{\dots} \\ \frac{d_1 x_1 \dots x_n}{\dots} \\ \frac{d_0 x_1 \dots x_n}{\dots} \end{array} \right\} (\vee\bullet), \quad (3.3)$$

$d_j \in \{0,1\}$ – конфигурационный бит, значение функции в соответствующей строке таблицы истинности функции z .

В связи с ограничениями Мида–Конвей [79] на число последовательно соединенных передающих МОП транзисторов (≤ 4) используют базовые деревья для $n=1,2,3$:

$$z(d_0 d_1 x) = \frac{d_1 x}{d_0 x} \left\{ (\vee\bullet), \right.$$

$$z(d_0 d_1 d_2 d_3 x_2 x_1) = \frac{\frac{d_3 x_2 x_1}{\dots}}{\frac{d_2 x_2 x_1}{\dots}} \left\{ (\vee\bullet), \right. \quad (3.4)$$

$$z(d_0 d_1 d_2 d_3 d_4 d_5 d_6 d_7 x_3 x_2 x_1) = \frac{\frac{\frac{d_7 x_3 x_2 x_1}{\dots}}{\frac{d_6 x_3 x_2 x_1}{\dots}}}{\frac{d_5 x_3 x_2 x_1}{\dots}} \left\{ (\vee\bullet). \right.$$

В выражениях (3.4) старшая переменная записана слева. LUT на одну переменную используется для объединения LUT на две и три переменные, а также

и как инвертор ($z(10x_1) = \overline{x_1}$) и как повторитель ($z(01x_1) = x_1$), в случае необходимости трассировки через него некоторой связи.

Из (3.4) строятся деревья для реализации функций для $n=4$, в том числе и для $n=5,6,7$ и более, которые используются в адаптивных логических модулях (АЛМ). В этом случае в промежуточных деревьях вместо настройки d_j указываются связи c_j . Кроме того, с целью восстановления уровня сигнала используют так называемые восстановители v (в простейших случаях это инверторы, NOT Gates) по входам настройки или коммутации и по выходам функции, например, LUT на четыре переменные будет представлять собой выражение (3.5):

$$\left. \begin{array}{l} \overline{d_{15}x_3x_2x_1} \\ \overline{d_{14}x_3x_2x_1} \\ \overline{d_{13}x_3x_2x_1} \\ \overline{d_{12}x_3x_2x_1} \\ \overline{d_{11}x_3x_2x_1} \\ \overline{d_{10}x_3x_2x_1} \\ \overline{d_9x_3x_2x_1} \\ \overline{d_8x_3x_2x_1} \end{array} \right\} (\vee \bullet) = c_1, \left. \begin{array}{l} \overline{d_7x_3x_2x_1} \\ \overline{d_6x_3x_2x_1} \\ \overline{d_5x_3x_2x_1} \\ \overline{d_4x_3x_2x_1} \\ \overline{d_3x_3x_2x_1} \\ \overline{d_2x_3x_2x_1} \\ \overline{d_1x_3x_2x_1} \\ \overline{d_0x_3x_2x_1} \end{array} \right\} (\vee \bullet) = c_0, z(d_0d_1\dots d_{15}x_4) = \frac{c_1vx_4}{c_0v\overline{x_4}} \left. \right\} (\vee \bullet), \quad (3.5)$$

где v – это функции повторения (восстановители) поскольку их нечетное число в цепочках (три), поэтому настройка функции инверсная. Настройка LUT на одну переменную не используется и такое устройство называется просто мультиплексором 2–1. Так, функция четырех переменных $z(d_0d_1d_2d_3d_4d_5d_6d_7\dots d_{15}x_4x_3x_2x_1)$ из двух 3–LUT и одного 1–LUT в режиме мультиплексора 2–1 (вместо данных настройки $d_j; j=0,1$ используются выходы двух 3–LUT $c_i; i=1,2$).

Казалось бы, дешифрация текущего набора переменных здесь выполняется. Однако, выделить единственную активную конституенту $x_1^{\delta_{1j}}x_2^{\delta_{2j}}\dots x_n^{\delta_{nj}}$, $\delta_i, i=1\dots n$,

$j=0..2^n$, где δ_i – показатель инверсирования j -ой конstituенты, не представляется возможным, поскольку они все объединены монтажным ИЛИ. Кроме того, на выход элемента выдается настроечное значение d_j , а оно может быть разным: либо нулем, либо единицей, поэтому однозначно идентифицировать набор переменных не получится.

Поэтому адаптивный элемент работает в двух режимах: вычисления одной заданной логической функции (конфигурационная переменная режима работы $k=1$) и дешифрации набора переменных (конфигурационная переменная режима работы $k=0$):

$$z(kd_0\dots d_{2^n-1}x_n\dots x_1) = \left. \begin{array}{l} \frac{d_{2^n-1}k}{(Ground)k} (\vee\bullet)x_1\dots x_n (\vee\bullet) \frac{k \Rightarrow z(d_{2^n-1}x_1\dots x_n)(\vee\bullet)}{\bar{k} \Rightarrow j(x_1\dots x_n)} \\ \frac{d_{2^n-2}k}{(Ground)k} (\vee\bullet)\bar{x}_1\dots x_n (\vee\bullet) \frac{k \Rightarrow z(d_{2^n-2}\bar{x}_1\dots x_n)(\vee\bullet)}{\bar{k} \Rightarrow j(x_1\dots x_n)} \\ \dots \\ \frac{d_1k}{(Ground)k} (\vee\bullet)x_1\dots \bar{x}_n (\vee\bullet) \frac{k \Rightarrow z(d_1x_1\dots \bar{x}_n)(\vee\bullet)}{\bar{k} \Rightarrow j(x_1\dots \bar{x}_n)} \\ \frac{d_0k}{(Ground)k} (\vee\bullet)\bar{x}_1\dots \bar{x}_n (\vee\bullet) \frac{k \Rightarrow z(d_0\bar{x}_1\dots \bar{x}_n)(\vee\bullet)}{\bar{k} \Rightarrow j(x_1\dots \bar{x}_n)} \end{array} \right\} (\vee\bullet)(NOT), \quad (3.6)$$

где $j(x_1\dots x_n)$ означает номер активируемой конstituенты. Режим выбирается при загрузке конфигурационного файла. Однако, выражение (3.6) не учитывает, что при активировании одной конstituенты, сигнал (ноль в случае подключения шины «Ноль вольт», «Ground») будет на всех выходах, поскольку они соединены. Для исключения такого явления возможна установка диодов q :

$$\begin{aligned}
& \left. \begin{aligned}
& \frac{d_{2^n-1}k}{(Ground)k} = (\vee\bullet)x_1\dots x_n(\vee\bullet) \frac{k \Rightarrow z(d_{2^n-1}x_1\dots x_n)q_{2^n-1}(\vee\bullet)}{k \Rightarrow j(x_1\dots x_n)} \\
& \frac{d_{2^n-2}k}{(Ground)k} = (\vee\bullet)\bar{x}_1\dots x_n(\vee\bullet) \frac{k \Rightarrow z(d_{2^n-2}\bar{x}_1\dots x_n)q_{2^n-2}(\vee\bullet)}{k \Rightarrow j(x_1\dots x_n)} \\
& \dots \\
& \frac{d_1k}{(Ground)k} = (\vee\bullet)x_1\dots \bar{x}_n(\vee\bullet) \frac{k \Rightarrow z(d_1x_1\dots \bar{x}_n)q_1(\vee\bullet)}{k \Rightarrow j(x_1\dots \bar{x}_n)} \\
& \frac{d_0k}{(Ground)k} = (\vee\bullet)\bar{x}_1\dots \bar{x}_n(\vee\bullet) \frac{k \Rightarrow z(d_0\bar{x}_1\dots \bar{x}_n)q_0(\vee\bullet)}{k \Rightarrow j(\bar{x}_1\dots \bar{x}_n)}
\end{aligned} \right\} (\vee\bullet)(NOT). \quad (3.7)
\end{aligned}$$

$$z(kd_0\dots d_{2^n-1}x_n\dots x_1) = \dots$$

Такая реализация будет препятствовать прохождению нулевого значения функции, поэтому следующим решением будет установка подтягивающего к нулю резистора R_0 на входе элемента NOT :

$$\begin{aligned}
& \left. \begin{aligned}
& \frac{d_{2^n-1}k}{(Ground)k} = (\vee\bullet)x_1\dots x_n(\vee\bullet) \frac{k \Rightarrow z(d_{2^n-1}x_1\dots x_n)q_{2^n-1}(\vee\bullet)}{k \Rightarrow j(x_1\dots x_n)} \\
& \frac{d_{2^n-2}k}{(Ground)k} = (\vee\bullet)\bar{x}_1\dots x_n(\vee\bullet) \frac{k \Rightarrow z(d_{2^n-2}\bar{x}_1\dots x_n)q_{2^n-2}(\vee\bullet)}{k \Rightarrow j(\bar{x}_1\dots x_n)} \\
& \dots \\
& \frac{d_1k}{(Ground)k} = (\vee\bullet)x_1\dots \bar{x}_n(\vee\bullet) \frac{k \Rightarrow z(d_1x_1\dots \bar{x}_n)q_1(\vee\bullet)}{k \Rightarrow j(x_1\dots \bar{x}_n)} \\
& \frac{d_0k}{(Ground)k} = (\vee\bullet)\bar{x}_1\dots \bar{x}_n(\vee\bullet) \frac{k \Rightarrow z(d_0\bar{x}_1\dots \bar{x}_n)q_0(\vee\bullet)}{k \Rightarrow j(\bar{x}_1\dots \bar{x}_n)}
\end{aligned} \right\} (\vee\bullet)R_0(NOT). \quad (3.8)
\end{aligned}$$

$$z(kd_0\dots d_{2^n-1}x_n\dots x_1) = \dots$$

В данном случае происходит увеличение задержки и потребляемой мощности [20]. Кроме того, резистор занимает относительно большую площадь кристалла. Другим решением может быть использование отключающих транзисторов $t(k)$, управляемых сигналом k :

$$\begin{aligned}
& \left. \begin{aligned}
& \frac{d_{2^n-1}k}{(Ground)k} = (\vee\bullet)x_1\dots x_n(\vee\bullet) \frac{k \Rightarrow z(d_{2^n-1}x_1\dots x_n)t_{2^n-1}(k)(\vee\bullet)}{k \Rightarrow j(x_1\dots x_n)} \\
& \frac{d_{2^n-2}k}{(Ground)k} = (\vee\bullet)\bar{x}_1\dots x_n(\vee\bullet) \frac{k \Rightarrow z(d_{2^n-2}\bar{x}_1\dots x_n)t_{2^n-2}(k)(\vee\bullet)}{k \Rightarrow j(\bar{x}_1\dots x_n)} \\
& \dots \\
& \frac{d_1k}{(Ground)k} = (\vee\bullet)x_1\dots \bar{x}_n(\vee\bullet) \frac{k \Rightarrow z(d_1x_1\dots \bar{x}_n)t_1(k)(\vee\bullet)}{k \Rightarrow j(x_1\dots \bar{x}_n)} \\
& \frac{d_0k}{(Ground)k} = (\vee\bullet)\bar{x}_1\dots \bar{x}_n(\vee\bullet) \frac{k \Rightarrow z(d_0\bar{x}_1\dots \bar{x}_n)t_0(k)(\vee\bullet)}{k \Rightarrow j(\bar{x}_1\dots \bar{x}_n)}
\end{aligned} \right\} (\vee\bullet)(NOT). \quad (3.9)
\end{aligned}$$

$$z(kd_0\dots d_{2^n-1}x_n\dots x_1) = \dots$$

В любом случае, целесообразно реализовать вычисление логической функции и дешифрацию набора переменных одновременно.

3.1.3. Разработка модели LUT, выполняющего дешифрацию набора переменных одновременно с использованием неактивной половины дерева транзисторов

Для одновременного вычисления логической функции и дешифрации набора переменных, то есть для получения унитарного кода активного набора, необходимо по той же цепочке $x_1^{\delta_{1j}} x_2^{\delta_{2j}} \dots x_n^{\delta_{nj}}$ передавать как значение функции d_j , так и некоторую константу, например, ноль (допустим, подключать шину «Ноль Вольт», «Ground»).

Анализ выражений (3.3), (3.4) позволяет установить факт обязательного активирования (истинности) двух подцепочек длиной $n-1$ при учете того, что n – старшая переменная:

$$x_1^{\delta_{1j}} x_2^{\delta_{2j}} \dots x_n^{\delta_{nj}}, x_1^{\delta_{1j}} x_2^{\delta_{2j}} \dots x_n^{\overline{\delta_{nj}}},$$

$$x_1^{\delta_{1j}} x_2^{\delta_{2j}} \dots x_{n-1}^{\delta_{(n-1)j}} = x_1^{\delta_{1j}} x_2^{\delta_{2j}} \dots x_{n-1}^{\delta_{(n-1)j}} \quad (3.10)$$

То есть, всегда используется только половина дерева. Это позволяет передавать константу не по текущей цепочке $x_1^{\delta_{1j}} x_2^{\delta_{2j}} \dots x_n^{\delta_{nj}}$, а по ее «зеркальному отражению» $x_1^{\delta_{1j}} x_2^{\delta_{2j}} \dots x_n^{\overline{\delta_{nj}}}$, эта цепочка активна только за счет старшей переменной. Но тогда требуется отключать настройку d_j и подключать шину «Ноль Вольт» (Ground), что может быть выполнено той же старшей переменной:

$$z(d_0 \dots d_{2^n-1} x_1 \dots x_n) = \left. \begin{array}{l} \frac{d_{2^n-1} x_n}{(Ground)x_n} (\vee \bullet) x_1 \dots x_{n-1} (\vee \bullet) \frac{x_n \Rightarrow z(d_{2^n-1} x_1 \dots x_n) (\vee \bullet)}{x_n \Rightarrow j(x_1 \dots x_n)} \\ \frac{d_{2^n-2} x_n}{(Ground)x_n} (\vee \bullet) \bar{x}_1 \dots x_{n-1} (\vee \bullet) \frac{x_n \Rightarrow z(d_{2^n-2} \bar{x}_1 \dots x_n) (\vee \bullet)}{x_n \Rightarrow j(x_1 \dots x_n)} \\ \dots \\ \frac{d_1 \bar{x}_n}{(Ground)x_n} (\vee \bullet) x_1 \dots \bar{x}_{n-1} (\vee \bullet) \frac{\bar{x}_n \Rightarrow z(d_1 x_1 \dots \bar{x}_n) (\vee \bullet)}{\bar{k} \Rightarrow j(x_1 \dots x_n)} \\ \frac{d_0 \bar{x}_n}{(Ground)x_n} (\vee \bullet) \bar{x}_1 \dots \bar{x}_{n-1} (\vee \bullet) \frac{\bar{x}_n \Rightarrow z(d_0 \bar{x}_1 \dots \bar{x}_n) (\vee \bullet)}{x_n \Rightarrow j(x_1 \dots x_n)} \end{array} \right\} (\vee \bullet), \quad (3.11)$$

где $j(x_1 \dots \bar{x}_n)$ означает номер, получаемый при инверсировании старшей переменной. Так, например, для двух переменных получим следующие выражения:

$$\left. \begin{array}{l} \frac{d_3 x_2}{(Ground)x_2} (\vee \bullet) x_1 (\vee \bullet) \frac{x_2 (\vee \bullet) \Rightarrow z(d_3 x_2 x_1) (\vee \bullet)}{x_2 \Rightarrow "1"} \\ \frac{d_2 x_2}{(Ground)x_2} (\vee \bullet) \bar{x}_1 (\vee \bullet) \frac{x_2 (\vee \bullet) \Rightarrow z(d_2 x_2 \bar{x}_1) (\vee \bullet)}{x_2 \Rightarrow "0"} \\ \frac{d_1 \bar{x}_2}{(Ground)x_2} (\vee \bullet) x_1 (\vee \bullet) \frac{\bar{x}_2 (\vee \bullet) \Rightarrow z(d_1 \bar{x}_2 x_1) (\vee \bullet)}{x_2 \Rightarrow "3"} \\ \frac{d_0 \bar{x}_2}{(Ground)x_2} (\vee \bullet) \bar{x}_1 (\vee \bullet) \frac{\bar{x}_2 (\vee \bullet) \Rightarrow z(d_0 \bar{x}_2 \bar{x}_1) (\vee \bullet)}{x_2 \Rightarrow "2"} \end{array} \right\} (\vee \bullet). \quad (3.12)$$

Для трех переменных аналогично получаем следующие выражения:

$$\left. \begin{array}{l}
\frac{d_7 x_3}{(Ground)x_3} = (\vee \bullet) x_2 x_1 (\vee \bullet) \frac{x_3 \Rightarrow z(d_7 x_3 x_2 x_1)(\vee \bullet)}{\bar{x}_3 \Rightarrow "111b"} \\
\frac{d_6 x_3}{(Ground)x_3} = (\vee \bullet) x_2 \bar{x}_1 (\vee \bullet) \frac{x_3 \Rightarrow z(d_6 x_3 x_2 \bar{x}_1)(\vee \bullet)}{\bar{x}_3 \Rightarrow "110b"} \\
\frac{d_5 x_3}{(Ground)x_3} = (\vee \bullet) \bar{x}_2 x_1 (\vee \bullet) \frac{x_3 \Rightarrow z(d_5 x_3 \bar{x}_2 x_1)(\vee \bullet)}{\bar{x}_3 \Rightarrow "101b"} \\
\frac{d_4 x_3}{(Ground)x_3} = (\vee \bullet) \bar{\bar{x}}_2 \bar{x}_1 (\vee \bullet) \frac{x_3 \Rightarrow z(d_4 x_3 \bar{\bar{x}}_2 \bar{x}_1)(\vee \bullet)}{\bar{x}_3 \Rightarrow "100b"} \\
\frac{d_3 \bar{x}_3}{(Ground)x_3} = (\vee \bullet) x_2 x_1 (\vee \bullet) \frac{\bar{x}_3 \Rightarrow z(d_3 \bar{x}_3 x_2 x_1)(\vee \bullet)}{x_3 \Rightarrow "011b"} \\
\frac{d_2 \bar{x}_3}{(Ground)x_3} = (\vee \bullet) x_2 \bar{x}_1 (\vee \bullet) \frac{\bar{x}_3 \Rightarrow z(d_2 \bar{x}_3 x_2 \bar{x}_1)(\vee \bullet)}{x_3 \Rightarrow "010b"} \\
\frac{d_1 \bar{x}_3}{(Ground)x_3} = (\vee \bullet) \bar{x}_2 x_1 (\vee \bullet) \frac{\bar{x}_3 \Rightarrow z(d_1 \bar{x}_3 \bar{x}_2 x_1)(\vee \bullet)}{x_3 \Rightarrow "001b"} \\
\frac{d_0 \bar{x}_3}{(Ground)x_3} = (\vee \bullet) \bar{\bar{x}}_2 \bar{x}_1 (\vee \bullet) \frac{\bar{x}_3 \Rightarrow z(d_0 \bar{x}_3 \bar{\bar{x}}_2 \bar{x}_1)(\vee \bullet)}{x_3 \Rightarrow "000b"}
\end{array} \right\} (\vee \bullet) \quad (3.13)$$

Масштабирование (наращивание разрядности) такого элемента помимо уже указанного примера расширения (3.6) должно учитывать распространение применяемой константы. Для этого предлагается использовать дополнительно новый элемент на одну переменную с дешифрацией:

$$z(d_0 d_1 x) = \left. \begin{array}{l}
\frac{x \Rightarrow 0}{d_1 x} \\
\frac{x \Rightarrow 1}{d_0 \bar{x}}
\end{array} \right\} (\vee \bullet), \quad (3.14)$$

где $x \Rightarrow 0$ означает выдачу нулевого сигнала на этом выходе, если x , а $\bar{x} \Rightarrow 1$ – выдачу нулевого сигнала на другом выходе, если не x . Кроме этого, есть еще и третий выход – значение функции.

В общем случае выражение, определяющее модель реализации логической функции и дешифрацию набора за счет неактивных ветвей дерева, выглядит следующим образом:

$$z(d_0 \dots d_{2^n-1} x_1 \dots x_n) = \left. \begin{array}{l} \frac{d_{2^n-1} (NOT) x_n}{(V_{cc}) x_n} (\vee \bullet) x_1 \dots x_{n-1} (\vee \bullet) \frac{x_n \Rightarrow z(d_{2^n-1} x_1 \dots x_n) (\vee \bullet)}{x_n (R_0) (NOT) \Rightarrow j(x_1 \dots x_n)} \\ \frac{d_{2^n-2} (NOT) x_n}{(V_{cc}) x_n} (\vee \bullet) \overline{x_1} \dots x_{n-1} (\vee \bullet) \frac{x_n \Rightarrow z(d_{2^n-2} \overline{x_1} \dots x_n) (\vee \bullet)}{x_n (R_0) (NOT) \Rightarrow j(x_1 \dots x_n)} \\ \dots \\ \frac{d_1 (NOT) x_n}{(V_{cc}) x_n} (\vee \bullet) x_1 \dots \overline{x_{n-1}} (\vee \bullet) \frac{x_n \Rightarrow z(d_1 x_1 \dots \overline{x_n}) (\vee \bullet)}{x_n (R_0) (NOT) \Rightarrow j(x_1 \dots x_n)} \\ \frac{d_0 (NOT) x_n}{(V_{cc}) x_n} (\vee \bullet) \overline{x_1} \dots \overline{x_{n-1}} (\vee \bullet) \frac{x_n \Rightarrow z(d_0 \overline{x_1} \dots \overline{x_n}) (\vee \bullet)}{x_n (R_0) (NOT) \Rightarrow j(x_1 \dots x_n)} \end{array} \right\} (\vee \bullet) (NOT). \quad (3.15)$$

В выражении (3.15) подтягивающие к нулю резисторы обозначены R_0 , вместо них можно использовать блоки конституент нуля БКН [44].

В качестве второго варианта рассмотрим выполнение дешифрации входного набора без использования неактивной половины дерева транзисторов с использованием выражения 3.14.

3.1.4. Разработка модели LUT, вычисляющего значение заданной логической функции и дешифрацию набора переменных одновременно без использования неактивной половины дерева транзисторов

Ранее предложенный вариант метода основан на реализации LUT в виде одноуровневого дерева так называемых (pass) передающих (ключевых) транзисторов – упрощенное выражение (3.12) для $n=2$. В выражении (3.16) d_i — это настройка, то есть значения функции в соответствующей строке таблицы истинности, записываемые в так называемую конфигурационную память.

$$z(d_0 d_1 d_2 d_3 x_2 x_1) = \left. \begin{array}{l} \overline{d_3 x_2 x_1} \\ \overline{d_2 x_2 x_1} \\ \overline{d_1 x_2 x_1} \\ \overline{d_0 x_2 x_1} \end{array} \right\} (\vee \bullet). \quad (3.16)$$

В зависимости от значений переменных, активируется одна из четырех цепочек, объединяемых символом $(\vee\bullet)$ – это операция монтажного ИЛИ (wired OR).

Однако лучшим решением, чем использование подтягивающих резисторов, было бы введение дополнительных блоков переменных, обеспечивающих подключение уровня нуля в случае неактивирования соответствующей цепочки к инвертору выхода дешифрации. В этом случае используется модель многоуровневого дерева.

Так, для $n = 1$ получим:

$$\begin{aligned} \frac{(V_{cc})\bar{x}}{(Ground)x}(\vee\bullet)(NOT) &= z_0(x) \\ \frac{(V_{cc})x}{(Ground)\bar{x}}(\vee\bullet)(NOT) &= z_1(x) \\ \frac{d_1(NOT)x}{d_0(NOT)\bar{x}}(\vee\bullet)(NOT) &= z(d_0d_1x) \end{aligned} \quad (3.17)$$

Использование (3.17) позволяет строить элементы для произвольного количества переменных с учетом ограничений по количеству последовательно соединенных передающих транзисторов. То есть предлагается суперпозиция 1–LUT по выражению (3.17) для синтеза LUT, вычисляющего значение заданной логической функции и дешифрацию набора переменных.

В общем случае модель многоуровневого дерева для n переменных реализуется следующим образом:

$$\begin{array}{l}
z_0(x_n \dots x_1) = \frac{[z_{2,0}(x_n x_{n-1} \dots x_2)]\bar{x}_1}{(Ground)x_1} (\vee \bullet)(NOT) \\
z_1(x_n \dots x_1) = \frac{[z_{2,0}(x_n x_{n-1} \dots x_2)]x_1}{(Ground)x_1} (\vee \bullet)(NOT) \\
\hline
\frac{d_0 x_1}{d_1 x_1} (\vee \bullet) \\
\vdots \\
\vdots \\
\vdots \\
\hline
z_0(x_n \dots x_1) = \frac{[z_{2,1}(x_n x_{n-1} \dots x_2)]\bar{x}_n}{(Ground)x_n} (\vee \bullet)(NOT) \\
z_1(x_n \dots x_1) = \frac{[z_{2,1}(x_n x_{n-1} \dots x_2)]x_n}{(Ground)x_n} (\vee \bullet)(NOT) \\
\hline
\frac{d_{2^n-2} x_1}{d_{2^n-1} x_1} (\vee \bullet)
\end{array}
\quad \dots \quad
\begin{array}{l}
z_{n-1,0}(x_n x_{n-1}) = \frac{[z_{n,0}(x_n)]\bar{x}_{n-1}}{(Ground)x_{n-1}} (\vee \bullet) \\
z_{n-1,1}(x_n x_{n-1}) = \frac{[z_{n,0}(x_n)]x_{n-1}}{(Ground)x_{n-1}} (\vee \bullet) \\
\hline
\frac{x_{n-1}}{x_{n-1}} (\vee \bullet) \\
\hline
z_{n-1,0}(x_n x_{n-1}) = \frac{[z_{n,1}(x_n)]\bar{x}_{n-1}}{(Ground)x_{n-1}} (\vee \bullet) \frac{x_n}{x_n} (\vee \bullet)(NOT) = z(d_0 \dots d_{2^n-1} x_n \dots x_1) \\
z_{n-1,1}(x_n x_{n-1}) = \frac{[z_{n,1}(x_n)]x_{n-1}}{(Ground)x_{n-1}} (\vee \bullet) \\
\hline
\frac{x_{n-1}}{x_{n-1}} (\vee \bullet)
\end{array}
\quad z_{n,0}(x_n) = \frac{(V_{cc})\bar{x}_n}{(Ground)x_n} (\vee \bullet) \\
z_{n,1}(x_n) = \frac{(V_{cc})x_n}{(Ground)x_n} (\vee \bullet)
\end{array} \quad (3.18)$$

В корне основного дерева (старшая переменная n), реализующего логическую функцию $z(d_0 \dots d_{2^n-1} x_n \dots x_1)$ в СДНФ, указана операция монтажного ИЛИ $\vee \bullet$ и восстановитель сигнала NOT:

$$\frac{x_n}{x_n} (\vee \bullet)(NOT) = z(d_0 \dots d_{2^n-1} x_n \dots x_1).$$

По этой же переменной дополнительно введен дешифратор:

$$z_{n,0}(x_n) = \frac{(V_{cc})\bar{x}_n}{(Ground)x_n} (\vee \bullet), \\
z_{n,1}(x_n) = \frac{(V_{cc})x_n}{(Ground)x_n} (\vee \bullet),$$

который реализует функции:

$$z_{n,0}(x_n) = \bar{x}_n; z_{n,1}(x_n) = x_n,$$

обеспечивающие дешифрацию старшей переменной. При этом, когда переменная, либо ее инверсия истинны, соответствующий корень $\vee \bullet$ принимает значение логической единицы за счет подключения константы V_{cc} (напряжение источника питания). Если же переменная ложна, то соответствующий корень $\vee \bullet$ принимает значение логического нуля за счет подключения константы $Ground$. Таким образом, ортогональность сигнала в так называемом обратном дереве (дереве дешифрации) на точках $\vee \bullet$ соблюдается (нет неопределенности

логического уровня при дешифрации). Использование (3.18) позволяет строить элементы для произвольного количества переменных с учетом ограничений по количеству последовательно соединенных передающих транзисторов (не более четырех, а в ПЛИС – не более трех).

То есть предлагается суперпозиция 1-LUT по выражению (3.18) для синтеза LUT, вычисляющих значение заданной логической функции и дешифрацию набора переменных [43].

Предложенные три варианта модели необходимо оценить с использованием схемотехнического и топологического моделирования.

3.2. Разработка электрических функциональных схем элементов LUT, вычисляющих значение заданной логической функции и дешифрацию набора переменных

3.2.1. Электрическая функциональная схема адаптивного элемента LUT с одноуровневым 2^n деревом

Известный LUT по выражениям (3.2), (3.3) представлен на рисунке 3.1.

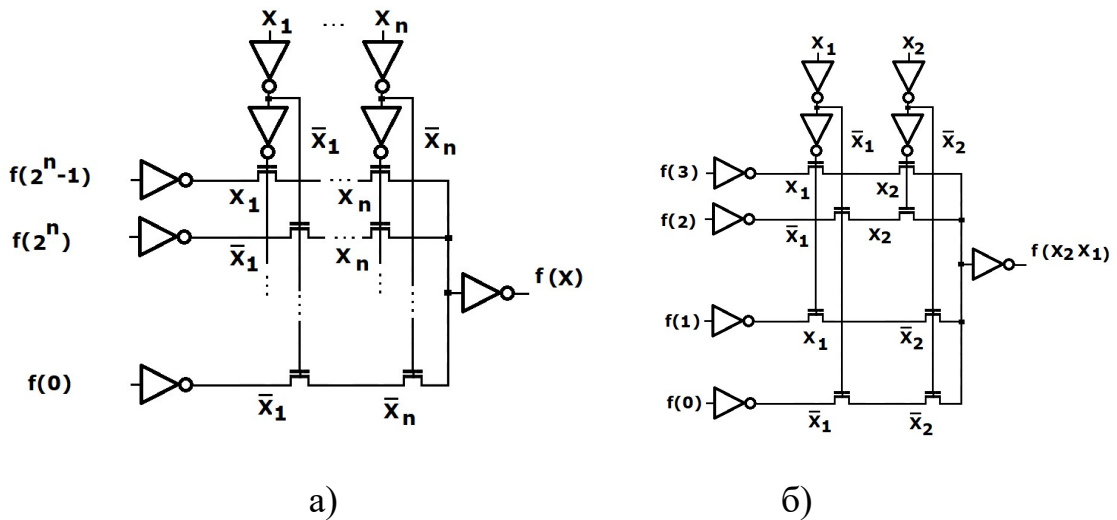
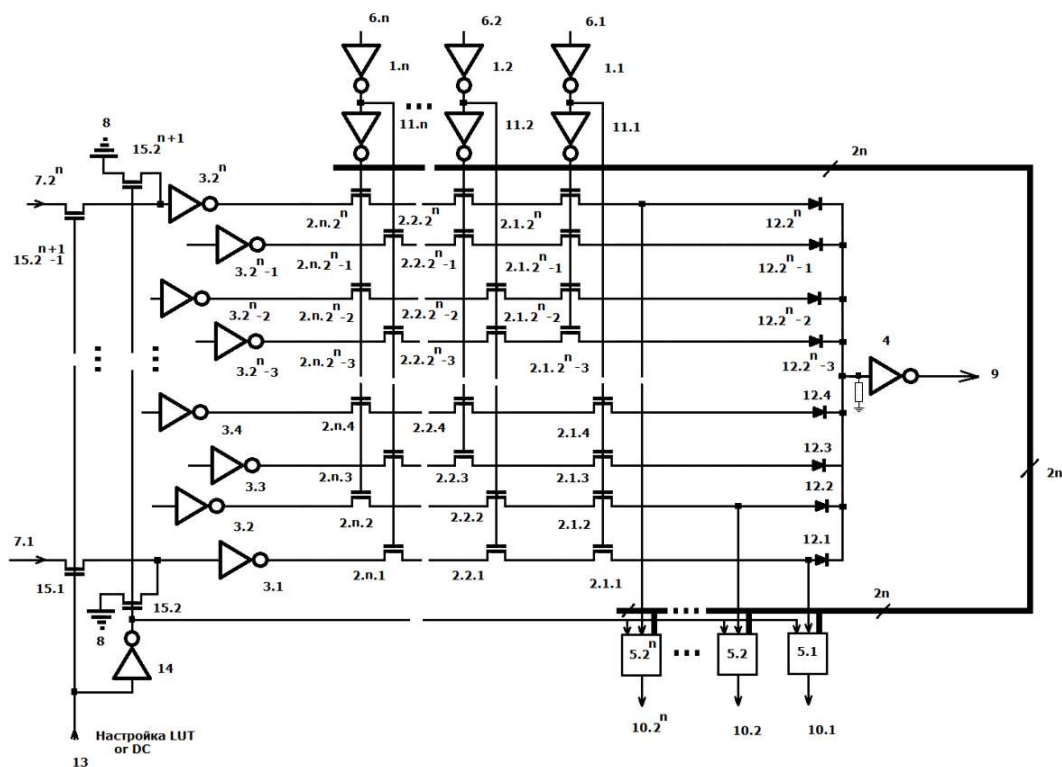


Рисунок 3.1 – LUT в виде одноуровневого дерева: а) n -LUT б) 2-LUT

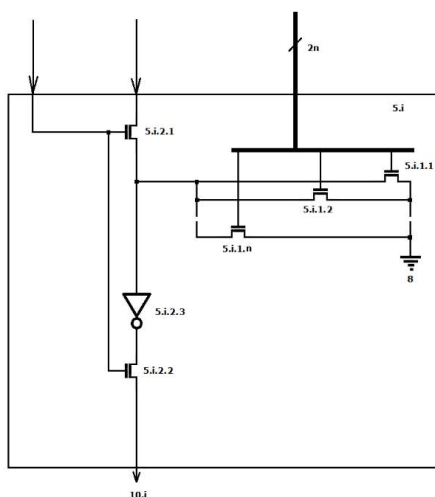
Пары инверторов по входам переменных необходимы для восстановления сигнала, уровень которого снижается после прохождения матриц коммутаций.

Предлагаемый адаптивный n -LUT [22,42], вычисляющий значение заданной логической функции или дешифрацию набора переменных в зависимости от

сигнала настройки с подтягивающим к уровню нуля резистором на входе инвертора показан на рисунке 3.2.



а)



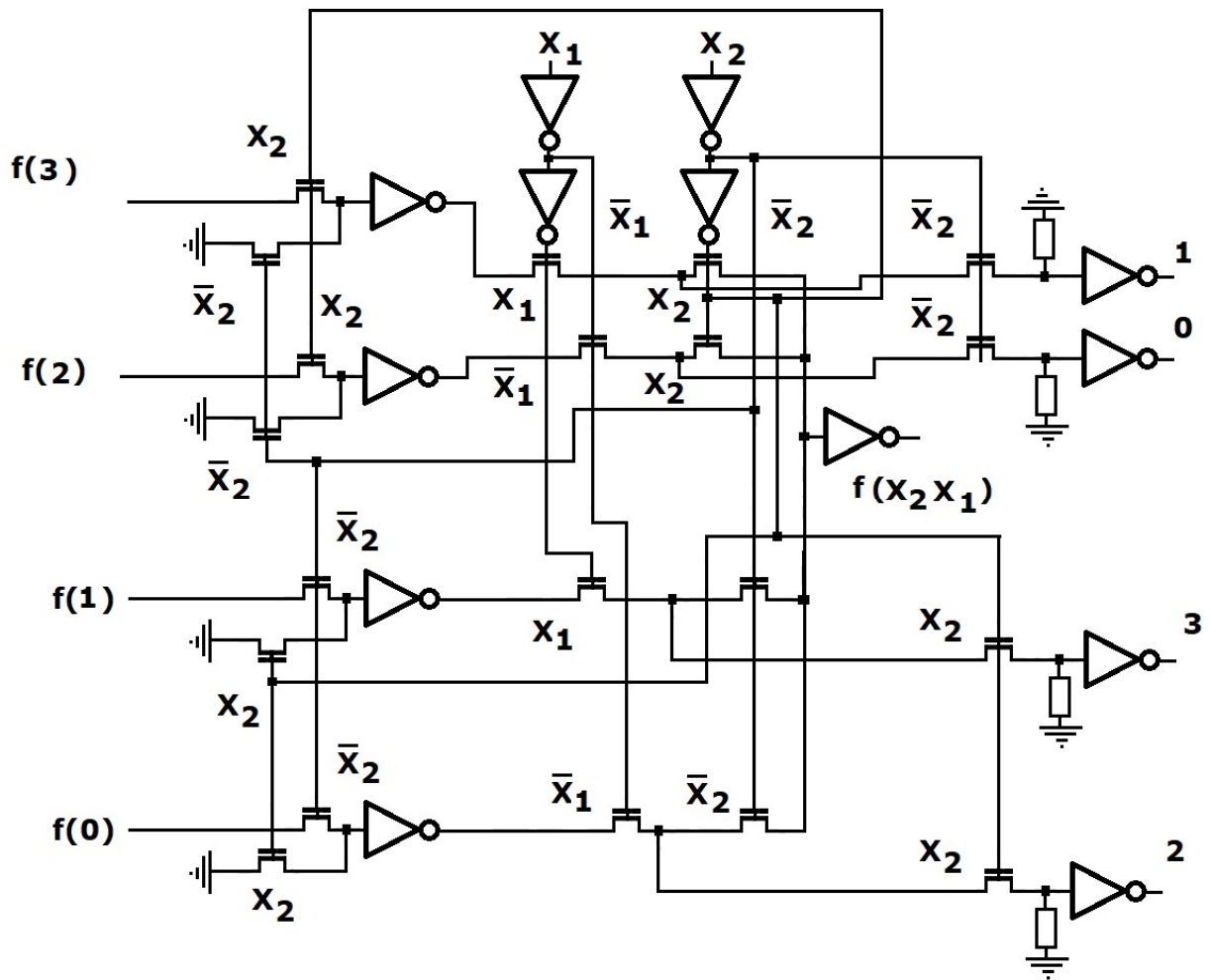
б)

Рисунок 3.2 – Адаптивный n -LUT, вычисляющий значение заданной логической функции или дешифрацию набора переменных в зависимости от сигнала настройки с подтягивающим к уровню нуля резистором на входе инвертора 4: а) схема электрическая функциональная; б) структура блоков 5 формирования конститuent нуля

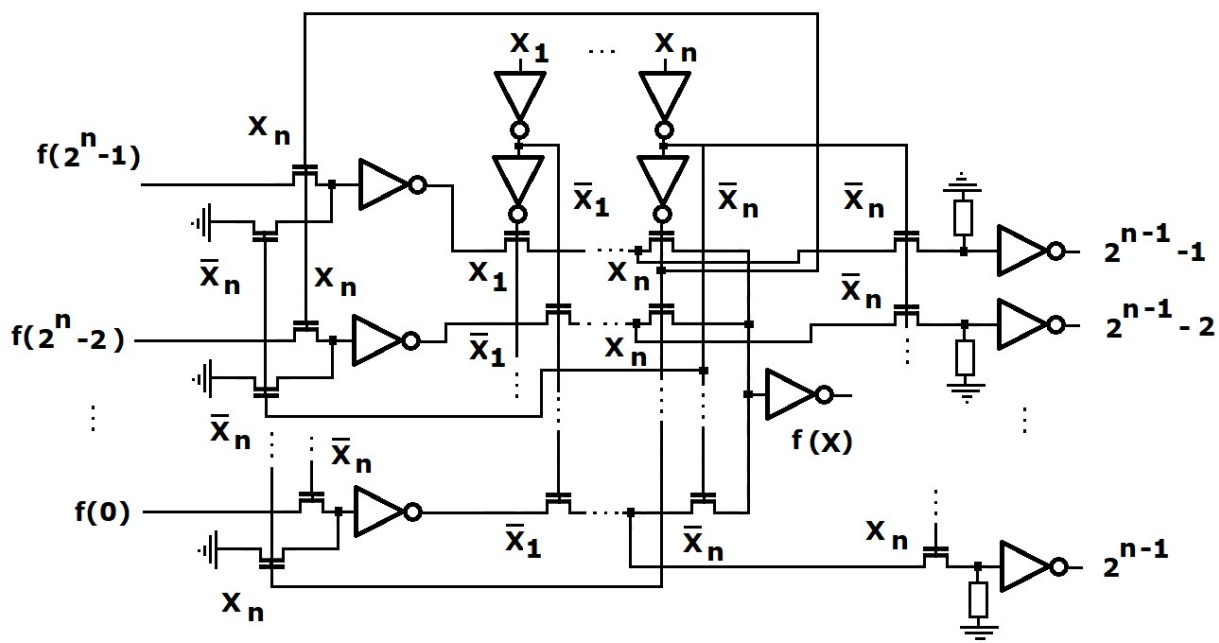
Однако, такой элемент не может выполнять одновременное вычисление логической функции и дешифрацию входного набора.

3.2.2. Электрическая функциональная схема LUT, вычисляющая значение заданной логической функции и дешифрацию набора переменных одновременно с использованием неактивной половины дерева транзисторов

Поэтому предлагается LUT, вычисляющий значение заданной логической функции и дешифрацию набора переменных одновременно [25]. Элемент показан на рисунке 3.3.



a)



б)

Рисунок 3.3 – LUT, вычисляющий значение заданной логической функции и дешифрацию набора переменных одновременно: а) 2–LUT б) n –LUT

Принцип формирования номера активного выхода (формируемой конституенты) для 2–LUT в двоичном коде поясняет таблица 3.1, в десятичном коде в таблице 3.2.

Таблица 3.1 – номер формируемой конституенты 2–LUT на выходе цепи данной конституенты в двоичном коде.

00b	01b	10b	11b
10b	11b	00b	01b

Таблица 3.2 – Номер формируемой конституенты 2–LUT на выходе цепи данной конституенты в десятичном коде.

0	1	2	3
2	3	0	1

Принцип формирования номера активного выхода (формируемой конституенты) для n –LUT в двоичном коде приведен в таблице 3.3, в десятичном коде в таблице 3.4.

Таблица 3.3 – Номер формируемой конституенты n -LUT на выходе цепи данной конституенты $0...2^n - 1$ в двоичном коде.

000...00b	000...01b	111...10b	111...11b
100...00b	100...01b	011...10b	011...11b

Таблица 3.4 – Номер формируемой конституенты n -LUT на выходе цепи данной конституенты $0...2^n - 1$ в десятичном коде.

0	1	$2^n - 2$	$2^n - 1$
2^{n-1}	$2^{n-1} + 1$	$2^{n-1} - 2$	$2^{n-1} - 1$

Разработанный 1-LUT, вычисляющий значение заданной логической функции и дешифрацию набора переменных одновременно без конфигурационных настроек приведен на рисунке 3.4:

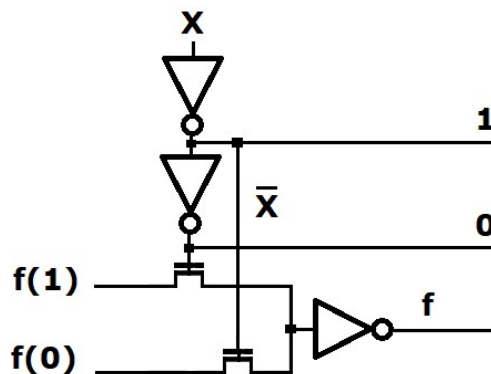


Рисунок 3.4 – 1-LUT, вычисляющий значение заданной логической функции и дешифрацию набора переменных одновременно без конфигурационных настроек

Разработанный 2-LUT, вычисляющий значение заданной логической функции и дешифрацию набора переменных одновременно с блоками конституент нуля (БКН) вместо подтягивающих резисторов показан на рисунке 3.5:

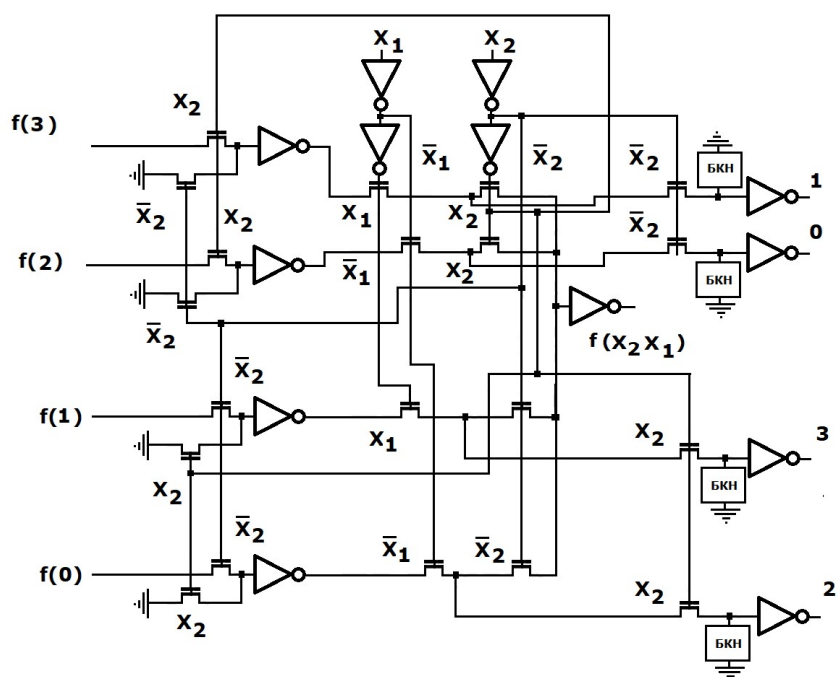


Рисунок 3.5 – 2–LUT, вычисляющий значение заданной логической функции и дешифрацию набора переменных одновременно с блоками конститuent

Предлагаемая схема 2–LUT по выражению с подтягивающими резисторами представлена на рисунке 3.6:

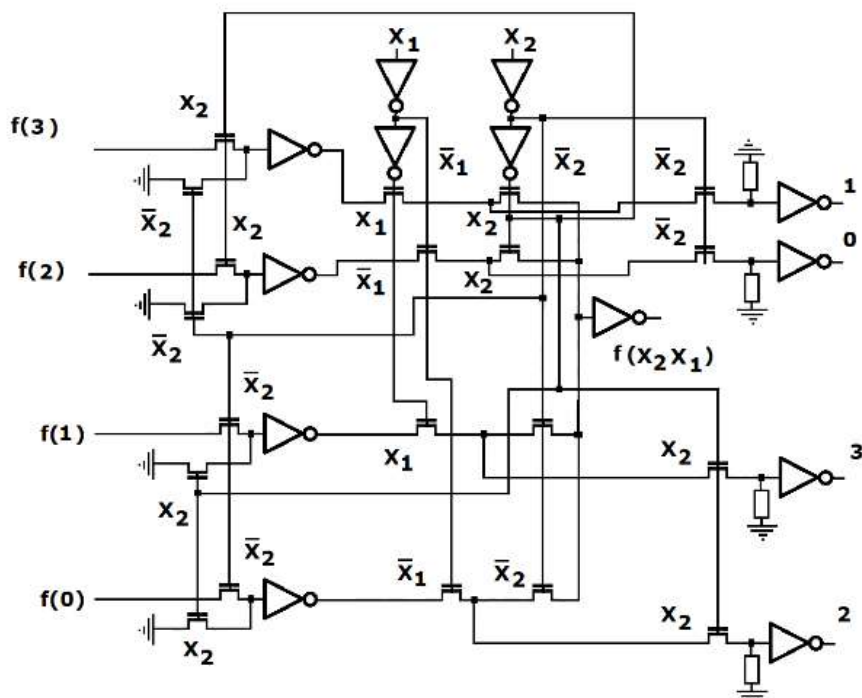


Рисунок 3.6 – 2–LUT, вычисляющий значение заданной логической функции и дешифрацию набора переменных одновременно с подтягивающими резисторами

Видно, что в верхней части схемы, где $x_2 = 1$ дешифрируются «младшие» наборы 1,0; а в нижней, где $x_2 = 0$, наоборот, «старшие» наборы 2,3.

Принцип наращивания разрядности для 3–LUT [47], вычисляющего значение заданной логической функции и дешифрацию набора переменных одновременно, построенного из двух 2–LUT с подтягивающими резисторами и одного 1–LUT показан на рисунке 3.7:

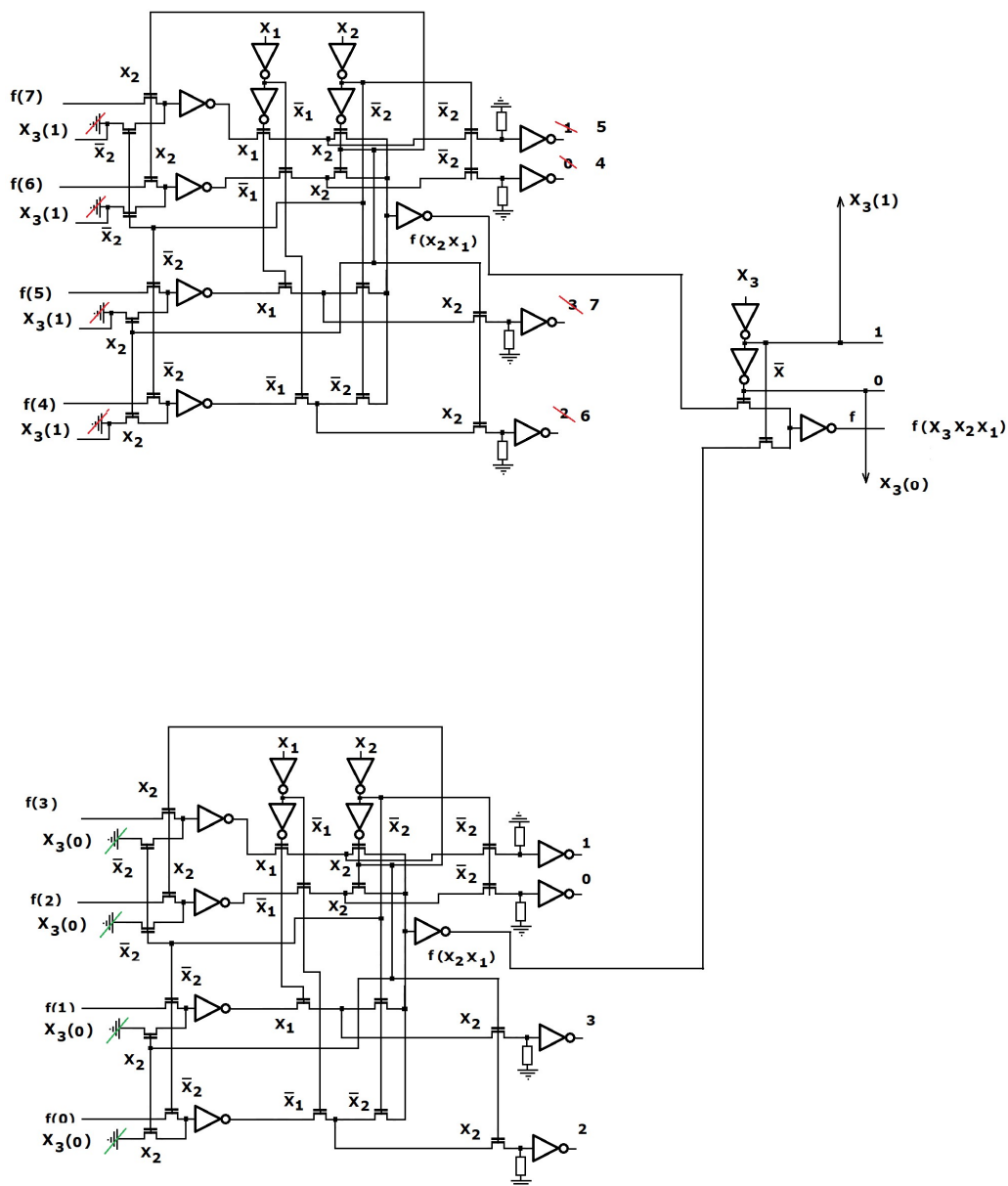


Рисунок 3.7 – 3–LUT, вычисляющий значение заданной логической функции и дешифрацию набора переменных одновременно, построенный из двух 2–LUT с подтягивающими резисторами и одного 1–LUT

3.2.3. Электрическая функциональная схема LUT вычисляющего значение заданной логической функции и дешифрацию набора переменных одновременно без использования неактивной половины дерева транзисторов

Схема 1-LUT по выражению (3.18) представляется следующим образом:

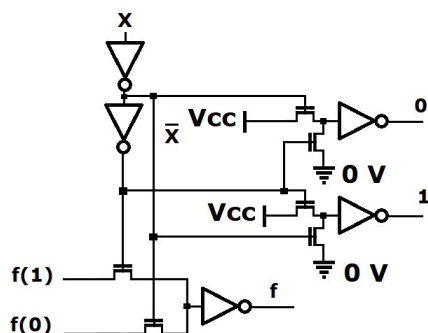


Рисунок 3.8 – 1-LUT_{DC} с обеспечением ортогональности сигналов по входам инверторов дешифрации

При построении 2-LUT_{DC} из трех 1-LUT_{DC} константы подключены только к «старшему» 1-LUT_{DC}, остальные получают эти константы от него:

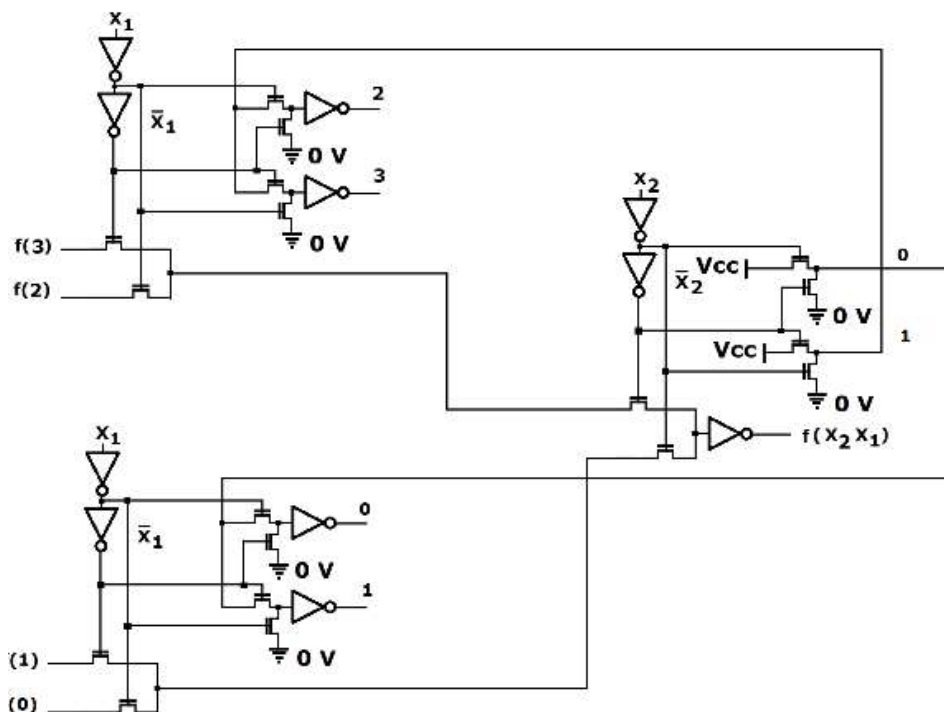


Рисунок 3.9 – 2-LUT_{DC}, построенный из предлагаемых 1-LUT_{DC}

Следует отметить, что по входу первой переменной достаточно только пары инверторов.

Разработанный 3-LUT_{DC}, построенный из предлагаемых 1-LUT_{DC}, изображен на рисунке 3.10:

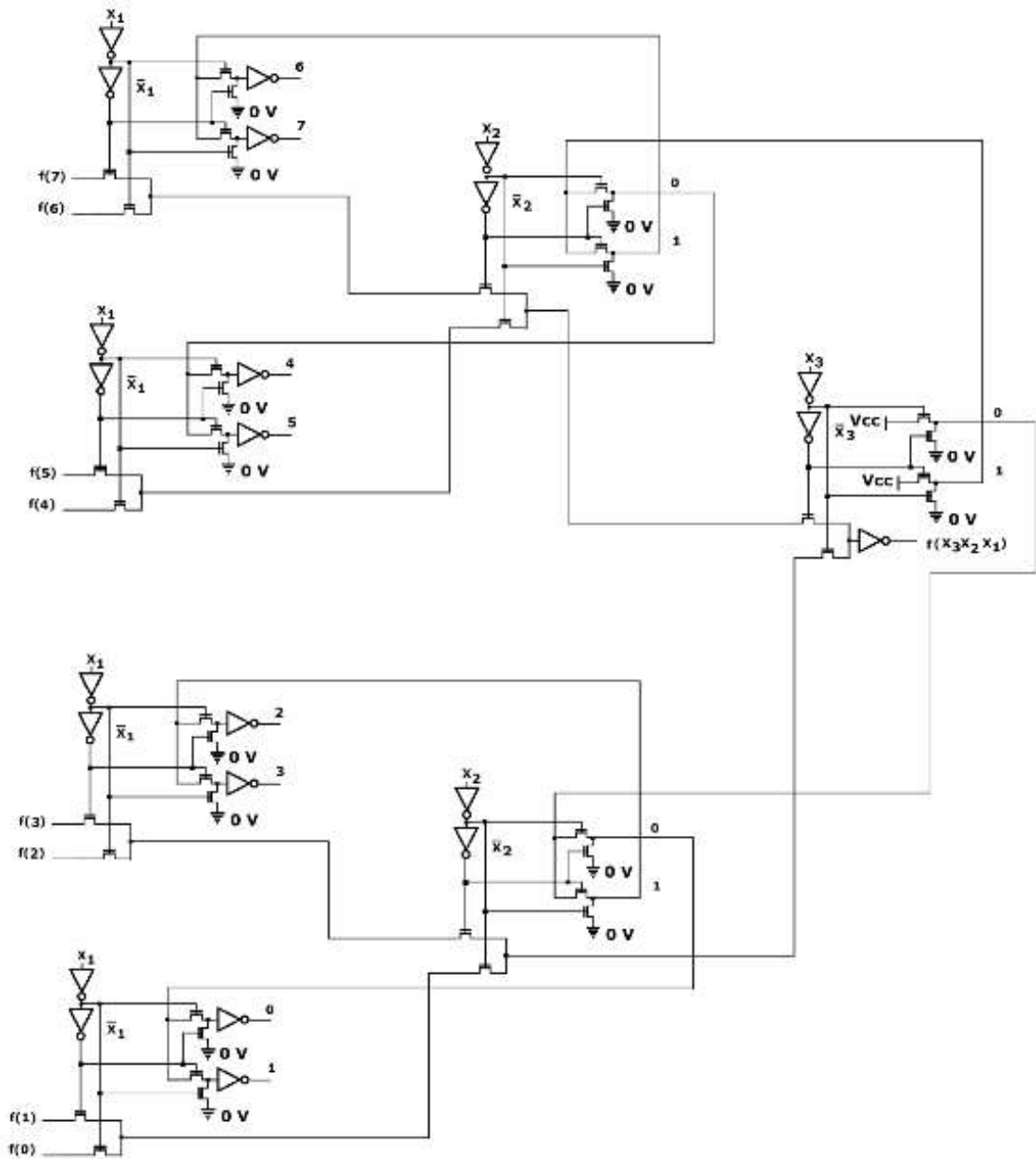


Рисунок 3.10 – 3-LUT+DC, построенный по третьей модели, $f(x_3, x_2, x_1)$ – основная функция; 0–7 – выходы дешифрации

Для синтеза описанных моделей были разработаны методы генерации LUT необходимой размерности с дешифрацией набора переменных.

3.3. Методы синтеза предложенных моделей

Метод синтеза в n -LUT* (звёздочка означает модификацию элемента в соответствие с выбранной моделью) логической функции и дешифрации набора переменных включает следующие шаги.

1. Задание параметра n
2. Синтез n -LUT из предлагаемого 1-LUT*
 - 2.1. Синтез 1-LUT*: подключить ко входам дешифрации первого 1-LUT* константы единицы («+» источника питания). Это корень дерева с дешифрацией.
 - 2.2. Синтез 2-LUT*: подключить ко входам дешифрации второго и третьего элементов 1-LUT* соответствующие выходы дешифрации первого элемента 1-LUT*. Информационные входы первого 1-LUT* подключаются к выходам соответствующих второго и третьего элементов 1-LUT*
 - 2.3. Выполнять пункты 2.1–2.2 подключения до достижения уровня n дерева.
3. На уровне n дерева подключить конфигурационные константы ко входам 1-LUT*.
4. Проверить корректность полученных номеров реализуемых конституент дешифрации.

Алгоритм соединения блоков функции–дешифрации для синтеза логического элемента LUT с дешифрацией входных переменных.

Реализованный алгоритм [47] обеспечивает синтез логического элемента LUT для определенного количества переменных, вычисляющий базовую логическую функцию и подключение дополнительных транзисторов для одновременного выполнения дешифрации входного набора.

Входными параметрами алгоритма являются: количество переменных n логического элемента LUT.

Общая схема алгоритма приведена на рисунке 3.11:

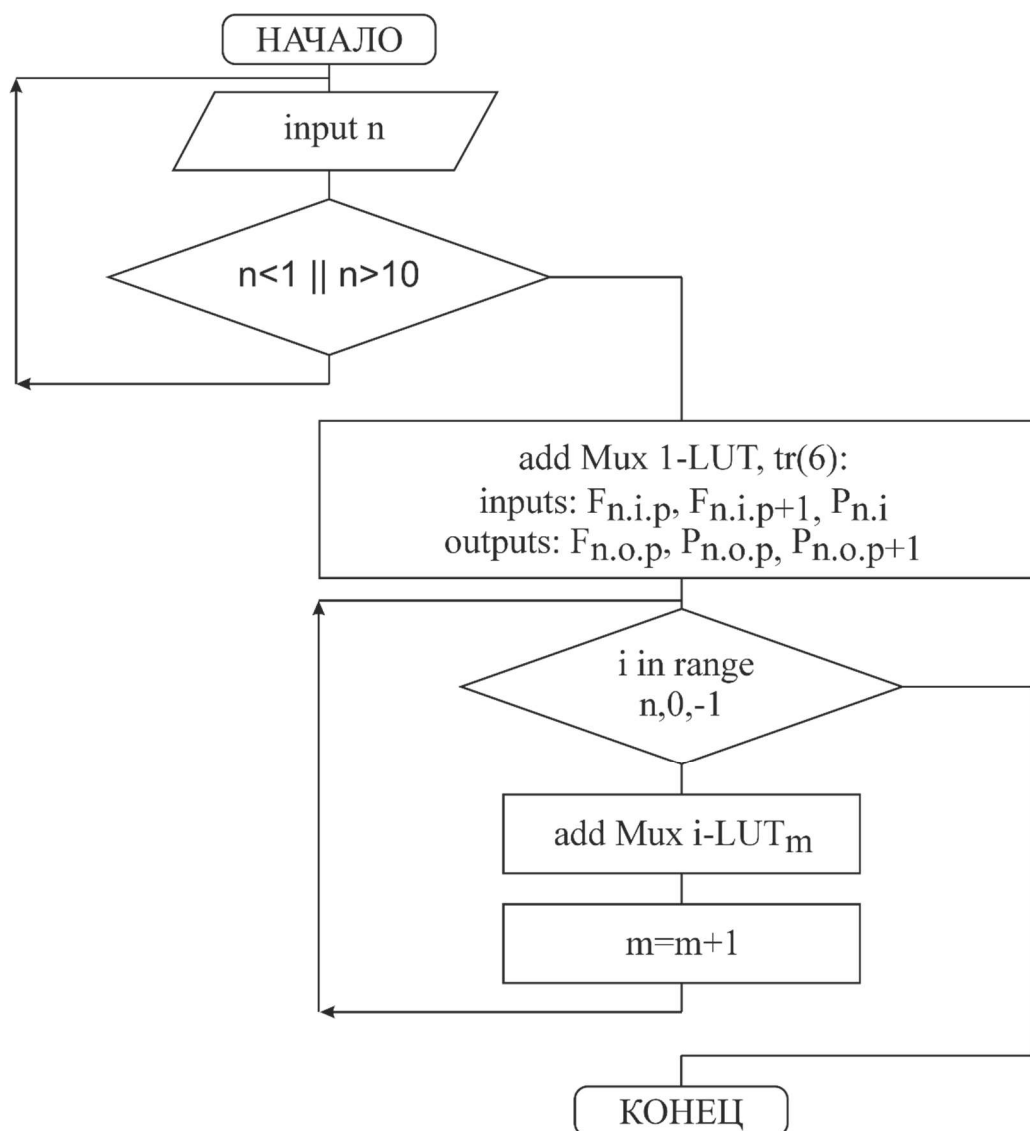


Рисунок 3.1 – Алгоритм подключения дополнительных транзисторов

Первым этапом производится синтез элемента 1–LUT с подключением дополнительных транзисторов, выполняющих дешифрацию. 1–LUT состоит из: входа переменной $X_{n,p}$, где p – индекс мультиплексора, определяющий подключение дешифрации; входа дешифрации $P_{n,i,2p}$, принимающий сигнал с предыдущих LUT; двух входов сигналов набора переменной или результатов вычисления предыдущих мультиплексоров; выхода (F_n) результата мультиплексирования; выходов дешифрации $P_{n,o,2p}$ $P_{n,o,2p+1}$, для подключения сигналов к предыдущим мультиплексорам. Дополнительно вводятся: два транзистора стоки которых объединены и подключены к порту ввода мультиплексора $P_{n,i,2p}$, и истоки объединены и подключены к выходу

мультиплексора $P_{n.o.2p}$, а затворы подключены к различным сигналам переменной – инвертированному и не инвертированному; два транзистора стоки которых объединены и подключены к нулю вольт (GROUND), и истоки объединены и подключены к выходу мультиплексора $P_{n.o.2p+1}$, а затворы подключены к различным сигналам переменной – инвертированному и не инвертированному.

Согласно количеству введенных переменных n синтезируется 2^n-1 мультиплексоров 1-LUT с дешифрацией. Начиная со старшей переменной X_n вводится первый ($m=1$) мультиплексор 1-LUT вход $P_{n.i.0}$, которого подключен к высокому уровню напряжения (VCC). Выход мультиплексора F_n является результатов вычисления логической функции. Вводятся мультиплексоры для предыдущей переменной X_{n-1} ($m=2$) в количестве 2^m-1 , где m – порядковый номер переменной от старшей к младшей. Осуществляется подключение выходов дешифрации старшего мультиплексора $P_{n.o.2p}$ к входу мультиплексора $P_{n-1.i.2p}$ и $P_{n.o.2p+1}$ к входу мультиплексора $P_{n-1.i.2p+1}$. Выходы результатов мультиплексирования переменной X_{n-1} подключаются к входам вычисления мультиплексирования старшей переменной X_n .

Синтез продолжается до достижения первой переменной X_1 , входы мультиплексоров которых подключены к набору переменных (SRAM). А выходы дешифрации являются результирующими сигналами дешифрации.

3.4. Выводы по главе 3

1. Предлагаемые модели и методы синтеза логических элементов LUT реализующих дешифрацию входного набора переменных и реализацию основной функции позволяют повысить функциональность известных элементов, сохраняя возможность вычисления основной функции, что отсутствовало в предыдущих моделях.

2. Разработанные модели делятся на два класса: 1 класс – двухрежимный вариант, когда реализуется основная функция в первом режиме и дешифрация переменных во втором режиме; 2 класс – однорежимный вариант, когда дешифрация и вычисление функции происходят одновременно.

3. Для реализации модели второго класса может быть использована неактивная часть дерева передающих транзисторов либо дешифрация осуществляется дополнительными средствами.

4. Для дальнейшего исследования работоспособности функциональных схем необходимо провести схемотехническое и топологическое моделирование.

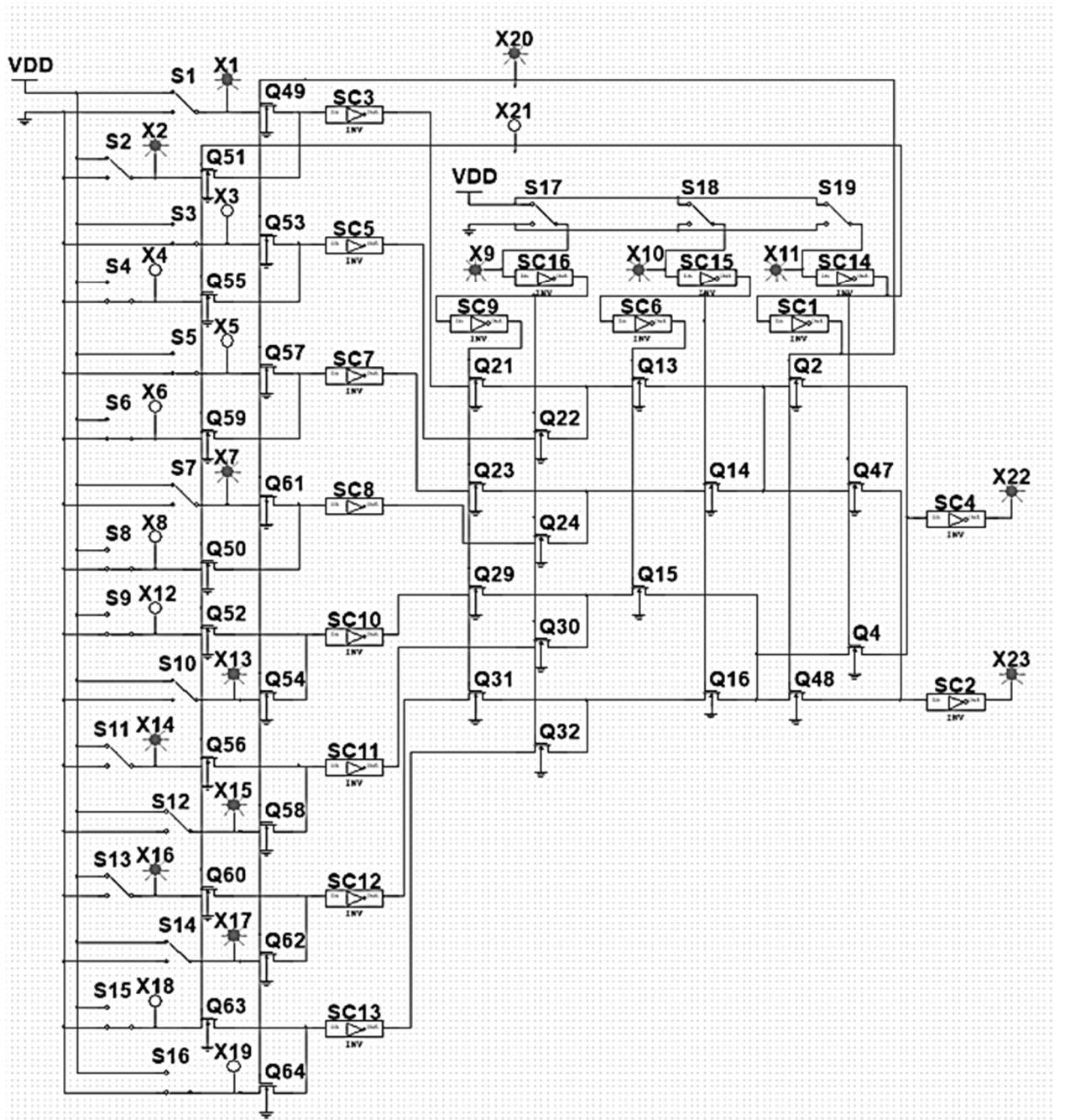
ГЛАВА 4. МОДЕЛИРОВАНИЕ РАЗРАБОТАННЫХ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ ПЛИС

4.1. Схемотехническое моделирование элемента n – LUT, реализующего одновременно 2^v функций от одних и тех же переменных

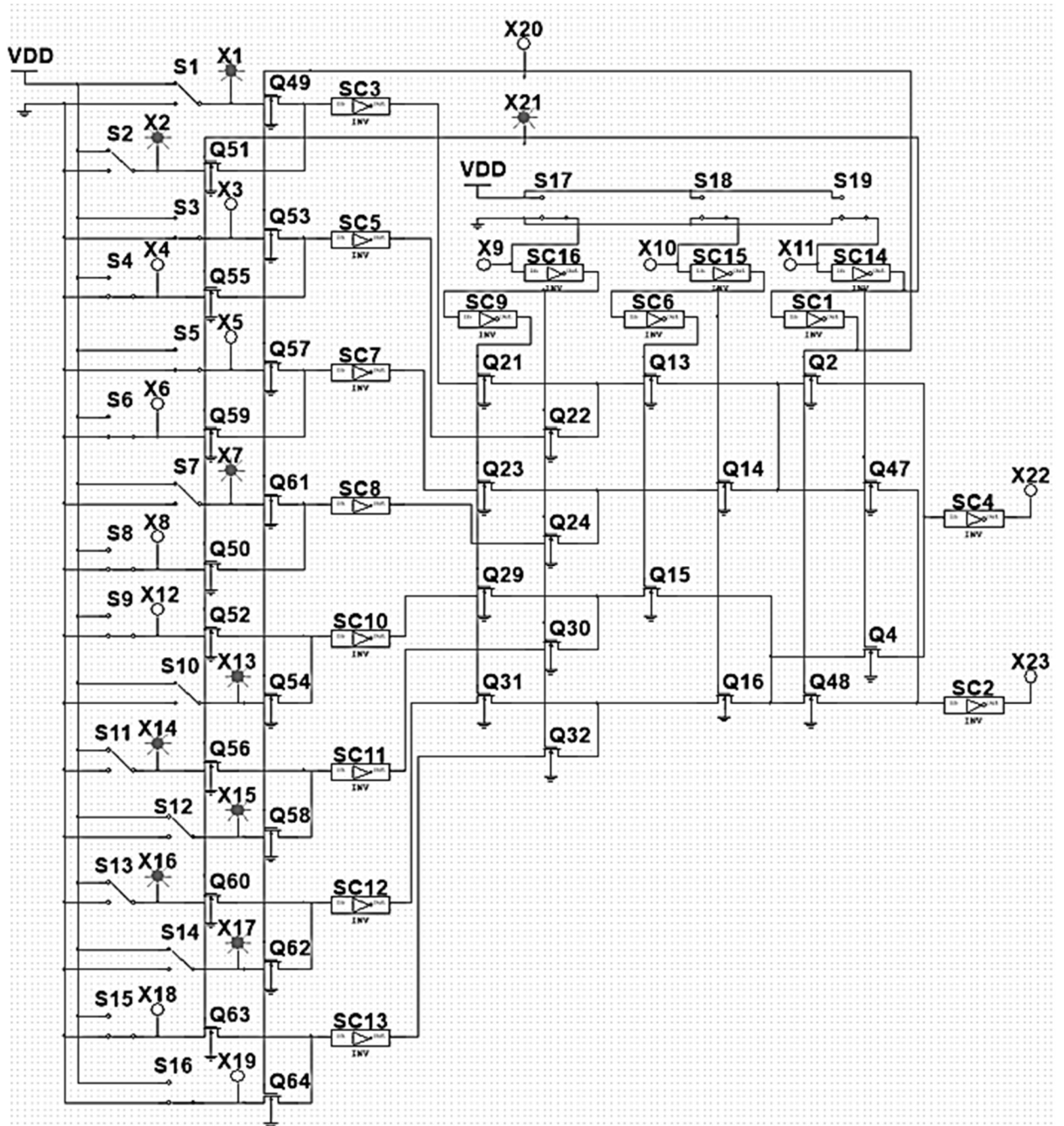
4.1.1. Статическое моделирование n – LUT, реализующего одновременно 2^v функций от одних и тех же переменных

Статическое моделирование по разработанным моделям проводилось в системе схемотехнического моделирования NI Multisim фирмы National Instruments Electronics Workbench Group [85]. Реализованная схема модели логического элемента на три переменные (3–LUT), выполняющего две функции одновременно представлена на рисунке 4.1. Базовый 3–LUT состоит из дерева передающих транзисторов (Q2, Q4, Q13–Q16, Q21–Q24, Q29–Q32), пар входных инвертеров переменных (SC1, SC6, SC9, SC14–SC16), которые восстанавливают сигнал после коммутации, выходного инвертора (SC4), инвертеров сигнала статической памяти SRAM (SC3, SC5, SC7, SC8, SC10–SC13). Ключи S17–S19 моделируют входные переменные логического элемента. Ключи S1–S8 моделируют ячейки памяти SRAM. Дополнительно введены передающие транзисторы Q47, Q48 истоки которых подключены к соответствующим истокам передающих транзисторов Q2, Q4, выходной инвертор дополнительной функции SC2, передающие транзисторы управления настройки Q49–Q64, управляемые старшей переменной, ключи ячеек памяти S9–S16. На рисунке 4.2 приведена схема инвертеров, используемых в схеме.

При подаче входного набора $A=1$, $B=1$, $C=1$ (рисунок 4.2 а) активируется ветвь базового LUT (Q2, Q13, Q21, SC3, Q49) передающая сигнал с ключа S1, для вычисления первой функции, значение которой фиксируется с помощью индикатора X22. Одновременно активируется вторая, параллельная ветвь дополнительной функции (Q48, Q15, Q29, SC10, Q54) передающая сигнал с ключа S10, для вычисления результата дополнительной функции, значение которой фиксируется с помощью индикатора X23.



a)



б)

Рисунок 4.1 – 3–LUT, вычисляющий значение двух логических функций одновременно: а) набор 111, выход 11; б) набор 000, выход 00

Аналогичным образом реализуется вычисление логических функций при подаче набора $A=0, B=0, C=0$ (рисунок 4.1 б). При активации соответствующих цепей происходит передача значений ключей S15, S8 на индикаторы вычисления логических функций.

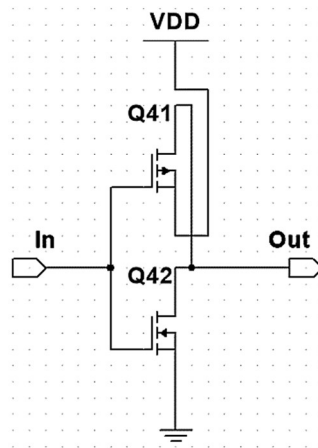
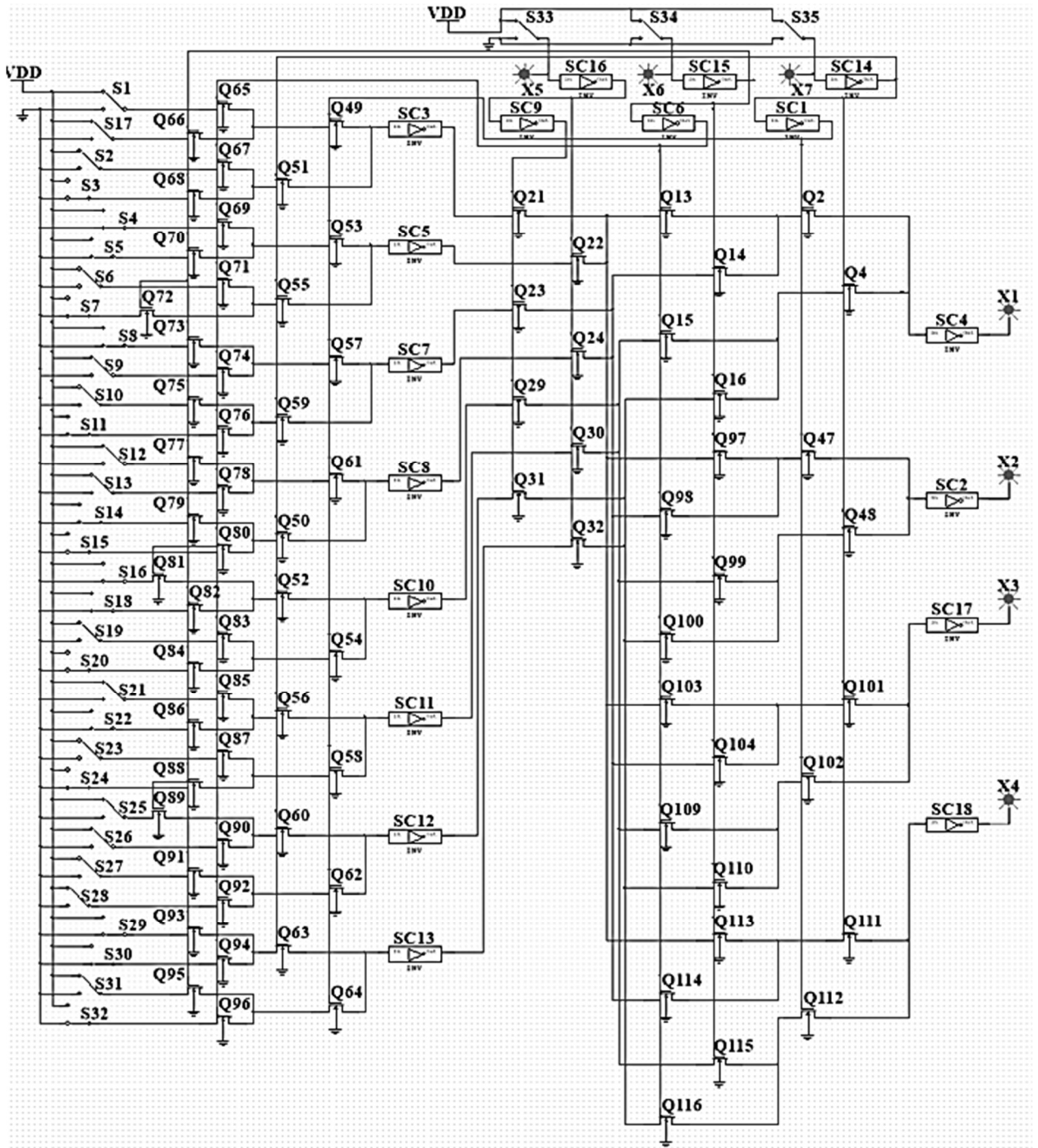


Рисунок 4.2 – Схема инвертора на полевых транзисторах

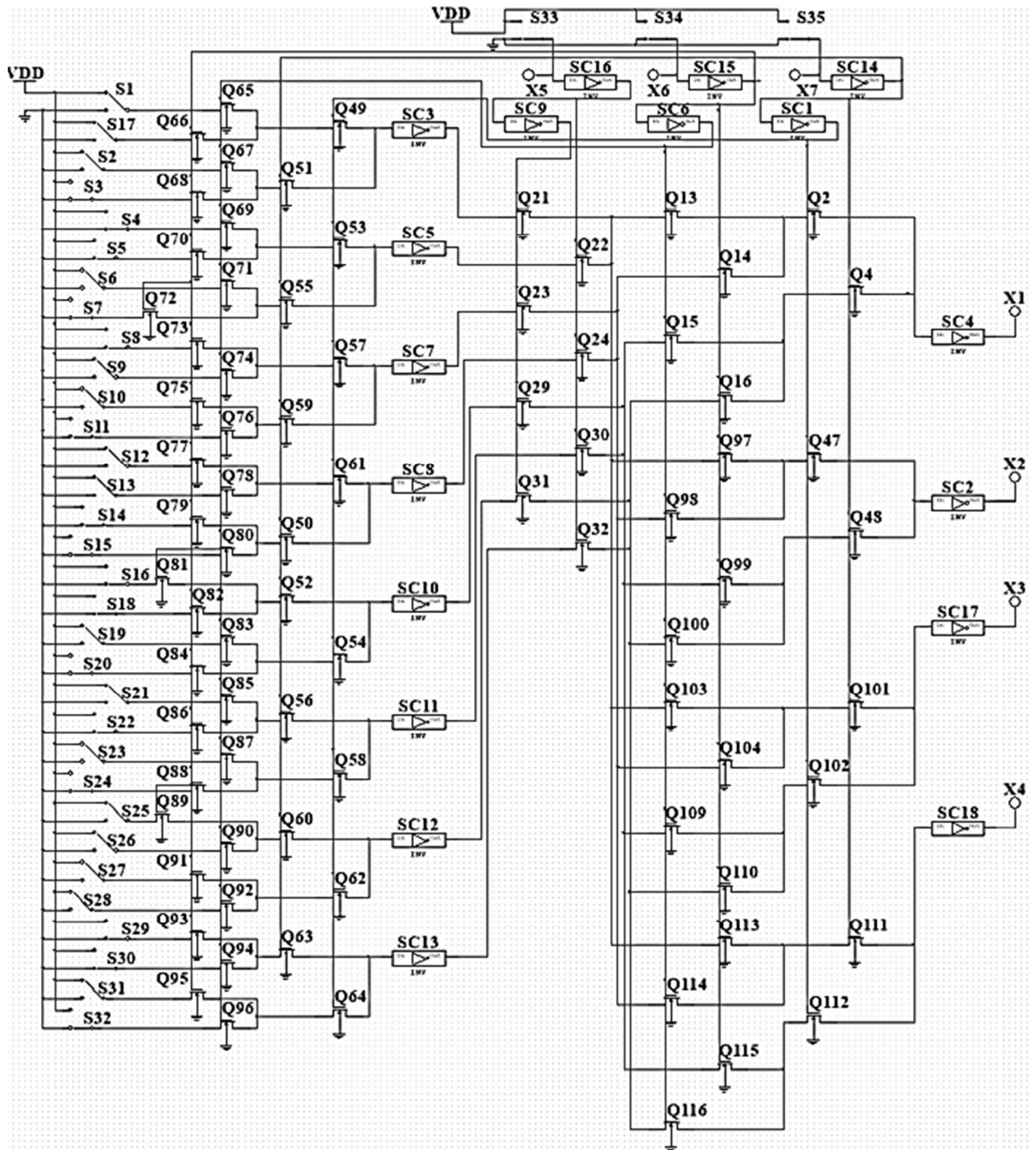
Реализованная схема модели логического элемента на три переменные (3–LUT), выполняющего четыре функции одновременно представлена на рисунке 4.3. К базовому 3–LUT дополнительно введены три 2–LUT, в которых истоки передающих транзисторов подключены к истокам передающих транзисторов на каскаде второй переменной два каскада транзисторов управления настройке. Первый каскад управляется старшей переменной, а второй каскад управляется второй переменной. Дополнительно введены ключи моделирующие ячейки памяти SRAM для одновременного вычисления дополнительных функций.

В случае установки входного набора $A=1$, $B=1$, $C=1$ (рисунок 4.3 а) активируются следующие пути вычисления логических функций: для $X1$ – Q2, Q13, Q21, SC3, Q49, Q65 сигнал ключа S1; для $X2$ – Q47, Q98, Q23, SC7, Q57, Q74 сигнал ключа S9; для $X3$ – Q102, Q109, Q29, SC10, Q54, Q83 сигнал ключа S19; для $X4$ – Q112, Q116, Q31, SC12, Q62, Q92 сигнал ключа S28.

Аналогичным образом реализуется вычисление логических функций при подаче набора $A=0$, $B=0$, $C=0$ (рисунок 4.3 б). При активации соответствующих цепей происходит передача значений ключей S29, S22, S14, S7 на индикаторы вычисления логических функций.



a)

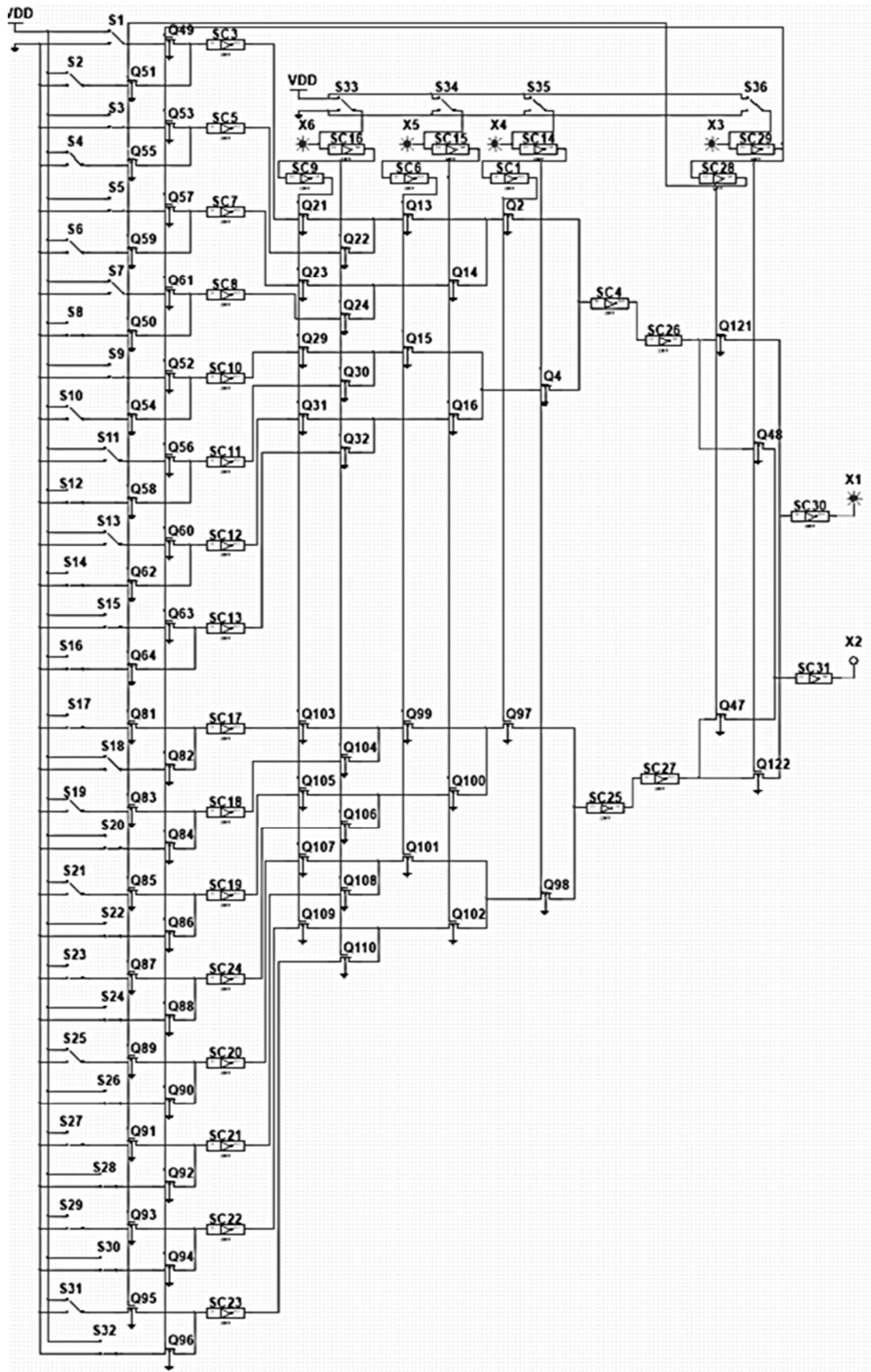


б)

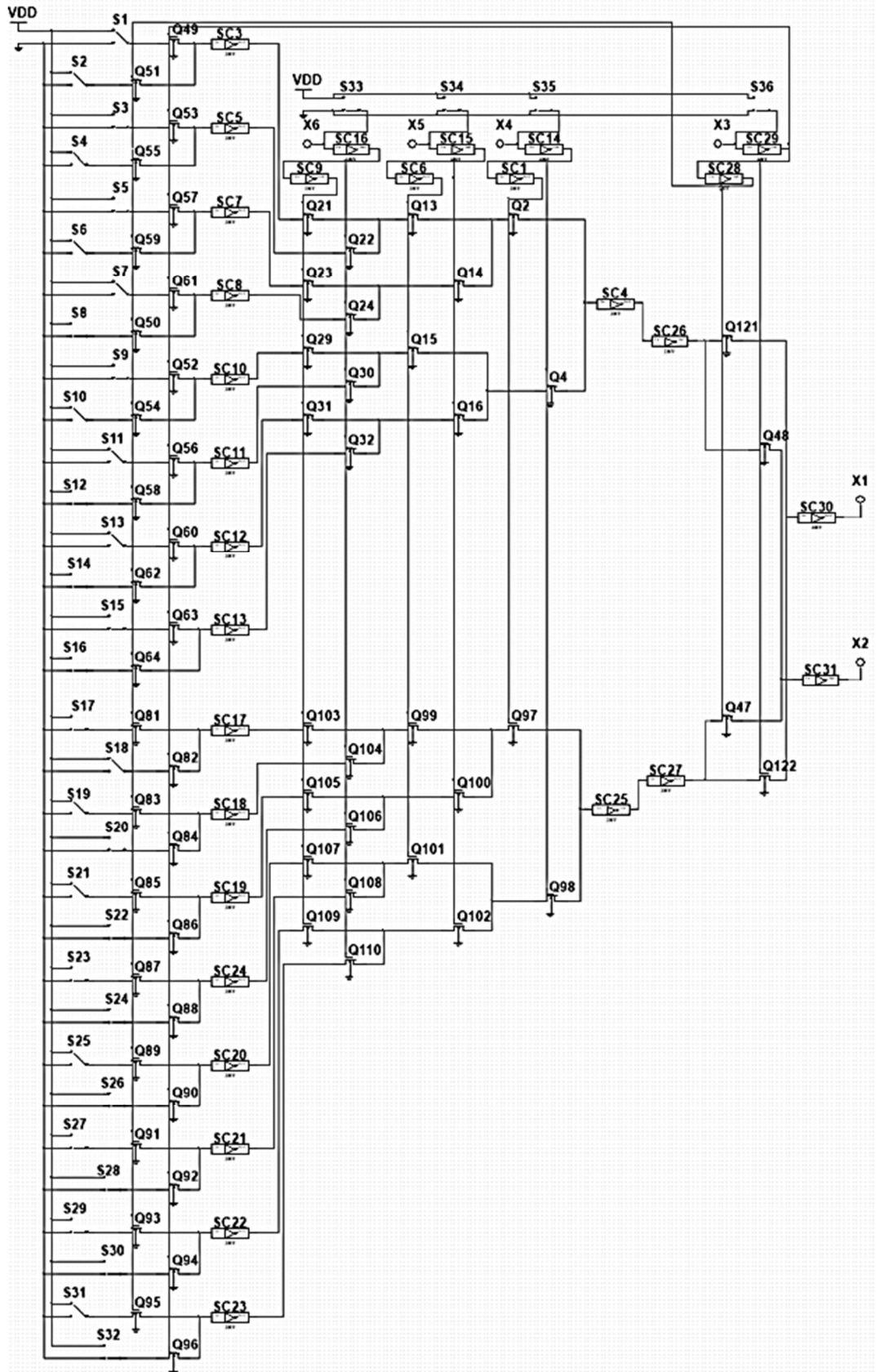
Рисунок 4.3 – 3-LUT, вычисляющий значение четырех логических функций одновременно: а) набор 111, выход 1111; б) набор 000, выход 0000

Схема логического элемента на четыре переменные (4-LUT), выполняющая две функции одновременно (рисунок 4.4), реализуется аналогичным образом схемы, изображенной на рисунке 4.1. Набор $A=1, B=1, C=1, D=1$ (рисунок 4.4 а)

реализует передачу значений ключей S2, S17, а набор A=0, B=0, C=0, D=0 (рисунок 4.4 б) передает значения ключей S32, S15.



a)



б)

Рисунок 4.4 – 4-LUT, вычисляющий значение двух логических функций одновременно: а) набор 1111, выход 10; б) набор 0000, выход 00

В результате статического моделирования продемонстрирована работоспособность предложенных методов и для дальнейшей оценки рекомендуется провести динамическое моделирование разработанных схем.

4.1.2. Динамическое моделирование n – LUT, реализующего одновременно 2^v функций от одних и тех же переменных

Для реализации динамического моделирования предложенных моделей статические ключи переменных заменены на генератор сигналов XWG1, выходные индикаторы выполнения функций заменены на осциллограф с двумя каналами XSC1. Результирующая схема моделирования логического элемента трех переменных (3–LUT) с одновременным вычислением двух логических функций приведена на рисунке 4.5. Моделью передающих транзисторов является BSIM4.8 с технологией изготовления 65 нм. Сигналы входных переменных заданы в генераторе в форме кода Грея и представлены на рисунке 4.6. Ключи статической памяти конфигурированы для реализации двух функций: F1 (SC4) – исключающее ИЛИ и F2 (SC2) – мажоритарной.

Результаты моделирования логического элемента 3–LUT представлены на рисунке 4.7, где верхняя осциллограмма является результатом вычисления логической функции «исключающее ИЛИ», а нижняя осциллограмма демонстрирует результат вычисления логической функции «мажоритарная».

Для моделирования логического элемента 3–LUT выполняющего четыре функции одновременно был установлен аналогичный генератор сигналов XWG1 вместо ключей переменных. Для записи осциллограмм четырех функций использовался четырехканальный осциллограф. Статические ячейки памяти SRAM конфигурированы для реализации четырех функций одновременно: исключающего ИЛИ, мажоритарной, дизъюнкции, конъюнкции.

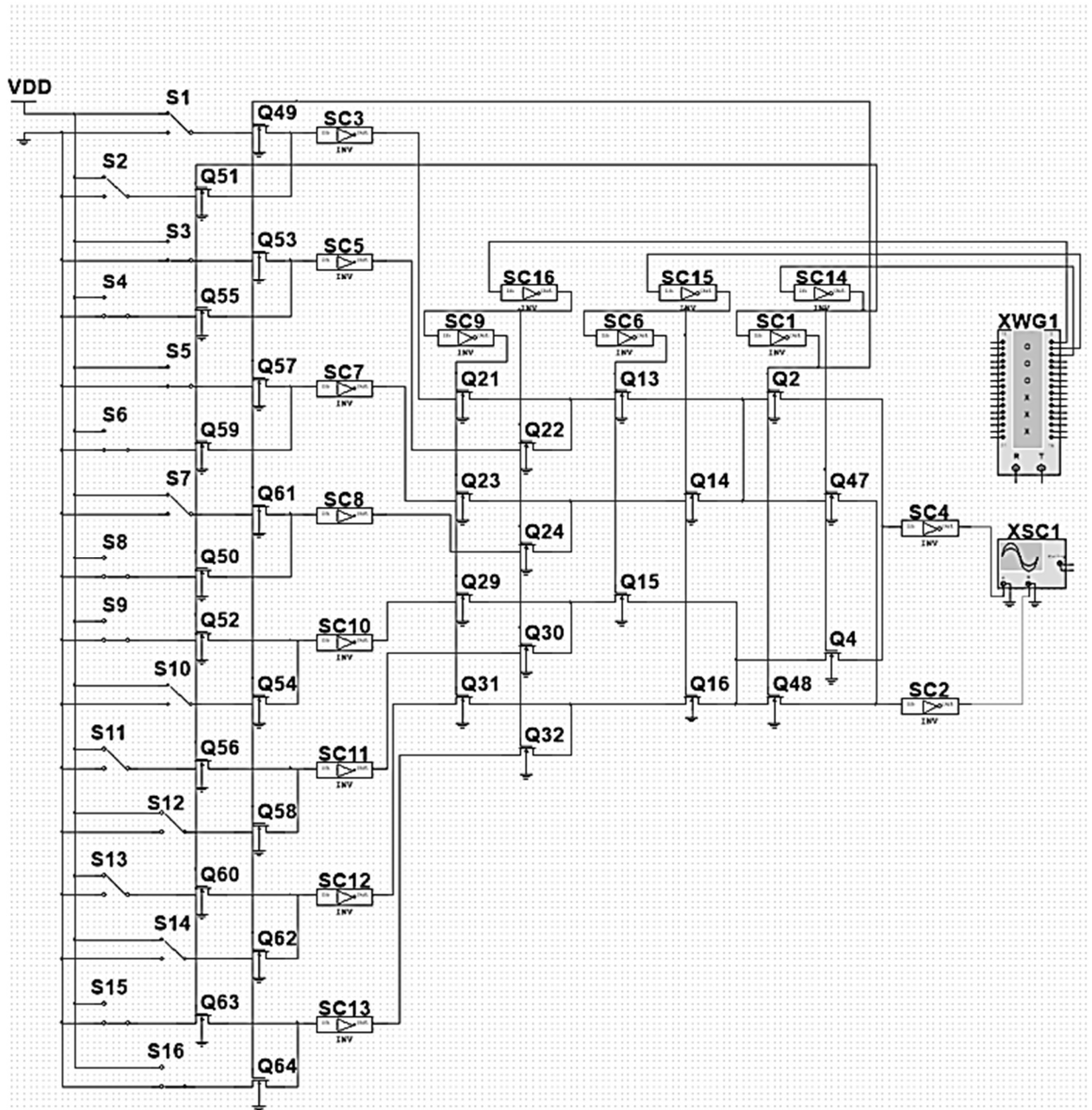


Рисунок 4.5 – Динамическое моделирование 3–LUT, вычисляющего значение двух логических функций одновременно

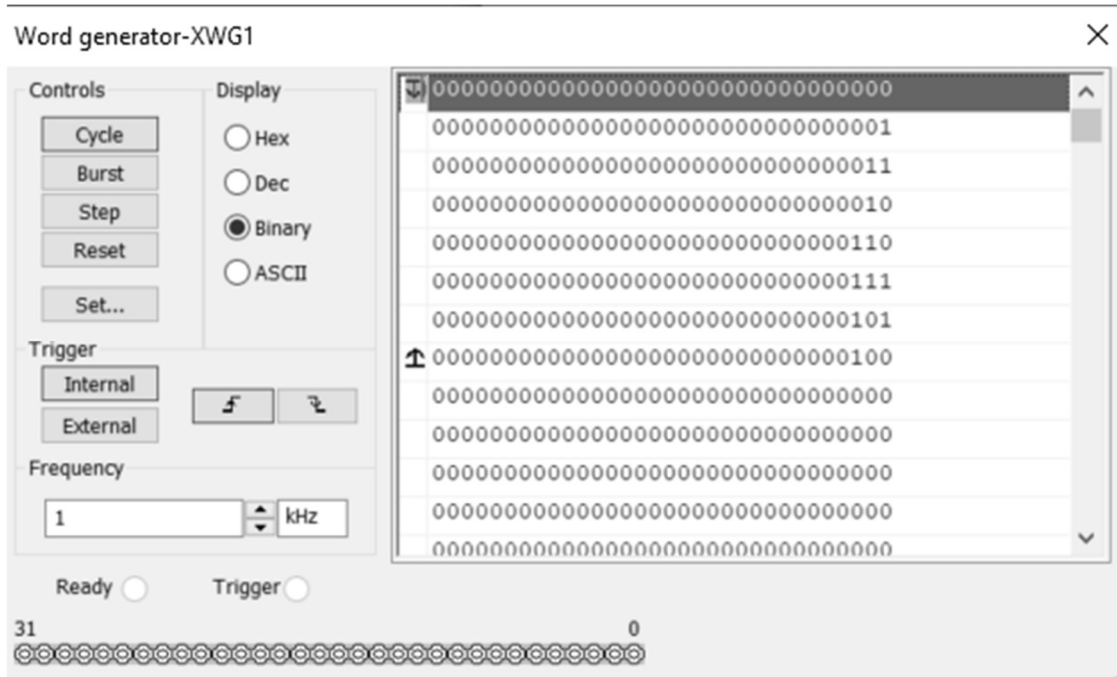


Рисунок 4.6 – Код входных сигналов переменных логического элемента 3–LUT в форме Грея

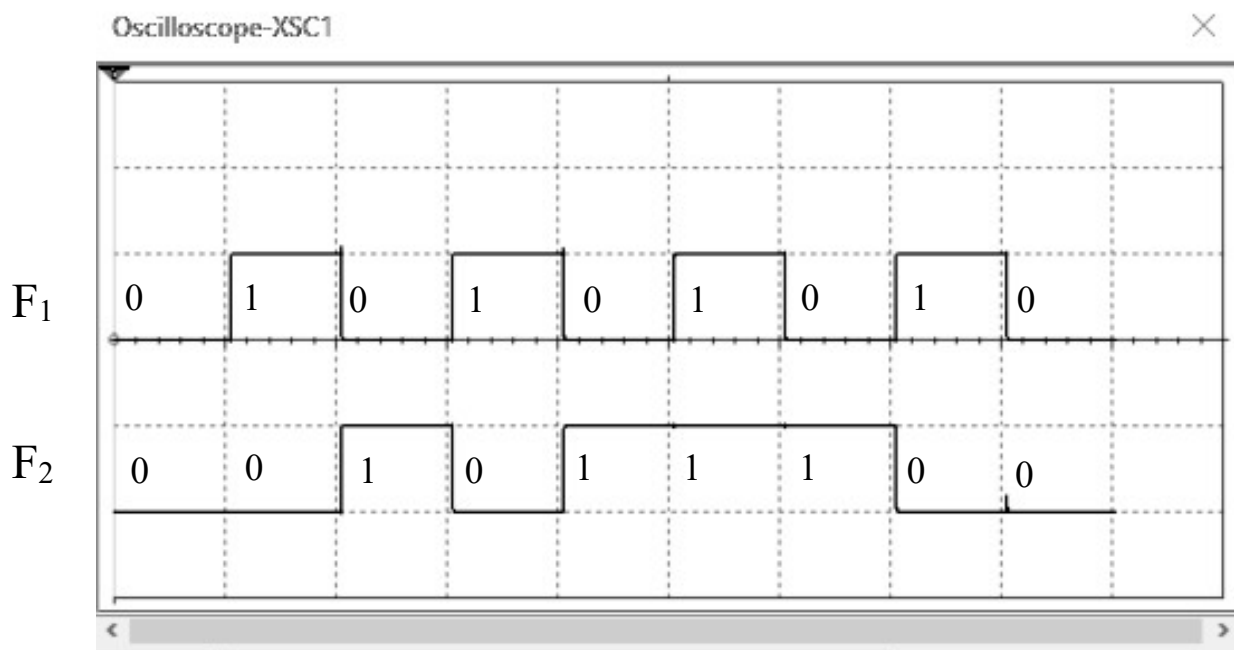


Рисунок 4.7 – Осциллограмма результата моделирования одновременного выполнения двух функций в 3–LUT

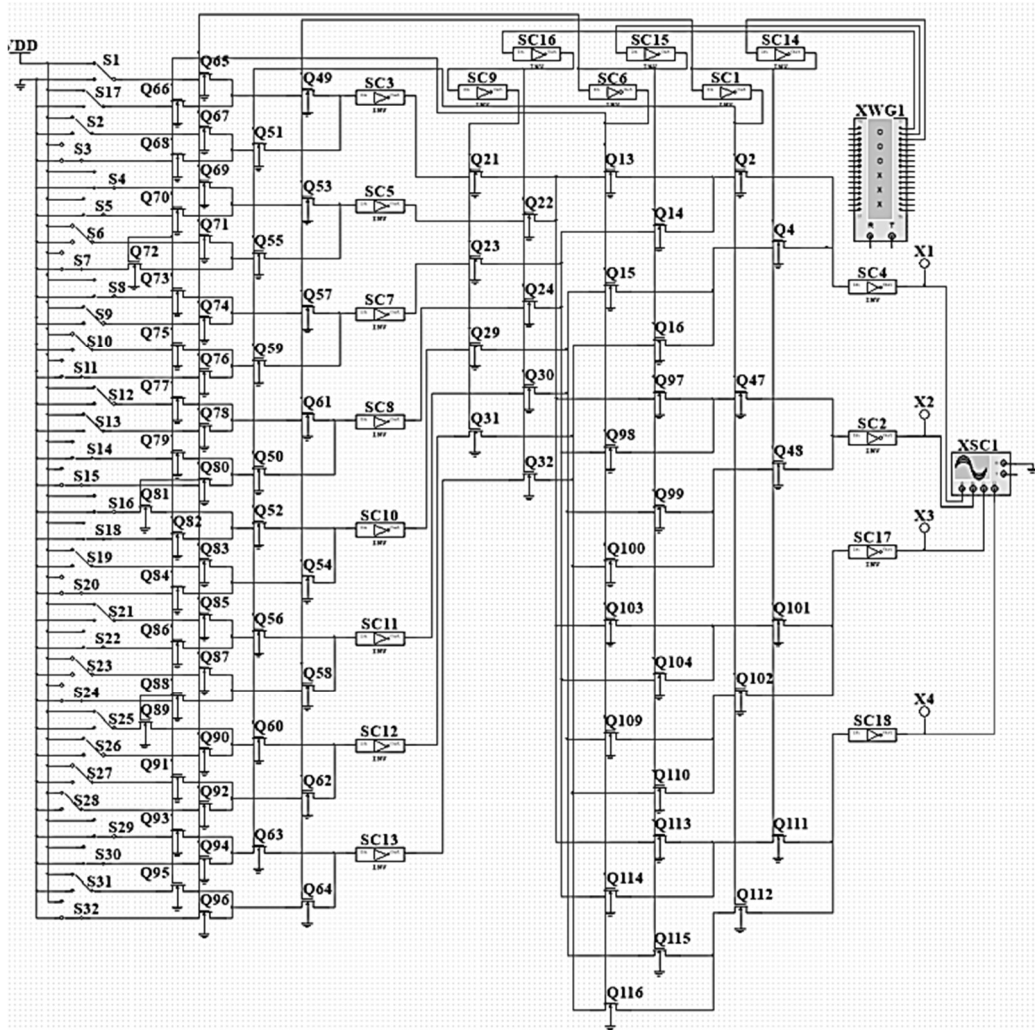


Рисунок 4.8 – Динамическое моделирование 3–LUT, вычисляющего значение четырех логических функций одновременно

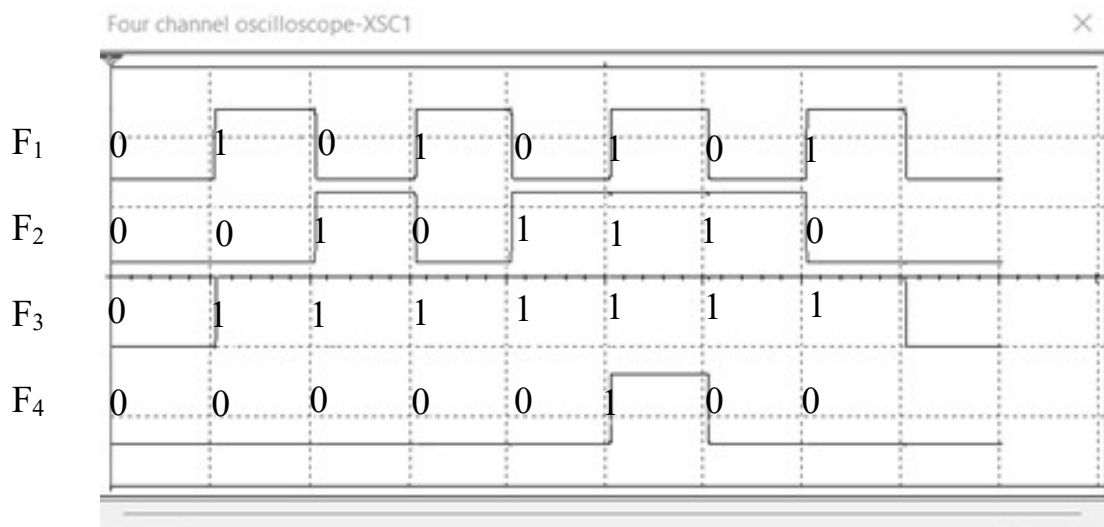


Рисунок 4.9 – Осциллограмма многофункционального логического элемента, выполняющая одновременное вычисление четырех функций: F₁ – исключающее ИЛИ, F₂ – мажоритарная, F₃ – дизъюнкция, F₄ – конъюнкция.

Для моделирования логического элемента 4-LUT выполняющего две функции одновременно был установлен генератор сигналов XWG1 вместо ключей переменных. Для записи осциллограмм двух функций использовался двухканальный осциллограф. Статические ячейки памяти SRAM конфигурированы для реализации двух функций одновременно: мажоритарной и исключающего ИЛИ. Общий вид схемы представлен на рисунке 4.11.

Результаты моделирования логического элемента 4-LUT представлены на рисунке 4.10, где верхняя осциллограмма является результатом вычисления логической функции «мажоритарная», а нижняя осциллограмма демонстрирует результат вычисления логической функции «исключающее ИЛИ».

Динамическое моделирование предлагаемых моделей демонстрирует работоспособность представленных схем. Рекомендуется дальнейшее исследование моделей по оценке быстродействия, потребляемой мощности и площади, занимаемой на кристалле по результатам топологического моделирования.

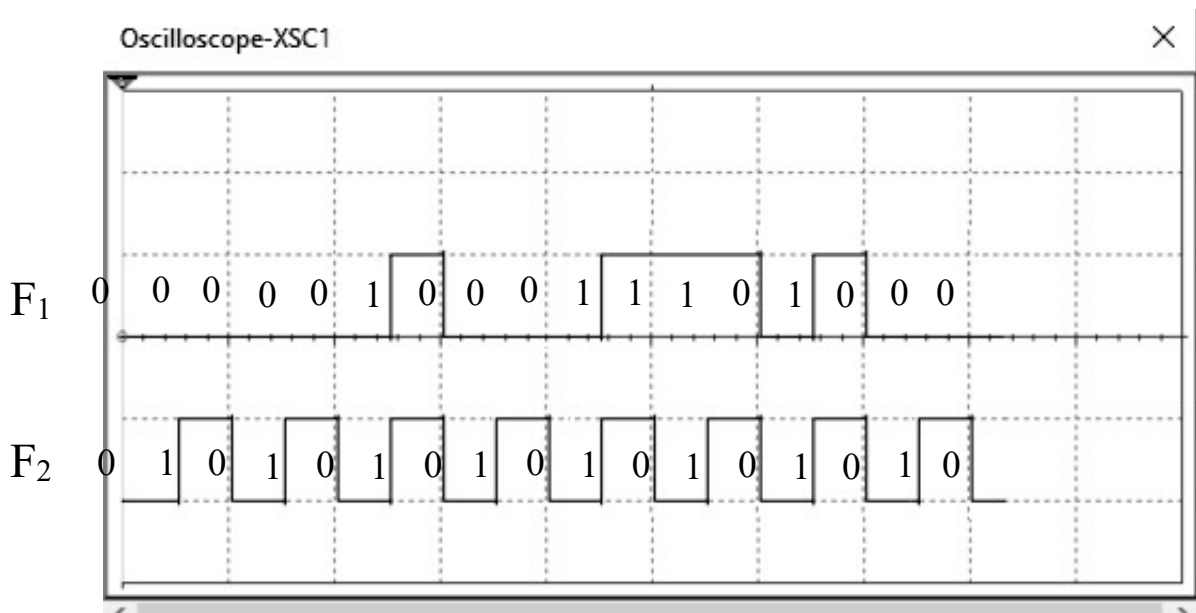


Рисунок 4.10 – Осциллограмма результата моделирования одновременного выполнения двух функций в 4-LUT

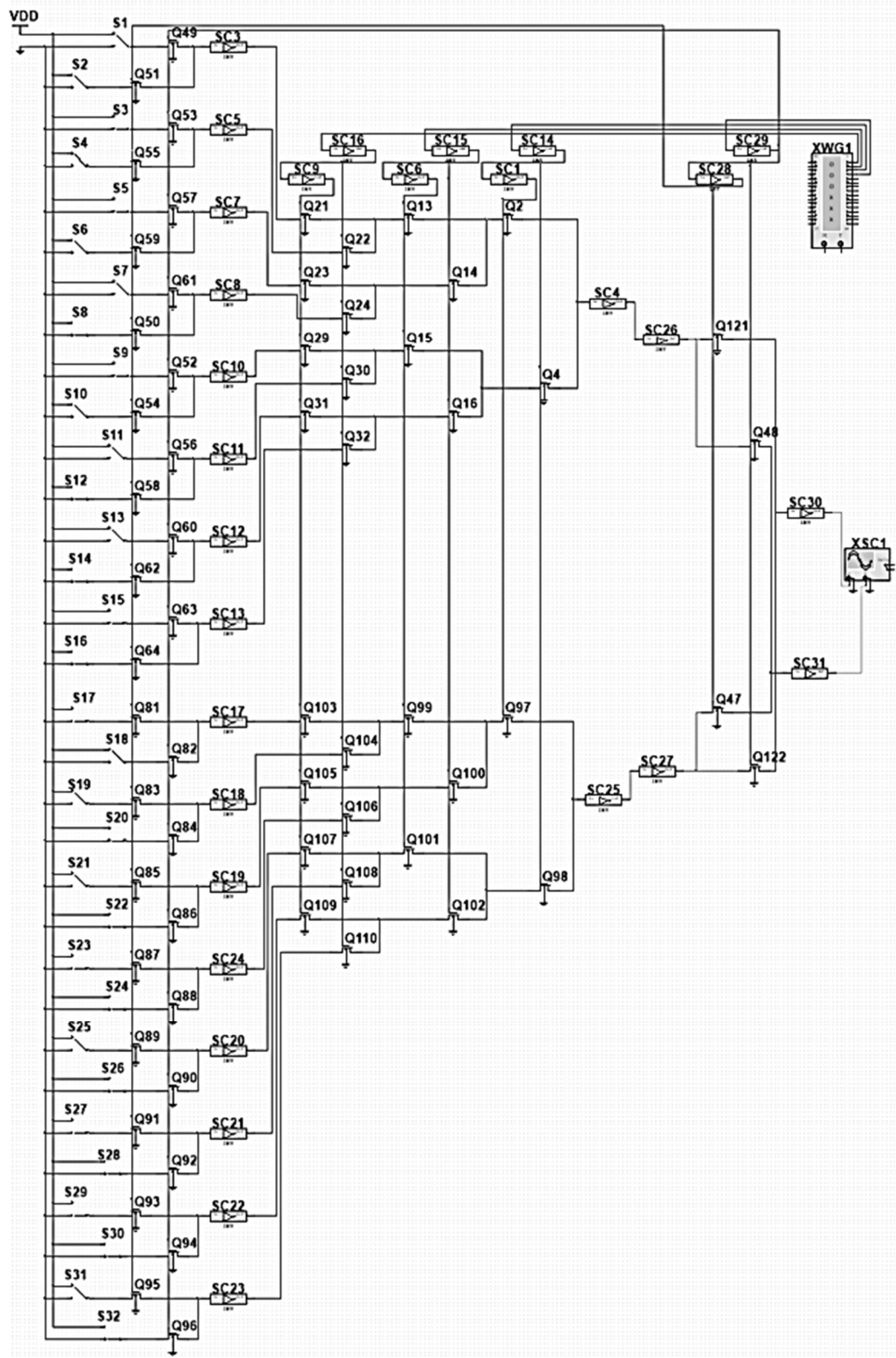


Рисунок 4.11 – Динамическое моделирование 4-LUT, вычисляющего значение двух логических функций одновременно

4.1.3. Топологическое моделирование n – LUT, реализующего одновременно 2^v функций от одних и тех же переменных

Топологии предлагаемых элементов разрабатывались в системе автоматизированного проектирования интегральных схем Microwind [83], с использованием планарной технологии 32 нм. Создание топологии представляет собой формирование слоев различной проводимости в масштабе, предусмотренной технологическими нормами. Топология транзистора n -типа (nMOS N-type metal-oxide-semiconductor) включает затвор, контактные окна сток-исток, металлизацию на контактных окнах и область n проводимости (рисунок 4.12 а). Топология транзистора p -типа (pMOS P-type metal-oxide-semiconductor) включает затвор, контактные окна сток-исток, металлизацию на контактных окнах, область p проводимости и карман n -типа (рисунок 4.12 б).

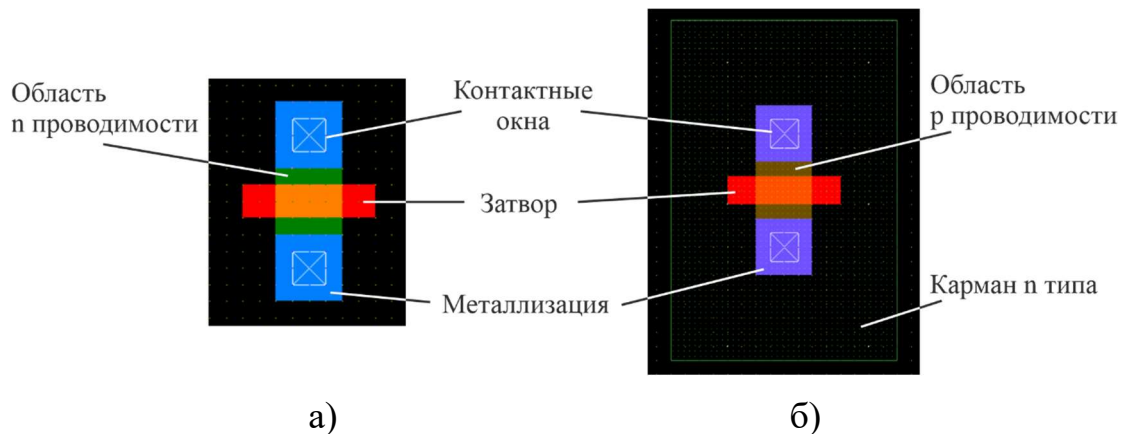


Рисунок 4.12 – Топологии nMOS и pMOS транзисторов

Топология инвертора реализуется с помощью комбинации транзисторов n и p типов, затворы которых объединены. Вход инвертора подключается к затвору, а выход подключается к общему стоку (рисунок 4.13).

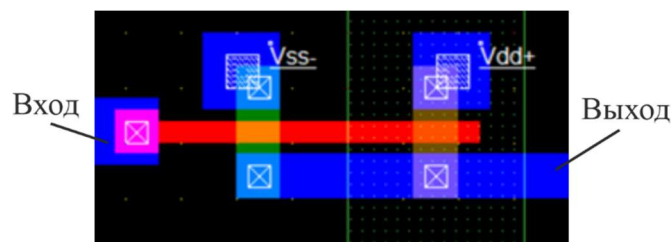


Рисунок 4.13 – Топология инвертора по технологии 32 нм

Топология 3-LUT [35] вычисляющая две функции одновременно представлена на рисунке 4.14. Сигналы переменных задаются с помощью $clock1$, $clock2$, $clock3$, которые формируют код Грея. Результат вычисления логических функций отображается на выходах $s1$, $s2$. Конфигурация SRAM задается с помощью высокого ($Vdd+$) и низкого ($Vss-$) уровня напряжения соответствующие логической «1» и «0».

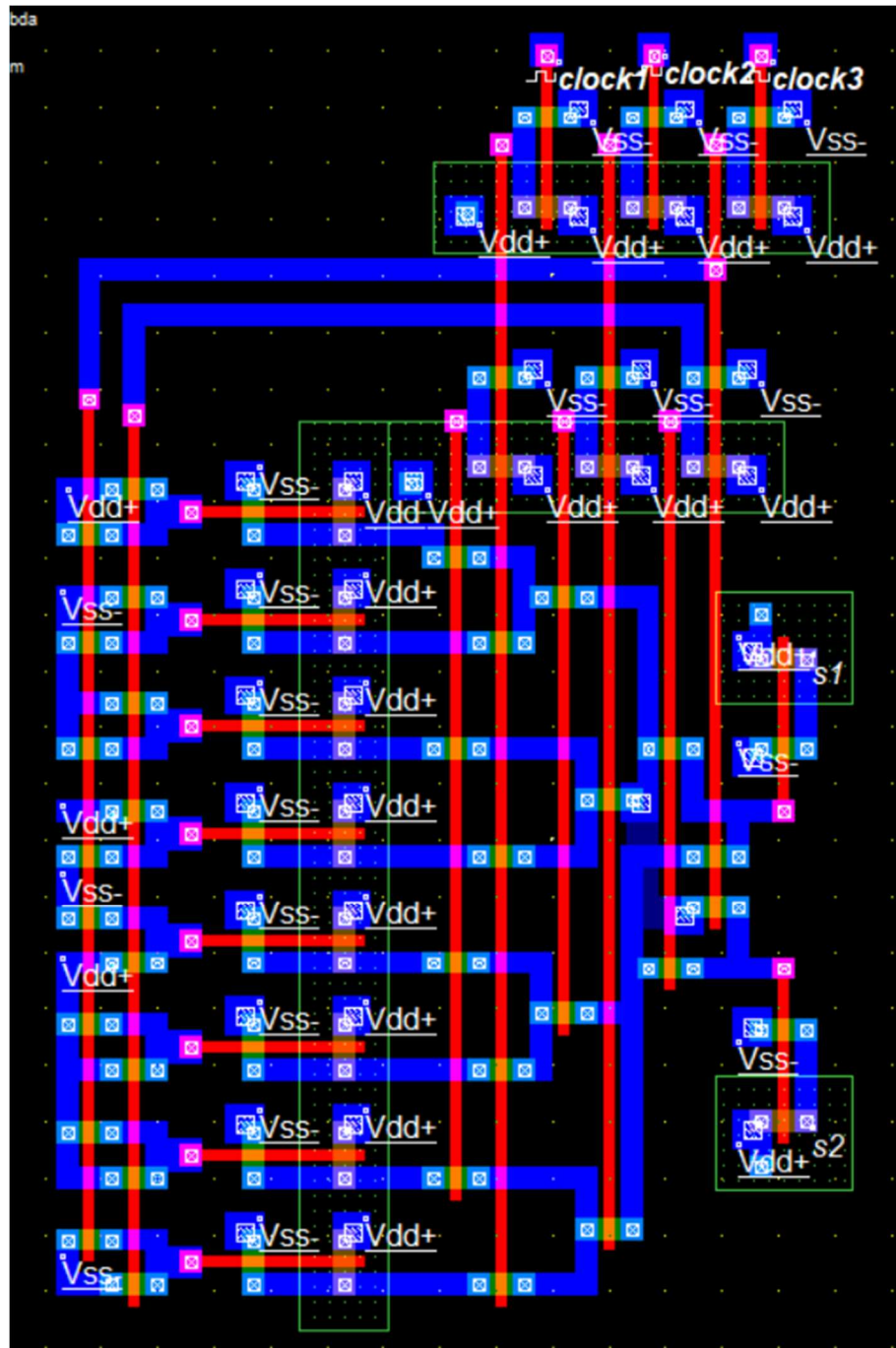


Рисунок 4.14 – Топология 3-LUT, реализующего две функции одновременно

Общее время моделирования составило 50 нс при установленном напряжении (V_{dd+}) в 0,8 В. Результат моделирования топологии 3-LUT реализующего функции исключающего «ИЛИ» и мажоритарной представлен на рисунке 4.15.

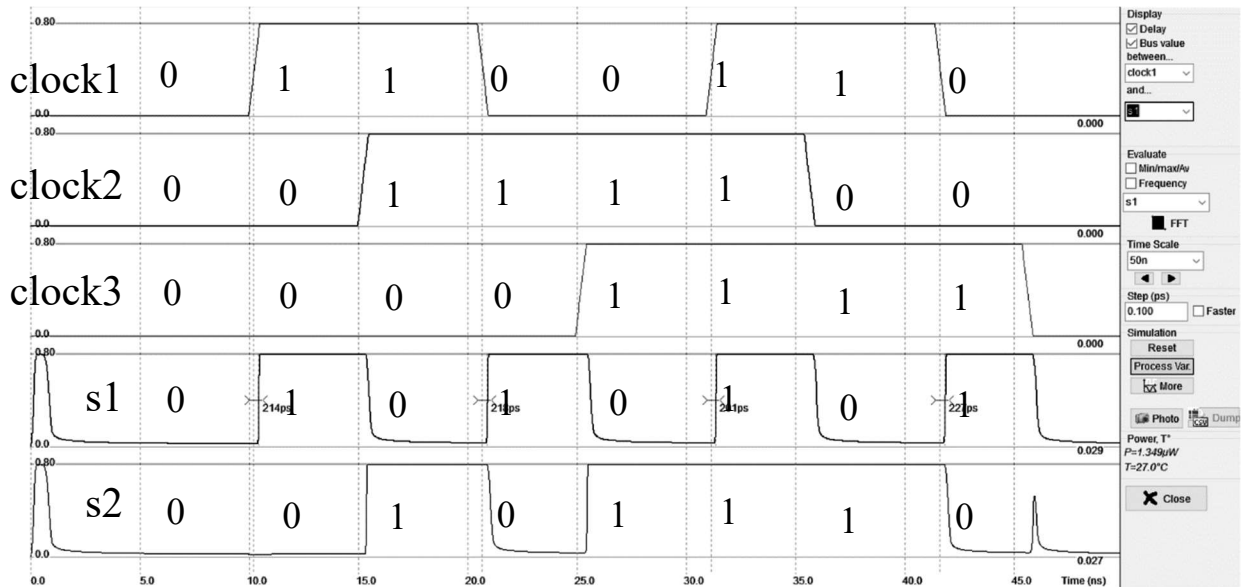


Рисунок 4.15 – Осциллограмма результата моделирования одновременного выполнения двух функций в 3-LUT

Максимальная задержка составила $T = 227$ пс. Общая потребляемая мощность $W = 1,349$ мкВт. Количество использованных транзисторов и количество занимаемой площади представлены на рисунке 4.16.

Layout Size	Electrical Properties
Width: 2.8 μm (141 lambda)	electrical nodes : 76/3000
Height: 4.6 μm (230 lambda)	nMOS devices : 48/2000
Surf: 13.0 μm ² (0.0 mm ²)	pMOS devices : 16/2000

Рисунок 4.16 – Количество используемых компонентов и занимаемой площади на кристалле 3-LUT с одновременной реализацией двух функций

Топология 3-LUT вычисляющая четыре функции одновременно представлена на рисунке 4.17. Сигналы переменных также задаются с помощью clock1, clock2, clock3. Результат вычисления логических функций отображается на

выходах s1, s2, s3, s4. Количество ячеек SRAM увеличено на 16 для конфигурирования дополнительных функций.

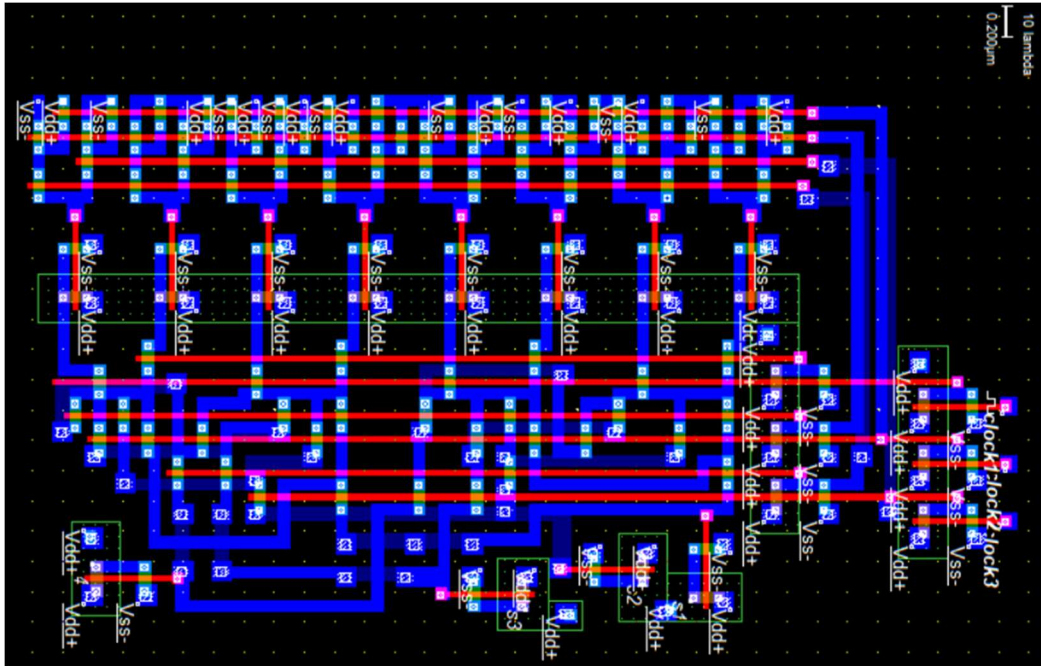


Рисунок 4.17 – Топология 3–LUT, реализующего четыре функции одновременно

Результат моделирования топологии 3–LUT реализующего функции исключающего «ИЛИ», мажоритарной, конъюнкции и дизъюнкции представлен на рисунке 4.18.

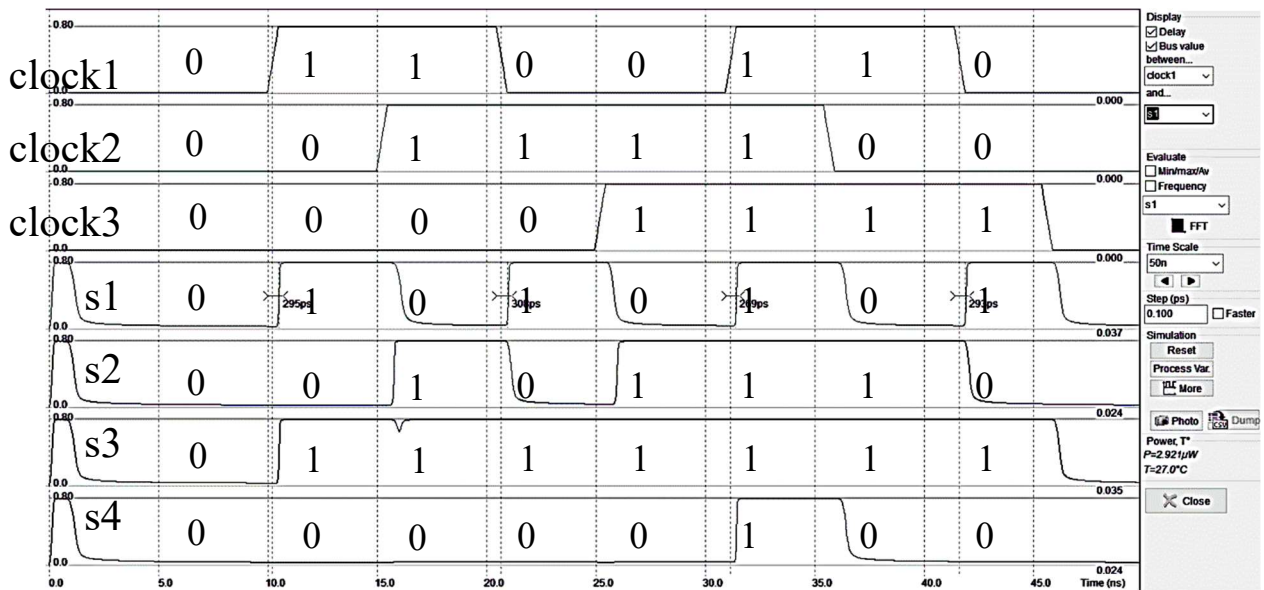


Рисунок 4.18 – Осциллограмма результата моделирования одновременного выполнения четырех функций в 3–LUT

Максимальная задержка составила $T = 308$ пс. Общая потребляемая мощность $W = 2,921$ мкВт. Количество использованных транзисторов и количество занимаемой площади представлены на рисунке 4.16.

Layout Size	Electrical Properties
Width: 3.5 μ m (177 lambda)	electrical nodes : 121/3000
Height: 6.6 μ m (328 lambda)	nMOS devices : 98/2000
Surf: 23.2 μ m ² (0.0 mm ²)	pMOS devices : 18/2000

Рисунок 4.19 – Количество используемых компонентов и занимаемой площади на кристалле 3–LUT с одновременной реализацией четырех функций

Топология 4–LUT, вычисляющая две функции одновременно представлена на рисунке 4.20. Сигналы переменных задаются с помощью clock1, clock2, clock3, clock4 которые формируют код Грея. Результат вычисления логических функций отображается на выходах s1, s2. Общее количество используемых ячеек памяти SRAM составляет 32. Дополнительно введены пара инверторов для восстановления сигнала после каждого третьего nMOS транзистора.

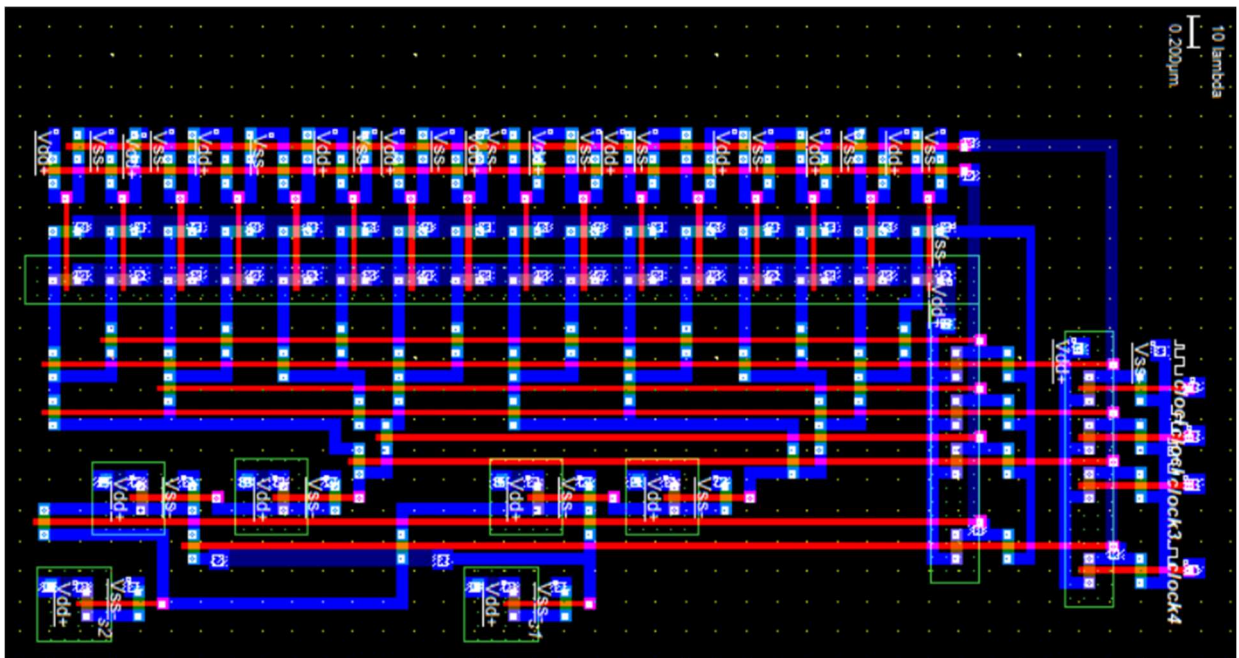


Рисунок 4.20 – Топология 4–LUT, реализующего две функции одновременно

Результат моделирования топологии 4-LUT реализующего функции мажоритарной и исключающего «ИЛИ» представлен на рисунке 4.21. Общее время моделирование составило 100 нс.

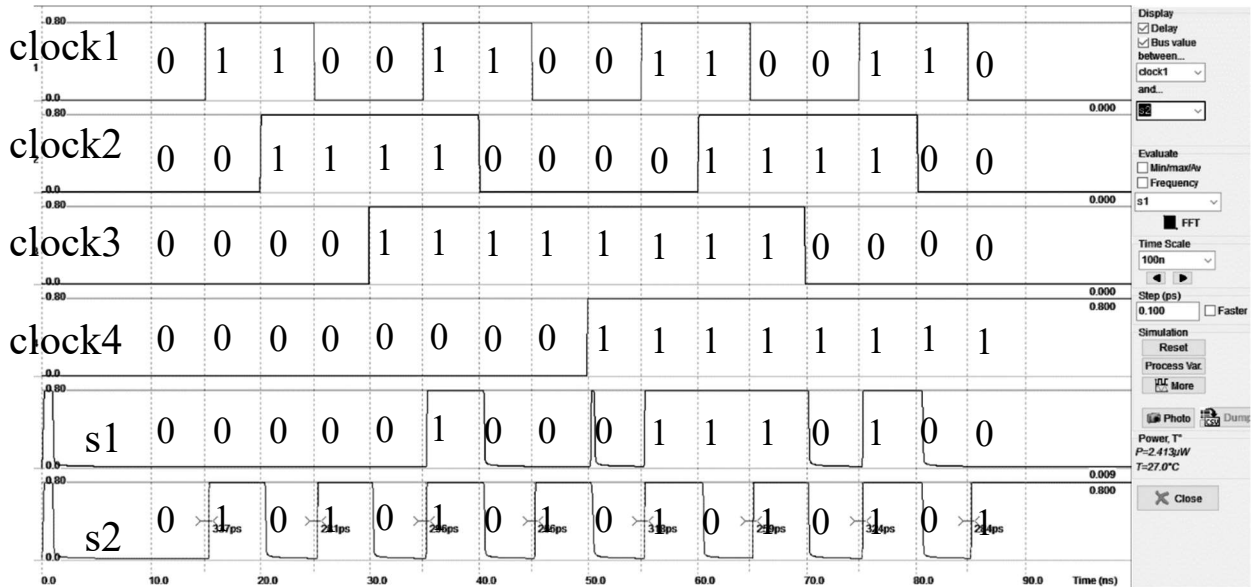


Рисунок 4.21 – Осциллограмма результата моделирования одновременного выполнения двух функций в 4-LUT

Максимальная задержка составила $T = 337$ пс. Общая потребляемая мощность $W = 2,413$ мкВт. Количество использованных транзисторов и количество занимаемой площади представлены на рисунке 4.22.

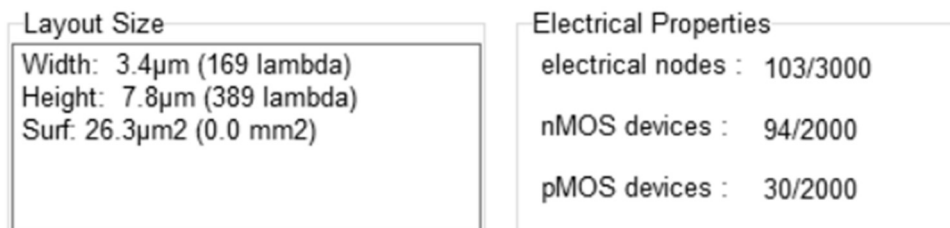


Рисунок 4.22 – Количество используемых компонентов и занимаемой площади на кристалле 4-LUT с одновременной реализацией двух функций

Топология 4-LUT, вычисляющая четыре функции одновременно представлена на рисунке 4.23. Сигналы переменных также задаются с помощью clock1, clock2, clock3, clock4. Результат вычисления логических функций отображается на выходах s1, s2, s3, s4. Количество ячеек SRAM увеличено до 64 для конфигурирования дополнительных функций.

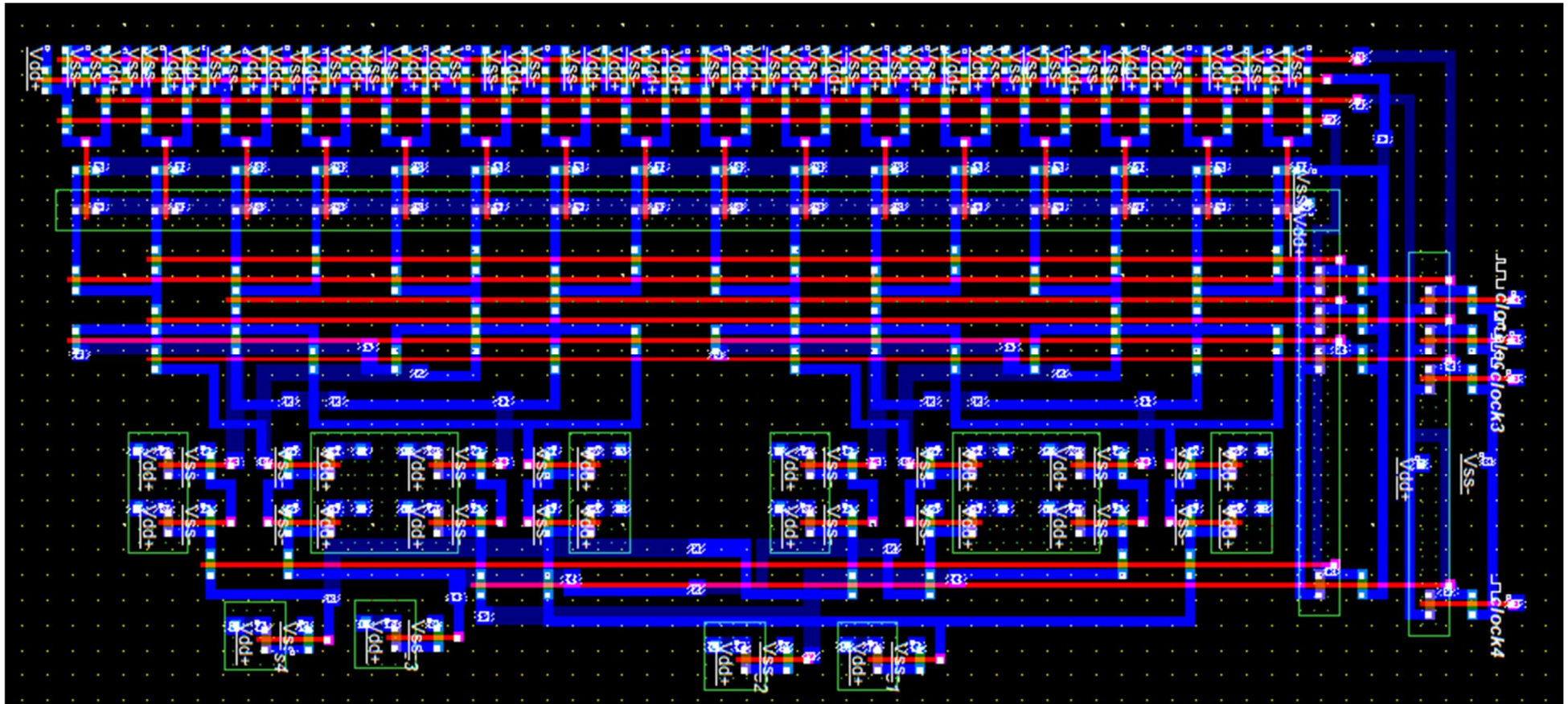


Рисунок 4.23 – Топология 4–LUT, реализующего четыре функции одновременно

Результат моделирования топологии 4-LUT, реализующего функции исключающего «ИЛИ», мажоритарной, дизъюнкции и конъюнкции представлен на рисунке 4.24.

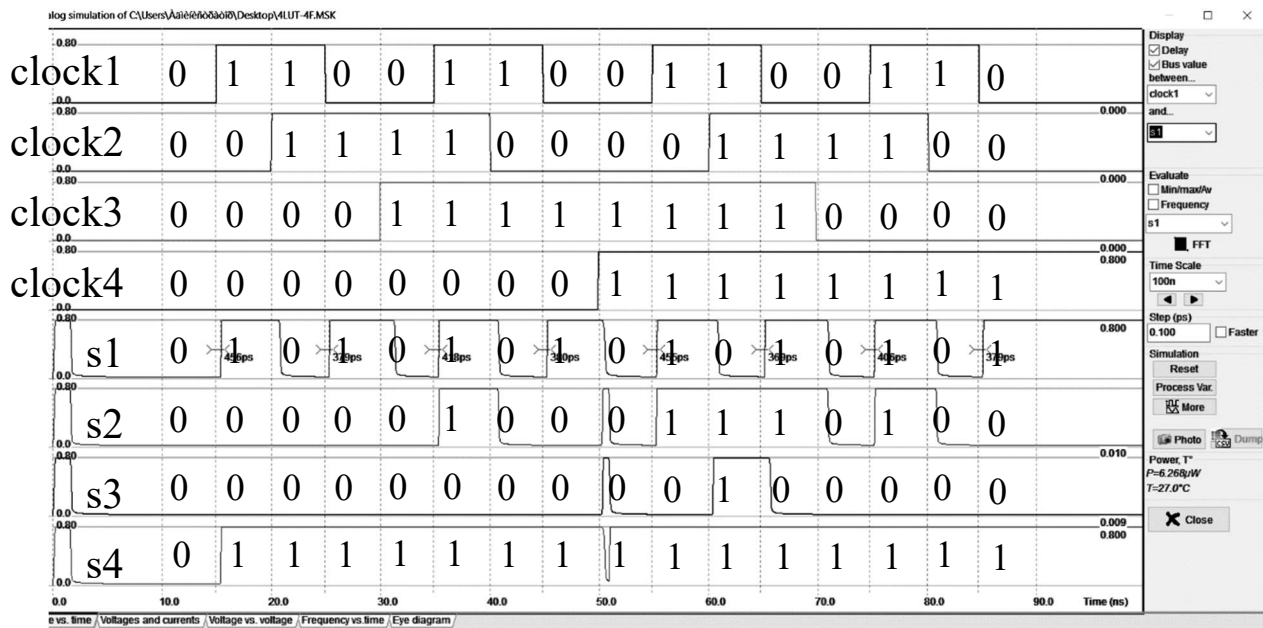


Рисунок 4.24 – Осциллограмма результата моделирования одновременного выполнения четырех функций в 4-LUT

Максимальная задержка составила $T = 456$ пс. Общая потребляемая мощность $W = 6,268$ мкВт. Количество использованных транзисторов и количество занимаемой площади представлены на рисунке 4.25.

Layout Size	Electrical Properties
Width: $5.2\mu\text{m}$ (258 lambda)	electrical nodes : 214/3000
Height: $11.9\mu\text{m}$ (596 lambda)	nMOS devices : 188/2000
Surf: $61.5\mu\text{m}^2$ (0.0 mm ²)	pMOS devices : 44/2000

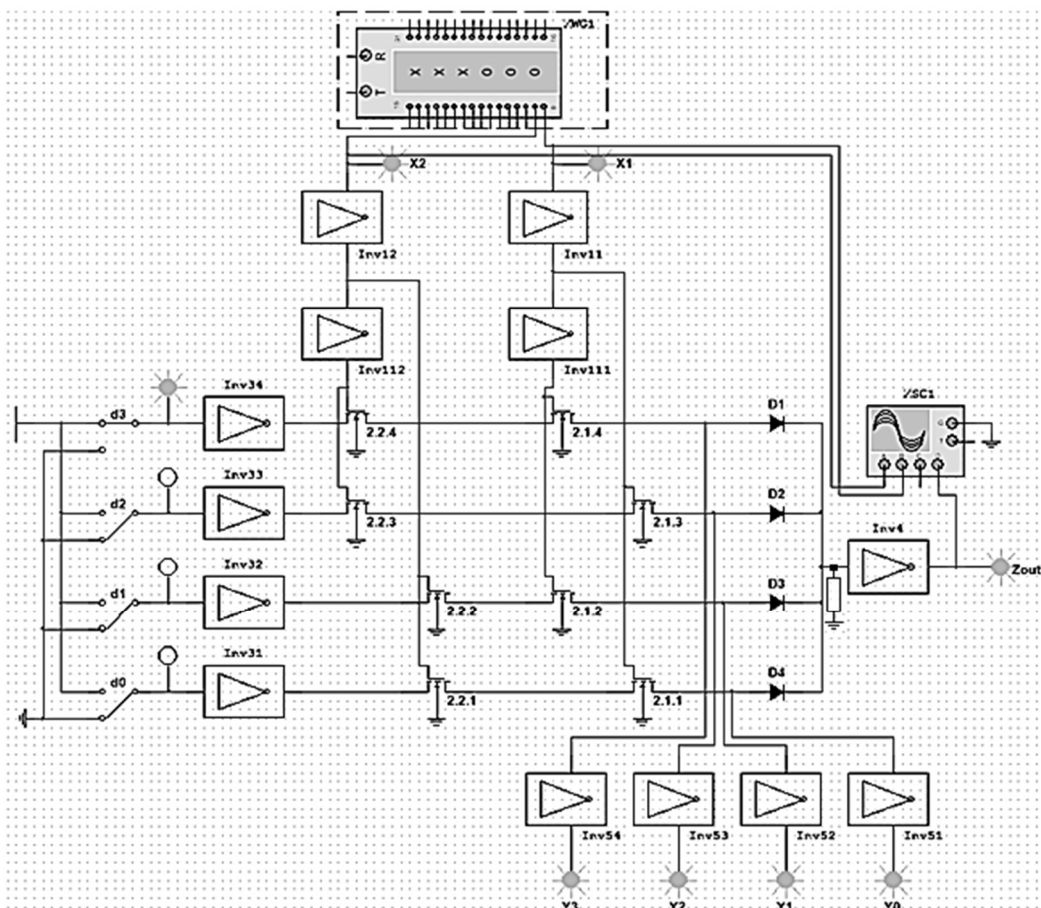
Рисунок 4.25 – Количество используемых компонентов и занимаемой площади на кристалле 4-LUT с одновременной реализацией четырех функций

4.2. Схмотехническое моделирование элементов LUT, вычисляющих значение заданной логической функции и дешифрацию набора переменных

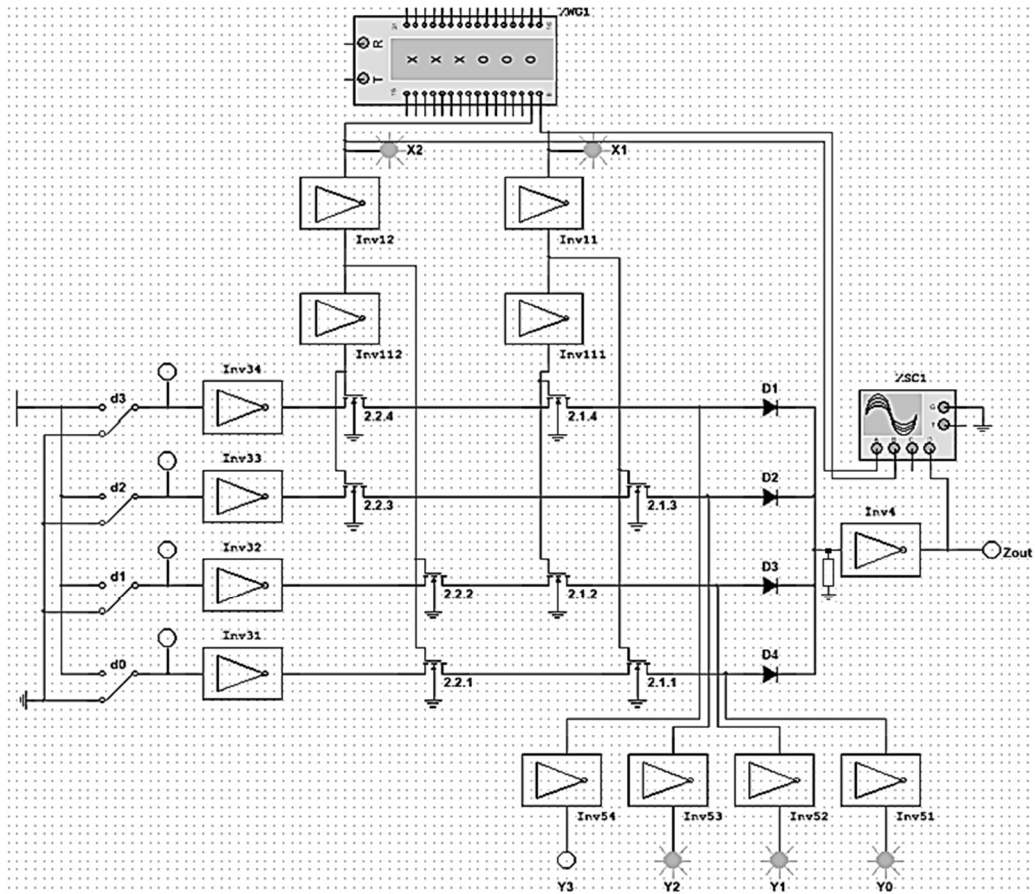
4.2.1. Моделирование адаптивного элемента LUT с одноуровневым 2^n деревом

Моделирование схемы адаптивного логического элемента 2-LUT вычисляющего значение заданной логической функции или дешифрацию набора переменных в зависимости от сигнала настройки с подтягивающим к уровню нуля резистором на входе инвертора представлено на рисунке 4.25.

При подаче входного набора $A=1, B=1$ активируется верхняя ветвь, передающая сигнал с ключа $d3$, для вычисления значения функции, отображаемой на Z_{out} (рисунок 4.25а). При изменении значения ключа $d3$ на «0» (рисунок 4.25б) происходит дешифрация, отображаемая на выводе $Y3$.



a)



б)

Рисунок 4.25 – Адаптивный 2–LUT с подтягивающим к уровню нуля резистором: а) в режиме вычисления заданной логической функции; б) в режиме дешифрации входного набора

Моделирование схемы адаптивного логического элемента 3–LUT вычисляющего значение заданной логической функции или дешифрацию набора переменных с помощью конstituент нуля представлено на рисунке 4.26.

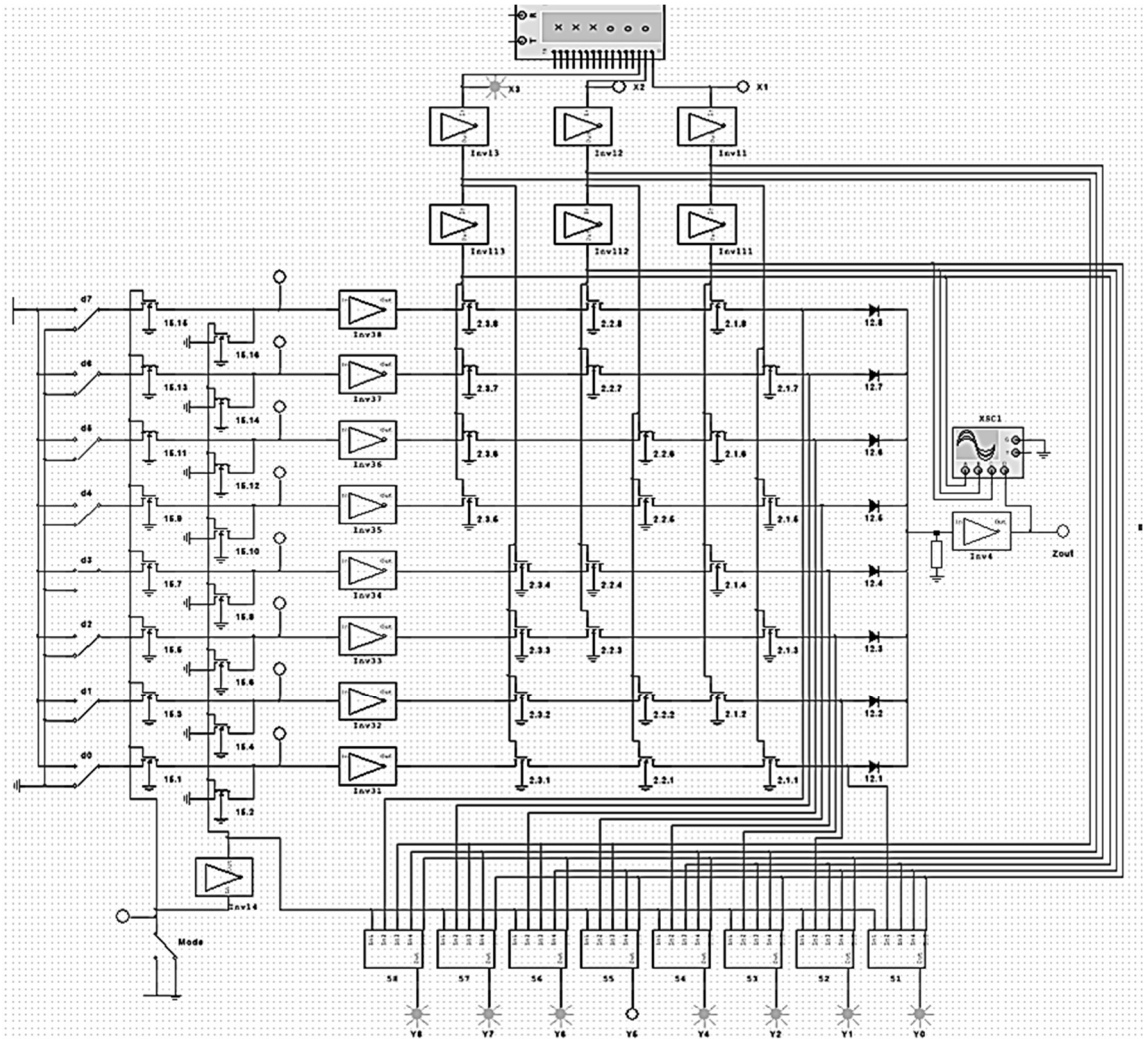


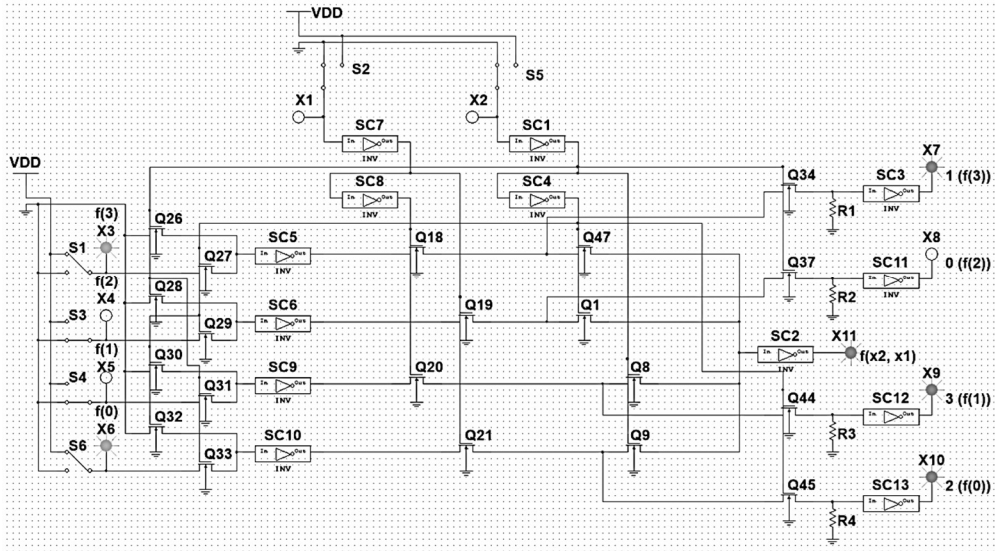
Рисунок 4.26 – Адаптивный 3–LUT с подтягивающим к уровню нуля резистором в режиме дешифрации входного набора

Управление режимом вычисления осуществляется с помощью ключа Mode. Каждый блок конститuenty S соответствует дешифрации ключа d эмитирующей ячейку памяти SRAM. При дешифрации набора невозможно одновременное вычисление основной функции.

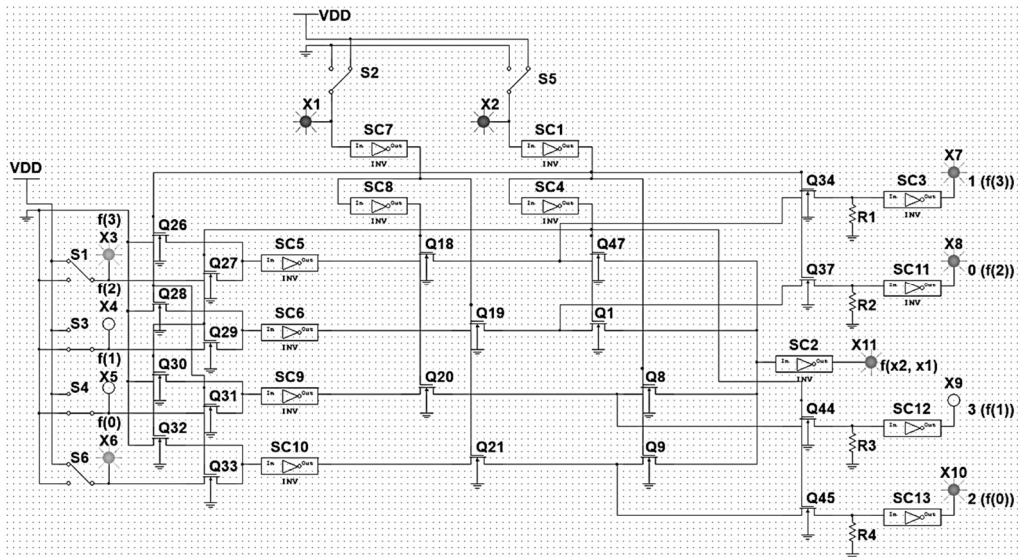
4.2.2. Статическое моделирование логического элемента, вычисляющего значение заданной логической функции и дешифрацию набора переменных одновременно с использованием неактивной половины дерева транзисторов

Моделирование схемы 2–LUT с подтягивающим к уровню нуля резистором, вычисляющий значение заданной логической функции и дешифрацию набора переменных одновременно представлено на рисунке 4.27. При установке ключей

$S2=0$, $S5=0$ (рисунок 4.27а), которые имитируют значения переменных активизируется цепь из транзисторов Q9, Q21, Q33 для передачи значения функции на выход X11 через инверторы SC10, SC2. В это же время активизируется цепь дешифрации Q37, Q19, Q28, исток последнего транзистора которой подключен к земле («0»).



а)



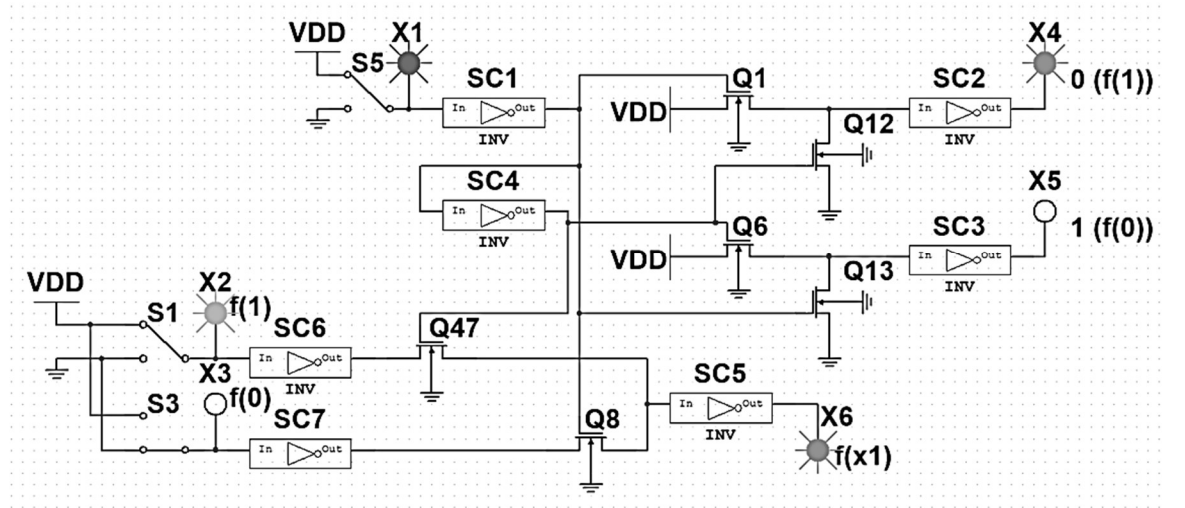
б)

Рисунок 4.27 – 2–LUT с подтягивающим к уровню нуля резистором, вычисляющий значение заданной логической функции и дешифрацию набора переменных одновременно: а) набор 00, выход функции 1, дешифрация 0; б) набор 11, выход функции 1, дешифрация 3

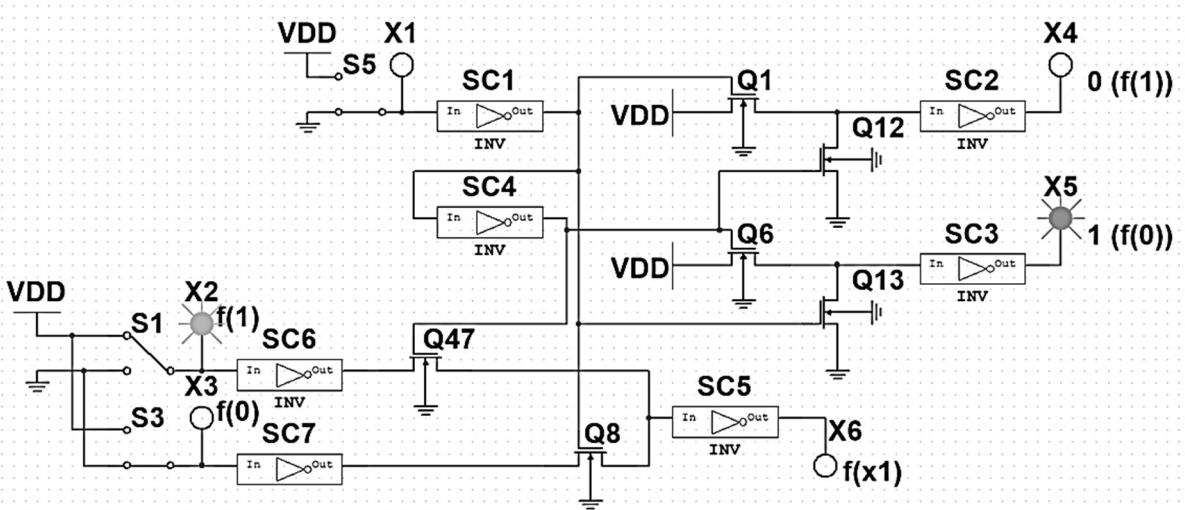
Резистор R2 способствует установке низкого уровня сигнала на индикаторе дешифрации X8, что сигнализирует об активной цепи $f(0)$, при использовании цепи второй половины ветви $f(2)$. Аналогичным образом реализуется вычисление основной функции и дешифрация набора при установке ключей $S2=1$, $S5=1$ (рисунок 4.27б). Вычисление основной функции происходит по цепи Q47, Q18, Q27 одновременно с дешифрацией переменной $f(3)$, которая использует ветвь $f(1)$. Использование резисторов при реализации схемы на кристалле приводит к значительному увеличению площади кристалла, что является одной из основных характеристик по оптимизации. Следовательно, следует рассмотреть моделирование схемы, которая реализует одновременное вычисление функции и дешифрацию набора без использования подтягивающих резисторов.

4.2.3. Статическое моделирование логического элемента выполняющего дешифрацию набора переменных одновременно без использования неактивной половины дерева транзисторов

Моделирование схемы 1-LUT выполняющего вычисление логической функции и дешифрацию набора с обеспечением ортогональности сигналов по входам инверторов дешифрации приведено на рисунке 4.28. Для каждого 1-LUT дополнительно используется 4 транзистора. Дополнительные инверторы соответствуют количеству используемых ячеек SRAM. При задании переменной $S5 = 1$ (рисунок 4.28 а), активируется Q47 и на выход функции поступает сигнал S1. При этом также происходит активация транзистора Q6 и Q12 к истокам которых подключены соответственно высокий уровень сигнала (VDD) и низкий (GND). Сигналы, проходя через инверторы SC2 и SC3 отображаются индикаторами X4 и X5 и демонстрируют дешифрацию набора 0 и 1 соответственно. Аналогично происходит вычисление функции и дешифрация набора для переменной $S5 = 0$ (рисунок 4.28 б).



а)

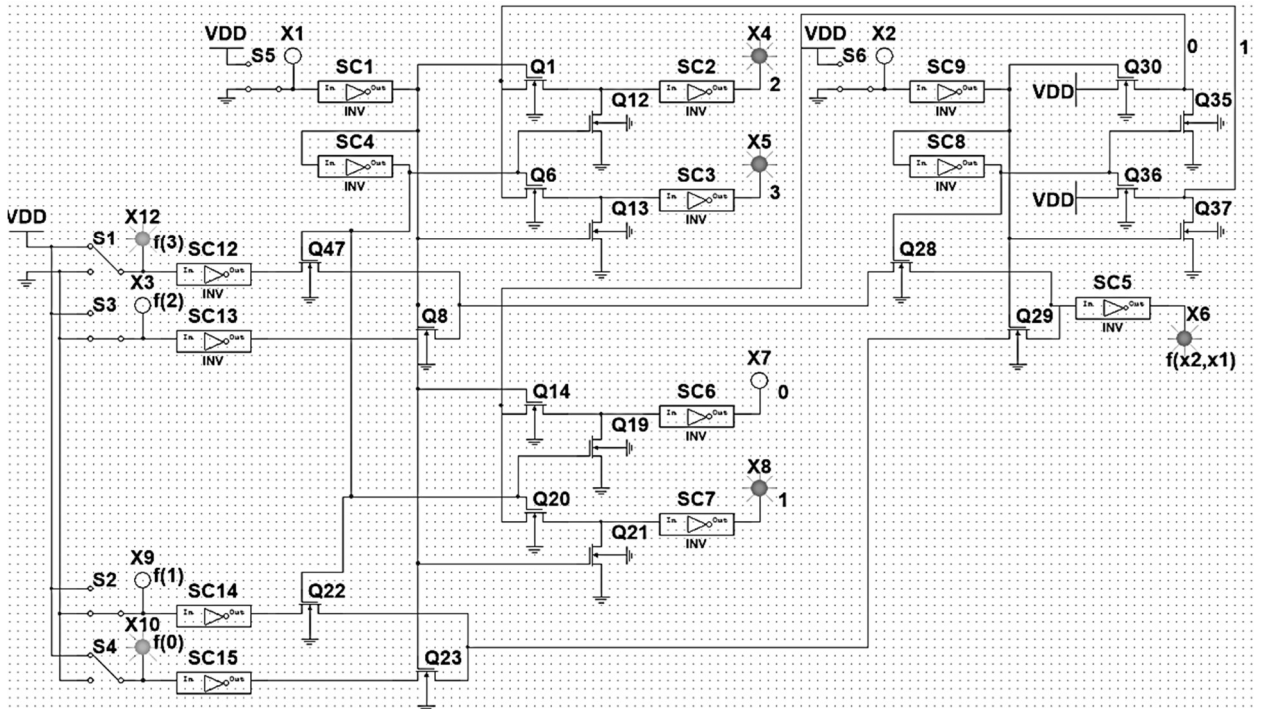


б)

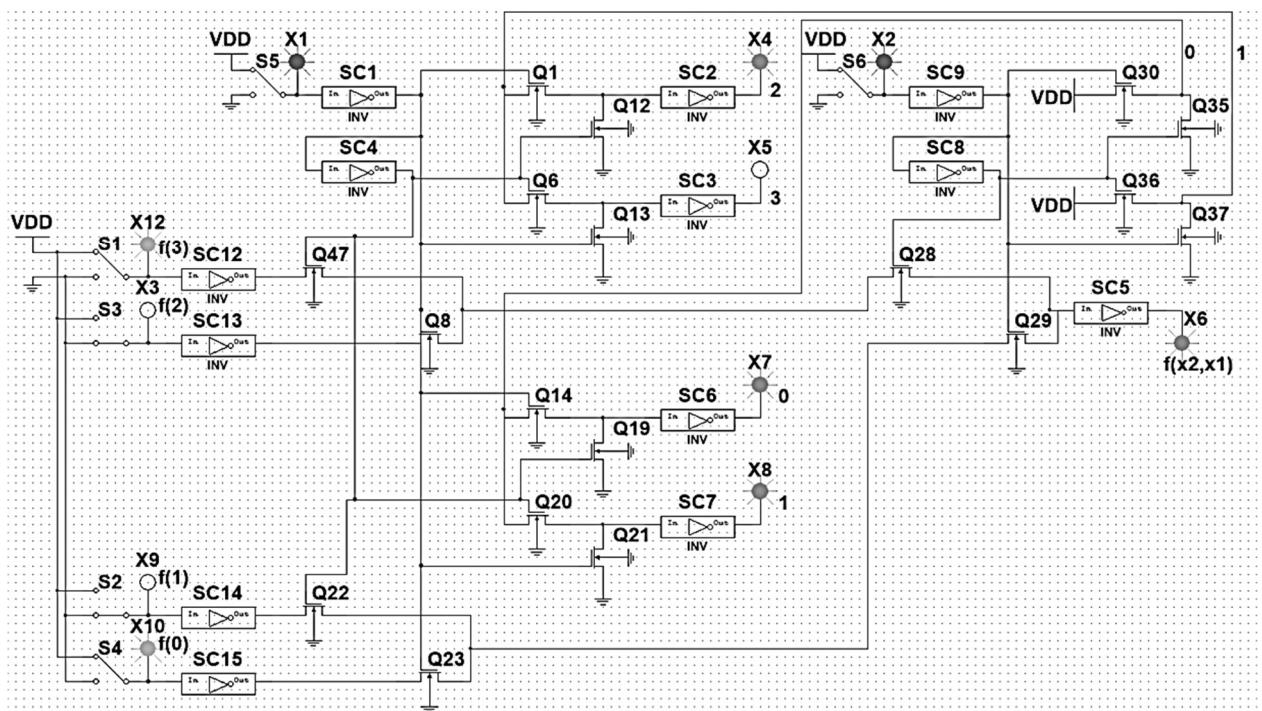
Рисунок 4.28 – 1–LUT с обеспечением ортогональности сигналов по входам инверторов дешифрации при задании функции отрицания: а) переменная = 1, выход функции 1, дешифрация 1; б) переменная = 0, выход функции 0, дешифрация 0

Моделирование схемы элемента 2–LUT, выполняющего вычисление логической функции с одновременной дешифровкой представлено на рисунке 4.29. Представленная схема 2–LUT состоит из трех элементов 1–LUT, причем стоки транзисторов дешифрации Q35 и Q37 подключаются к истокам транзисторов первого каскада Q1, Q6 и Q14, Q20 соответственно. Со стоков транзисторов

дешифрации первого каскада сигнал поступает уже непосредственно на выходной инвертор дешифрации.



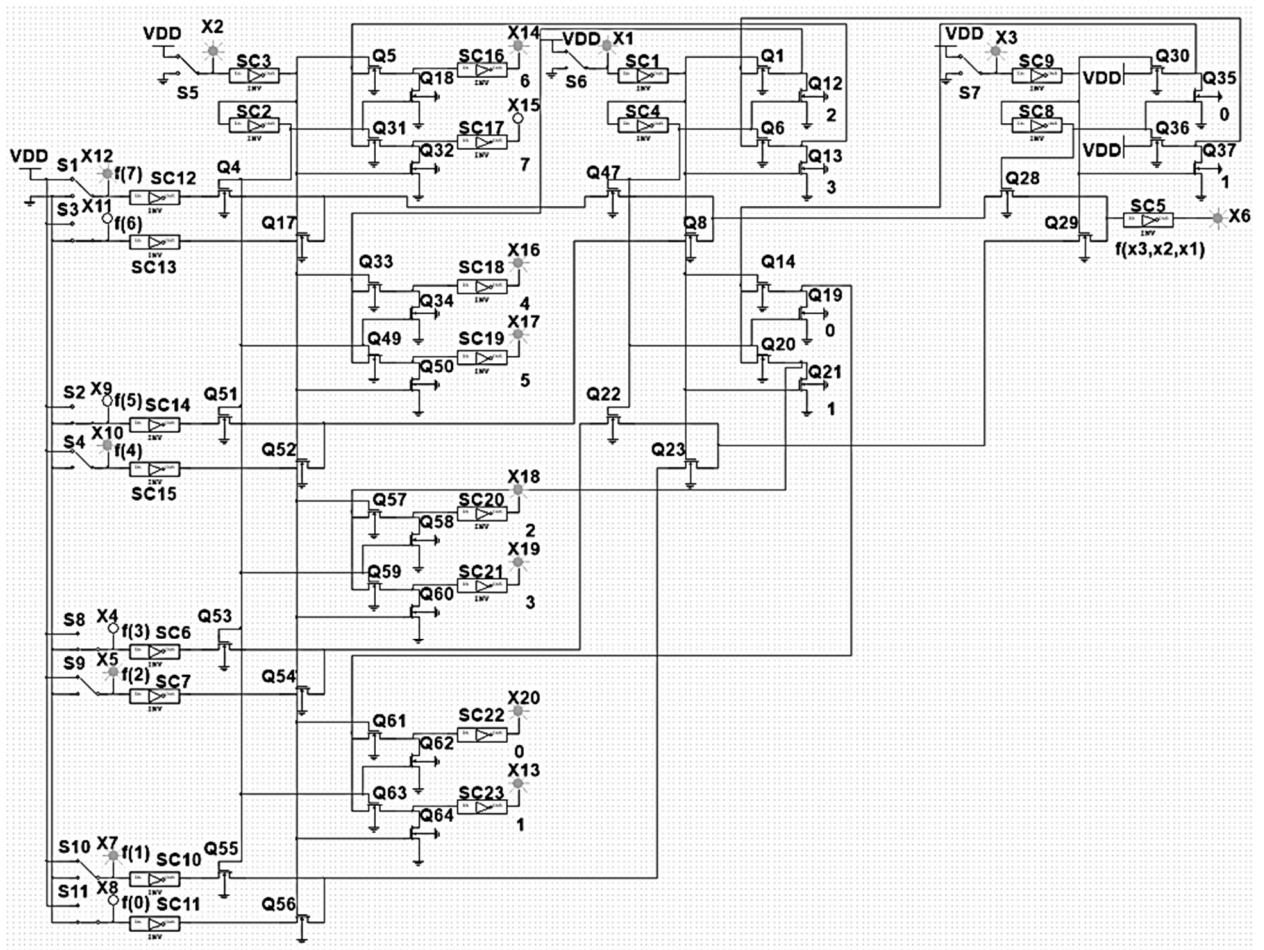
а)



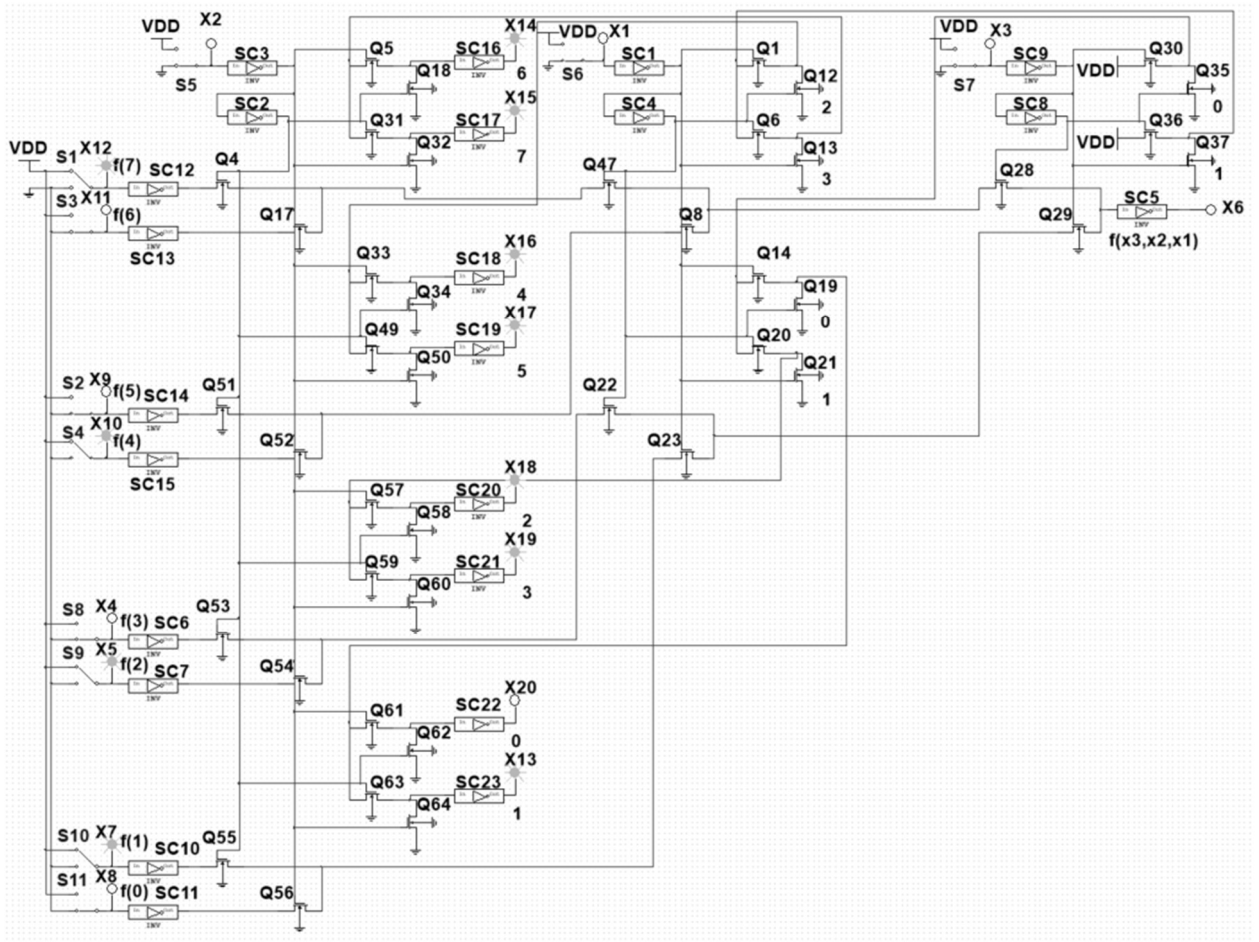
б)

Рисунок 4.29 – 2-LUT+DC а) набор 00, выход функции 1, дешифрация 0; б) набор 11, выход функции 1, дешифрация 3

Моделирование схемы элемента 3-LUT, выполняющего вычисление логической функции с одновременной дешифровкой продемонстрировано на рисунке 4.30. Схема 3-LUT состоит из семи элементов 1-LUT с каскадным подключением дополнительных транзисторов дешифровки. Таким образом задание набора $S5=1, S6=1, S7=1$ активирует ветвь $Q28, Q47, Q4$ для вычисления функции и одновременно активируется ветвь $Q36, Q6, Q31$ для выполнения дешифровки.



a)



б)

Рисунок 4.30 – 3–LUT+DC, построенного из предлагаемых 1–LUT а) набор 111, выход функции 1, дешифрация 7; б) набор 000, выход функции 0, дешифрация 0

Статическое моделирование по всем возможным наборам подтвердило правильность вычисления функции и дешифрацию входных наборов. Рекомендуется провести динамическое моделирование предложенных решений.

4.2.4. Динамическое моделирование логического элемента, вычисляющего значение заданной логической функции и дешифрацию набора переменных одновременно с использованием неактивной половины дерева транзисторов

Для реализации динамического моделирования предложенных моделей статические ключи переменных заменены на генератор сигналов XWG1, выходной индикатор выполнения функции заменен на осциллограф с двумя каналами XSC2. Выходные индикаторы дешифрации заменены на четырехканальный осциллограф XSC1. Результирующая схема моделирования логического элемента двух

переменных (2-LUT) с одновременным выполнением дешифрации приведена на рисунке 4.31.

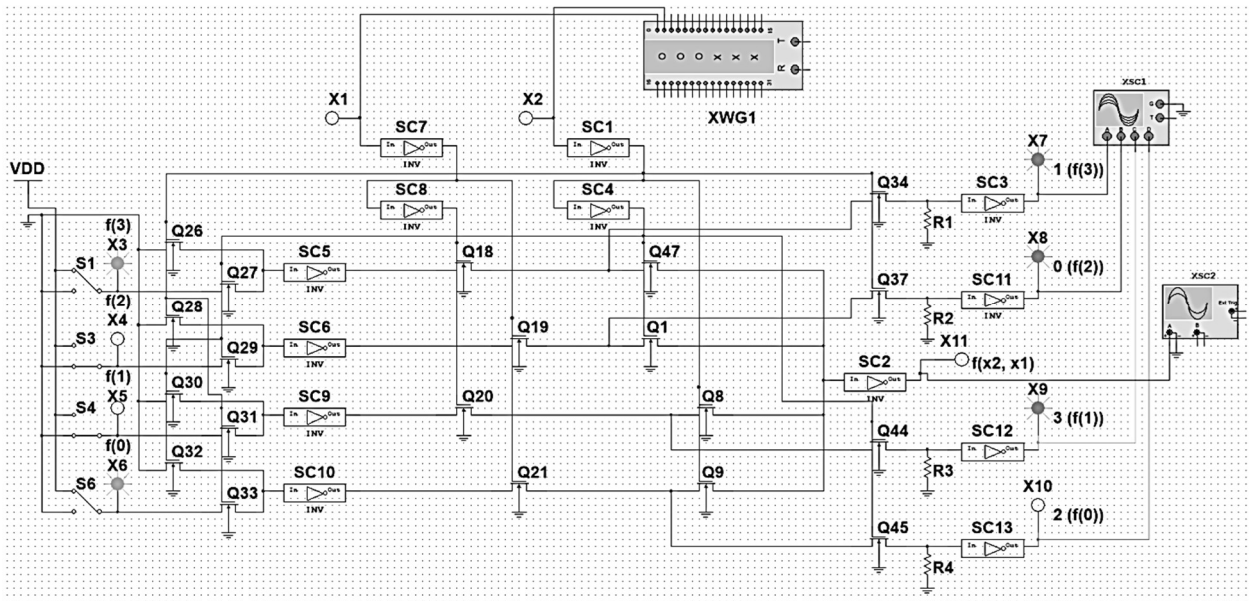


Рисунок 4.31 – 2-LUT с подтягивающим к уровню нуля резистором, вычисляющий значение заданной логической функции и дешифрацию набора переменных одновременно

Сигналы входных переменных заданы в генераторе XWG1 в форме кода Грея и представлены на рисунке 4.32.

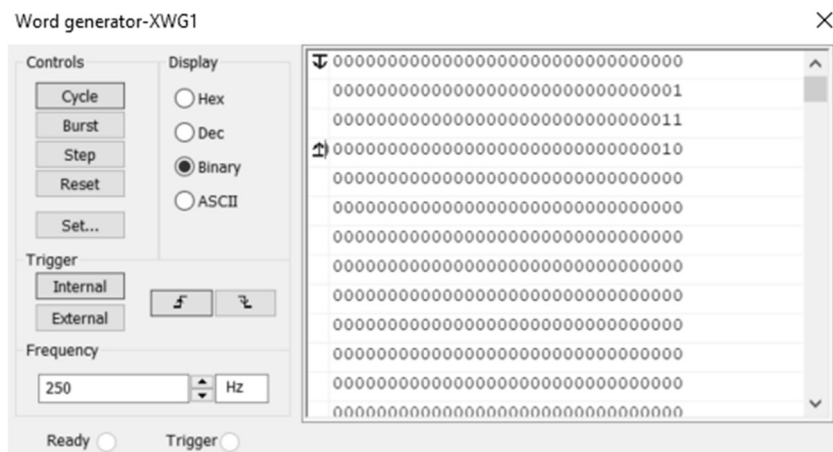


Рисунок 4.32 – Код входных сигналов переменных логического элемента 2-LUT с дешифрацией в форме кода Грея

Результаты моделирования логического элемента 2-LUT представлены на рисунке 4.33. В части а) приведен результат выполнения логической функции. В

части б) приведена осциллограмма дешифрации выполнения логической функции, где «0» является индикатором передачи сигнала.

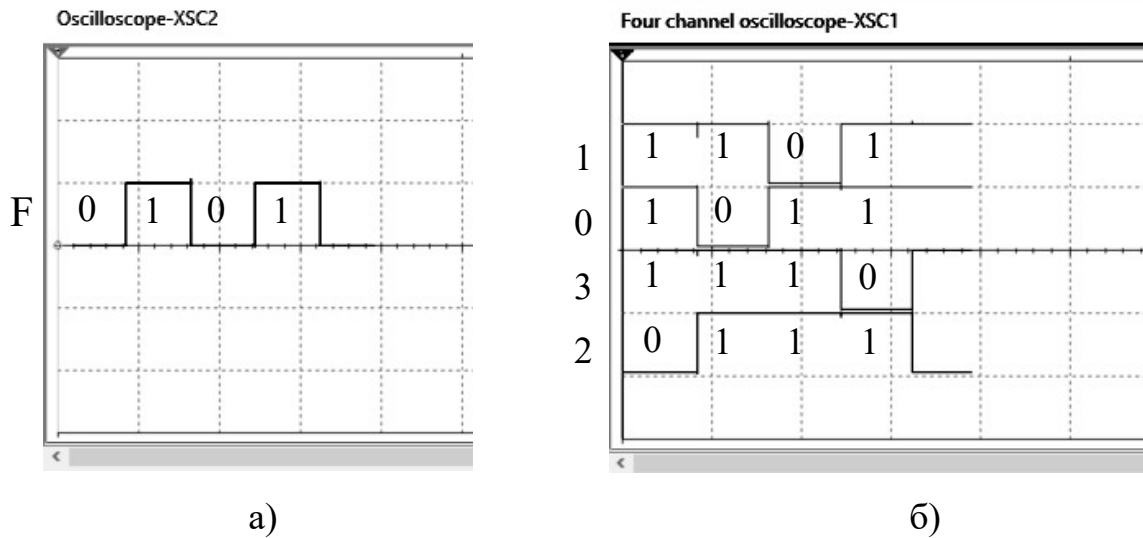


Рисунок 4.33 – Осциллограммы результатов моделирования одновременного выполнения логической функции (а) и дешифрации (б) в 2–LUT

4.2.5. Динамическое моделирование логического элемента выполняющего дешифтацию набора переменных одновременно без использования неактивной половины дерева транзисторов

Динамическое моделирование модели 2–LUT с использованием неактивной половины дерева транзисторов приведено на рисунке 4.34. Статические ключи переменных заменены на генератор сигналов XWG1, выходной индикатор выполнения функции заменен на осциллограф с двумя каналами XSC2. Выходные индикаторы дешифрации заменены на четырехканальный осциллограф XSC1.

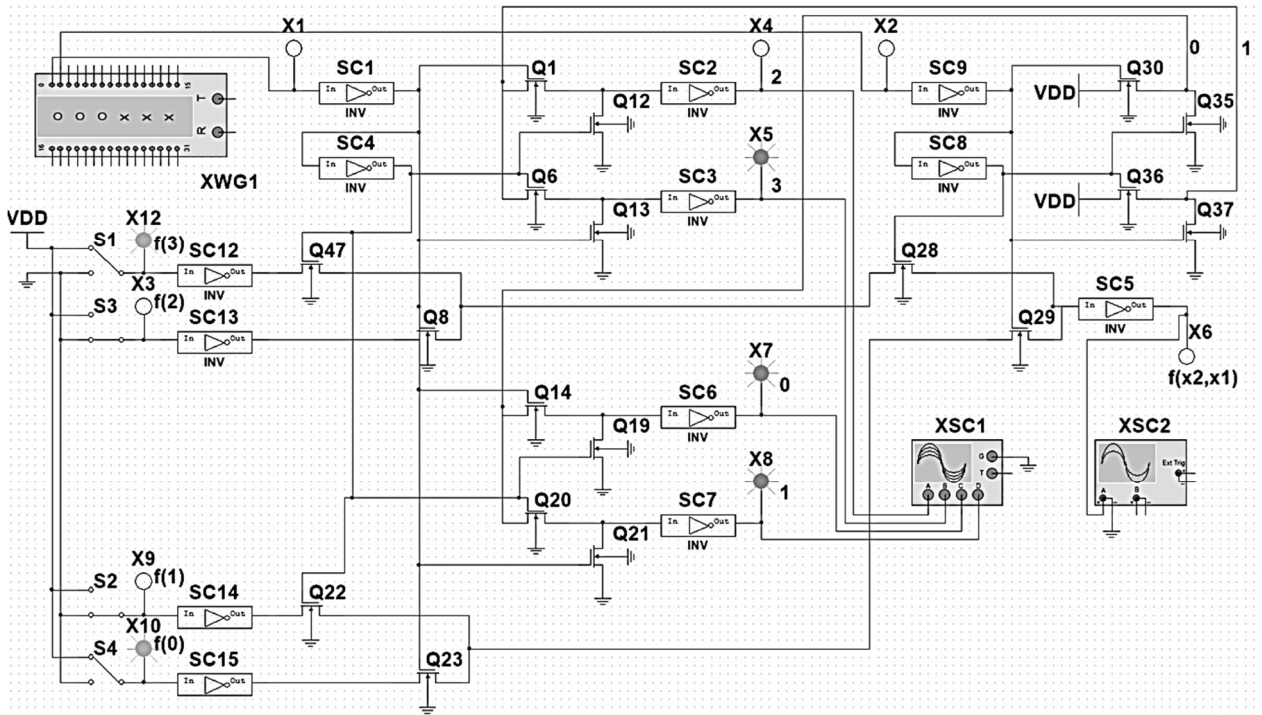


Рисунок 4.34 – 2–LUT, вычисляющий значение функции и дешифрацию набора переменных с использованием неактивной половины дерева транзисторов

Сигналы входных переменных заданы в генераторе XWG1 в форме кода Грея и представлены на рисунке 4.35.

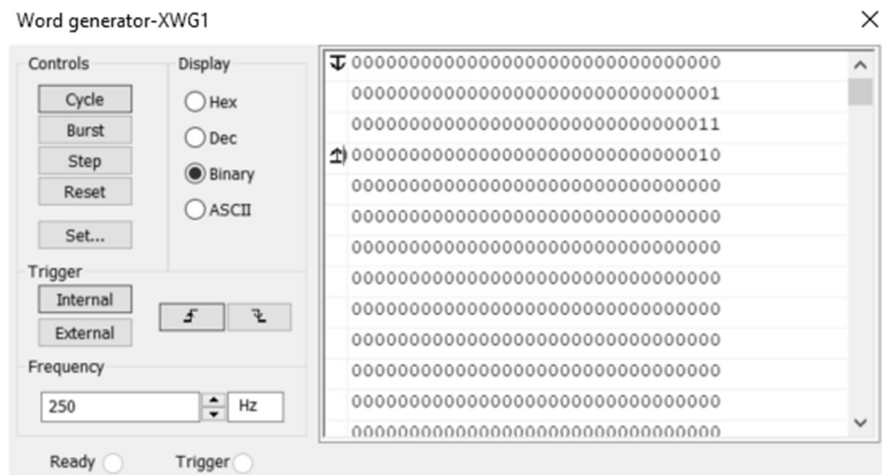


Рисунок 4.35 – Код входных сигналов переменных логического элемента 2–LUT с дешифрацией в форме кода Грея

Результаты моделирования логического элемента 2–LUT представлены на рисунке 4.36. В части а) приведен результат выполнения логической функции. В

части б) приведена осциллограмма дешифрации выполнения логической функции, где «0» является индикатором передачи сигнала.

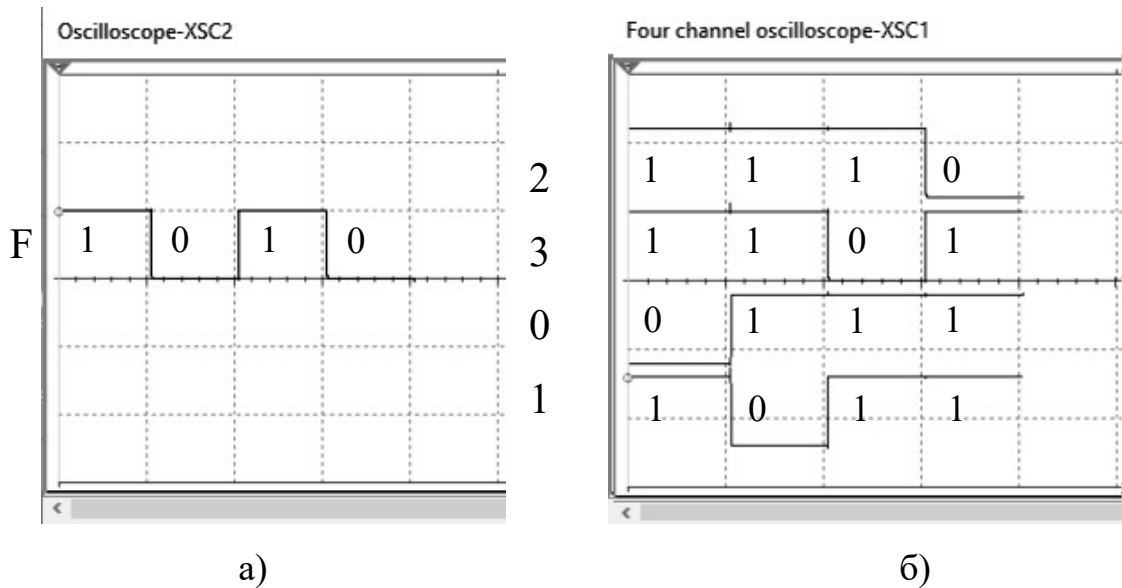


Рисунок 4.36 – Осциллограммы результатов моделирования одновременного выполнения логической функции (а) и дешифрации (б) в 2–LUT

Динамическое моделирование модели 3–LUT с использованием неактивной половины дерева транзисторов приведено на рисунке 4.37. Статические ключи переменных заменены на генератор сигналов XWG1, выходной индикатор выполнения функции заменен на осциллограф с двумя каналами XSC3. Выходные индикаторы дешифрации заменены на четырехканальные осциллографы XSC1 и XSC2, где первая половина дешифрации 0,1,2,3 подключена к осциллографу XSC2, а вторая половина 4,5,6,7 ко осциллографу XSC1.

Сигналы входных переменных заданы в генераторе XWG1 в форме кода Грея и представлены на рисунке 4.38.

Результаты моделирования логического элемента 3–LUT представлены на рисунке 4.39. В части а) приведен результат выполнения логической функции. В части б) приведена осциллограмма дешифрации выполнения логической функции.

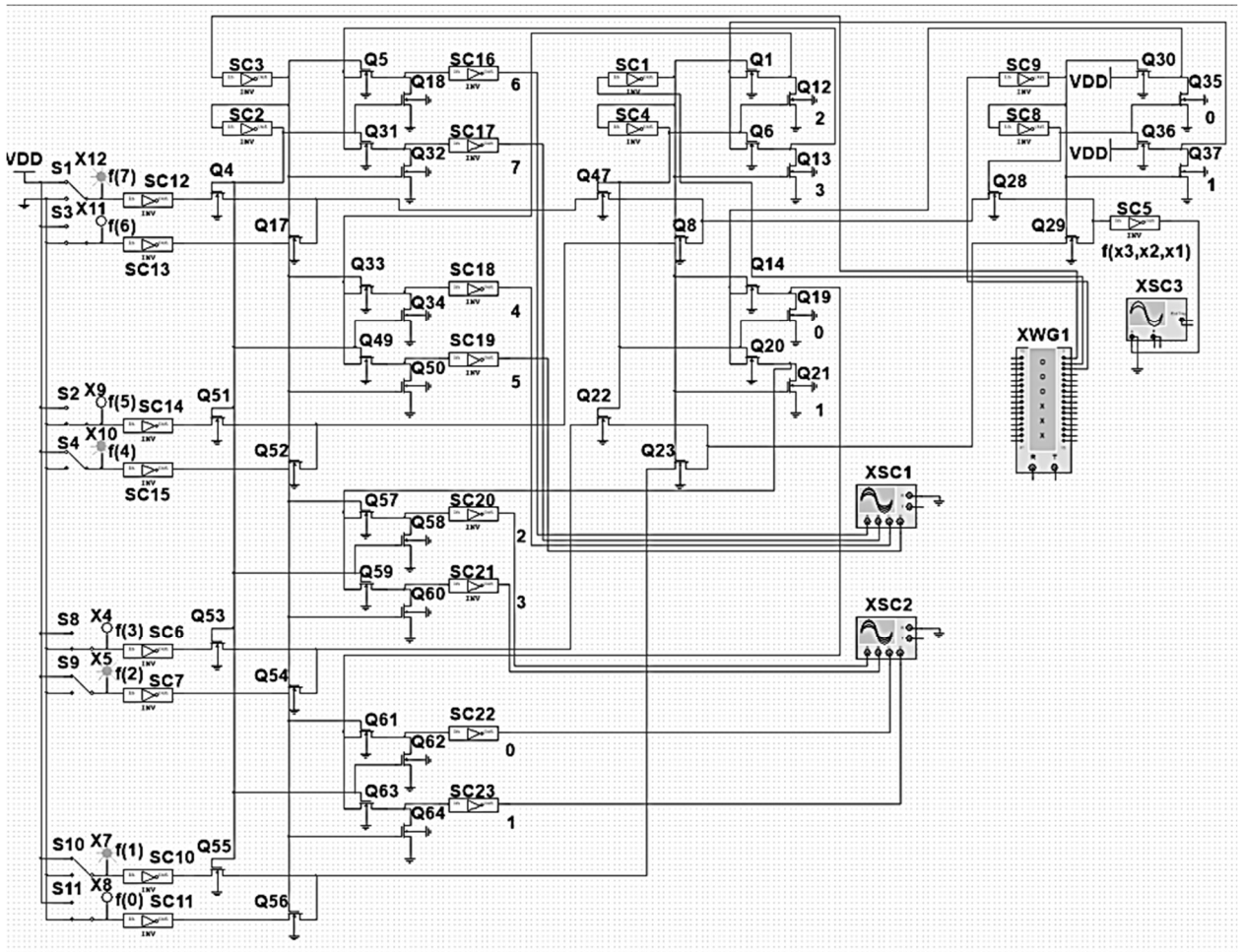


Рисунок 4.37 – 3–LUT, вычисляющий значение функции и дешифрацию набора переменных с использованием неактивной половины дерева транзисторов

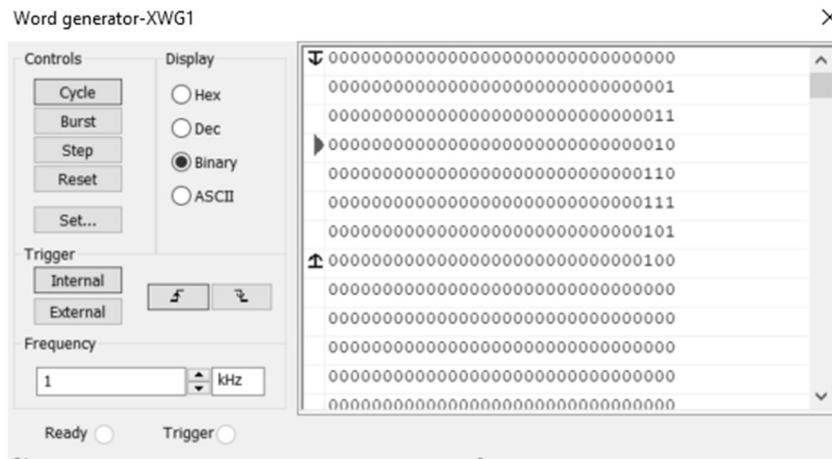


Рисунок 4.38 – Код входных сигналов переменных логического элемента 3–LUT с дешифрацией в форме кода Грея

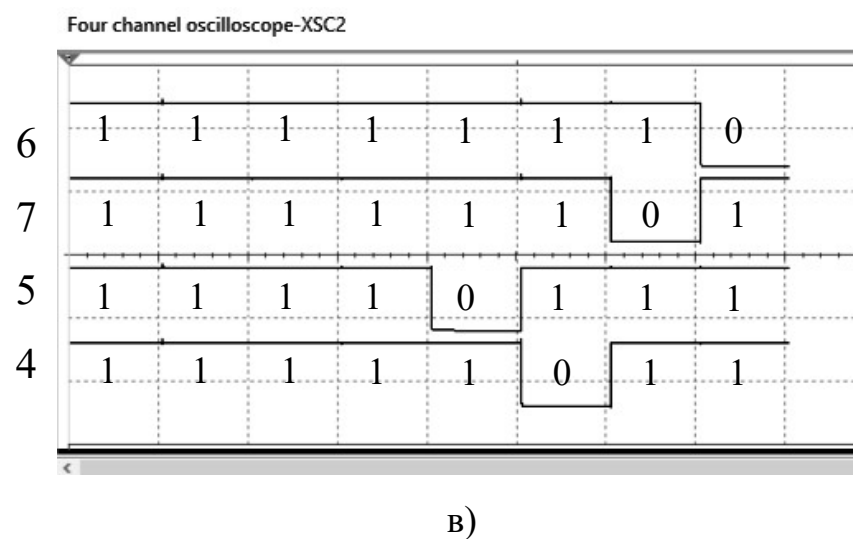
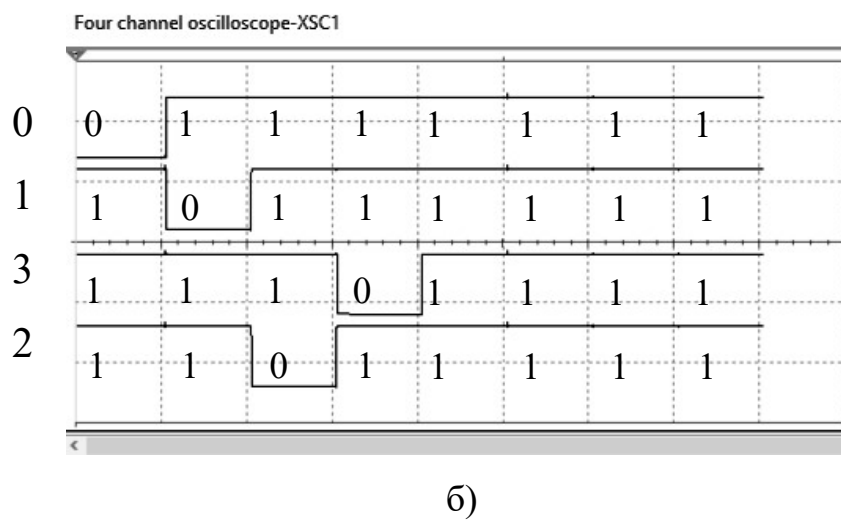
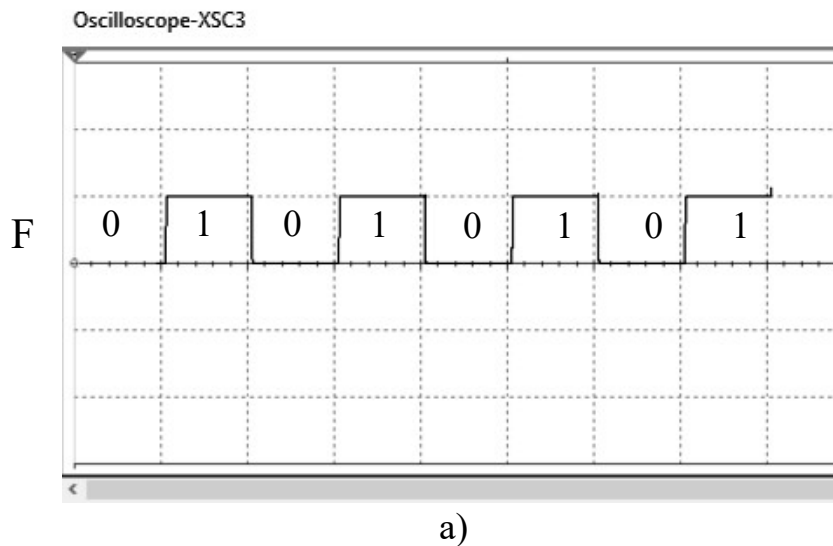


Рисунок 4.39 – Осциллограммы вычисления логической функции, исключающего ИЛИ (а) и одновременной дешифрации набора переменных набора 0–3 (б), набора 4–7 (а); Исходных код представлен в виде кода Грея (0,1,3,2,6,7,5,4)

4.2.6. Топологическое моделирование элементов LUT, вычисляющих значение заданной логической функции и дешифрацию набора переменных одновременно без использования неактивной половины дерева транзисторов

Топология 3-LUT, вычисляющего логическую функцию одновременно с дешифрацией входных переменных представлена на рисунке 4.40. Сигналы переменных задаются с помощью clock1, clock2, clock3, которые формируют код Грея. Результат вычисления логической функции отображается на выходе s1. Результат дешифрации отображается на выходах d0–d7. Конфигурация SRAM задается с помощью высокого (Vdd+) и низкого (Vss-) уровня напряжения соответствующие логической «1» и «0».

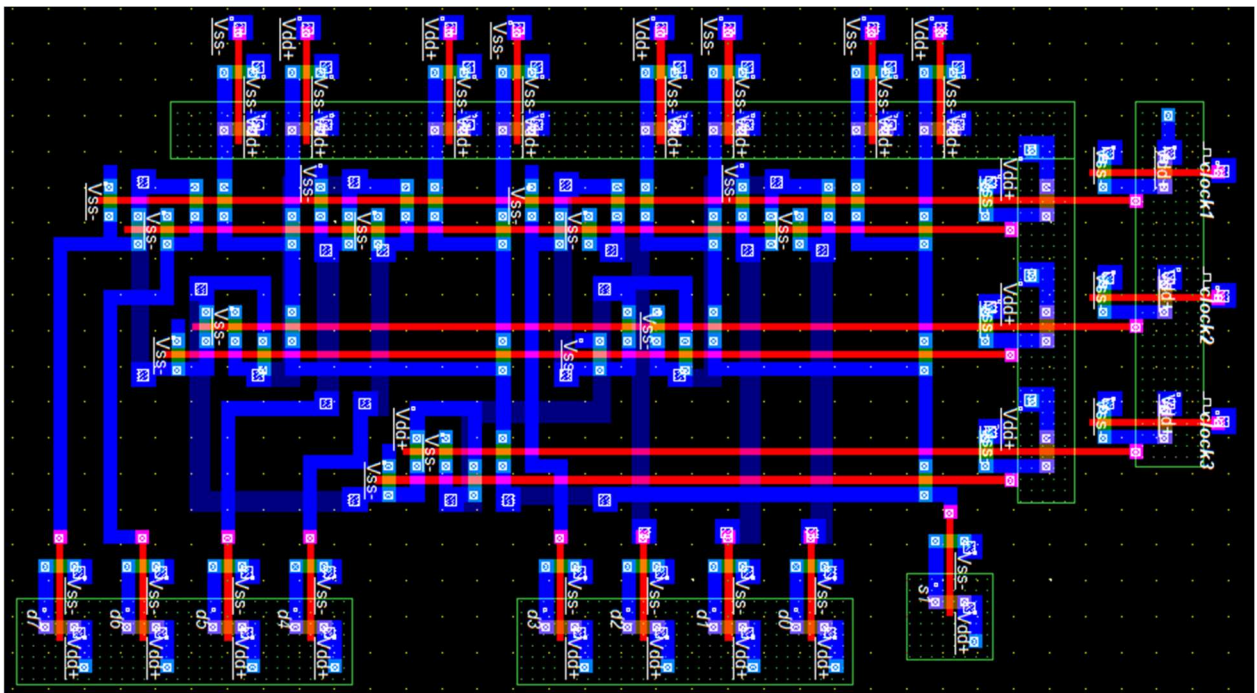


Рисунок 4.40 – Топология 3-LUT, реализующего вычисление логической функции и дешифрацию одновременно

Общее время моделирования составило 50 нс при установленном напряжении (Vdd+) в 0,8 В. Результат моделирования топологии 3-LUT, реализующего логическую функцию и дешифрацию представлен на рисунке 4.41.

Максимальная задержка составила $T = 283$ пс. Общая потребляемая мощность $W = 1,326$ мкВт. Количество использованных транзисторов и количество занимаемой площади представлены на рисунке 4.42.

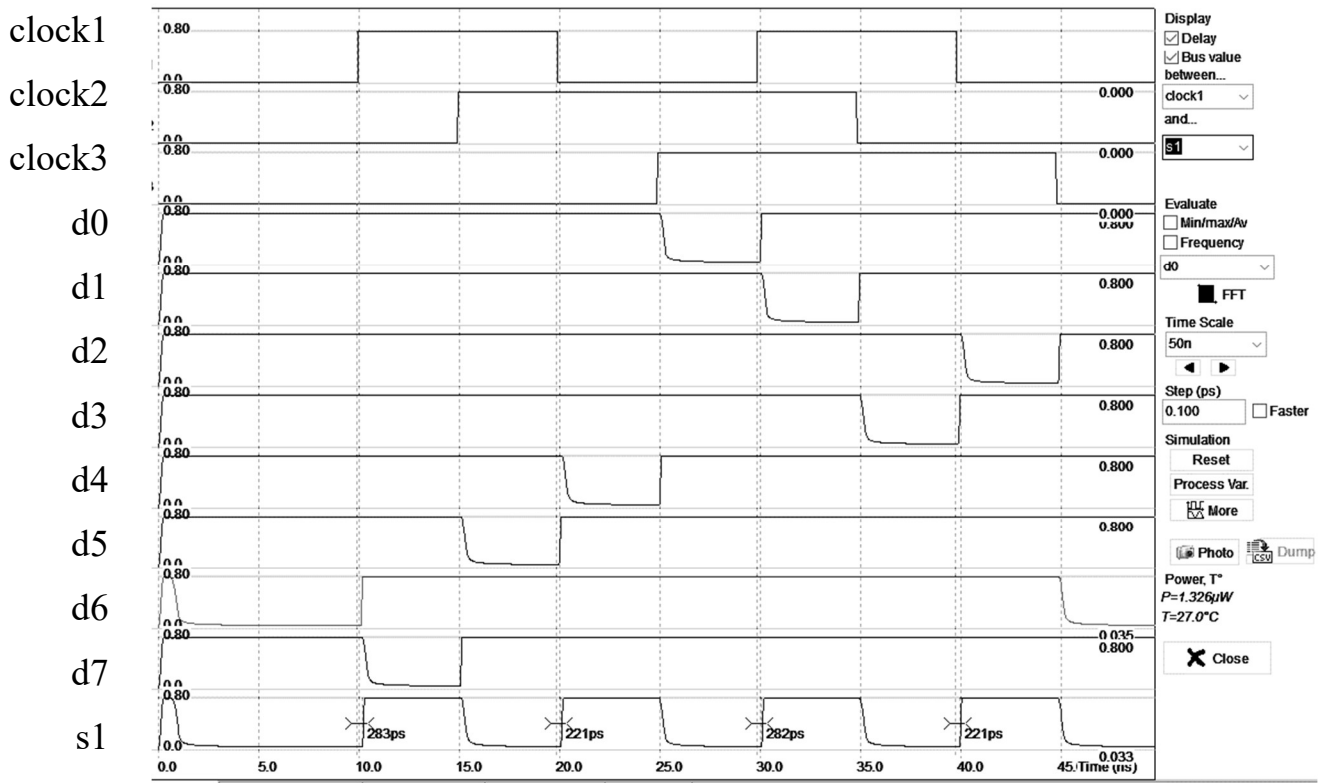


Рисунок 4.41 – Осциллограмма результата моделирования одновременного вычисления логической функции и дешифрации в 3–LUT

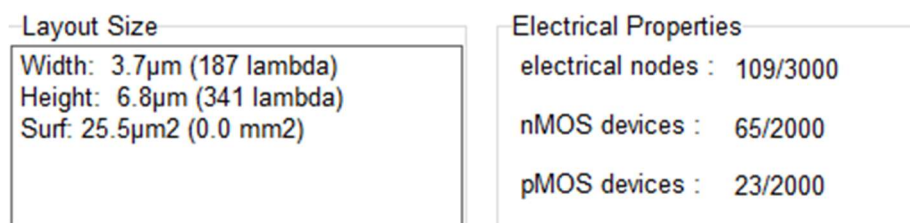


Рисунок 4.42 – Количество используемых компонентов и занимаемой площади на кристалле 3–LUT с реализацией логической функции и дешифрации

Топология 2–LUT, вычисляющего логическую функцию одновременно с дешифрацией входных переменных представлена на рисунке 4.43. Сигналы переменных задаются с помощью clock1, clock2 которые формируют код Грея.

Результат вычисления логической функции отображается на выходе s1. Результат дешифрации отображается на выходах d0–d3.

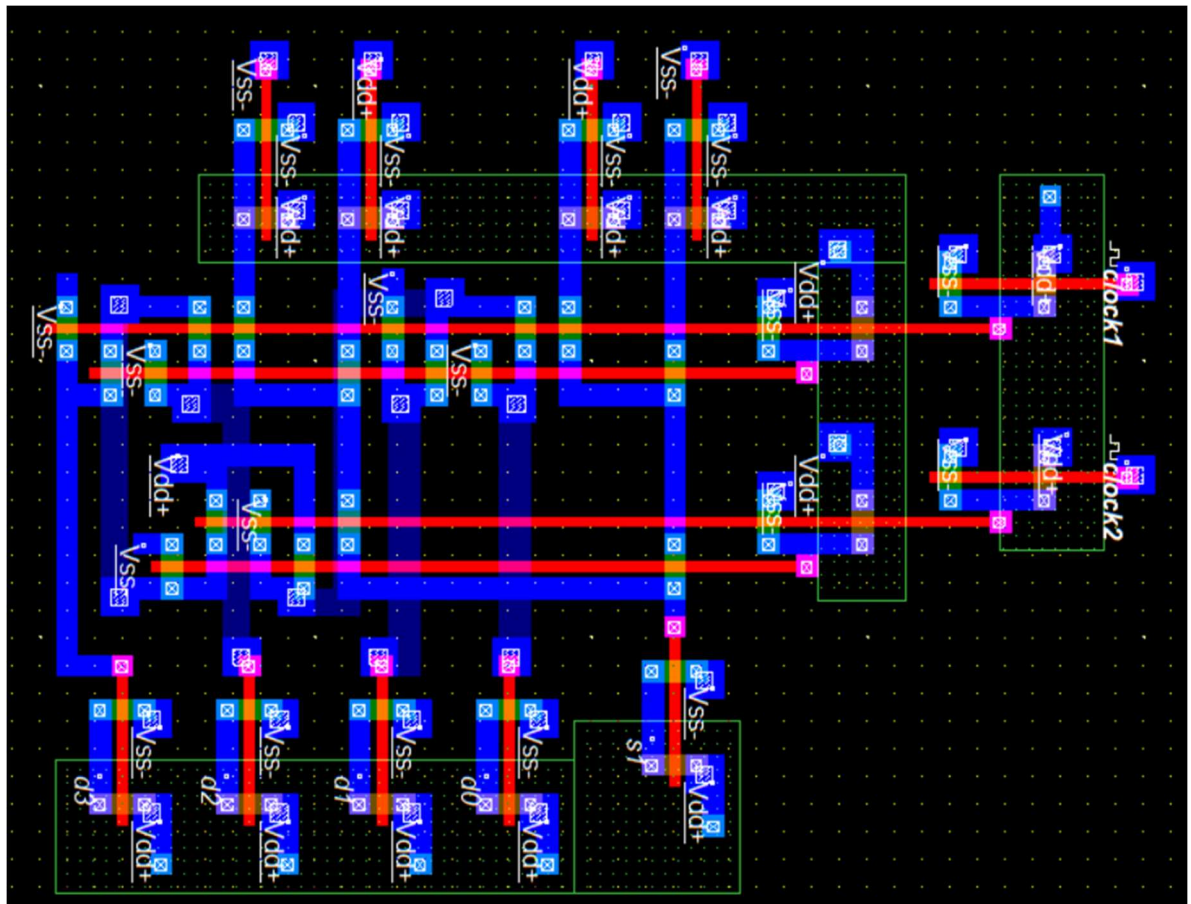


Рисунок 4.43 – Топология 2–LUT, реализующего вычисление логической функции и дешифрацию одновременно

Общее время моделирования составило 30 нс при установленном напряжении (V_{dd+}) в 0,8 В. Результат моделирования топологии 2–LUT, реализующего логическую функцию и дешифрацию представлен на рисунке 4.44.

Максимальная задержка составила $T = 165$ пс. Общая потребляемая мощность $W = 0,825$ мкВт. Количество использованных транзисторов и количество занимаемой площади представлены на рисунке 4.45.

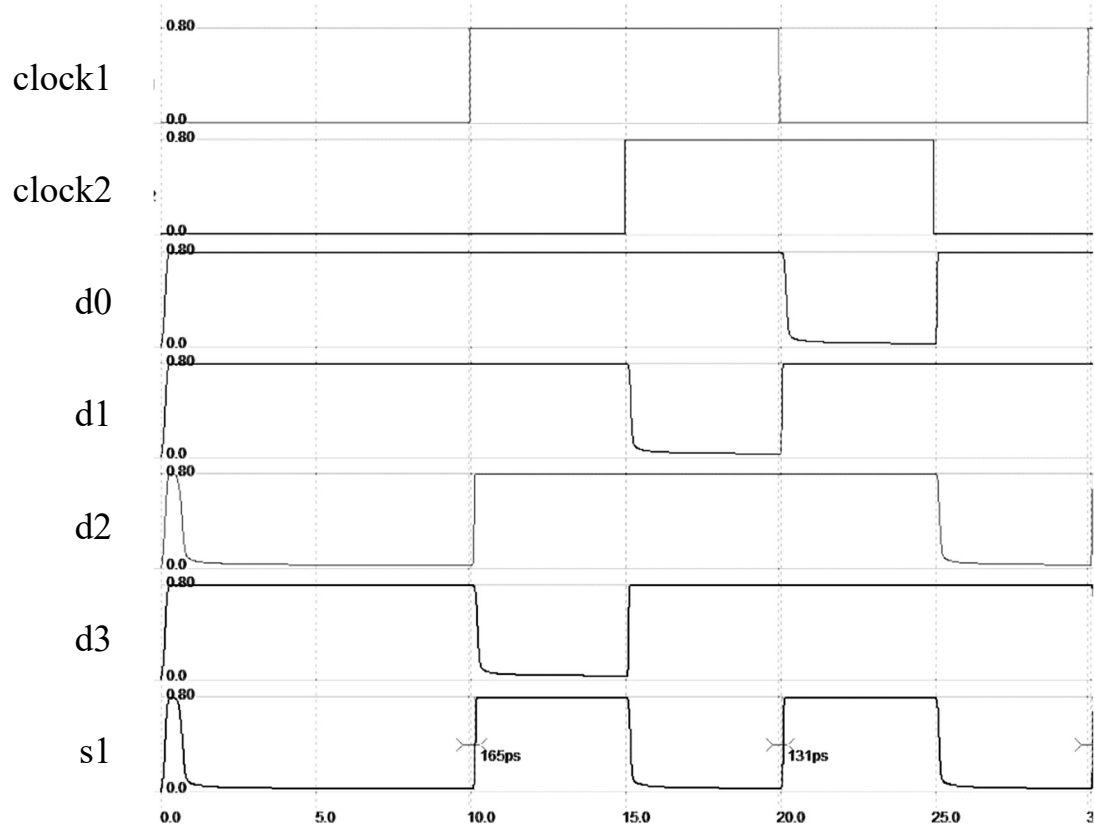


Рисунок 4.44 – Осциллограмма результата моделирования одновременного вычисления логической функции и дешифрации в 2–LUT

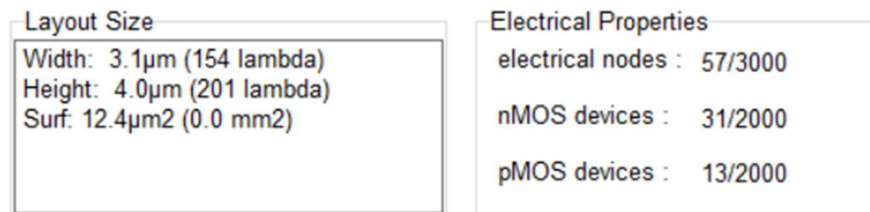


Рисунок 4.45 – Количество используемых компонентов и занимаемой площади на кристалле 3–LUT с реализацией логической функции и дешифрации

Результаты топологического моделирования продемонстрировали работоспособность вычисления логической функции и выполнения дешифрации входного набора одновременно, что подтверждает корректность предложенной модели с реверсивным деревом передающих транзисторов.

4.3. Выводы по главе 4

1. Выполненное моделирование в системах схемотехнического и топологического моделирования подтверждает работоспособность n -LUT реализующего несколько функций одновременно и вычисление n -LUT с одновременной дешифрацией входного набора переменных по трем моделям (3.9, 3.15, 3.18).

2. Проведенное моделирование в различных системах моделирования: Microwind, Multisim, Cadence Virtuoso показало непротиворечивые результаты по предлагаемым моделям.

3. Полученные результаты характеристик предлагаемых логических элементов по временной задержке, потребляемой мощности и используемой площади кристалла целесообразно использовать для сравнительной оценки эффективности.

ГЛАВА 5. СРАВНИТЕЛЬНЫЕ ОЦЕНКИ ЭФФЕКТИВНОСТИ РАЗРАБОТАННЫХ МЕТОДОВ

5.1. Оценка эффективности предложенного метода реализации нескольких логических функций одновременно по количеству транзисторов и площади, занимаемой на кристалле

В отличие от известных решений, в которых используется одна половина (в зависимости от значения старшей переменной) группы передающих транзисторов, в предлагаемом n -LUT используются обе половины группы передающих транзисторов по значению не одной, а нескольких (2^v , $v = 2, 3, \dots, n-1$) старших переменных.

То есть одновременно реализуются не две, а 2^v , $v = 2, 3, \dots, n-1$ логических функций, что увеличивает функциональность устройства. Для реализации того же количества логических функций с помощью предлагаемого n -LUT требуется 2^{v-1} устройств. Несмотря на дополнительные затраты в предлагаемом n -LUT, получается выигрыш по количеству требуемых транзисторов.

Так, сложность известного логического элемента LUT в транзисторах в зависимости от числа переменных n оценивается как:

$$\begin{aligned} L_{v1}(n) &= (2 + 2) \cdot 2^n + (2^{n+1} - 2) + 2n + 2 + 4 = \\ &= 2^{n+2} + 2^{n+1} + 2n + 4 = 3 \cdot 2^{n+1} + 2n + 4. \end{aligned} \quad (5.1)$$

Для вычисления 2^v функций в известном логическом элементе LUT необходимо взять 2^{v-1} известных LUT, суммарная сложность которых будет равна:

$$L_1(v, n) = 2^{v-1} \cdot L_{v1}(n) = 2^{v-1} \cdot (3 \cdot 2^{n+1} + 2n + 4). \quad (5.2)$$

Для вычисления такого же количества логических функций в предлагаемом логическом элементе n -LUT получаем сложность:

$$\begin{aligned} L_2(v, n) &= (2^{v+1} - 2 + 2) \cdot 2^n + 2^{n+1} + 2n + (2^{v+1} - 2)(2^v - 1) + 2(2^v - 1) = \\ &= (2^v + 1) \cdot 2^{n+1} + 2n + 2^{v+1} \cdot (2^v - 1). \end{aligned} \quad (5.3)$$

Например, для реализации четырёх логических функций ($v=2$) от одних и тех же переменных при $n=4$ необходимы два 4-LUT при общих затратах, равных 216 транзисторам, а в предлагаемом 4-LUT, реализующем все четыре функции одновременно, они равны 192 транзисторам.

Выигрыш растет при увеличении n и v . При $n > 4$ в качестве альтернативы предлагаемому n -LUT рассматривается схема, реализующая принципы построения и использования групп передающих транзисторов, заложенные в известные решения, сложность которых в транзисторах соответствует формуле (5.1). Так, при $n=5$ затраты на реализацию восьми логических функций от одних и тех же переменных ($v=3$) равны соответственно 824 в аналоге прототипа и 698 в предлагаемом устройстве.

Изменение сложности известного n -LUT $L_1(v, 8)$ и предлагаемого n -LUT $L_2(v, 8)$ при реализации $2^v, v = 2, 3, 4, 5$ логических функций ($n=8$) представлены на рисунке 5.1.

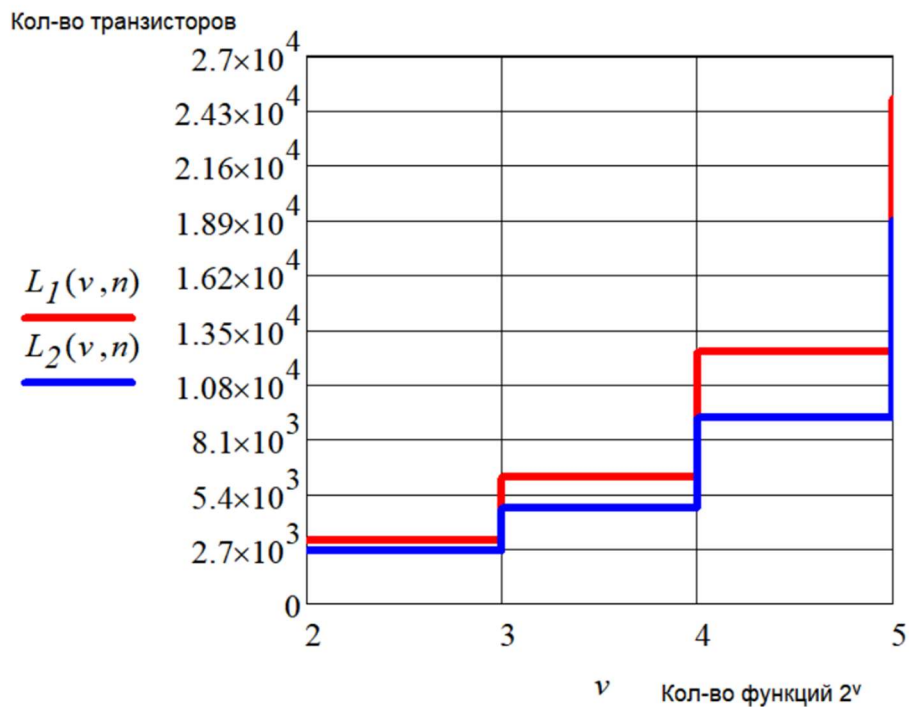


Рисунок 5.1 – Сравнение сложности многофункционального логического элемента (L_2) с известными логическими элементами (L_1)

Проведем сравнение занимаемой площади (S) на кристалле реализованных LUT, вычисляющих несколько функций одновременно с известными решениями. Для этого разработана топология базовых логических элементов LUT для трех, четырех, пяти и шести переменных, вычисляющих одну логическую функцию. Разработана топология логических элементов LUT вычисляющих две логические

функции одновременно для того же количества переменных. Проведено сравнение (рисунок 5.2) характеристик разработанных топологий логических элементов LUT, вычисляющих четыре логические функции для трех и четырех переменных.

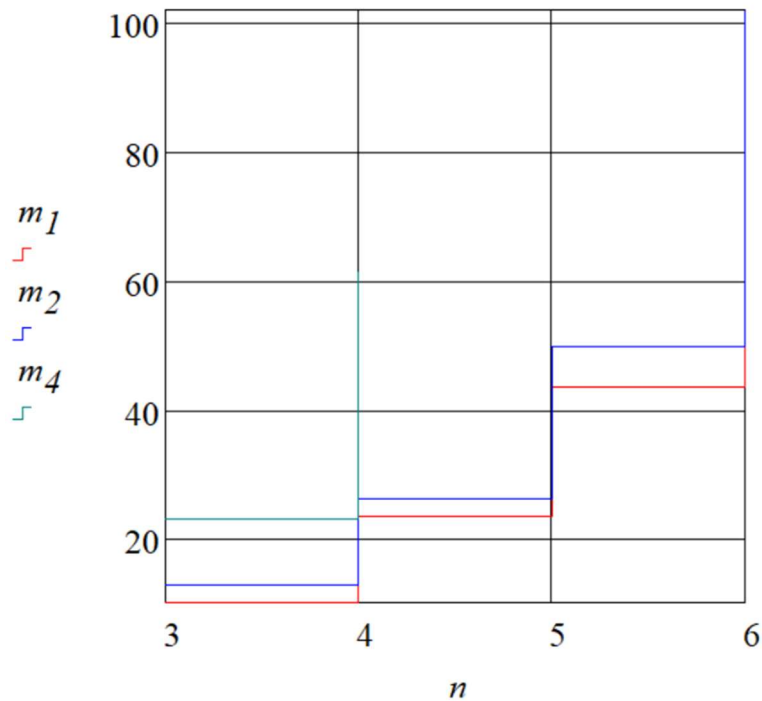


Рисунок 5.2 – Сравнение занимаемой площади (мкм²) кристалла смоделированных топологий для m_1 – вычисление одной функции, m_2 – вычисление двух функций одновременно, m_4 – вычисление четырех функций одновременно от количества переменных (n)

Значительное повышение занимаемой площади кристалла при реализации четырех переменных связано с дублированием инвертеров для восстановления сигнала. Для реализации четырех функций в логическом элементе пяти переменных восстанавливающие инверторы не дублируются, поскольку четыре функции реализуются за счет двух старших переменных, тогда как инверторы реализуются после третьей переменной.

Для вычисления четырех логических функций с помощью известного логического элемента LUT, реализующего одну логическую функцию, необходимо использовать дублирование известного LUT. При анализе известных LUT с

предлагаемым LUT используется как множитель количества реализуемых функций, так и моделирование топологий последовательно коммутируемых LUT. Анализ используемых ресурсов в сравнении с известными решениями логических элементов, реализующих одну, две и четыре функции представлен в таблице 5.1.

Таблица 5.1 – Сравнение занимаемой площади кристалла и количества используемых транзисторов известных логических элементов с предлагаемым:

	2^v известных LUT $m=1$	Предлагаемый LUT $m=4$
Площадь кристалла S ($n=3$)	40,8 мкм ²	23,2 мкм ²
Площадь кристалла S ($n=4$)	94 мкм ²	61,5 мкм ²
Кол-во транзисторов L ($n=3$)	176	116
Кол-во транзисторов L ($n=4$)	352	232
Потребляемая мощность W ($n=3$)	2,06 мкВт	2,921 мкВт
Потребляемая мощность W ($n=4$)	4,064 мкВт	6,862 мкВт
Задержка T ($n=3$)	210 пс	308 пс
Задержка T ($n=4$)	307 пс	386 пс
Интегральный показатель S·W ($n=3$)	84 мкВт*мкм ²	67,8 мкВт*мкм ²
Интегральный показатель S·W ($n=4$)	382 мкВт*мкм ²	340 мкВт*мкм ²

В приведенном анализе не учитываются коммутируемые узлы логических элементов, что еще больше увеличивает используемые площади кристалла. В результате полученных оценок предлагаемый LUT, который реализует четыре функции одновременно по сравнению с четырьмя известными LUT имеет выигрыш

по показателям площади кристалла (S) и количества транзисторов (L) от 34%. А по показателям потребляемой мощности (W) и временной задержки (T) имеет ухудшение характеристик известному LUT от 26%. Однако, интегральный показатель площади, занимаемой на кристалле (S) и потребляемой мощности (W) имеет выигрыш перед известными решениями от 11%.

5.2. Оценка эффективности предлагаемого метода реализации дешифрации набора переменных с использованием неактивной половины дерева транзисторов элемента LUT

Сравним сложность известного решения логического элемента LUT с n уровневым деревом транзисторов DC-LUT $_n$ (выражение 3.3) и предлагаемого логического элемента ADC-LUT $_n$ (выражение 3.8). Сложность DC-LUT $_n$, реализующего одну функцию определяется выражением:

$$\begin{aligned} L_{DC-LUT_n} &= 2^{n+1} + 2n + 2n \cdot 2^n + n \cdot 2^n + m \cdot 2^n + 2m = \\ &= 2^{n+2} + (n + m)(2 + 2^n), \end{aligned} \quad (5.4)$$

где m – число логических функций, n – число переменных; $2^{n+1} - 2 + 2 = 2^{n+1}$ – число транзисторов в дереве передающих транзисторов плюс два транзистора в инверторе 4; $2n$ – число транзисторов в группе инверторов 1; $2 \cdot 2^n$ – число транзисторов в группе инверторов 3; $n \cdot 2^n$ – число транзисторов в блоках конститuent нуля 5; $m \cdot (2^n + 2)$ – число транзисторов в m блоках вычисления функций 6.

В предлагаемом устройстве ADC-LUT $_n$:

$$\begin{aligned} L_{ADC-LUT_n} &= n \cdot 2^n + 2 + 4n + 2 \cdot 2^n + (n + 2) \cdot 2^n + 2 \cdot 2^n + 2 + 2^n = \\ &= (7 + 2n) \cdot 2^n + 4n + 4, \end{aligned} \quad (5.5)$$

где $n \cdot 2^n + 2$ – число транзисторов в дереве передающих транзисторов 2 и в выходном инверторе 4; $4n$ – число транзисторов в группах инверторов 1 и 11; $2 \cdot 2^n$ – число транзисторов в группе инверторов 3; $(n + 1) \cdot 2^n$ – число транзисторов в блоках дешифрации 5; $2 \cdot 2^n + 2$ – число транзисторов в дополнительной группе передающих транзисторов 15 плюс 2 транзистора в инверторе выбора режима

работы; 2^n – число транзисторов в дополнительной отключающей группе транзисторов 12.

В результате выигрыш предлагаемого ADC–LUT n определяется следующим образом:

$$\delta(m) = \frac{2^{n+2} + (n+m)(2+2^n)}{(7+2n) \cdot 2^n + 4n + 4}. \quad (5.6)$$

График зависимости выигрыша от m при $n=4$ представлен на рисунке 5.3.

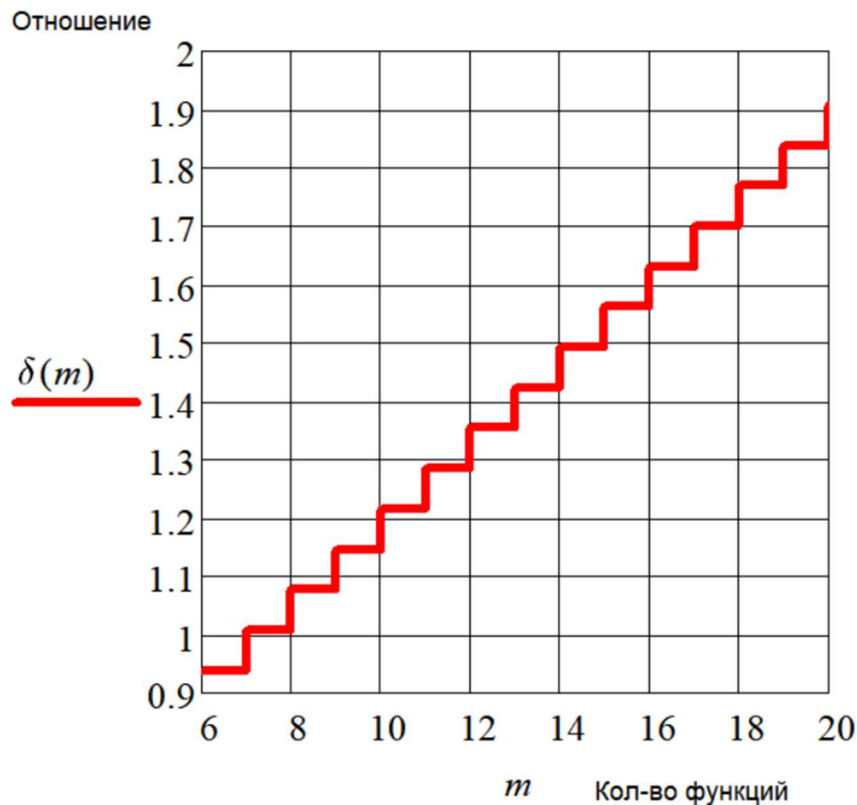


Рисунок 5.3 – Выигрыш ADC–LUT n для $n=4$ в зависимости от m

При этом задержка увеличивается на значение задержки конфигурационного транзистора 15 (рисунок 3.2), диода и подтягивающего резистора. Однако, в предложенном LUT по выражению (3.18), вычисляющем значение заданной логической функции и дешифрацию набора переменных одновременно задержка уменьшается за счет использования второй, неактивной половины дерева. Причем оба режима выполняются одновременно, что невозможно в ADC–LUT n , хотя затраты в количестве транзисторов сравнимы с затратами ADC–LUT n . Таким

образом, LUT_{DC} по выражению (3.18), предпочтительней $ADC-LUT_n$ по выражению (3.8).

5.3. Оценка эффективности предлагаемого метода реализации дешифрации набора переменных одновременно с вычислением логической функции

Проведем оценку сложности предлагаемого LUT_{DC} , в котором выполняется вычисление основной функции и дешифрация набора переменных одновременно с известным базовым LUT_0 и базовым LUT_{0DC} с элементами дешифрации. В $1-LUT_0$ всего 8 транзисторов. Если учитывать инверторы по входам настройки, то число транзисторов увеличивается до 12. Дешифрация требует дополнительных 8 транзисторов повышая общее количество транзисторов до 20. Таким образом сложность возрастает по отношению к исходному, известному LUT_0 . Для реализации вычисления функции и дешифрации набора на основе известного LUT_{0DC} необходимо $8+8+8$ (всего три $1-LUT$), что увеличивает количество используемых транзисторов до 24.

Для построения $2-LUT_{0DC}$ необходимо три таких элемента, в одном из которых инверторы по выходам дешифрации исключены, а в двух других исключены инверторы по выходам функции. При этом сложность известного $2-LUT_0$ составляет 24 транзистора (с инверторами по входам настройки). Получаем $24*5=120$ транзисторов для реализации функции двух переменных и дешифратора на четыре выхода. В то же время предлагаемый элемент содержит всего 48 транзисторов (рисунок 3.9), выигрыш существенный, даже с учетом повторения пар инверторов по входам младшей переменной.

$3-LUT_{DC}$ (рисунок 3.10) требует $72+16=88$ транзисторов с учетом повторения пар инверторов по входам младшей переменной, в тоже время известный $3-LUT_0$ имеет $14+8+2+6=30$ транзисторов, но таких элементов нужно уже 9, то есть всего 270 транзисторов против 88. Очевидно, что выигрыш увеличивается при увеличении числа переменных n и увеличивается экспоненциально.

С учетом конфигурационной памяти и инверторов-восстановителей обычный LUT_0 имеет сложность:

$$L_0(n) = 2^{n+1} - 2 + 8 \cdot 2^n + 4n + 2 = 10 \cdot 2^n + 4n, \quad (5.7)$$

где $2^{n+1} - 2$ – сложность n уровневого дерева передающих транзисторов, $8 \cdot 2^n$ – сложность настройки (конфигурационной памяти) дерева с учетом инверторов – восстановителей, $4n$ – сложность инверторов – восстановителей по входным переменным дерева, 2 – сложность выходного инвертора (из корня дерева). Оценка сложности приведена без учета декомпозиции деревьев при $n > 3$, когда требуются дополнительные восстановители.

При дешифрации переменных и реализации основной функции сложность LUT_{0DC} рассчитывается по следующему выражению:

$$L_{0DC}(n) = (2^n + 1)(10 \cdot 2^n + 4n). \quad (5.8)$$

Предлагаемый элемент LUT_{DC} , реализующий дешифрацию набора одновременно с реализацией логической функции (по выражению 3.9), построенный из модифицированных $1-LUT_{DC}$ характеризуется следующей оценкой сложности: к основному LUT_0 дополнительно вводятся инверторы по выходам дешифрации $2 \cdot 2^n = 2^{n+1}$, к каждому транзистору дерева добавляется еще два, в сумме получаем $(2^{n+1} - 2) + 2(2^{n+1} - 2) = 3(2^{n+1} - 2)$. Общее выражение, характеризующее сложность LUT_{DC} :

$$L_{dc}(n) = 10 \cdot 2^n + 4n + 2 \cdot 2^n + 2(2^{n+1} - 2). \quad (5.9)$$

График сравнения сложности базового LUT_0 (L_0) без дешифрации, LUT_{0DC} (L_{0DC}) с дешифрацией и предлагаемый LUT_{DC} (L_{DC}) с дешифрацией представлен на рисунке 5.4 и 5.5:

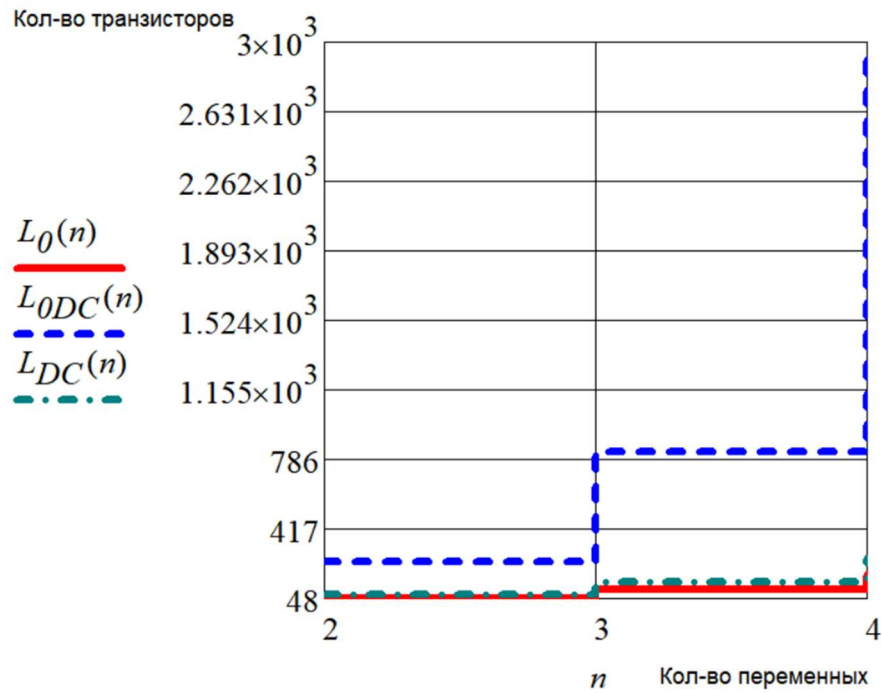


Рисунок 5.4 – Сравнение сложности известных LUT₀ (L_0), LUT_{0DC} (L_{0DC}) и предлагаемого LUT_{DC} (L_{DC}) при $n=2,3,4$

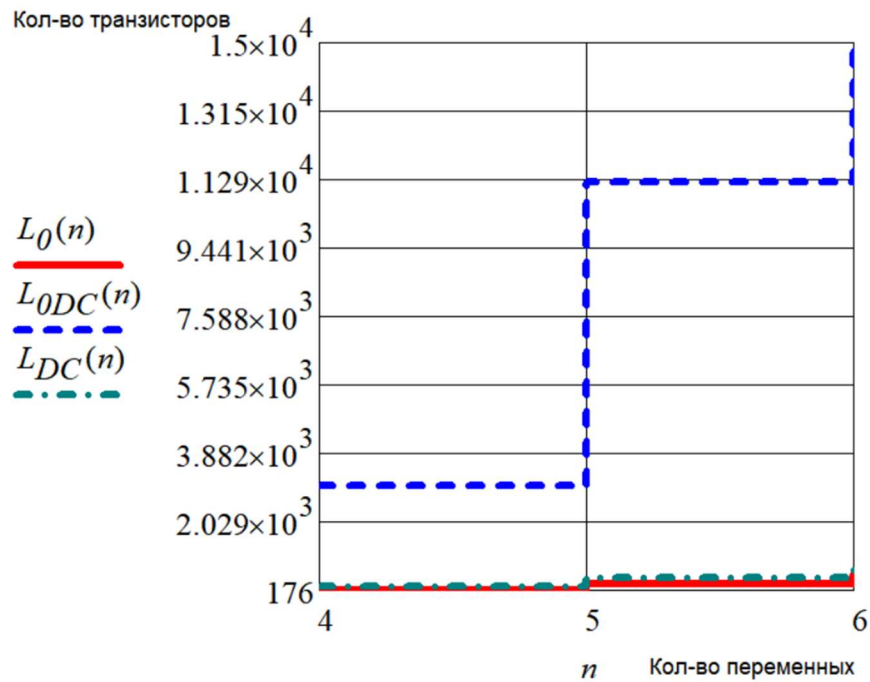


Рисунок 5.5 – Сравнение сложности известных LUT₀ (L_0), LUT_{0DC} (L_{0DC}) и предлагаемого LUT_{DC} (L_{DC}) при $n=4,5,6$

Таким образом, предложенный LUT_{DC} (L_{DC}) по сложности предпочтительнее, чем LUT_{0DC} (L_{0DC}). Оценки по временной задержке, площади кристалла и

потребляемой мощности сравнения двух моделей (3.15, 3.18) представлены в таблице 5.4. Сложность LUT_{0DC} в диапазоне переменных 5,6 очень высокая, а сложность LUT_{DC} близка к сложности LUT_0 .

5.4. Оптимизация по Парето блоков логических элементов

5.4.1 Оптимизация по Парето блоков логических элементов из известных и предложенных многофункциональных элементов

Для выбора предпочтительных по выражениям (1.3), (1.4) вариантов блоков логических элементов, включающих и предлагаемые многофункциональные, выполняется оптимизация по Парето [6]. При этом задается значение количества переменных n и количества g функций в блоке из нескольких логических элементов.

Множество несравнимых вариантов (множество Парето) строится по соответствующим n и g параметрам количества транзисторов (L), площади кристалла (S), временной задержке (T) и потребляемой мощности (W), полученные в системе топологического моделирования в Microwind (глава 4). Оцениваются комбинации предлагаемого многофункционального элемента на 2^v функций, например, ЛЭ на одну ($v=0$, известный элемент), две ($v=1$), четыре ($v=2$) и так далее. Из полученного множества вариантов для заданных параметров n и g строится Парето – множество, являющееся результатом многокритериальной оптимизации. Из этого множества может быть осуществлен выбор варианта если заданы ограничения, например, минимальный по площади кристалла и не превышении заданной временной задержки.

Рассмотрим пример. Пусть задано $n=5$, $g=7$. Блок на пять переменных в котором реализуется семь функций возможно построить шестью различными способами (таблица 5.2).

Таблица 5.2 – Комбинации ЛЭ на семь функций

№	Варианты комбинаций ЛЭ ($g=7$)	L, шт	S, мкм ²	T, пс	W, мкВт
1	1,1,1,1,1,1,1	966	304,5	350	13,26

2	1,1,1,1,1,2	896	267,3	460	12,8
3	1,1,1,2,2	826	230,1	460	12,34
4	1,1,1,4	768	203,8	694	13,08
5	1,2,2,2	756	192,9	460	11,88
6	1,2,4	698	167,6	694	12,62

Множество Парето выделено в таблице 5.2 зеленым цветом. Полученное множество отобразим графически (рисунок 5.6). Если, например, необходимо синтезировать блок элементов с минимальным количеством используемых транзисторов при не превышении временной задержки 500 пс, то это соответствует пятому варианту.

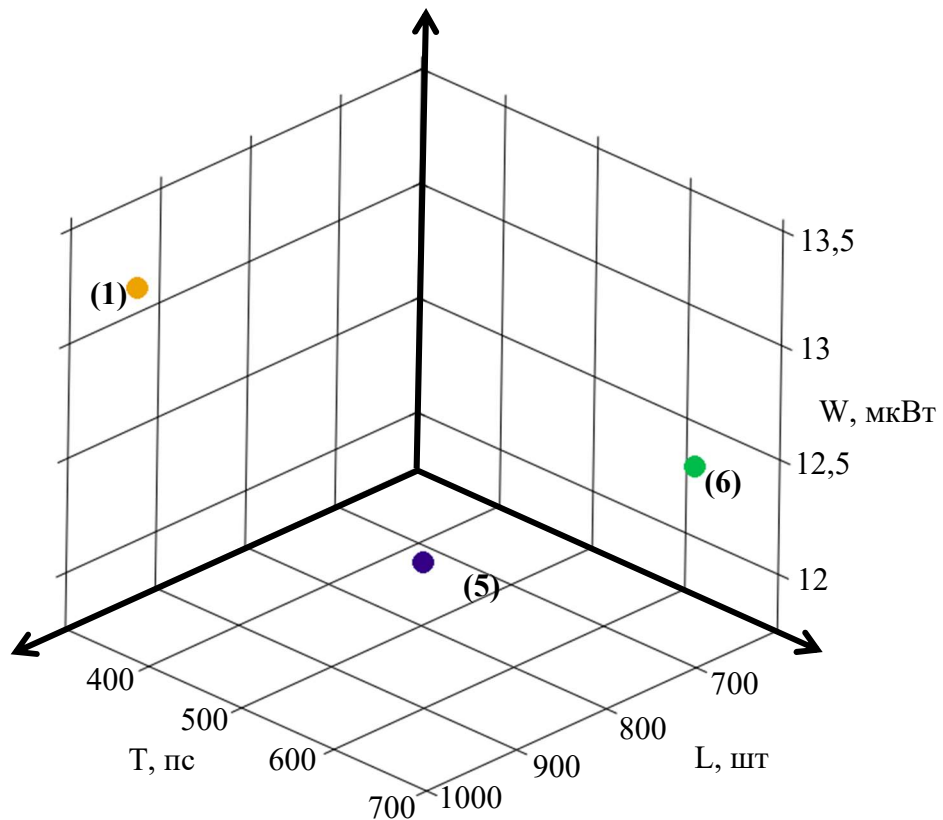


Рисунок 5.6 – Множество Парето при $n=5, g=7$ (L,T,W)

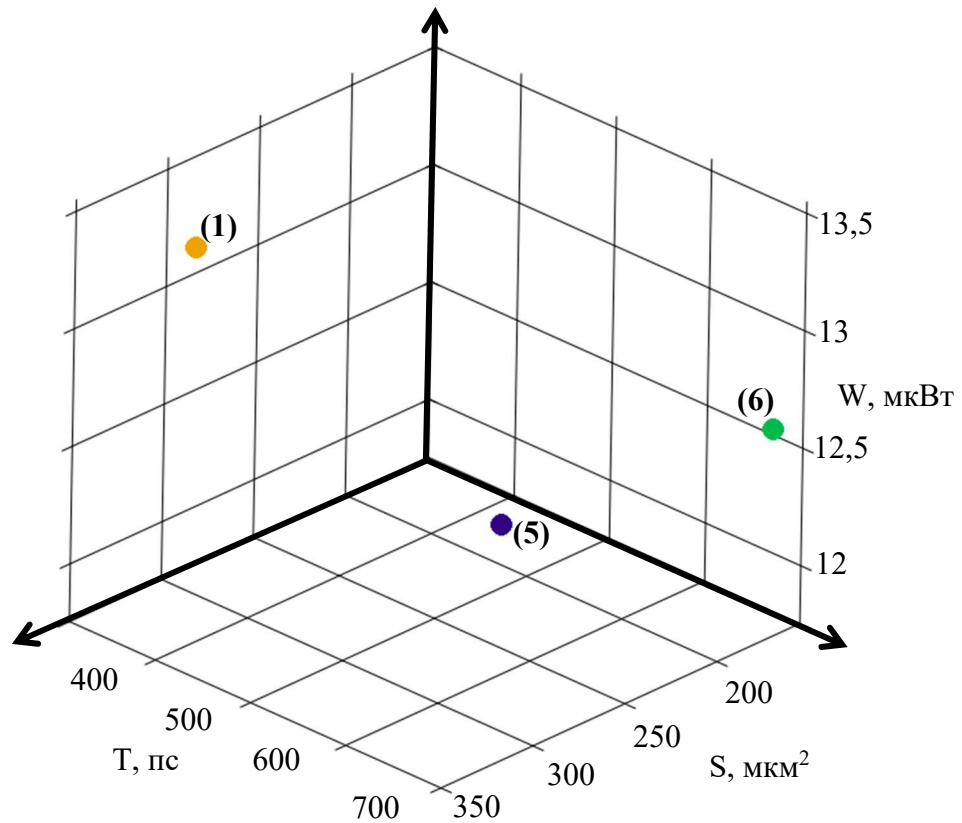


Рисунок 5.7 – Множество Парето при $n=5, g=7$ (S,T,W)

Пусть $n=5, g=12$. В этом случае возможно 20 комбинаций – таблица 5.3.

Парето-оптимальные варианты выделены зеленым цветом.

Таблица 5.3 – Комбинации ЛЭ на 12 функций

№	Варианты комбинаций ЛЭ ($g=12$)	L, шт	S, ккм ²	T, пс	W, кВт
1	1,1,1,1,1,1,1,1,1,1,1,1	1656	522	350	22,73
2	1,1,1,1,1,1,1,1,1,1,2	1586	484,8	460	22,27
3	1,1,1,1,1,1,1,1,2,2	1516	447,6	460	21,81
4	1,1,1,1,1,1,1,1,4	1458	423,3	694	22,55
5	1,1,1,1,1,1,2,2,2	1446	410,4	460	21,35
6	1,1,1,1,1,1,2,4	1388	384,1	694	22,09
7	1,1,1,1,2,2,2,2	1376	373,2	460	20,9
8	1,1,1,1,2,2,4	1318	362,9	694	21,63
9	1,1,1,1,4,4	1260	359,6	694	22,37

10	1,1,1,1,8	1250	354,1	976	23
11	1,1,2,2,2,2,2	1306	336	460	20,44
12	1,1,2,2,2,4	1248	317,7	694	21,17
13	1,1,2,4,4	1190	309,4	694	21,91
14	1,1,2,8	1180	307,9	976	22,55
15	2,2,2,2,2,2	1236	298,8	460	19,98
16	2,2,2,2,4	1178	291,5	694	20,72
17	2,2,4,4	1120	283,2	694	21,45
18	2,2,8	1110	279,7	976	22,09
19	4,4,4	1062	269,9	694	22,19
20	4,8	1052	262,4	976	22,82

Множество Парето выделено в Таблице 5.3 зеленым цветом. Полученное множество отобразим графически (рисунок 5.8 (L,T,W), рисунок 5.9 (S,T,W)).

Множество Парето при $n=5, g=12$ показано на рисунках 5.8, 5.9.

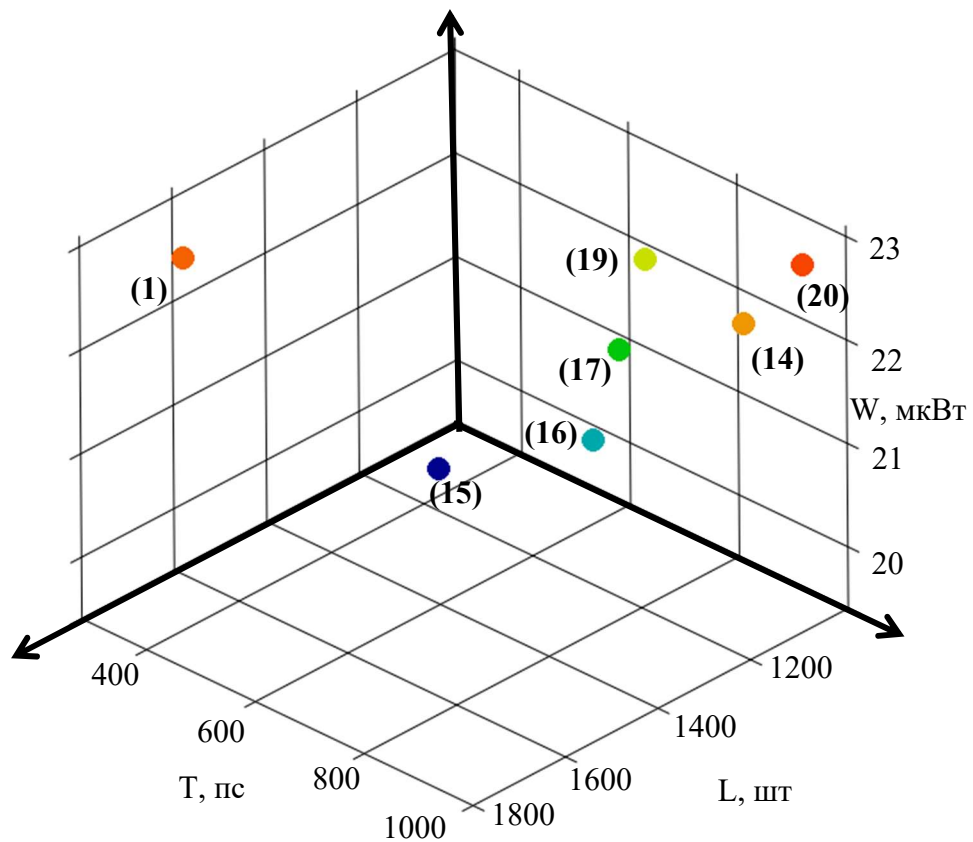


Рисунок 5.8 – Множество Парето при $g = 12, n=5$ (L,T,W)

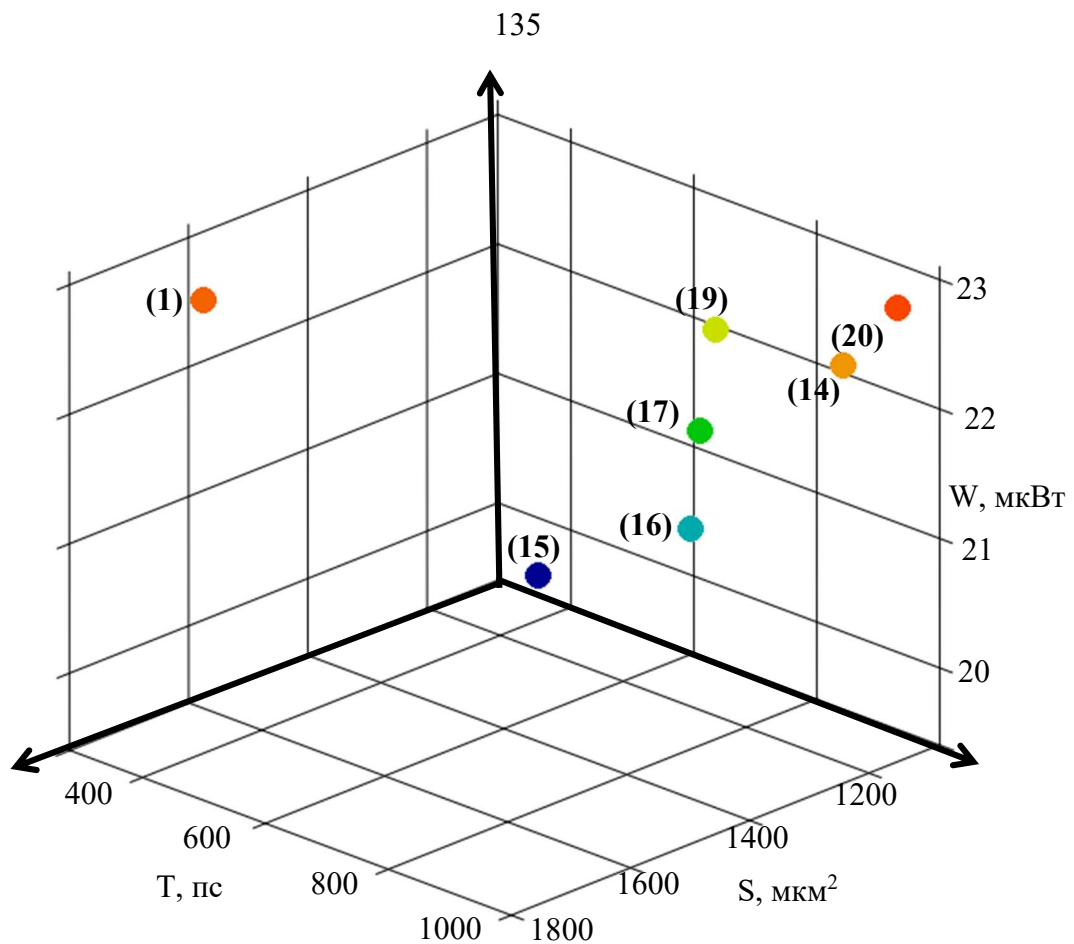


Рисунок 5.9 – Множество Парето при $g=12, n=5 (S, T, W)$

Если, например, необходимо синтезировать блок элементов с минимальным количеством используемых транзисторов при не превышении временной задержки 700 пс, то это соответствует варианту 19.

Если минимизируется потребляемая мощность при том же ограничении, то это вариант 16.

Таким образом, при заданных некоторых других значениях g строятся таблицы возможных вариантов, по которым определяются Парето – оптимальные. Далее по этому множеству выполняется оптимизация с заданными ограничениями, например, минимизация задержки при не превышении площади кристалла, либо минимизация количества транзисторов (с целью повышения вероятности безотказной работы) при не превышении заданной временной задержки.

5.4.2 Оптимизация по Парето блоков логических элементов, обеспечивающего одновременное вычисление логической функции и дешифрацию набора переменных без использования неактивной половины дерева транзисторов с известными элементами и предложенным многофункциональным элементом

Исследование показало, что элемент с двумя режимами работы (3.9) проигрывает по всем характеристикам, элементам, обеспечивающим одновременное вычисление логической функции с использованием неактивной половины дерева транзисторов (3.15) и без использования неактивной половины дерева транзисторов (3.18).

Моделирование логических элементов с одновременной реализацией логической функции и дешифрацией набора переменных с использованием неактивной части дерева транзисторов и без использования неактивной части дерева транзисторов показало (таблица 5.4), что всем показателям логический элемент с использованием неактивной части дерева транзисторов проигрывает элементу без использования неактивной части дерева транзисторов.

Таблица 5.4 – Сравнение занимаемой площади кристалла, количества используемых транзисторов, потребляемой мощности и временной задержки предлагаемых логических элементов с одновременной реализацией логической функции и дешифрацией набора переменных с использованием неактивной части дерева транзисторов (модель 3.15) и без использования неактивной части дерева транзисторов (модель 3.18):

	Модель 3.15 (LUT_{0DC})	Модель 3.18 (LUT_{DC})
Площадь кристалла S ($n=2$)	14,4 мкм ²	12,4 мкм ²
Площадь кристалла S ($n=3$)	28 мкм ²	25,5 мкм ²
Кол-во транзисторов L ($n=2$)	46 шт.	43 шт.

Кол–во транзисторов L ($n=3$)	96 шт.	88 шт.
Потребляемая мощность W ($n=2$)	1,705 мкВт	0,825 мкВт
Потребляемая мощность W ($n=3$)	2,848 мкВт	1,326 мкВт
Задержка основной функции T ($n=2$)	188 пс	165 пс
Задержка основной функции T ($n=3$)	265 пс	283 пс
Задержка по дешифрации ($n=2$)	271 пс	249 пс
Задержка по дешифрации ($n=3$)	488 пс	447 пс

В результате полученных оценок LUT по модели (3.18), который реализует одновременно вычисление основной функции и дешифрацию набора переменных без использования неактивной половины дерева передающих транзисторов по сравнению с LUT по модели (3.15) в которой используется неактивная ветвь передающих транзисторов имеет выигрыш по всем показателям от 7%, кроме временной задержки выполнения основной функции (T), показатели которой ухудшаются на 6%.

Сравнение элемента, обеспечивающего одновременное вычисление логической функции и дешифрацию набора переменных без использования неактивной части дерева транзисторов (выражение 3.18, схема в разделе 3.2.3, моделирование в разделе 4.2.6) с эквивалентным количеством известных элементов выполняется с использованием таблицы 5.5.

Для $n = 2$ требуется пять известных элементов LUT, четыре для реализации дешифрации и один для реализации основной функции, для $n = 3$ требуется девять

известных элементов LUT, восемь для реализации дешифрации и один для реализации основной функции.

При использовании предложенного многофункционального элемента (выражение 2.5, схема в разделе 2.3, моделирование в разделе 4.1.3) для $n = 2$ требуется один многофункциональный элемент при $v = 2$ и еще один элемент, либо один известный элемент LUT, для $n = 3$ требуется один многофункциональный элемент при $v = 3$ и еще один элемент, либо один известный элемент LUT.

Таблица 5.5 – Оценка занимаемой площади кристалла, количества используемых транзисторов, потребляемой мощности и временной задержки (имеется в виду максимальная временная задержка блока) количества известных элементов LUT, предложенных многофункциональных элементов, эквивалентных элементу, обеспечивающему одновременное вычисление логической функции и дешифрацию набора переменных без использования неактивной части дерева транзисторов.

	Известные LUT	Варианты предложенных многофункциональных		Предложенные многофункциональные + известный LUT для реализации основной функции
Обозначение	(2.1)	(2.5 $n+1$)	(2.5)	(2.5+2.1)
Количество элементов ($n=2$)	5	2 ($n=3$, $v=2$)	3 ($n=2$, $v=1$)	2 ($n=2$, $v=1$) + 1 (2.1)
Количество элементов ($n=3$)	9	2 ($n=4$, $v=3$)	3 ($n=3$, $v=2$)	2 ($n=3$, $v=2$) + 1 (2.1)
Площадь кристалла S ($n=2$)	47 мкм ²	46,4 мкм ²	32,1 мкм ²	30,8 мкм ²
Площадь кристалла S ($n=3$)	91,8 мкм ²	246 мкм ²	92,8 мкм ²	56,6 мкм ²

Кол-во транзисторов L ($n=2$)	120 шт.	232 шт.	108 шт.	96 шт.
Кол-во транзисторов L ($n=3$)	396 шт.	816 шт.	348 шт.	276 шт.
Потребляемая мощность W ($n=2$)	1,335 мкВт	5,842 мкВт	2,532 мкВт	1,955 мкВт
Потребляемая мощность W ($n=3$)	4,635 мкВт	27,054 мкВт	8,763 мкВт	6,357 мкВт
Временная задержка T ($n=2$)	113 пс	308 пс	128 пс	128 пс
Временная задержка T ($n=3$)	210 пс	612 пс	308 пс	308 пс

Графическое сравнение показателей предложенного элемента, обеспечивающего одновременное вычисление логической функции и дешифрацию набора переменных без использования неактивной части дерева транзисторов с показателями эквивалентного количества известных элементов LUT и предложенных многофункциональных элементов при $n=2$ и $n=3$, показано на рисунках 5.10 (L,T,W), 5.11 (S,T,W) и рисунках 5.12 (L,T,W), 5.13 (S,T,W) соответственно.

Таким образом, во множество Парето при $n = 2$ и $n = 3$ входят известные элементы LUT (2.1) и предлагаемый элемент (3.18), выполняющий дешифрацию входного набора без использования неактивной части дерева транзисторов. Многофункциональные элементы (2.5, 2.5 $n+1$, 2.5+2.1) при реализации дешифрации переменных существенно проигрывают по всем характеристикам.

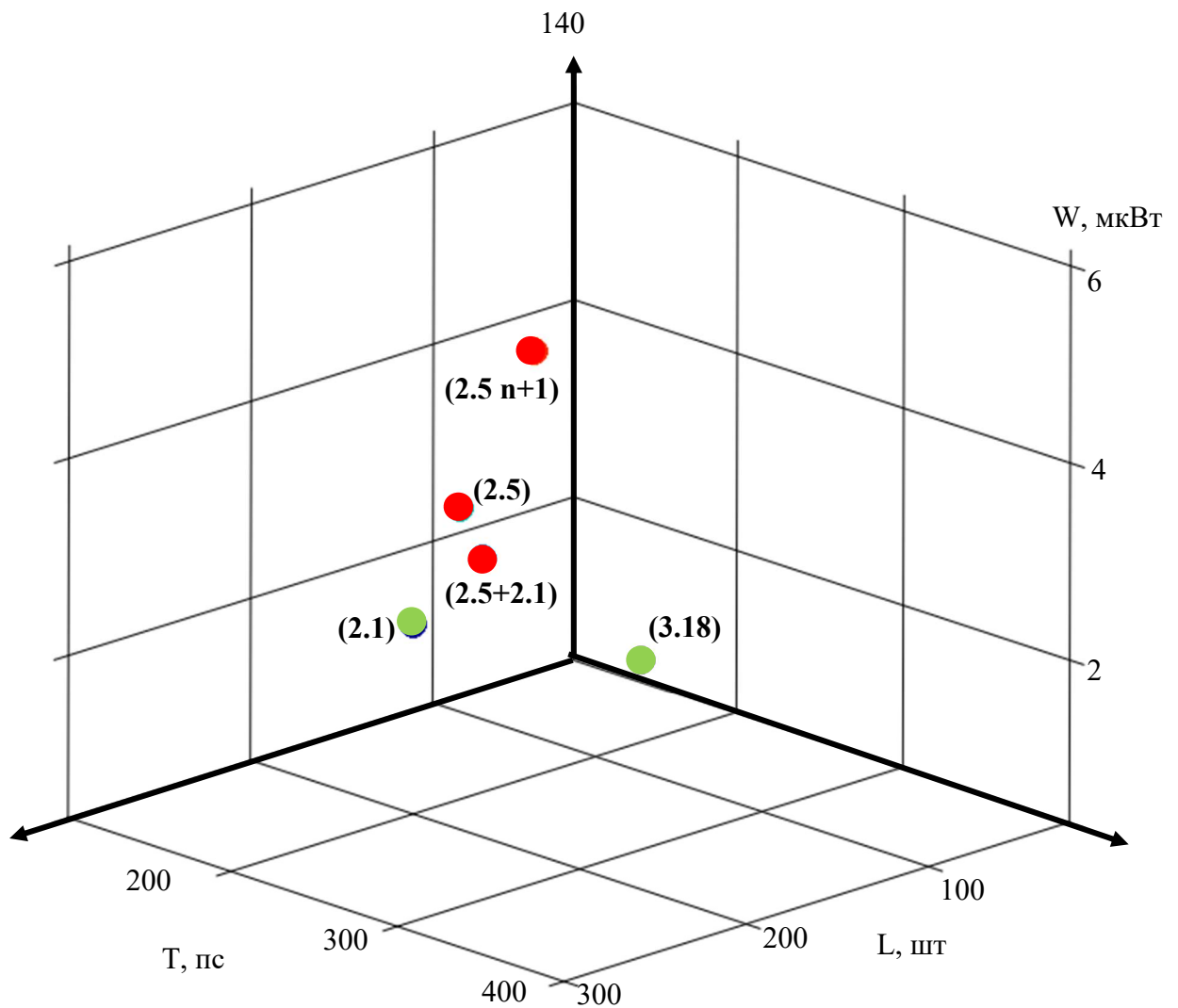


Рисунок 5.10 – Сравнение по показателям L, W, T блоков элементов, обеспечивающих одновременное вычисление логической функции и дешифрацию набора переменных: элемент 3.18 (для $n=2$) – без использования неактивной части дерева транзисторов, 2.1 – девять известных элементов LUT; 2.5 $n+1$ – один многофункциональный элемент на 3 переменных; 2.5 – два многофункциональных элемента на 2 переменных; 2.5+2.1 – один многофункциональный на три переменные и один известный LUT (множество Парето выделено зеленым)

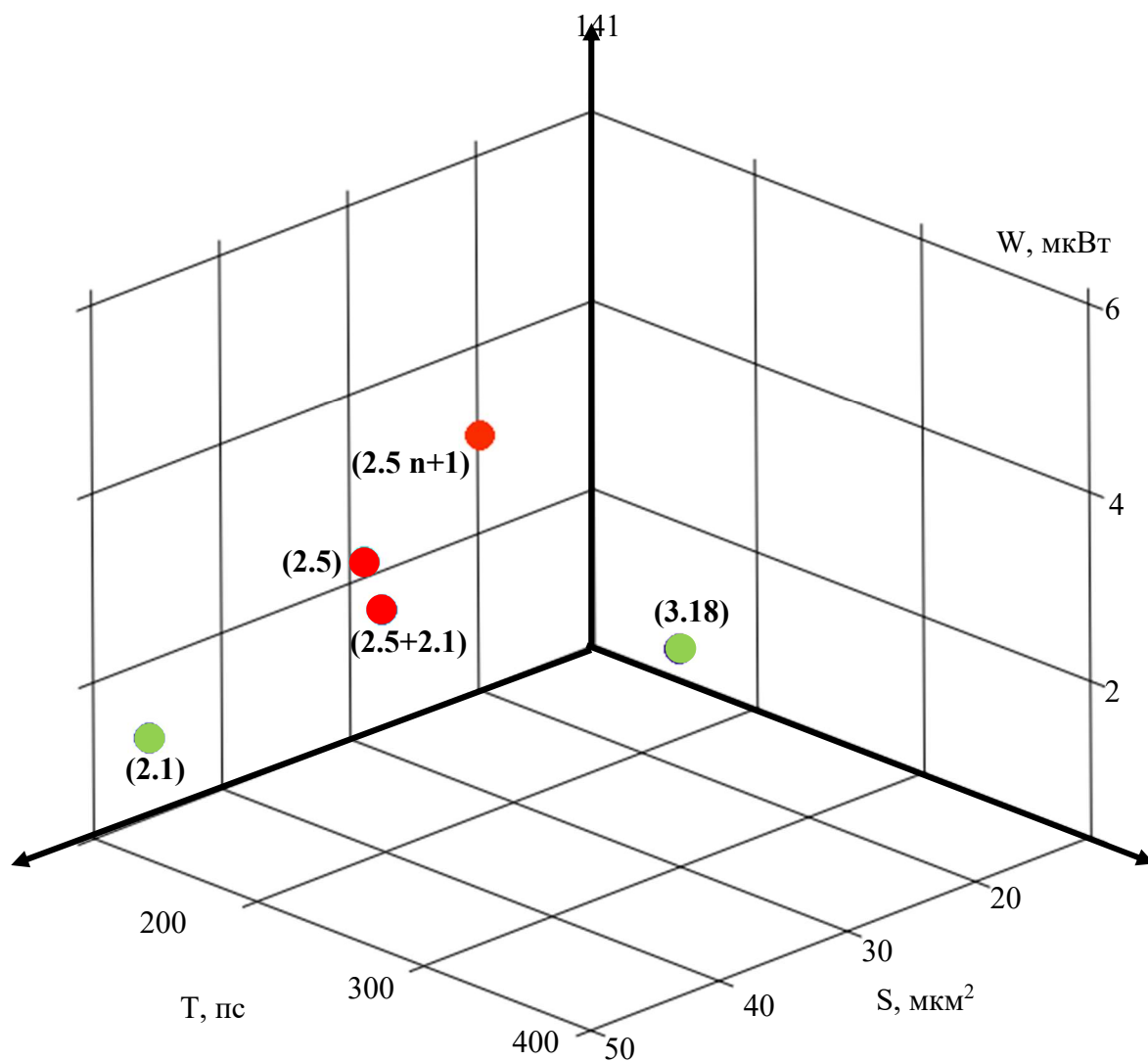


Рисунок 5.11 – Сравнение по показателям S, W, T блоков элементов, обеспечивающих одновременное вычисление логической функции и дешифрацию набора переменных: элемент 3.18 (для $n=2$) – без использования неактивной части дерева транзисторов, 2.1 – девять известных элементов LUT; $2.5 n+1$ – один многофункциональный элемент на 3 переменных; 2.5 – два многофункциональных элемента на 2 переменных; $2.5+2.1$ – один многофункциональный на три переменные и один известный LUT (множество Парето выделено зеленым)

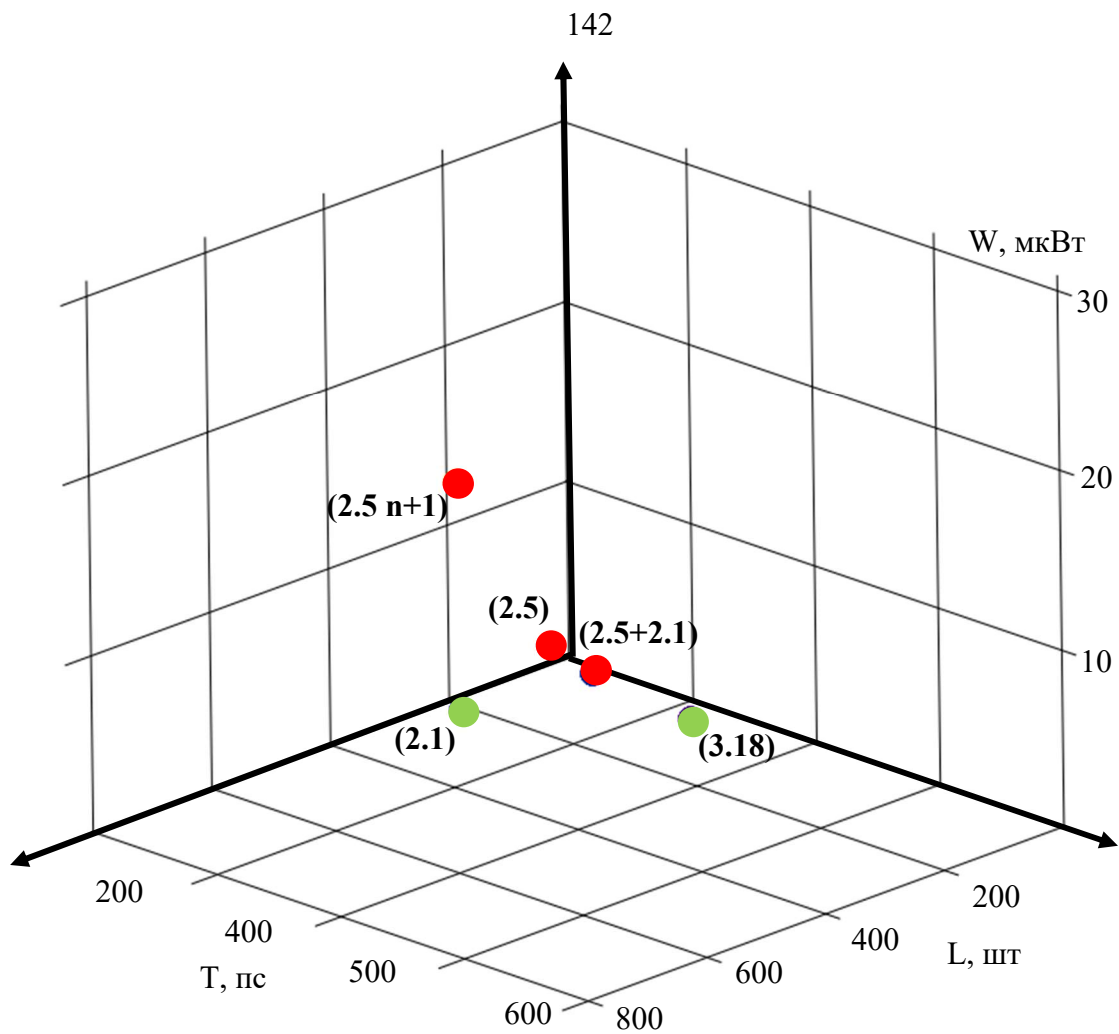


Рисунок 5.12 – Сравнение по показателям L, W, T блоков элементов, обеспечивающих одновременное вычисление логической функции и дешифрацию набора переменных: элемент 3.18 (для $n=3$) – без использования неактивной части дерева транзисторов, 2.1 – девять известных элементов LUT; 2.5 $n+1$ – один многофункциональный элемент на 3 переменных; 2.5 – два многофункциональных элемента на 2 переменных; 2.5+2.1 – один многофункциональный на три переменные и один известный LUT (множество Парето выделено зеленым)

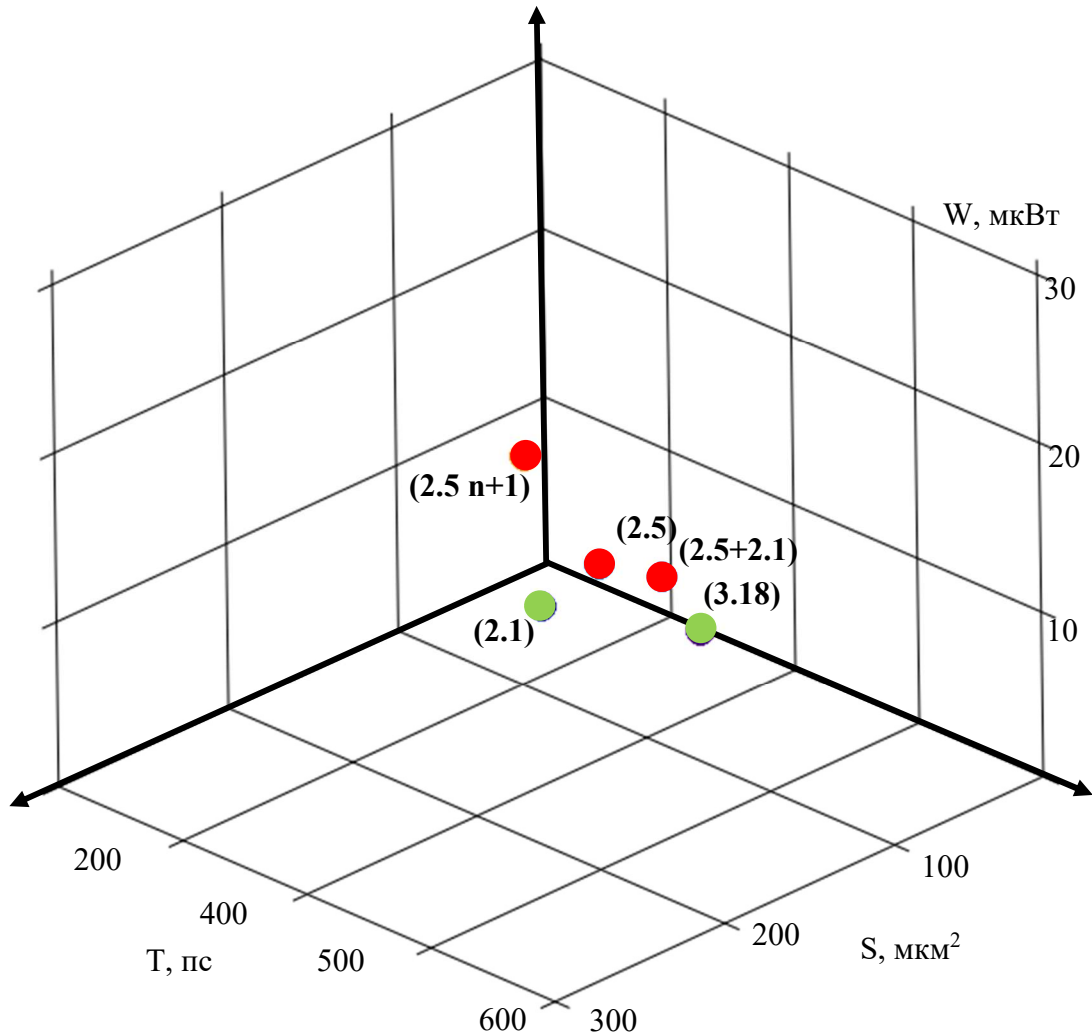


Рисунок 5.13 – Сравнение по показателям S, W, T блоков элементов, обеспечивающих одновременное вычисление логической функции и дешифрацию набора переменных: элемент 3.18 (для $n=3$) – без использования неактивной части дерева транзисторов, 2.1 – девять известных элементов LUT; $2.5n+1$ – один многофункциональный элемент на 3 переменных; 2.5 – два многофункциональных элемента на 2 переменных; $2.5+2.1$ – один многофункциональный на три переменные и один известный LUT (множество Парето выделено зеленым)

Синтез элементов, обеспечивающих одновременное вычисление логической функции и дешифрацию набора переменных без использования неактивной части дерева с большим значением n может осуществляться путем каскадирования предложенных элементов с $n = 2, 3$.

5.5. Выводы по главе 5

1. Предложенные многофункциональные элементы требуют значительно меньше количества транзисторов по сравнению с известными LUT (от 15% до 30%). Однако, по временной задержке они проигрывают.

2. Многофункциональные элементы целесообразны там, где требуется снижение аппаратных затрат, например, в задачах повышения вероятности безотказной работы, так как при снижении количества транзисторов снижается количество отказов.

3. Среди предложенных элементов реализующих дешифрацию входного набора по количеству транзисторов лучшие показатели имеет вариант, обеспечивающий одновременное вычисление логической функции и дешифрацию набора переменных (3.18). Этот же вариант выигрывает и по потребляемой мощности, и по площади кристалла и по временной задержке.

4. Парето оптимизация блоков логических элементов из известных LUT и предложенных многофункциональных элементов позволяет для заданного количества реализуемых функций g получить варианты предпочтительные либо по временной задержке, либо по другим параметрам используя заданные ограничения по времени.

5. Парето оптимизация блоков логических элементов из известных LUT, предложенных многофункциональных и предложенных элементов, обеспечивающих одновременное вычисление логической функции и дешифрацию набора переменных: элемент 3.18 без использования неактивной части дерева транзисторов (для $n = 2,3$) показала, что многофункциональные элементы для дешифрации переменных нецелесообразны.

ЗАКЛЮЧЕНИЕ

Представленная диссертационная работа посвящена решению актуальной научной задачи разработки моделей и методов синтеза логических элементов LUT ПЛИС FPGA, многофункциональных логических элементов: логических элементов, реализующих несколько функций при одной конфигурационной настройке; логических элементов выполняющих дешифрацию входного набора переменных и реализацию основной функции. В диссертационной работе поставлены и решены следующие задачи исследования:

1. В процессе анализа моделей, методов и алгоритмов синтеза логических элементов ПЛИС, выявлены особенности и недостатки известных логических элементов. Выбраны направления диссертационного исследования.

2. Разработаны модели, реализующие вычисление нескольких функций одновременно и дешифрацию набора переменных вместе с вычислением основной функции, позволяющие получить выигрыш по количеству транзисторов и площади кристалла по сравнению с соответствующим количеством известных логических элементов LUT.

3. Разработан метод синтеза многофункционального логического элемента, реализующего одновременное вычисление 2^v , $v = 1, 2, 3, \dots, n-1$ логических функций, позволяющий синтезировать требуемый многофункциональный логический элемент.

4. Разработан метод синтеза логического элемента, реализующего вычисление основной логической функции одновременно с дешифрацией набора переменных, позволяющий синтезировать логический элемент, реализующий дешифрацию набора переменных одновременно с вычислением основной функции.

5. Разработаны алгоритмы подключения дополнительных транзисторов в логических элементах LUT, реализующих вычисление нескольких функций одновременно и дешифрацию набора переменных вместе с вычислением основной функции, позволяющие создать структуру заданного многофункционального

логического элемента с заданным количеством входов переменных и необходимым количеством функций.

6. Получены оценки сложности по показателям количества транзисторов, площади, занимаемой на кристалле, потребляемой мощности и временной задержке предлагаемых логических элементов LUT, которые используются при синтезе функциональных и принципиальных электрических схем и топологий логических элементов с заданными параметрами.

7. Выполнена апробация и внедрение разработанных моделей, методов и алгоритмов синтеза предлагаемых логических элементов, реализующих вычисление нескольких функций одновременно и дешифрацию набора переменных вместе с вычислением основной функции в НИР ФИЦ ИУ РАН. Установлено, что сложность в количестве транзисторов и площади кристалла на реализацию логических функций уменьшены более чем на 15%. Внедренные результаты готовы к ОКР.

Направление дальнейших исследований целесообразно продолжить в области использования средств моделирования транзисторов, используемых в микросхемах с топологическими нормами менее 10 нм для уточнения показателей разработанных логических элементов.

Список сокращений

АЛМ – адаптивный логический модуль;

БМК – базовый матричный кристалл;

ДНФ – дизъюнктивная нормальная форма;

ИС – интегральная схема;

КМОП – комплементарная структура металл–оксид–полупроводник;

МОП – металл–оксид–полупроводник;

ПЗУ – постоянное запоминающее устройство;

ПЛУ – программируемое логическое устройство;

ПЛУ–АДС – программируемое логическое устройство с дешифрацией входных переменных;

ППЗУ – перепрограммируемое постоянное запоминающее устройство;

ПЛМ – программируемая логическая матрица;

ПМЛ – программируемая матрица логики;

ПЛИС – программируемая логическая интегральная схема;

САПР – Система автоматизированного проектирования;

СДНФ – совершенная дизъюнктивная нормальная форма;

ТТЛ – транзистор–транзисторная логика;

ТЦ МИЭТ – технологический центр Московского института электронной техники;

ФИЦ ИУ РАН – Федеральный исследовательский центр "Информатика и управление" Российской академии наук;

ALM – Adaptive logic module;

CLB – Configurable Logic Block;

CMOS – Complementary Metal–Oxide–Semiconductor;

CPLD – Complex Programmable Logic Device;

DSP – Digital Signal Processing;

finFET – fin Field–Effect Transistor;

FPGA – Field–Programmable Gate Array;

GA – Gate Array;

GAA – Gate All Around;

LAB – Logic Array Block;

LCA – Logic Cell Array;

LUT – Look Up Table, таблица поиска;

MOSFET – Metal–Oxide–Semiconductor Field–Effect Transistor;

PLA – Programmable logic array;

PLL – Phase Locked Loop;

SRAM – Static Random Access Memory;

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Арбузов, И. М. Пример разработки проекта в базисе ПЛИС 5578ТС024 / И. М. Арбузов, А. В. Строгонов, П. С. Городков // Компоненты и технологии. – 2019. – № 7. – С. 66–69.
2. Балашов, Е. П. Проектирование информационно–управляющих систем / Е. П. Балашов, Д. В. Пузанков // Радио и связь. – 1987. – 256 с.
3. Балашов, Е. П. Микро– и мини–ЭВМ : учеб. пособие / Е. П. Балашов, В. Л. Григорьев, Г. А. Петров. – Ленинград : Энергоатомиздат: Ленинградское отделение. – 1984. – 376 с.
4. Бахтин, В. В. Алгоритм разделения монолитной нейронной сети для реализации туманных вычислений в устройствах на программируемой логике / В. В. Бахтин // Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления. – 2022. – № 41. – С. 123–145.
5. Берски, Д. Быстродействующие ППВМ–кристаллы с повышенной плотностью упаковки уверенно теснят вентильные матрицы / Д. Берски // Электроника. – 1993. – № 18. – С.44–57.
6. Вихорев, Р. В. Логические элементы ПЛИС FPGA для реализации систем функций : автореф. дис. ... канд. тех. наук : 05.13.05 / Р. В. Вихорев. – Пермь, 2017. – 21 с.
7. Водяхо, А. И. Высокопроизводительные системы обработки данных: учеб. пособие / А. И. Водяхо, Н. Н. Горпец, Д. В. Пузанков. – М. : Высшая шк. – 1997. – 304 с.
8. Городилов, А. Ю. Методы и алгоритмы диагностирования и реконфигурации логики высоконадёжных ПЛИС : автореф. дис. ... канд. тех. наук : 05.13.05 / А. Ю. Городилов. – Пермь, 2016. – 19 с.
9. Денисов, А. Н. Методология проектирования аппаратуры по технологии БМК–ПЛИС–БМК / А. Н. Денисов // Известия высших учебных заведений. Электроника. – 2009. – № 5 (79). – С. 85–86.

10. Евреинов, Э. В. Однородные универсальные вычислительные системы высокой производительности / Э. В. Евреинов, Ю. Г. Косарев // Новосибирск: Наука, Сибирское отделение. – 1966. – 308 с.
11. Евреинов, Э. В. Цифровые автоматы с настраиваемой структурой / Э. В. Евреинов, И.В. Прангишвили // Энергия. – 1976. – 240 с.
12. Жеребцов, И. П. Основы электроники. / И. П. Жеребцов. – Ленинград : Энергоатомиздат. – 1989. – 352 с.
13. Зубчук, В. И. Справочник по цифровой схемотехнике / В. И. Зубчук, В. П. Сигорский, А. Н. Шкуро. – 1990. – 448 с.
14. Кабанчик, Д. Искусственный интеллект в промышленных граничных вычислениях / Д. Кабанчик // Современная электроника и технологии автоматизации. – 2022. – № 2.
15. Каменских, А. Н. Комбинированное резервирование самосинхронных схем : дис. ... канд. тех. наук / А. Н. Каменских. – Пермь, 2016. – 185 с.
16. Ковач, Н. Архитектура ПЛИС (FPGA) : сайт. – 2014. – URL: <https://marsohod.org/11-blog/265-fpga> (дата обращения 06.03.2024). – Режим доступа: свободный.
17. Кузелин, М. ПЛИС CPLD компании Xilinx с малым потреблением. Серия CoolRunner / М. Кузелин // Компоненты и технологии. – 2001. – №5.
18. Микушин, А. В. Программируемые логические матрицы / А. В. Микушин // Вычислительная техника и информационные технологии. – 2012.
19. Московский институт электронной техники : сайт. – URL: <https://miet.ru> (дата обращения: 06.03.2024). – Режим доступа: свободный.
20. Новая концепция ПЛИС с выбором режима работы и двухрежимный базисный логический элемент / И.А. Соколов, С.Ф. Тюрин, Ю.А. Степченков, Ю.Г. Дьяченко, М.С. Никитин, С.И. Советов // Системы высокой доступности. – 2024. – Т. 20. – № 2. – С. 56–64.
21. Панчул, Ю. ASIC и FPGA: сорок лет эволюции : сайт. – 2023. – URL: <https://engineer.yadro.com/article/asic-and-fpga-evolution/> (дата обращения: 06.03.2024). – Режим доступа: свободный.

22. Патент № 2637462 Российская Федерация, МПК G11C 17/00 (2006.01). Программируемое логическое устройство : № 2016131738 : заявл. 27.10.2023 : опубл. 06.05.2024 / С. Ф. Тюрин, Ю. Г Дьяченко, С. И. Советов, Ю. А. Степченков.
23. Патент № 2826302 Российская Федерация, МПК G11C 17/00 (2006.01). Программируемое логическое устройство : № 2023127768 : заявл. 27.10.2023 : опубл. 09.09.2024 / С. Ф. Тюрин, Ю.А. Васенин, Ю. А. Степченков, Ю. Г Дьяченко, С. И. Советов.
24. Патент № 2818802 Российская Федерация, МПК G06F 7/57 (2006.01). Программируемое логическое устройство : № 2016131738 : заявл. 01.08.2016 : опубл. 04.12.2017 / С. Ф. Тюрин, А. С Прохоров // Yandex.ru : патенты. URL: https://yandex.ru/patents/doc/RU2637462C1_20171204 (дата обращения: 06.03.2024).
25. Патент № 2811404 Российская Федерация, МПК G06F 7/57 H03K 19/173. Программируемое логическое устройство : заявл. 02.08.2023 : опубл. 11.01.2024 / С. Ф. Тюрин, И. А. Васенин, Ю. А. Степченков, Ю. Г. Дьяченко, С. И. Советов.
26. Патент № SU 1335974 A1 СССР. Универсальный логический модуль : заявл. 02.01.1986 : опубл. 07.09.1987 / Л. Ф. Викентьев, Ю. А. Аляев, А. А. Шалыто.
27. Патент № 1444892 СССР, G11C17/00, G06F7/00. Программируемое логическое устройство : опубл. 07.09.1988 / С. Ф. Тюрин, В. С. Харченко, С. Н. Ткаченко, В. Я. Жихарев, В. П. Улитенко.
28. Рабаи, М. Цифровые интегральные схемы. Методология проектирования / М. Рабаи, А. Чандракасан, Б. Николич. – М. : Вильямс, 2007. – 912 с.
29. Садчев, П. Особенности построения цифровой системы питания FPGA / П. Садчев // Компоненты и Технологии. – 2020. – №6.
30. Салломи, П. Высокие технологии, телекоммуникации, развлечения и СМИ / П. Салломи, П. Ли, О. Табакова // Deloitte. – 2018.
31. Семенов, Н. Технология FPGA для тысячи применений : сайт. – 2023. – URL: <https://habr.com/ru/articles/505838/> (дата обращения: 06.03.2024). – Режим доступа: свободный.

32. Симонов, Б. Базовые матричные кристаллы : сайт. – 2014. – URL: <http://www.chipinfo.ru/literature/chipnews/200006/18.html> (дата обращения: 06.03.2024). – Режим доступа: свободный.
33. Советов, С. И. Алгоритм подключения дополнительных транзисторов в схеме логического элемента ПЛИС / С. И. Советов // Инновационные технологии: теория, инструменты, практика. – 2022. – Т. 1. – С. 51–58.
34. Советов, С. И. Метод синтеза логического элемента, реализующего несколько функций одновременно / С. И. Советов, С. Ф. Тюрин. – DOI 10.32362/2500–316X–2023–11–3–46–55 // Russian Technological Journal. – 2023. – № 11 (3). – С. 46–55.
35. Советов, С. И. Разработка топологии многофункционального логического элемента ПЛИС / С. И. Советов. – DOI 10.15593/2224–9397/2023.4.02 // Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления. – 2023. – № 48. – С. 30–49.
36. Строгонов, А. В. Системное проектирование программируемых логических интегральных схем : учеб. пособие / А. В. Строгонов. – Воронеж. – ФГБОУ ВПО «Воронежский государственный технический университет». – 2012. – 322 с.
37. Строгонов, А. Современные тенденции развития ПЛИС: от системной интеграции к искусственному интеллекту / А. Строгонов, П. Городков // Электроника: Наука, технология, бизнес. – 2020. – № 4 (195). – С. 46–56.
38. Строгонов, А. Программируемая коммутация ПЛИС: взгляд изнутри. / А. Строгонов С. Цыбин // Компоненты и Технологии. – 2010. – №10.
39. Тарасов, И. Методология проектирования для ПЛИС Xilinx: организационные аспекты / И. Тарасов // Компоненты и Технологии. – 2015. – №1.
40. Технология устройств CPLD : сайт . – URL: <https://parallel.ru/fpga/cpld.html> (дата обращения: 06.03.2024). – Режим доступа: свободный.

41. Тюрин, С. Ф. Особенности архитектуры Гиперфлекс / С. Ф. Тюрин // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – 2018. – № 1. – С. 56–62.

42. Тюрин, С. Ф. Логические элементы ПЛИС FPGA на основе комбинированного кодирования переменных / С. Ф. Тюрин, И. А. Васенин, С. И. Советов. – DOI 10.15593/2224–9397/2023.2.04 // Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления. – 2023. – № 46. – С. 83–107.

43. Тюрин, С. Ф. Логический элемент ПЛИС FPGA, реализующий функцию и дешифрацию набора переменных / С. Ф. Тюрин, С. И. Советов. – DOI 10.15593/2224–9397/2023.3.01 // Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления. – 2023. – № 47. – С. 5–31.

44. Тюрин, С.Ф. Логический элемент программируемых логических интегральных схем FPGA, вычисляющий функцию одновременно с дешифрацией входных переменных / С.Ф. Тюрин, С.И. Советов // Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления. – 2024. – № 50. – С.216–234.

45. Тюрин С. Ф. Логический элемент FPGA, вычисляющий две функции одновременно / С. Ф. Тюрин, А. С. Прохоров // Проектирование и технология электронных средств. – 2016. – № 2. – С. 18–23.

46. Тюрин, С. Ф. Программа для подключения дополнительных транзисторов в многофункциональном логическом элементе ПЛИС «МФЛУТ» / С. Ф. Тюрин, С. И. Советов // Свидетельство о регистрации программы для ЭВМ № 2024616246. – 19.03.2024. – Заявка № 2024615319 от 19.03.2024.

47. Тюрин, С. Ф. Программа соединения блоков функции–дешифрации на одну переменную по уровням дерева транзисторов элемента LUT «ДШФЛУТ» / С. Ф. Тюрин, С. И. Советов // Свидетельство о регистрации программы для ЭВМ № 2024663917. – 14.06.2024. – Заявка № 2024663917 от 14.06.2024.

48. Тюрин, С. Ф. FPGA LUT с двумя выходами декомпозиции по Шеннону / С. Ф. Тюрин, М. А. Чудинов // Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления. – 2019. – № 29. – С. 136–147.
49. Угрюмов, Е. П. Цифровая схемотехника / Е. П. Угрюмов. – СПб. : БХВ–Петербург, 2010. – 816 с.
50. Хаханов, В. И. Инфраструктура диагностического обслуживания SoC / В. И. Хаханов // Вестник Томского университета. – 2008. – №4 (5). – С. 74–101.
51. Цифровая и вычислительная техника / Э. В. Евреинов, Ю. Т. Бутыльский, И. А. Мамзелев, и др. – М.: Радио и связь, 1991. – 464 с.
52. Шаурман, А. А. Архитектура ПЛИС. Часть 1. Логический элемент : сайт . – URL: http://www.labfor.ru/articles/fpga_arch_le (дата обращения: 06.03.2024). – Режим доступа: свободный.
53. A Survey of FPGA Logic Cell Designs in the Light of Emerging Technologies / R. Shubham, N. Pallab, R. Ansh, K. Santosh, K. Akash. – DOI 10.1109/ACCESS.2021.3092167 // IEEE Access. – 2021. – Vol. 9. – P. 91564 – 91574.
54. Ahmed, E. The effect of LUT and cluster size on deepsubmicron FPGA performance and density / E. Ahmed, J. Rose // IEEE Trans. Very Large Scale Integr. (VLSI) Syst. – 2004. – Vol. 12, no. 3. – P. 288–298.
55. An FPGA Architecture for ASIC–FPGA Co–design to Streamline Processing of IDSs / S. Tomoaki, C. Sorawat, M. Phichet, H. Kohji. – DOI 10.1109/CTS.2016.0079 // 2016 International Conference on Collaboration Technologies and Systems (CTS). – 2016.
56. An automatic RTL compiler for high–throughput FPGA implementation of diverse deep convolutional neural networks / Y. Ma, Y. Cao, S. Vrudhula, J.–S. Seo, // Proc. 27th Int. Conf. Field Program. Logic Appl. (FPL). –2017. – P. 1–8.
57. Anderson, J. H. Raising FPGA logic density through synthesis–inspired architecture / J. H. Anderson, Q. Wang, C. Ravishankar // IEEE Trans. Very Large Scale Integr. (VLSI) Syst. – 2012. – Vol. 20, no. 3, P. 537–550.

58. Chi Wai, Yu. Hybrid FPGA: Architecture and Interface / Chi Wai, Yu. – London : Imperial College of London Department of Computing, 2010. – 178 p.
59. Cong Jason : сайт . – URL: <https://ieeexplore.ieee.org/author/37276009100> (дата обращения: 06.03.2024). – Режим доступа: свободный.
60. Danilova, E. Y. FPGAs Logic Checking Method by Genetic Algorithms / E. Y. Danilova // Proceedings of the 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering. – 2020. – С. 1787–1790.
61. Dillien, P. And the Winner of Best FPGA of 2016 is... : сайт . – URL: <https://www.eetimes.com/and-the-winner-of-best-fpga-of-2016-is/> (дата обращения: 06.03.2024). – Режим доступа: свободный.
62. Drozd, A. Use of Natural LUT Redundancy to Improve Trustworthiness of FPGA Design / A. Drozd, M. Drozd, M. Kuznietsov // CEUR Workshop Proceedings. – 2016. – Vol. 1614. – PP. 322–331.
63. Feng, W. Improving FPGA performance with a S44 LUT structure,” / W. Feng, J. Greene, A. Mishchenko. – DOI 10.1145/3174243.3174272 // Proc. ACM/SIGDA Int. Symp. FieldProgram. Gate Arrays. – 2018. – P. 61–66.
64. Fohl, W. An FPGA-based virtual reality audio system / W. Fohl, D. Hemmer // Audio Eng. Soc. – 2015.
65. Fossum, J. G. Fundamentals of Ultra-Thin-Body MOSFETs and FinFETs / J. G. Fossum, V. P. Trivedi. – Cambridge University Press. – 1st edition. – 2013. – 226 p.
66. FPGA infrastructure for the development of augmented reality applications / G. F. Guimarães, J. P. S. M. Lima, J. M. X. N. Teixeira, G. D. Silva, and others // Proc. 20th Annu. Conf. Integr. Circuits Syst. Design (SBCCI). – 2007. – p. 336.
67. FPGA Architecture White paper : офиц. сайт . – URL: https://altera.com/content/dam/altera-www/global/en_US/pdfs/literature/wp/wp-01003.pdf (дата обращения: 06.03.2024). – Режим доступа: свободный.
68. Freeman, R. Field Programmable Gate Array (FPGA) U.S. Patent No. 4,870,302. – URL: <https://www.invent.org/inductees/ross-freeman#:~:text=Ross%20Freeman.%20Field%20Programmable%20Gate,make%20last>

%20minute%20design%20changes (дата обращения: 01.09.2024). – Режим доступа: свободный.

69. Hidden fault analysis of FPGA projects for critical applications / O. Drozd, I. Perebeinos, O. Martynyuk, K. Zashcholkin, O. Ivanova, M. Drozd. – DOI 10.1109/TCSET49122.2020.235591 // 15th Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering. – 2020. – P. 128–132.

70. Intel® FPGAs and SoC FPGAs : офиц. сайт . – URL: <https://www.intel.in/content/www/in/en/products/details/fpga.html> (дата обращения: 06.03.2024). – Режим доступа: свободный.

71. Intel Agilex 7 Logic Array Blocks and Adaptive Logic Modules User Guide : офиц. сайт . – URL: <https://cdrdv2.intel.com/v1/dl/getContent/667015?fileName=ug-ag-lab-683577-667015.pdf> (дата обращения: 06.03.2024). – Режим доступа: свободный.

72. Introduction of Gate–All–Around Field–Effect Transistor / X. Chengzhen, Z. Yuxin, Z. Kexin, Z.–R. Leon. – DOI 10.54254/2755–2721/28/20230354 // Proc. of the 2023 International Conference on Mechatronics and Smart Systems. – 2023. – P. 164–175.

73. Kajal, FinFET: A Beginning of Non–planar Transistor Era / Kajal, V. Sharma. – DOI 10.1007/978–981–15–7937–0_8 // Nanoscale VLSI. – 2020. P. 139–159.

74. Kaviani, A. Hybrid FPGA architecture / A. Kaviani, S. Brown // University of Toronto, Canada. – 1996. – PP. 1–7.

75. Kharchenko, V. Design and testing technique of FPGA–based critical systems / V. Kharchenko, O. Siora, V. Sklyar // 10th International Conference – The Experience of Designing and Application of CAD Systems in Microelectronics. – 2009. – PP. 107–132.

76. Liang, K. High–Performance Constant–Time Discrete Gaussian Sampling. / K. Liang, L. Shuguo, L. Ruirui. – DOI 10.1109/TC.2020.3001170 // IEEE Transactions on Computers. – 2021. – Vol.70, iss. 7. – P. 1019 – 1033.

77. Libero SoC : офиц. сайт . – URL: <https://www.microsemi.com/product-directory/vectorblox-ai/5598-libero-soc> (дата обращения: 06.03.2024). – Режим доступа: свободный.

78. Maxfield, C. The Design Warrior's Guide to FPGAs: Devices, Tools and Flows / C. Maxfield. – Elsevier. – 2004. – 560 p.
79. Mead, C. A. Introduction to VLSI Systems. / C. A. Mead, L. Conway. – 259 с. – URL: https://www.researchgate.net/publication/234388249_Introduction_to_VLSI_systems (дата обращения: 06.03.2024). – Режим доступа: свободный.
80. Mehta, N. An ultra-low-energy, variation-tolerant FPGA architecture using component-specific mapping / N. Mehta // Dissertation (Ph.D.). – California Institute of Technology. – 2013.
81. Monther, A. A comparison of FinFET based FPGA LUT / A. Monther, P. Khatri. – DOI:10.1145/2591513.2591596 // Great Lakes Symposium on VLSI. – 2014.
82. Dehon, A. Exploiting partially defective LUTs: Why you don't need perfect fabrication / A. Dehon, N. Mehta – DOI: 10.1109/FPT.2013.6718323 // International Conference on Field-Programmable Tehnology (FPT). – 2013.
83. Microwind & Dsch Version 3.8 : офиц. сайт . – URL: <https://www.microwind.net/> (дата обращения: 06.03.2024). – Режим доступа: свободный.
84. Morgan, T. P. How Microsoft Is Using FPGAs To Speed Up Bing Search. / T. P. Morgan // Enterprise Tech. – 2014. – URL: <https://www.enterpriseai.news/2014/09/03/microsoft-using-fpgas-speed-bing-search/> (дата обращения: 06.03.2024). – Режим доступа: свободный.
85. National Instruments : офиц. сайт . – URL: <http://www.ni.com/multisim/> (дата обращения: 06.03.2024). – Режим доступа: свободный.
86. Quartus Prime : офиц. сайт . – URL: <https://www.intel.com/content/www/us/en/products/details/fpga/development-tools/quartus-prime.html> (дата обращения: 06.03.2024). – Режим доступа: свободный.
87. Simpson, P. A. FPGA Design, Best Practices for Team Based Reuse / P. A. Simpson. – Switzerland : Springer International Publishing AG. – 2nd edition. – 2015. – 268 p.

88. Skornyakova, A. Yu. Self-Timed LUT Layout Simulation. / A. Yu. Skornyakova, R.V. Vikhorev. – DOI 10.1109/EIConRus49466.2020.9039374 // Conference of Russian Young Researchers in Electrical and Electronic Engineering. – 2020. – P. 176–179.
89. Skorobogatov, S. Breakthrough Silicon Scanning Discovers Backdoor in Military Chip / S. Skorobogatov, C. Woods // Cryptographic Hardware and Embedded Systems – CHES 2012. Lecture Notes in Computer Science. – 2012. – Vol. 7428. – P. 23–40.
90. Sovetov, S.I. Multi-Function LUT for FPGAs / S.I. Sovetov, S.F. Tyurin // Proceedings of the Seminar on Microelectronics, Dielectrics and Plasmas, Theory and Practical Applications, MDP. – 2023. – P.122–126. DOI 10.1109/MDP60436.2023.10424229.
91. Tyurin, S. F. Green Logic: Green LUT FPGA Concepts, Models and Evaluations / S. F. Tyurin. – DOI 10.1007/978-3-319-55595-9 // Green IT Engineering: Components, Networks and Systems Implementation. – 2017. – Vol. 105. – P. 241–261.
92. Tyurin, S. F. LUT's Sliding Backup / S. F. Tyurin. – DOI 10.1109/TDMR.2019.2898724 // IEEE transactions on device and materials reliability. – 2019. – Vol. 19. – P. 221–225.
93. UltraScale Architecture: Highest Device Utilization, Performance, and Scalability White Paper (WP455) : офиц. сайт . – URL: <https://docs.xilinx.com/v/u/en-US/wp455-utilization> (дата обращения: 06.03.2024). – Режим доступа: свободный.
94. Vasenin, I. A. Advanced Logic Gates for FPGAs / I. A. Vasenin, S. I. Sovetov, N. E. Oputin, S. F. Tyurin // International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices, EDM. – 2023. – P.110–115. DOI 10.1109/EDM58354.2023.10225215.
95. Vikhorev, R. Universal logic cells to implement systems functions / R. Vikhorev. – DOI 10.1109/EIConRusNW.2016.7448197 // Conference of Russian Young Researchers in Electrical and Electronic Engineering. – 2016. – P. 404–406.

96. Vikhorev, R. Improved FPGA logic elements and their simulation / R. Vikhorev. – DOI 10.1109/EIConRus.2018.8317080 // Conference of Russian Young Researchers in Electrical and Electronic Engineering. – 2016. – P. 275–280.
97. Vivado Design Suite : офиц. сайт . – URL: <https://www.xilinx.com/products/design-tools/vivado.html> (дата обращения: 06.03.2024). – Режим доступа: свободный.
98. Wanlass, F. Complementary Metal Oxide Semiconductor (CMOS) U.S. Patent No. 3,356,858. – URL: <https://www.invent.org/inductees/frank-wanlass> (дата обращения: 02.09.2024). – Режим доступа: свободный.
99. Yervant, Z. Gest editors' introduction: Design for Yield and reliability / Z. Yervant, G. Dmytris // IEEE Design & Test of Computers. – May–June 2004. – Pp. 177–182.
100. Zhou, Y. An FPGA-based accelerator implementation for deep convolutional neural networks / Y. Zhou, J. Jiang // Proc. 4th Int. Conf. Comput. Sci. Netw. Technol. (ICCSNT). – Harbin, China. – 2015. – P. 829–832.
101. Zhong, C. High-Speed Phase Structured Light Integrated Architecture on FPGA / C. Zhong, H. Tianlong, H. Yuya, Z. Xianmin. – DOI 10.1109/TIE.2023.3250771 // IEEE Transactions on Industrial Electronics. – Vol. 71, Iss.: 1. – 2024. – P. 1017 – 1027.

Приложение А

Программа для подключения дополнительных транзисторов в многофункциональном логическом элементе ПЛИС «МФЛУТ»

1. Описание программы

1.1. Общие сведения.

Программа для подключения дополнительных транзисторов в многофункциональном логическом элементе ПЛИС предназначена для описания реализации дополнительных функций в логическом элементе ПЛИС за счет подключения транзисторов к сигналам переменных, к базовому дереву транзисторов и подключение транзисторов настройки к статической памяти. Для работы программы требуется интерпретатор языка Python с библиотекой math.

1.2. Функциональное назначение.

В зависимости от введенных данных по количеству переменных (n) и необходимых функций (m) отображается базовое дерево транзисторов логического элемента, дополнительные транзисторы для реализации дополнительных функций и подключение к сигналам переменных.

1.3. Описание логической структуры.

В программе реализованы функции: ввода количества переменных, ввода количества реализуемых функций, вывод построчный в консоль, вывод выходов переменных, вывод дерева таблицы поиска, вывод дополнительных деревьев, вывод транзисторов настройки.

1.4. Используемые технические средства.

Для исполнения программы достаточно ЭВМ с интерпретатором языка Python. Программа может работать на любой ЭВМ под управлением операционных систем семейств Windows, macOS, GNU/Linux, BSD и др.

1.5. Вызов и загрузка.

Вызов осуществляется из консоли с помощью интерпретатора Python.

1.6. Входные данные.

Количество переменных – n и количество реализуемых функций – m .

1.7. Выходные данные

Выводится структура деревьев таблицы поиска с отображением подключения основных транзисторов к сигналам переменных так и дополнительных деревьев и транзисторов настройки. Используется псевдографика консоли.

2. Листинг программы.

```
import math
def inputN():
    #ввод и проверка количества переменных n
    while True:
        print("Введите количество переменных для LUT (1 .. 10)")
        try:
            n = int(input())
            if n < 1 or n > 10:
                raise ValueError("Количество переменных должно быть от
1 до 10")
            break
        except ValueError as e:
            print(e)
    return n
def inputM(n):
    #ввод и проверка количества функций m
    while True:
        print("Введите количество функций для n =", n, "(1", end="")
        stringM = "1"
        if n < 6:
            for i in range(2,n+1):
                print(", ", 2**(i-1), end="")
                stringM += " "+str(2**(i-1))
            print(")")
        else:
```

```

    for i in range(2,6):
        print(" ",2**(i-1), end="")
        stringM += " "+str(2**(i-1))
    print(")")
try:
    m = int(input())
    if not(str(m) in stringM) or m==6:
        raise ValueError("Возможное количество функций: " +
stringM)
        break
except ValueError as e:
    print(e)
return m
def printLines(n,shift,beginline,data,index,lines,noIndex,startNumber=1):
    for j in range(0,lines):
        if index < 0:
            indexStr = ""
        else:
            indexStr = "."+str(index+j)
        print(beginline, end=")
        bit = 0
        for i in range(startNumber,startNumber+n):
            print("\t", end="")
            bit += 1
            if format(2**(n+2)+shift,"b")[-bit] == "1":
                if noIndex:
                    print("\t"+data, end="") #справа
                else:
                    print("\t"+data+str(i)+indexStr, end="") #справа
            else:

```

```

        print(data+str(i)+indexStr, end="\t") #слева
#инвертировать биты для второй строки
shift = 2**n-1-shift
print("")
def printLUTvar(n,startNumber=1):
    printLines(n,0,"Переменные:", "|X",-1,1,False,startNumber)
    printLines(n,2**n-1,"      ", "|__",-1,1,True)
    printLines(n,2**n-1,"      ", "|X",-1,1,False,startNumber)
def printLUTtree(n,numLUT,lines,startNumber=1):
    mas = []
    for k in range(0,n):
        mas.append((2**(n-k-1)))
    for j in range(0,mas[0]):
        count = 0
        for i in range(0,len(mas)):
            if mas[i] > 0:
                count += 1
                mas[i] = mas[i] - 1
            if startNumber!=1 and lines<=2:
                printLines(count,numLUT,"      ", "|T", (j+1)*2-
1,lines,False,startNumber)
            elif startNumber!=1 and lines>2:
                printLines(count,numLUT,"      ", "|T", ((j+1)*2-
1)+numLUT*lines,lines,False,startNumber)
            else:
                printLines(count,0,"      ", "|T", (j+1)*2-1,lines,False)

def printAdditionalLUT(m,n):
    print("Дополнительные LUT:")
    powM = int(math.log2(m))

```

```

for i in range(1,m):
    printLUTvar(powM,n-powM+1)
    printLUTtree(powM,i,2,n-powM+1)
    print("—————")

def printConfigTrans(m,n):
    print("Транзисторы настройки:")
    powM = int(math.log2(m))
    printLUTvar(powM,n-powM+1)
    for i in range(0,m):
        printLUTtree(powM,i,int(2**(n+1)/m),n-powM+1)

#begin
n = inputN()
m = inputM(n)
printLUTvar(n)
printLUTtree(n,1,2)
printAdditionalLUT(m,n)
printConfigTrans(m,n)

```

Программа соединения блоков функции-дешифрации на одну переменную по уровням дерева транзисторов элемента LUT "ДШФЛУТ"

1. Описание программы

1.1. Общие сведения.

Программа соединения блоков функции-дешифрации на одну переменную по уровням дерева транзисторов элемента LUT предназначена для описания реализации дешифрации входного набора в логическом элементе ПЛИС за счет подключения дополнительных транзисторов, обеспечивающих ортогональный обратный сигнал к используемой ячейке памяти и отображения её использования.

1.2. Функциональное назначение.

В зависимости от введенных данных количества переменных отображается базовый мультиплексор на одну переменную, соединения необходимого количества мультиплексоров для реализации LUT на n переменных по индексам входов и выходов.

1.3. Описание логической структуры.

В программе реализованы функции: ввода количества переменных, вывод построчный в консоль, вывод выходов переменных, вывод дерева таблицы поиска, вывод дополнительных деревьев, вывод транзисторов настройки.

1.4. Используемые технические средства.

Для исполнения программы достаточно ЭВМ с интерпретатором языка Python. Программа может работать на любой ЭВМ под управлением операционных систем семейств Windows, macOS, GNU/Linux, BSD и др.

1.5. Вызов и загрузка.

Вызов осуществляется из консоли с помощью интерпретатора Python.

1.6. Входные данные.

Количество переменных – n и количество реализуемых функций – m .

1.7. Выходные данные

Выводится структура деревьев таблицы поиска с отображением подключения основных транзисторов к сигналам переменных так и дополнительных деревьев и транзисторов настройки. Используется псевдографика консоли.

2. Листинг программы.

```
import math
```

```
def inputN():
```

```
    #ввод и проверка количества переменных n
```

```
    while True:
```

```
        print("Введите количество переменных для LUT (1 .. 10)")
```

```
        try:
```

```
            n = int(input())
```

```

        if n < 1 or n > 10:
            raise ValueError("Количество переменных должно быть от
1 до 10")
            break
    except ValueError as e:
        print(e)
    return n

```

```
def printBaseLUT():
```

```
    #базовая схема 1-LUT
```

```
    print("1-LUT:\t\t |X_n_p |")
```

```
    print("\t\t |t |_")
```

```
    print("\t\t |t |X_n_p")
```

```
    print("F_n_i_0 _____|T1_____|_____")
```

```
    print("F_n_i_1 _____|_____|T2____|---F_n_o_p")
```

```
    print(" _____|_____|T3____")
```

```
    print("P_n_i-| gnd_____|T4_____|_____|---P_n_o_0")
```

```
    print(" |_____|T5_____|_____")
```

```
    print(" gnd_____|_____|T6____|---P_n_o_1")
```

```
def printMUX(n,m):
```

```
    #мультиплексор
```

```
    for p in range(0,2**(m+1),+2):
```

```
        print(f"\t |X_{n}_{int(p/2)}")
```

```
        print(" |\\")
```

```
        if n == 1:
```

```
            print(f"SRAM_{p}_____|\\")
```

```
            if m == 0:
```

```
                print(f"SRAM_{p+1}_____| |__F_out")
```

```
                print(f" VCC_____| |__D_o_{p}")
```

else:

```
print(f"SRAM_{p+1}_____ | ___F_{n}_o_{int(p/2)}")
```

```
print(f"P_{n}_i_{int(p/2)}_____ | ___D_o_{p}")
```

```
print(f"          | ___D_o_{p+1}")
```

else:

```
print(f"F_{n-1}_i_{p}_____ | \")
```

if m == 0:

```
print(f"F_{n-1}_i_{p+1}_____ | ___F_out")
```

```
print(f" VCC_____ | ___P_{n-1}_o_{p}")
```

else:

```
print(f"F_{n-1}_i_{p+1}_____ | ___F_{n}_o_{int(p/2)}")
```

```
print(f"P_{n}_i_{int(p/2)}_____ | ___P_{n-1}_o_{p}")
```

```
print(f"          | ___P_{n-1}_o_{p+1}")
```

```
print("          | /")
```

```
print("          | /")
```

```
print("")
```

```
#begin
```

```
n = inputN()
```

```
printBaseLUT()
```

```
m = 0;
```

```
for i in range(n,0,-1):
```

```
    printMUX(i,m)
```

```
    m = m + 1
```

Приложение Б

Дополнительные результаты топологического моделирования в системе Microwind

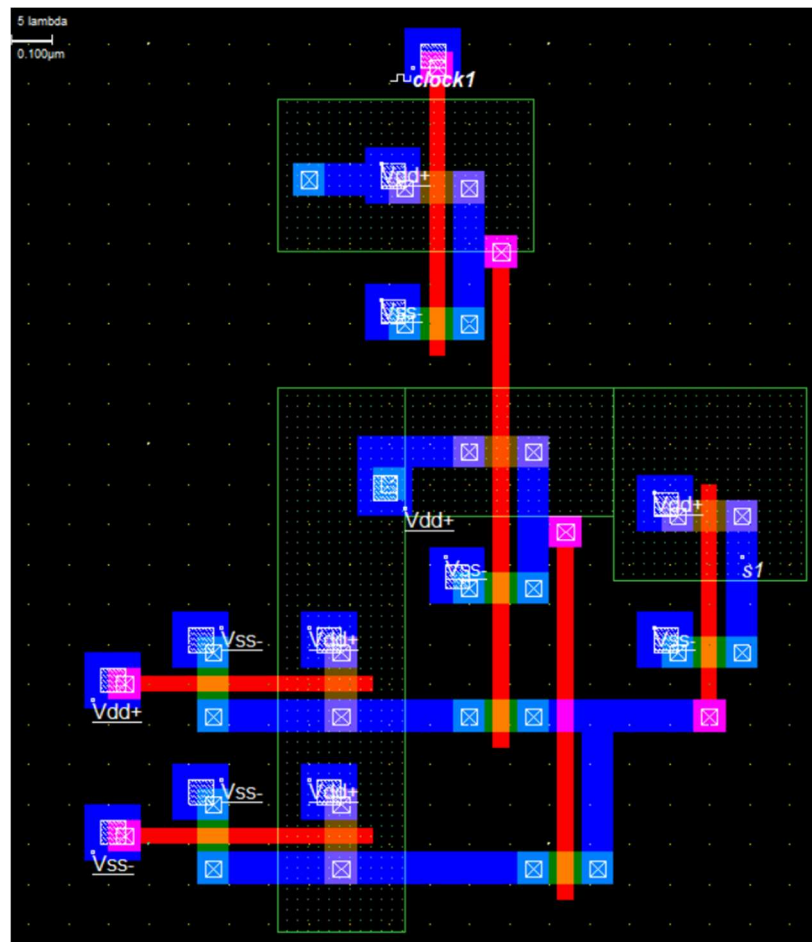


Рисунок ПБ 1а. Топология 1-LUT

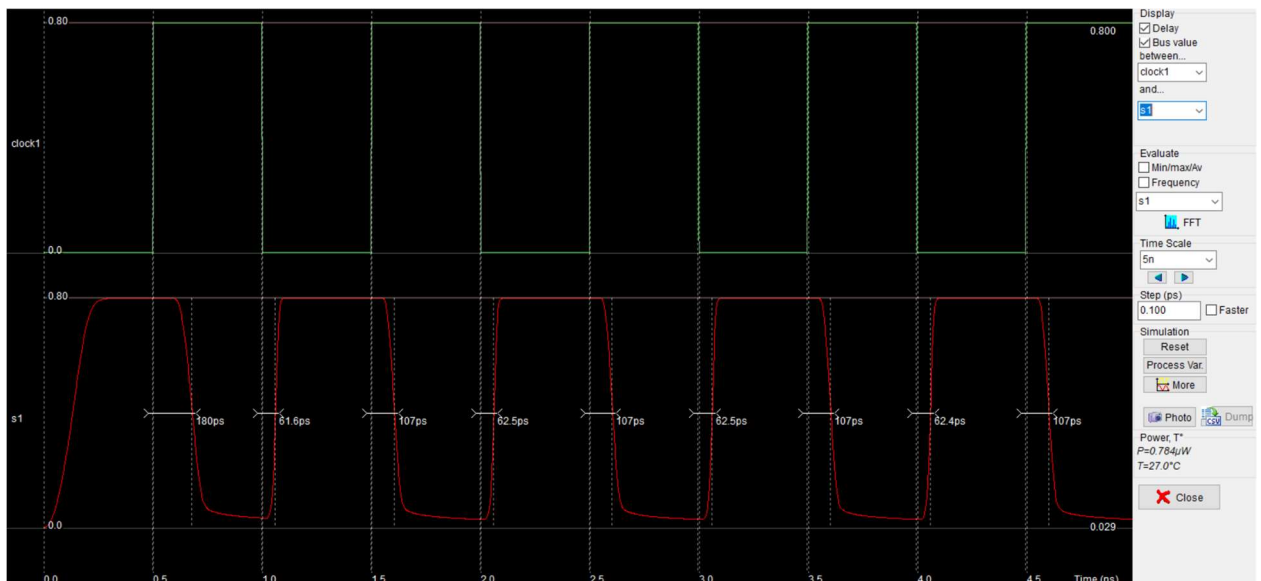


Рисунок ПБ 1б – Результат моделирования 1-LUT

Layout Size	Electrical Properties
Width: 1.8 μm (90 lambda)	electrical nodes : 20/3000
Height: 2.3 μm (113 lambda)	nMOS devices : 7/2000
Surf: 4.1 μm^2 (0.0 mm 2)	pMOS devices : 5/2000

Рисунок ПБ 1в – Занимаемая площадь на кристалле 1-LUT

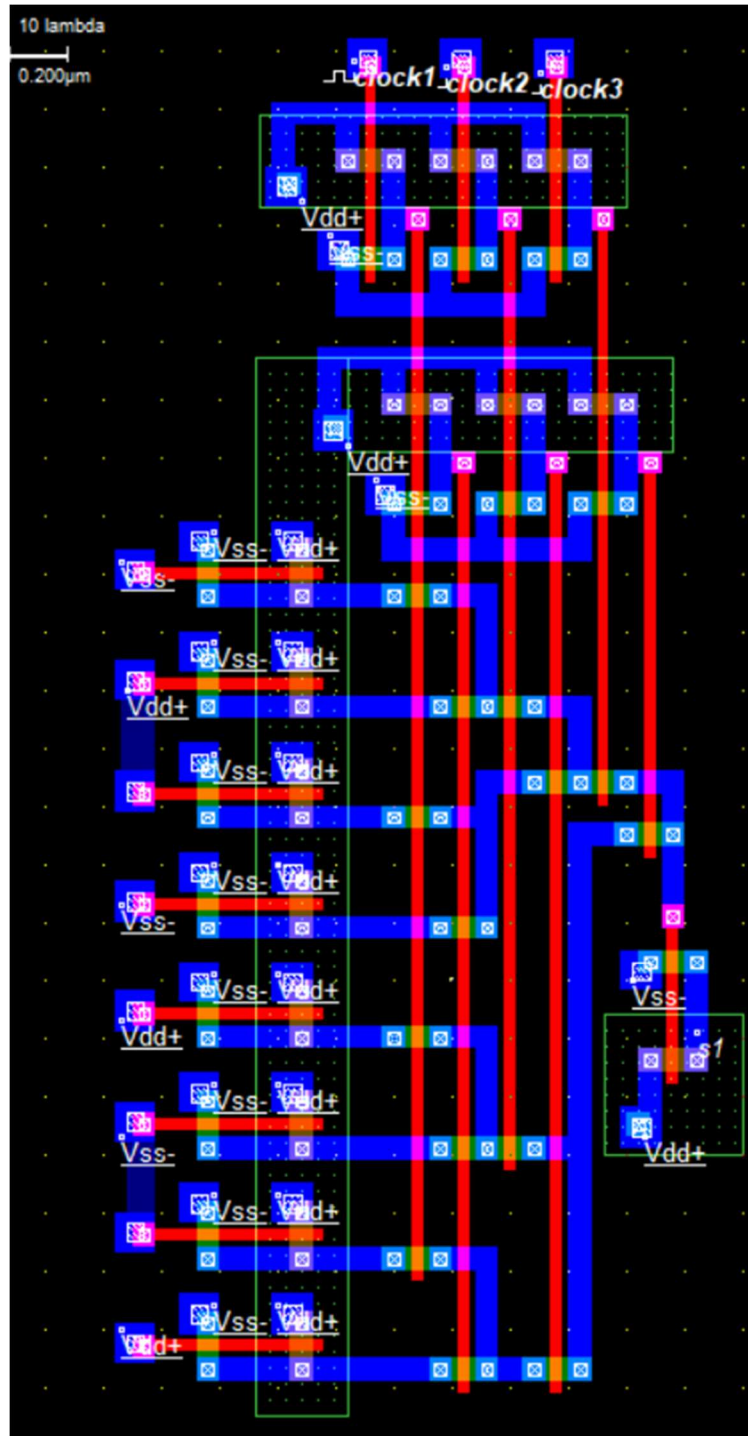


Рисунок ПБ 2а – Топология 3-LUT

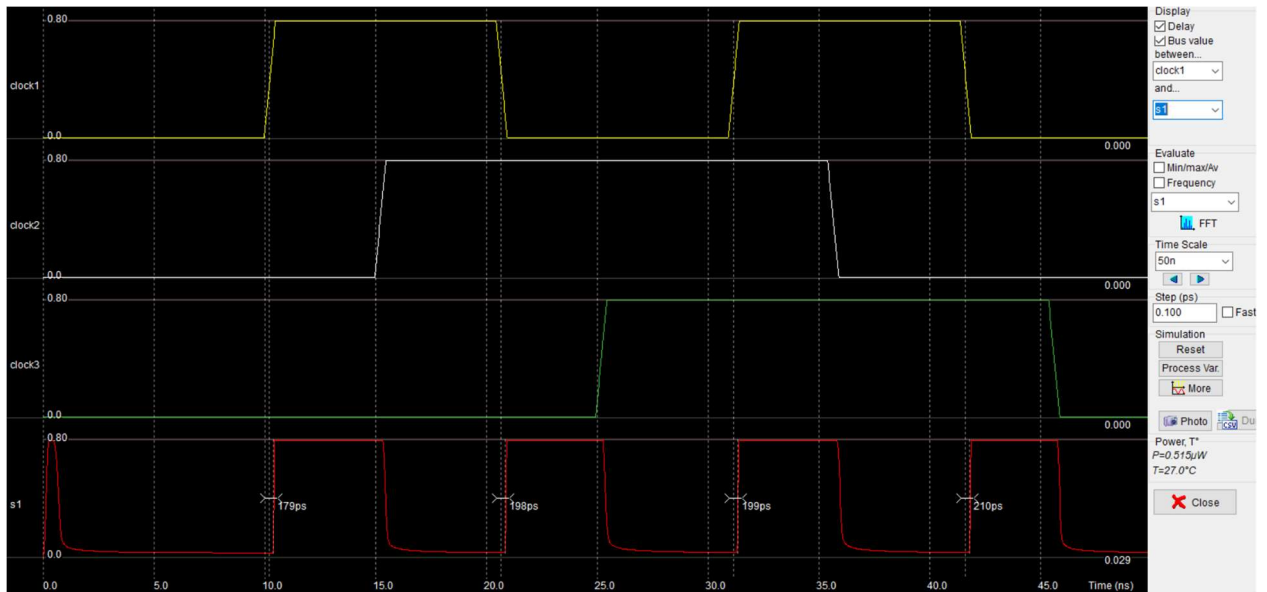


Рисунок ПБ 2б – Результат моделирования 3-LUT

Layout Size	Electrical Properties
Width: 2.2 μ m (108 lambda)	electrical nodes : 54/3000
Height: 4.7 μ m (237 lambda)	nMOS devices : 29/2000
Surf. 10.2 μ m ² (0.0 mm ²)	pMOS devices : 15/2000

Рисунок ПБ 2в – Занимаемая площадь на кристалле 3-LUT

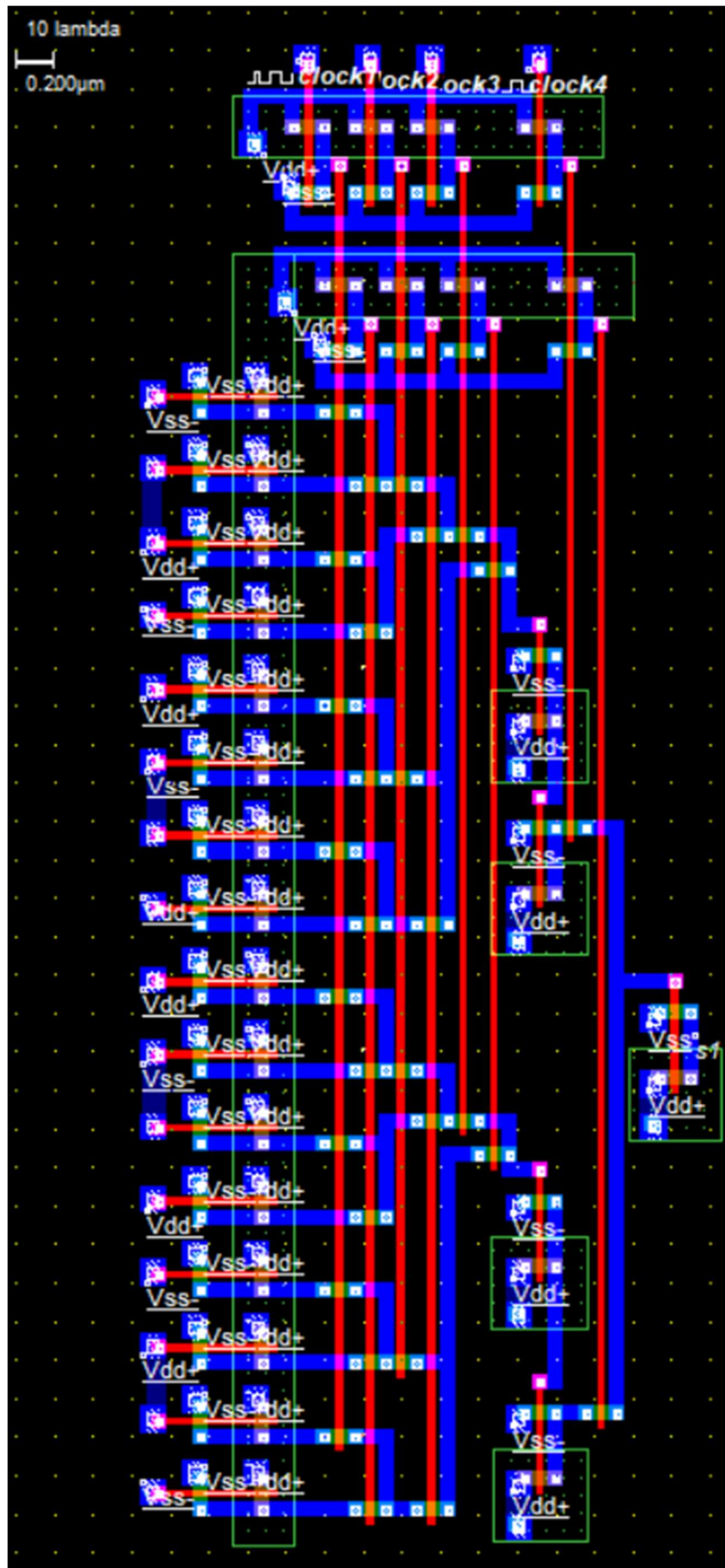


Рисунок ПБ 3а – Топология 4-LUT

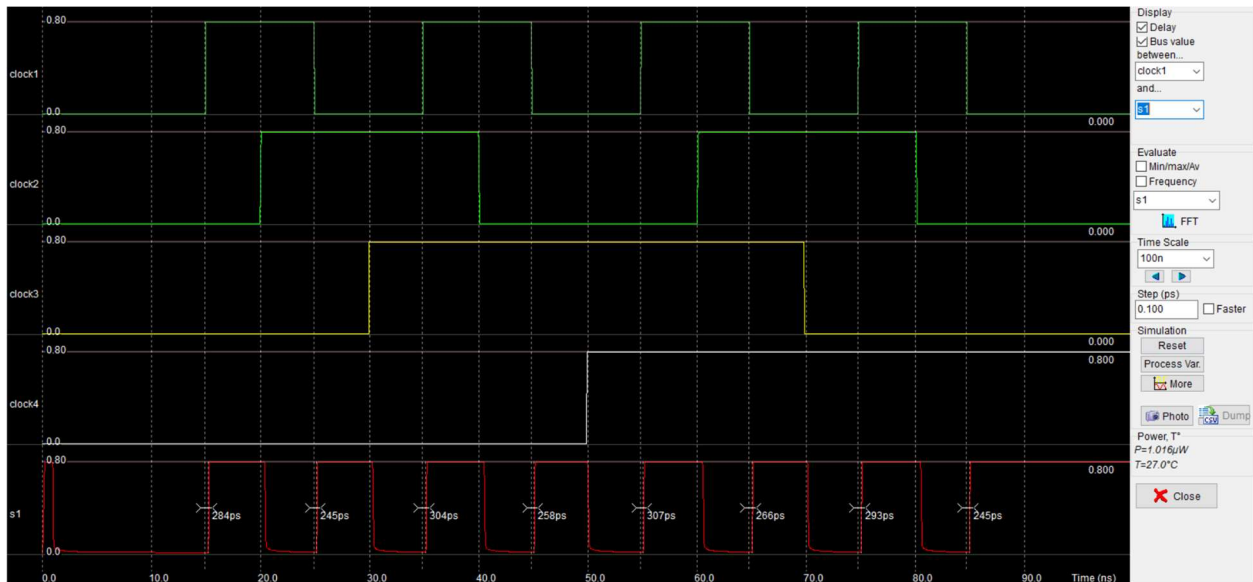


Рисунок ПБ 3б – Результат моделирования 4–LUT

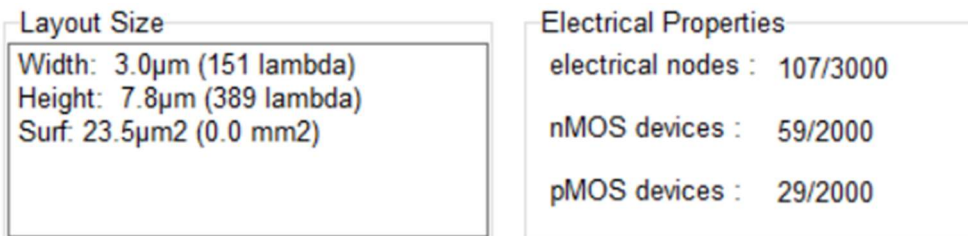


Рисунок ПБ 3в – Занимаемая площадь на кристалле 4–LUT

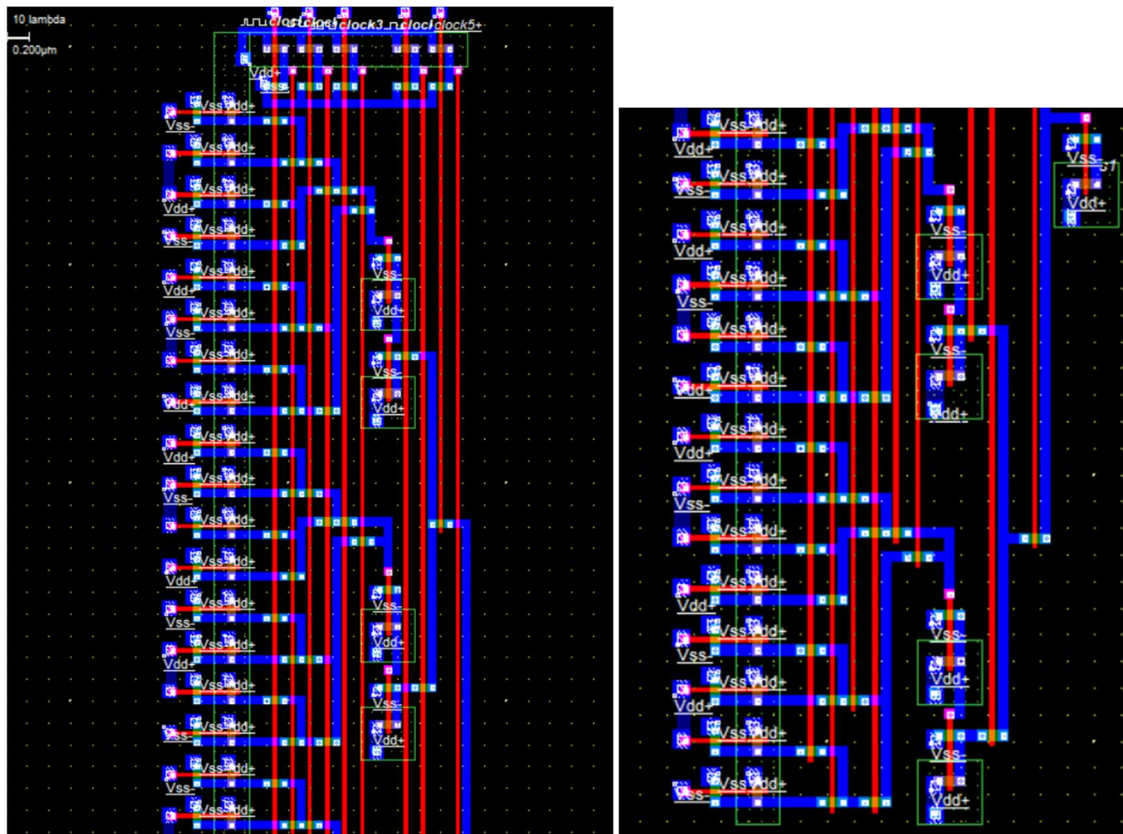


Рисунок ПБ 4а – Топология 5–LUT

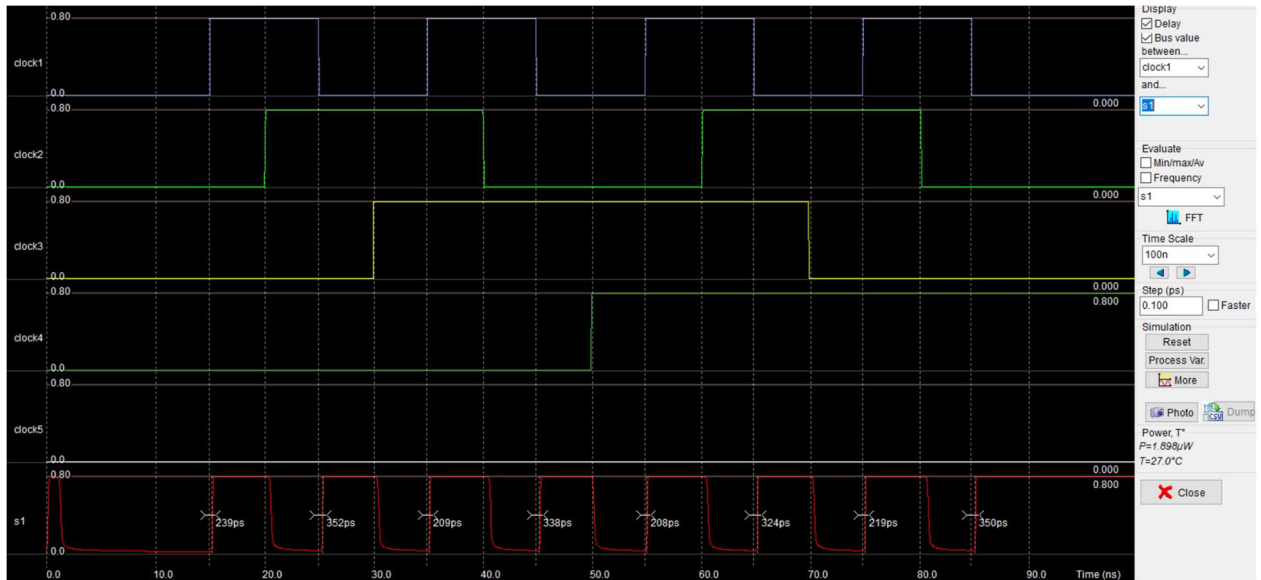
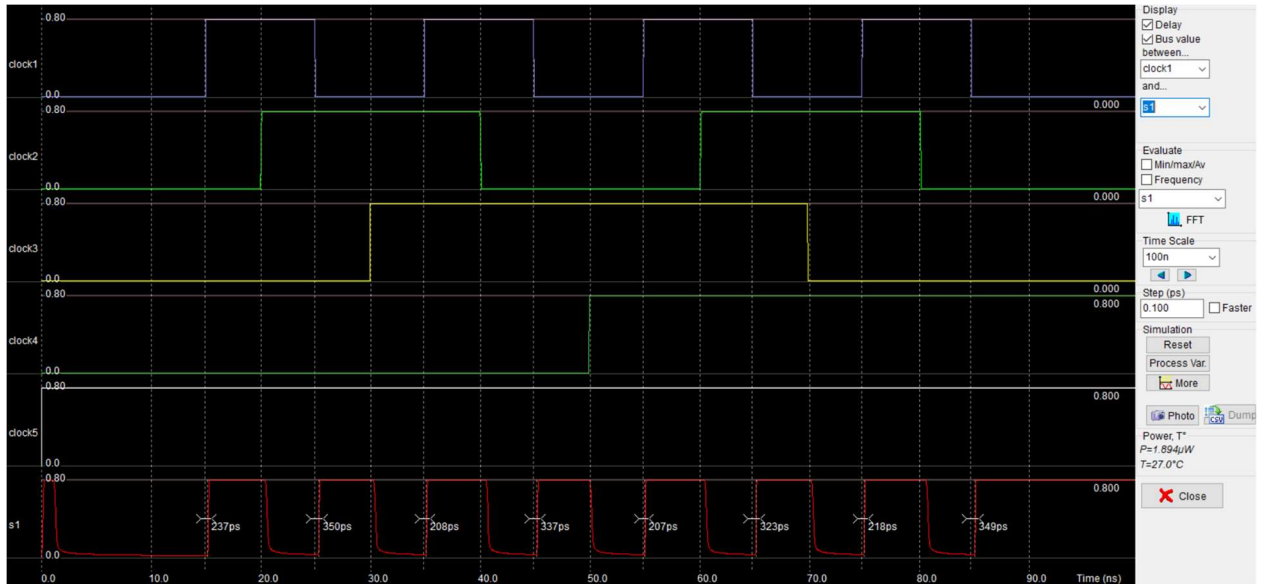


Рисунок ПБ 4б – Результат моделирования 5-LUT

Layout Size	Electrical Properties
Width: 3.3 μ m (167 lambda)	electrical nodes : 191/3000
Height: 13.0 μ m (651 lambda)	nMOS devices : 108/2000
Surf: 43.5 μ m ² (0.0 mm ²)	pMOS devices : 46/2000

Рисунок ПБ 4в – Занимаемая площадь на кристалле 5-LUT

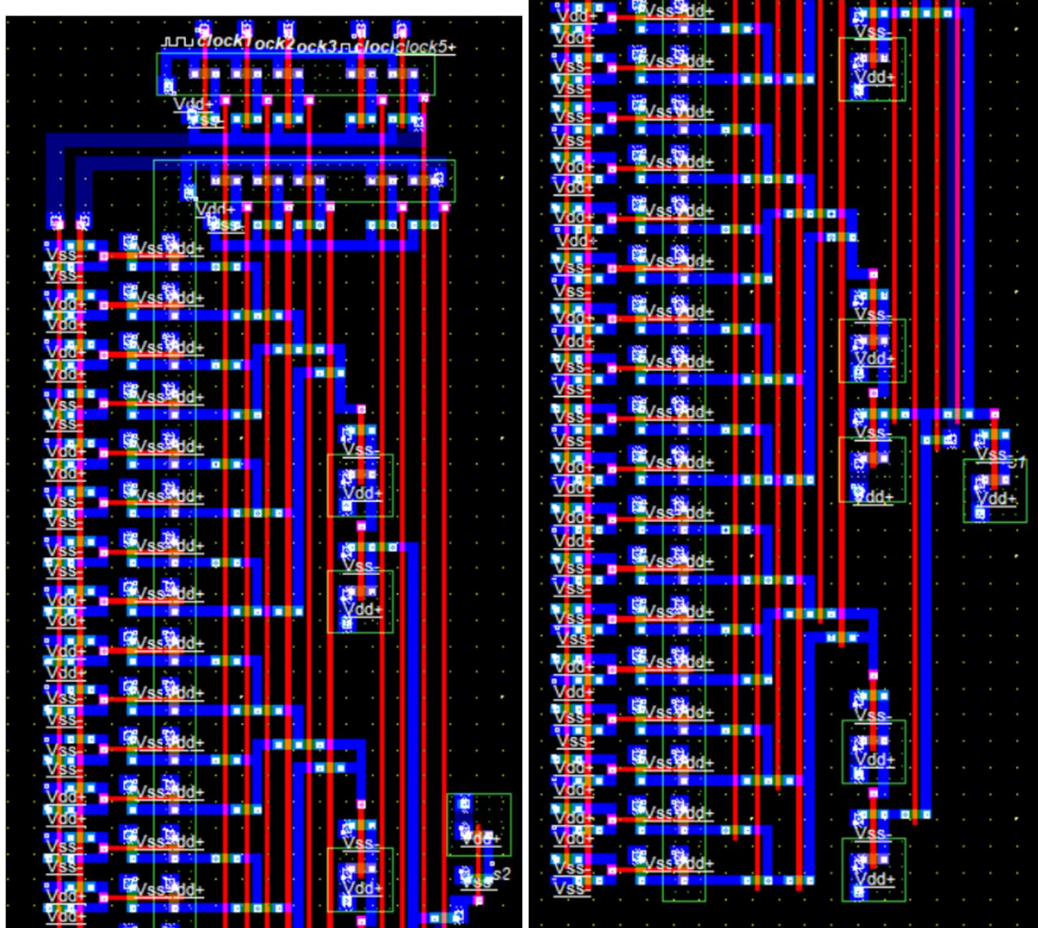
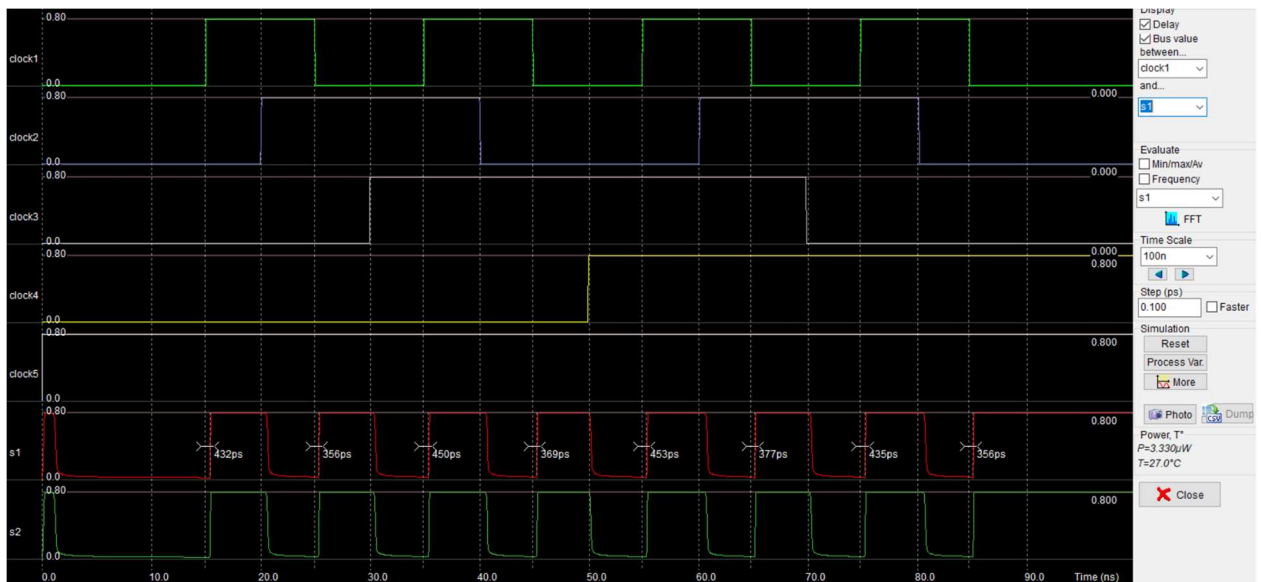


Рисунок ПБ 5а – Топология 5–LUT реализующая две функции одновременно



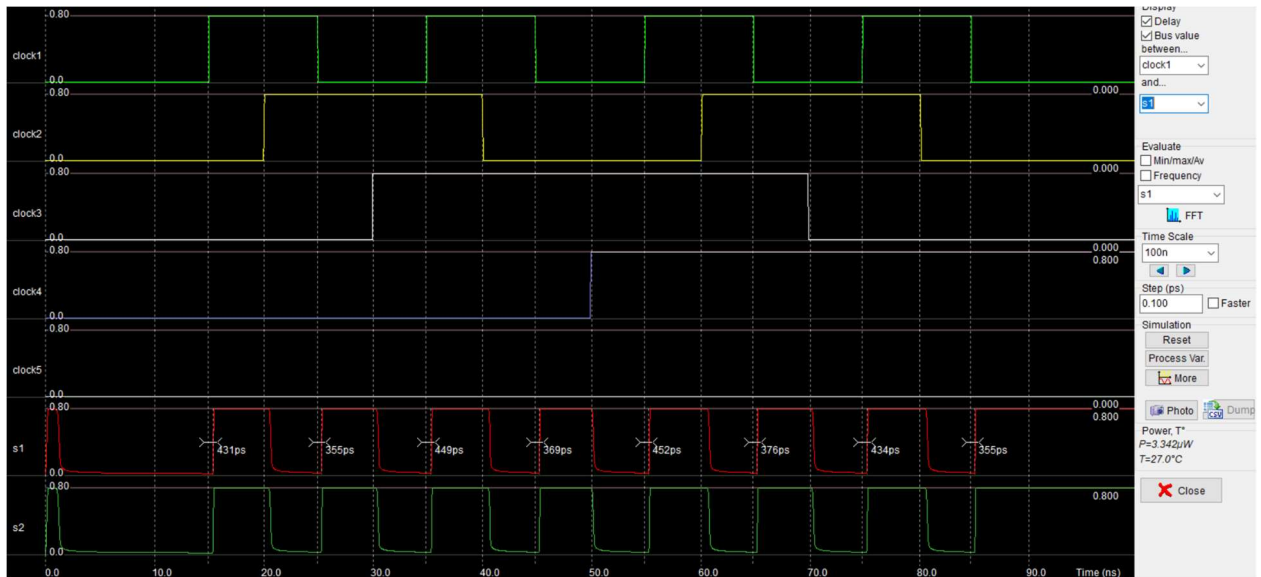


Рисунок ПБ 5б – Результат моделирования двухфункционального 5–LUT

Layout Size	Electrical Properties
Width: 3.6 μ m (180 lambda)	electrical nodes : 274/3000
Height: 13.8 μ m (692 lambda)	nMOS devices : 180/2000
Surf. 49.8 μ m ² (0.0 mm ²)	pMOS devices : 52/2000

Рисунок ПБ 5в – Занимаемая площадь на кристалле двухфункционального 5–LUT

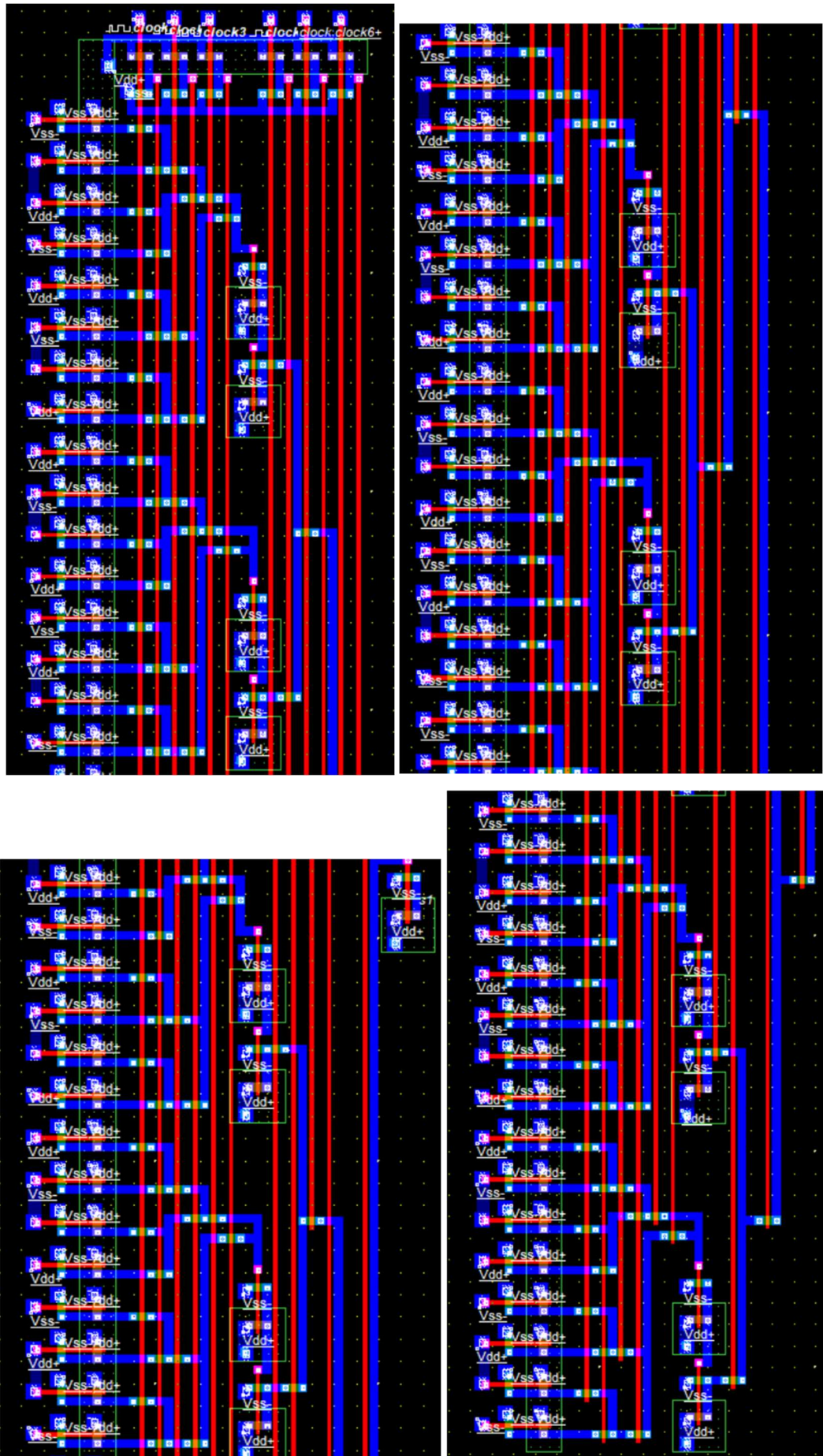
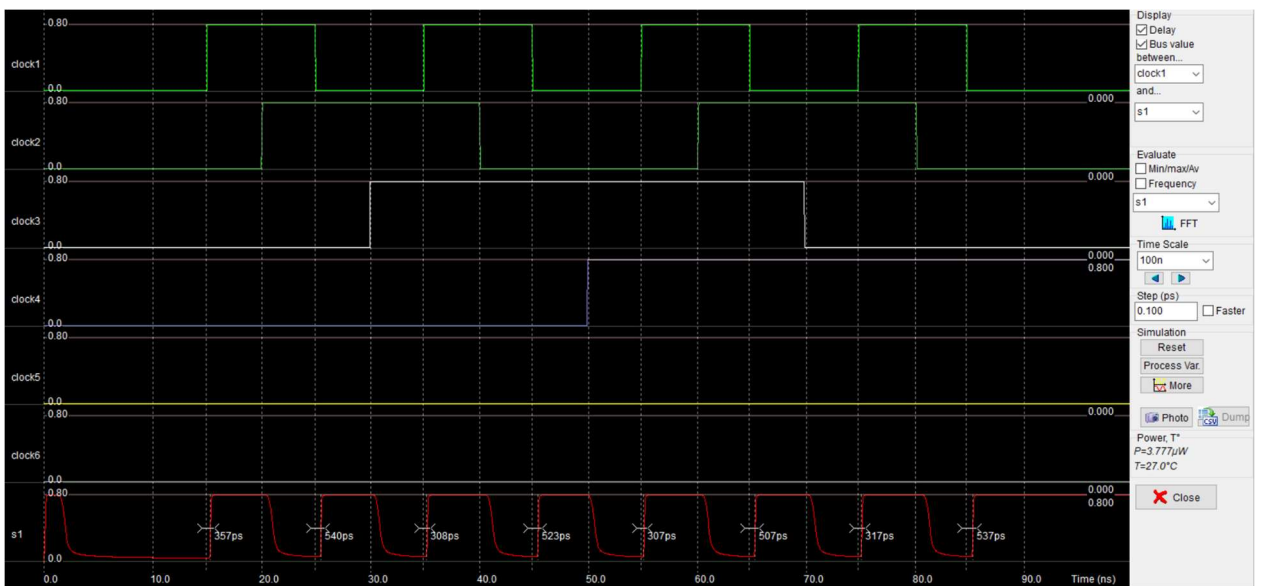
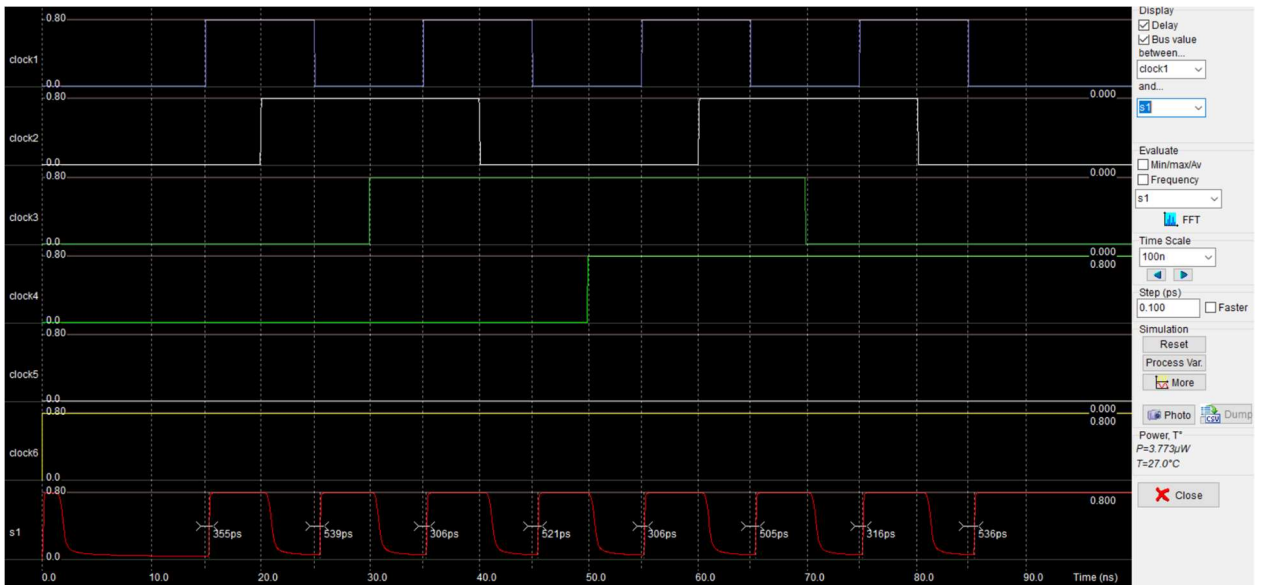
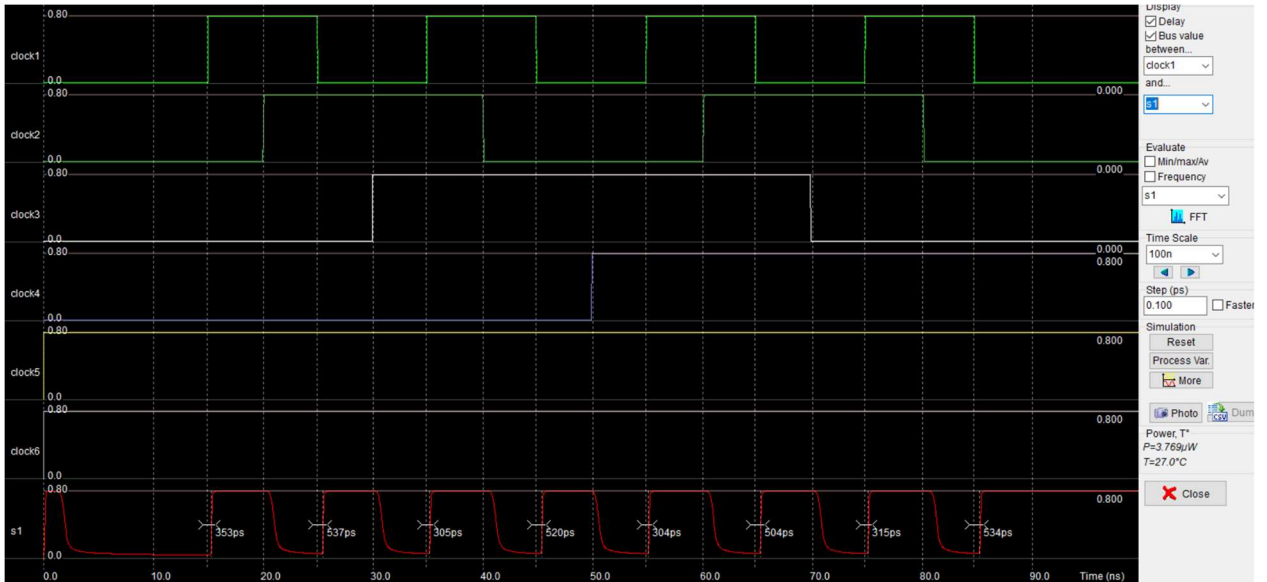


Рисунок ПБ 6а – Топология 6-LUT



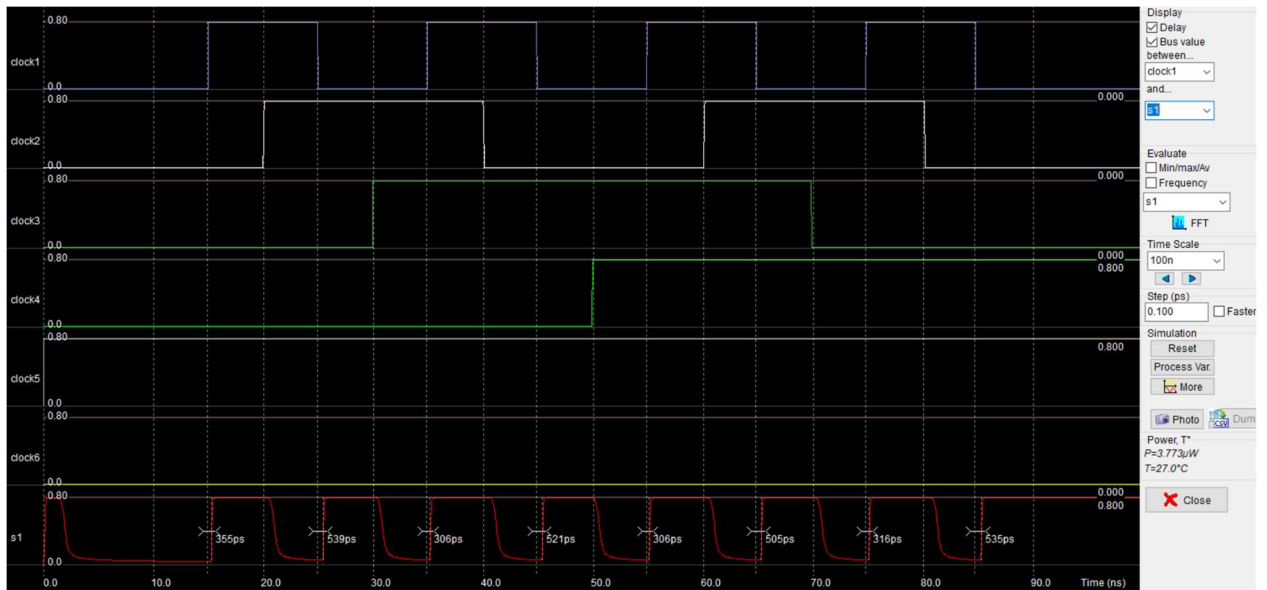
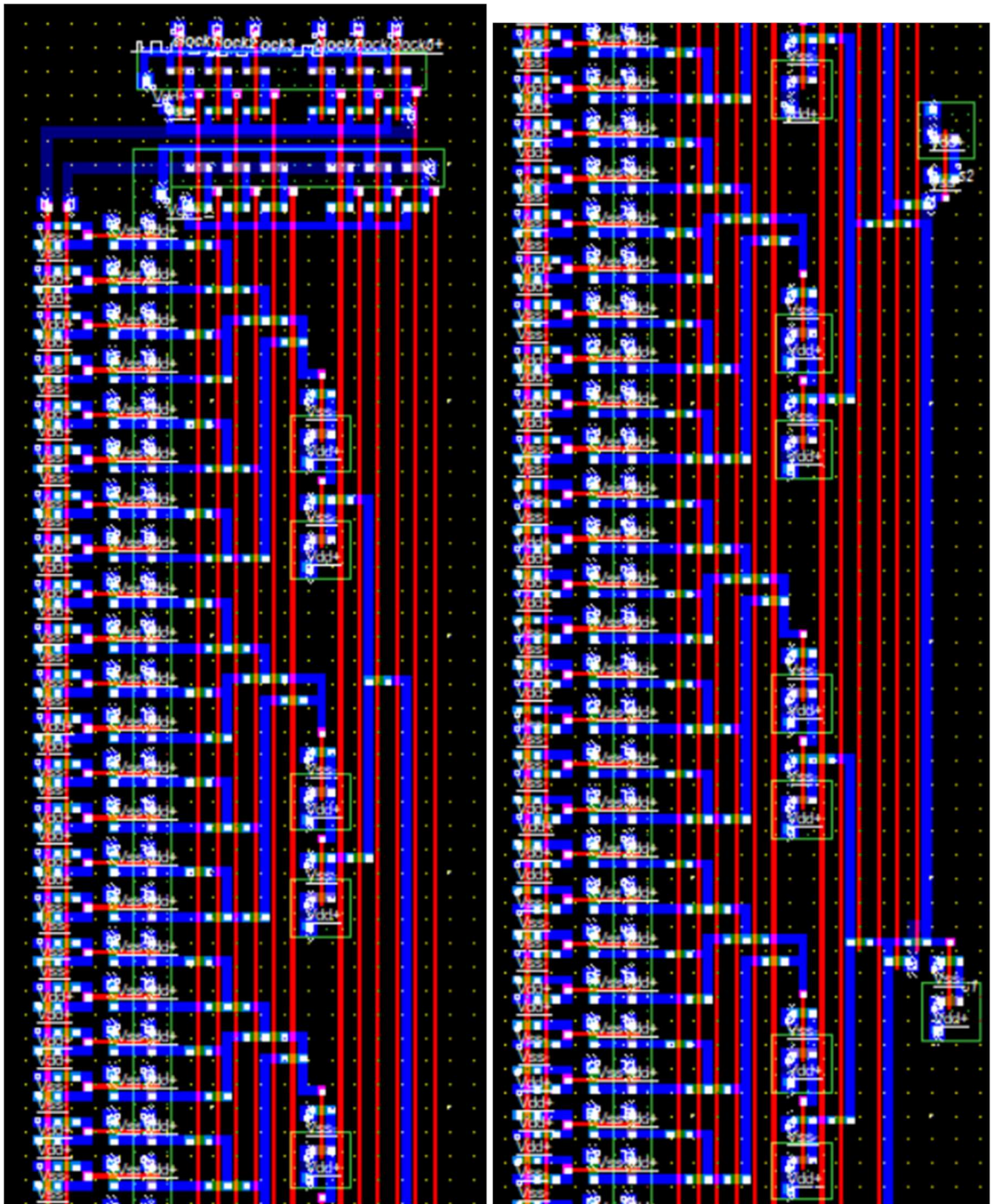


Рисунок ПБ 6б – Результат моделирования 6-LUT

Layout Size	Electrical Properties
Width: 3.7 μ m (183 lambda)	electrical nodes : 369/3000
Height: 25.2 μ m (1259 lambda)	nMOS devices : 213/2000
Surf: 92.2 μ m ² (0.0 mm ²)	pMOS devices : 87/2000

Рисунок ПБ 6в – Занимаемая площадь на кристалле 6-LUT



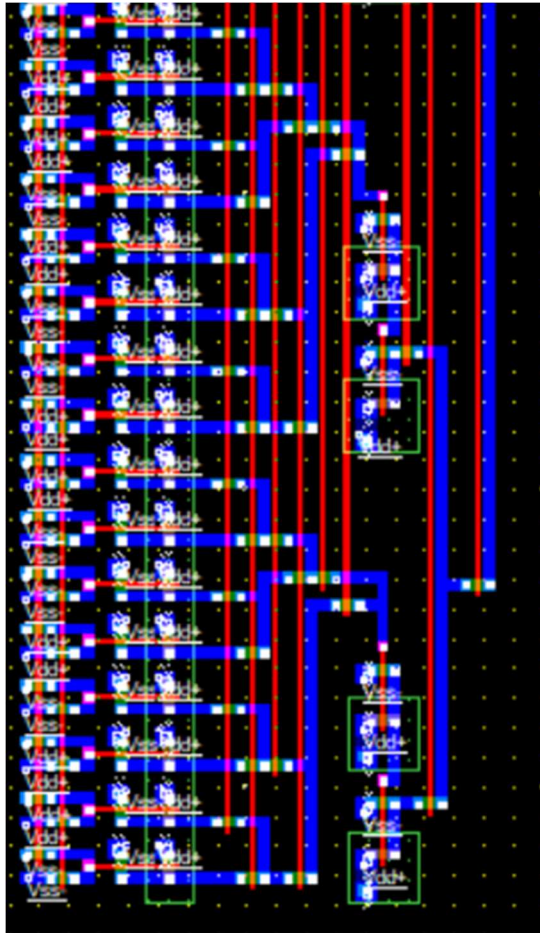


Рисунок ПБ 7а – Топология 6–LUT реализующая две функции одновременно

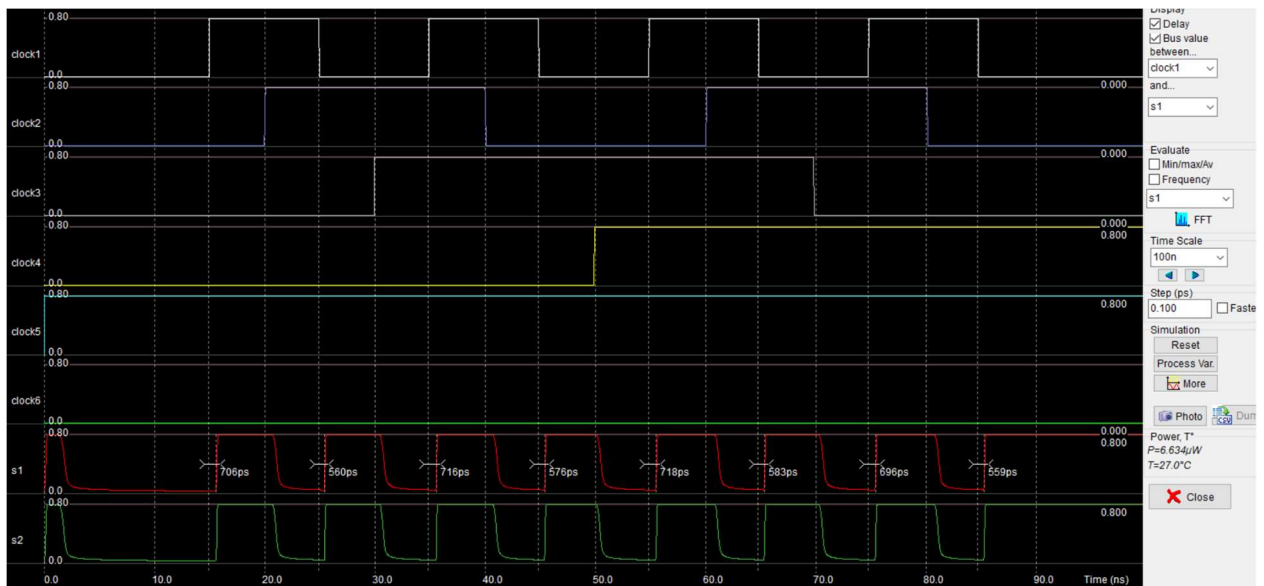




Рисунок ПБ 7б – Результат моделирования двухфункционального 6-LUT

Layout Size	Electrical Properties
Width: 3.9 μm (196 lambda)	electrical nodes : 525/3000
Height: 26.0 μm (1300 lambda)	nMOS devices : 350/2000
Surf: 101.9 μm^2 (0.0 mm 2)	pMOS devices : 94/2000

Рисунок ПБ 7в – Занимаемая площадь на кристалле двухфункционального 6-LUT

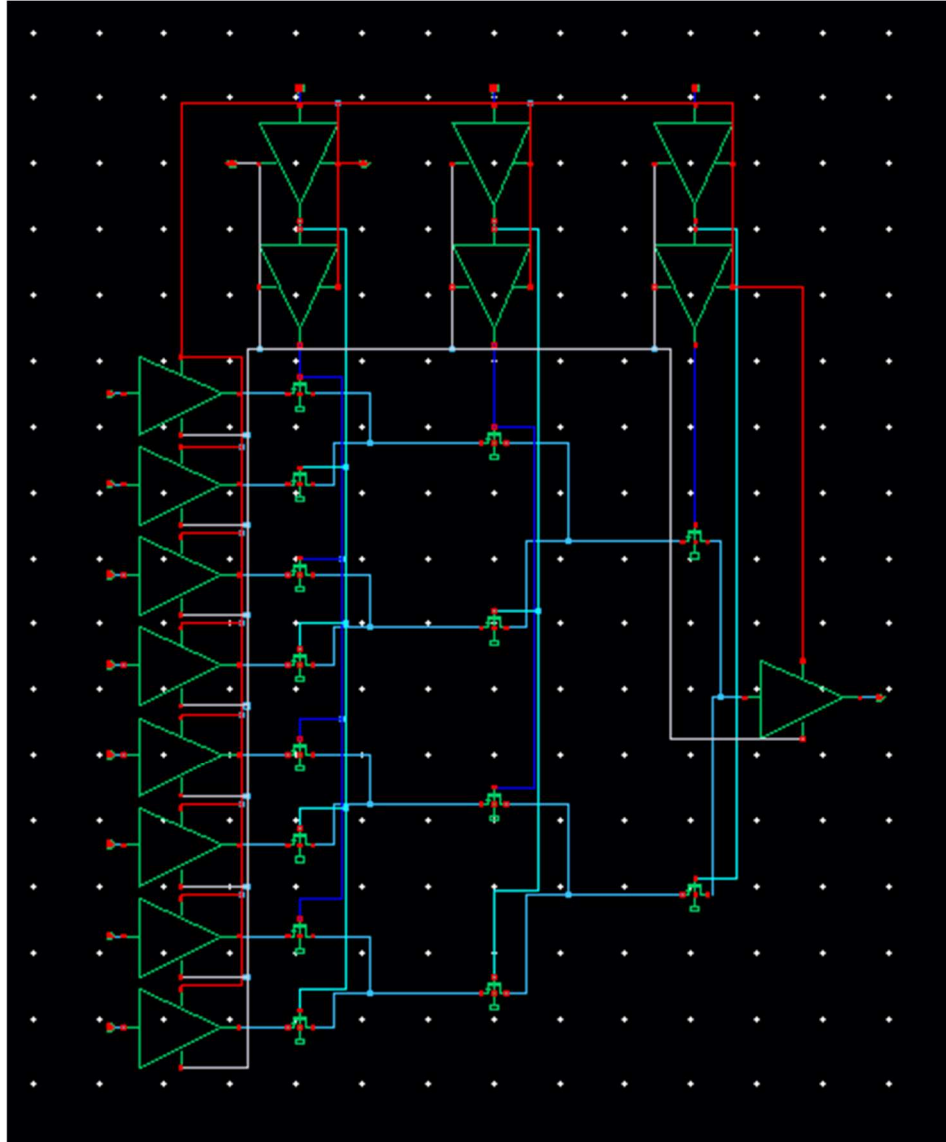


Рисунок ПБ 8а – Схема реализации 3-LUT в системе Cadence Virtuoso с использованием grdk по технологии 45 нм.

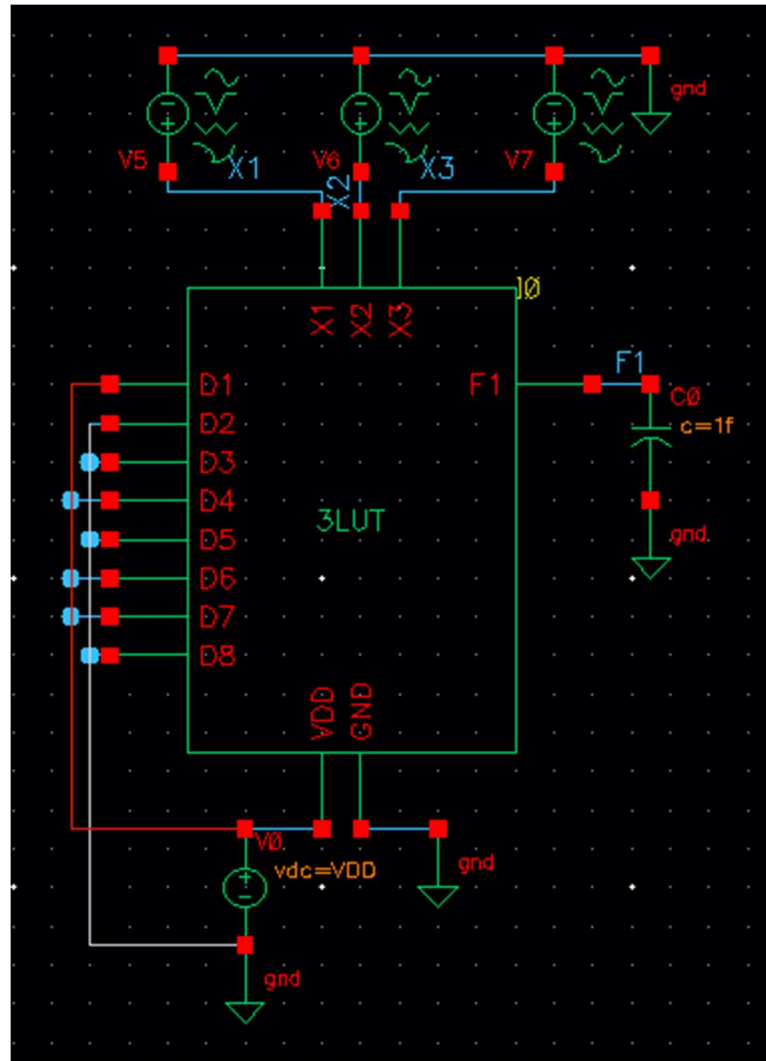


Рисунок ПБ 8б – Test bench 3-LUT, где X1–X3 – сигналы переменных, D1–D8 – статическая память настройки, F1 – выход логической функции.

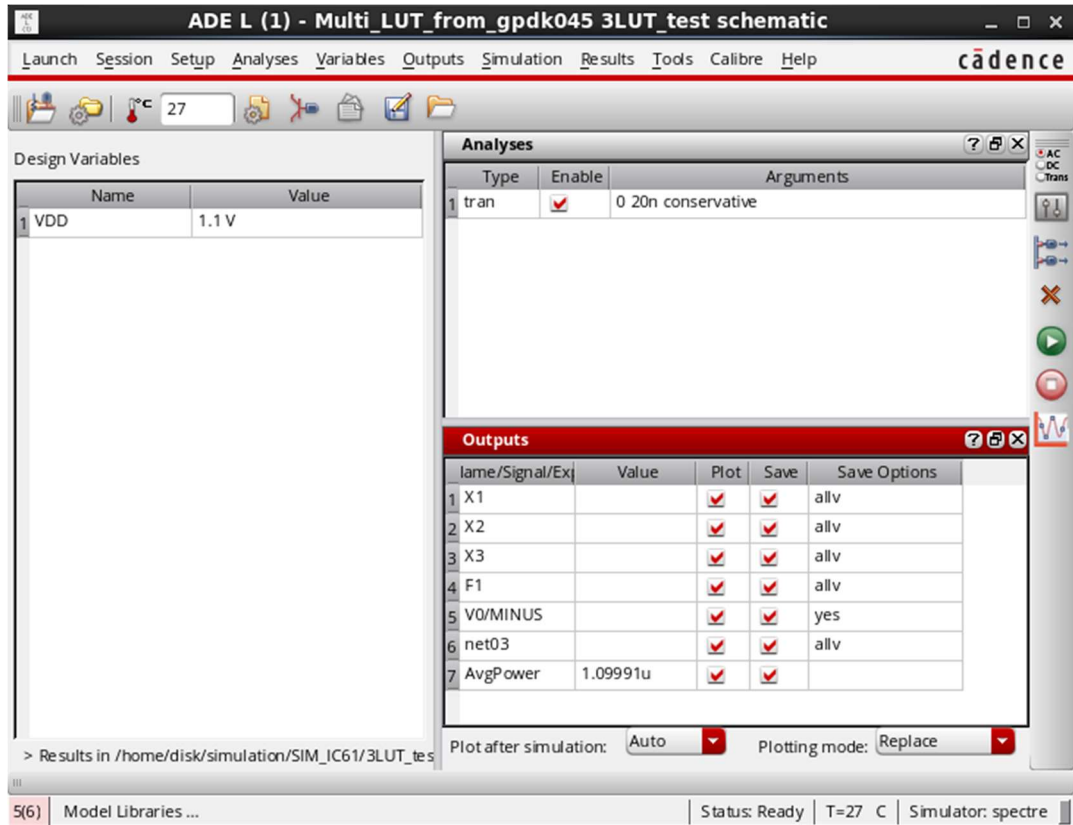


Рисунок ПБ 8в – Настройки моделирования в симуляторе Spectre

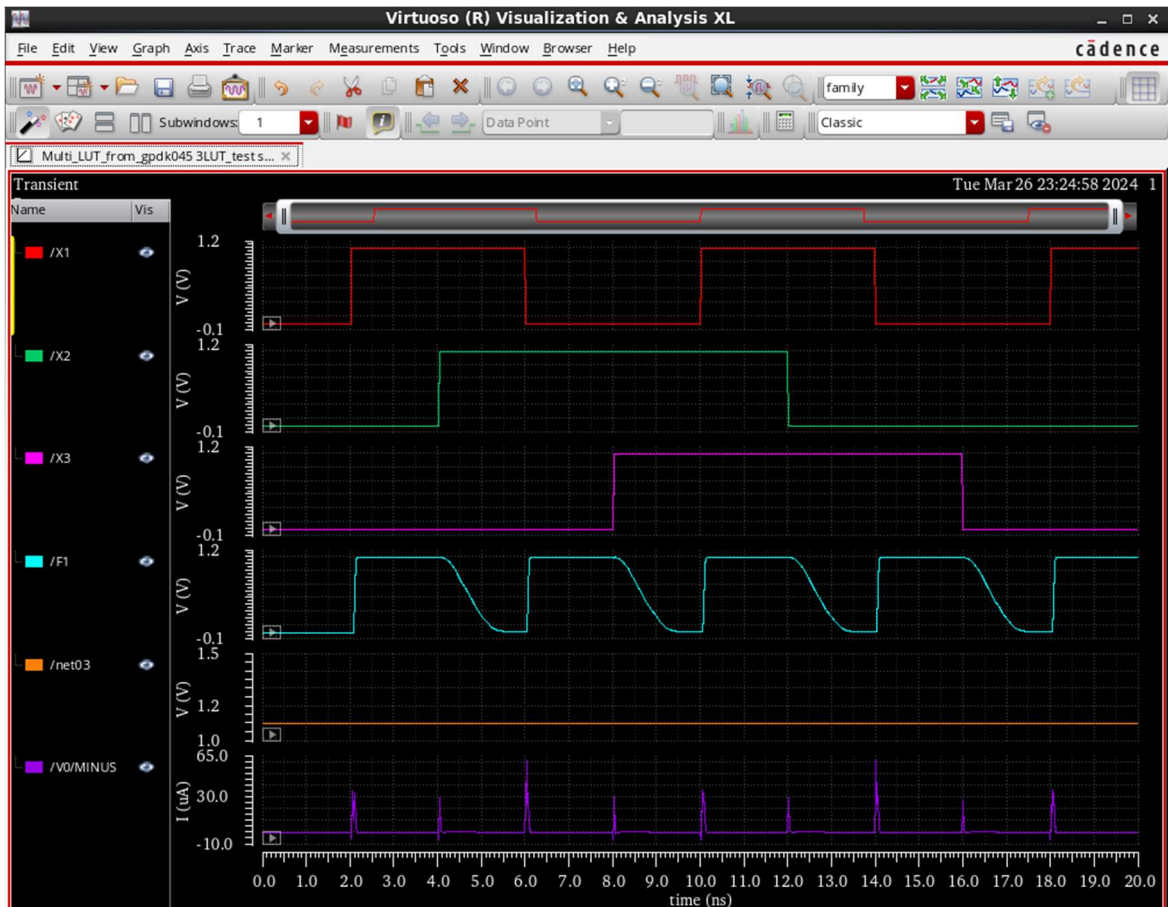


Рисунок ПБ 8г – Результат моделирования 3-LUT, реализующего одну функцию

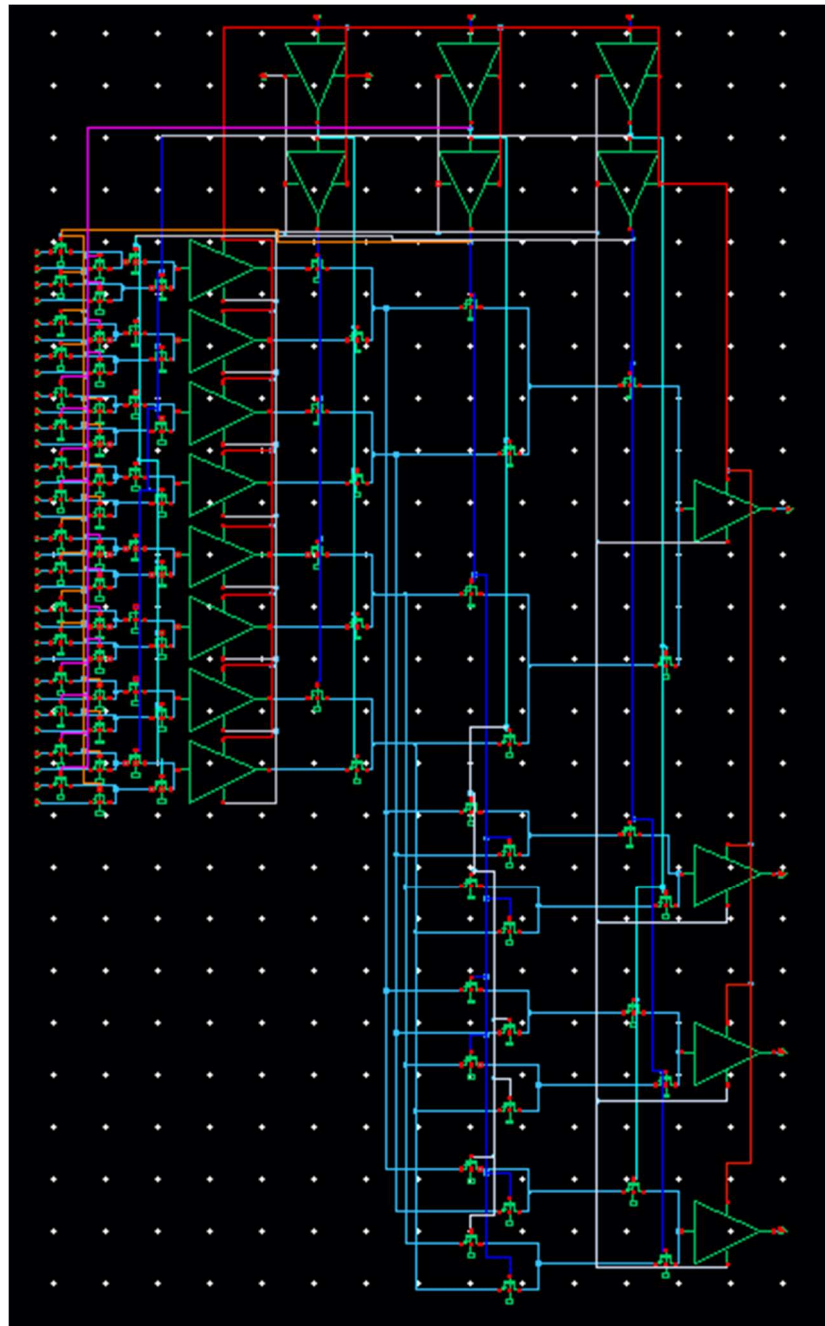


Рисунок ПБ 9а – Схема 3–LUT, реализующего четыре функции в системе Cadence Virtuoso с использованием grdk по технологии 45 нм.

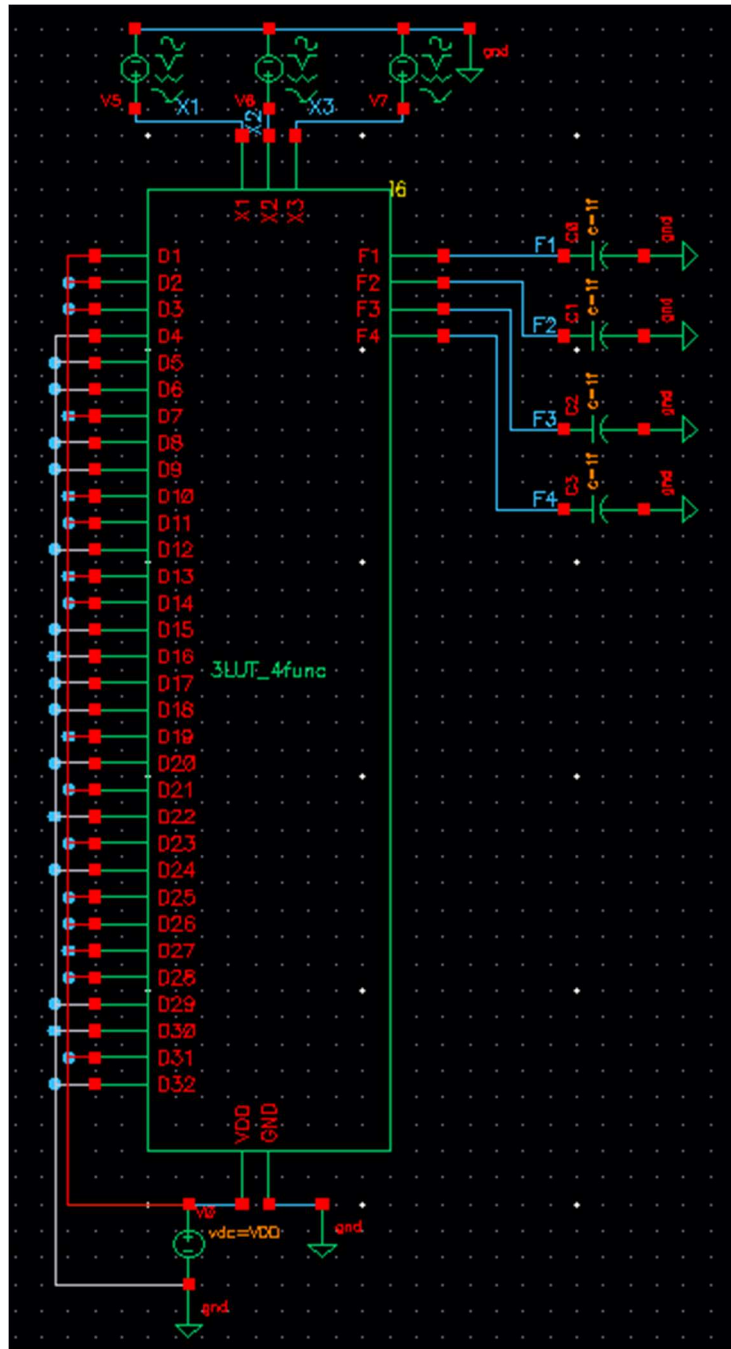


Рисунок ПБ 96 – Test bench четырехфункционального 3–LUT, где X1–X3 – сигналы переменных, D1–D32 – статическая память настройки, F1–F4 – выходы логических функций.

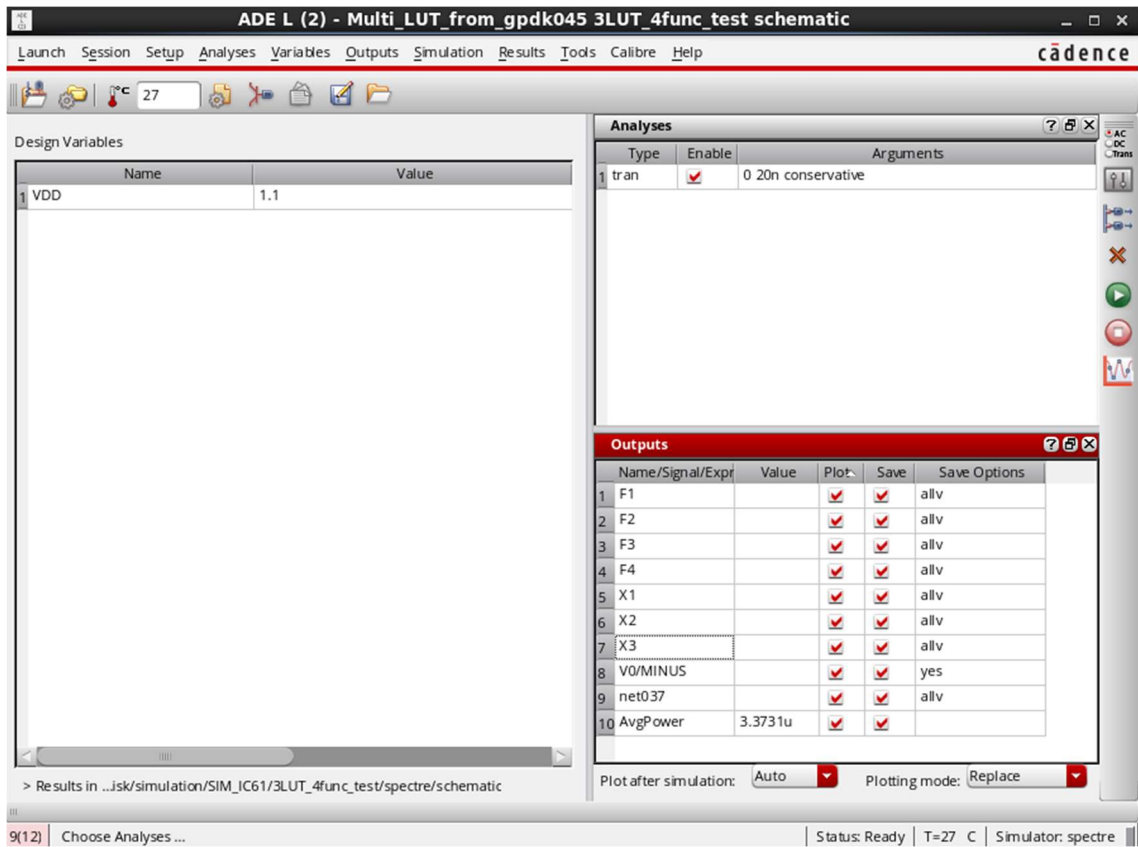


Рисунок ПБ 9в. – Настройки моделирования в симуляторе Spectre



Рисунок ПБ 9г. – Результат моделирования 3-LUT, реализующего четыре функции одновременно

Приложение В

Акты внедрения результатов диссертационного исследования

УТВЕРЖДАЮ



Проректор по образовательной
деятельности ПНИПУ

И.Ю. Черникова

«25» 09 2024 г.

АКТ

внедрения в учебный процесс кафедры «Автоматика и телемеханика» ФГАОУ ВО ПНИПУ результатов диссертационной работы Советова Станислава Игоревича на тему «Логические элементы ПЛИС FPGA, реализующие несколько функций одновременно»

Комиссия в составе:

Председатель: Южаков А.А. – д.т.н., проф., зав. кафедрой «Автоматика и телемеханика»,

Члены комиссии:

Фрейман В.И. – д.т.н., проф. кафедры «Автоматика и телемеханика»,

Гончаровский О.В. – к.т.н., доц. кафедры «Автоматика и телемеханика»

составили настоящий акт о том, что результаты диссертационного исследования Советова Станислава Игоревича внедрены в учебный процесс кафедры «Автоматика и телемеханика» ФГАОУ ВО «Пермский национальный исследовательский политехнический университет» в рамках практических занятий профильных дисциплин «Дискретная математика и математическая логика», «Цифровая схемотехника» для бакалавриата направлений подготовки 11.03.02 «Инфокоммуникационные технологии и системы связи», 15.03.06 «Мехатроника и робототехника», 27.03.04 «Управление в технических системах».

Разработанные результаты диссертационной работы:

1. Модели многофункционального логического элемента LUT, реализующих вычисление нескольких функций одновременно, и логического элемента LUT, реализующего одновременно вычисление логической функции и дешифрацию набора переменных;

2. Методы синтеза многофункционального логического элемента LUT, реализующего вычисление нескольких функций одновременно и логического элемента LUT, реализующего одновременно вычисление логической функции и дешифрацию набора переменных;

3. Алгоритмы подключения дополнительных транзисторов в многофункциональном логическом элементе LUT, который реализует 2^x логических функций при одной заданной конфигурации.

4. Оценки сложности логических элементов LUT, реализующих несколько функций одновременно, а также реализация логической функции совместно с дешифрацией входных переменных;

внедрены в практических и лабораторных занятиях.

Эффект от внедрения результатов диссертационной работы заключается в повышении уровня знаний, умений и владений в соответствии со стандартами ФГОС ВО по указанным выше направлениям подготовки.


Председатель комиссии:

д.т.н., проф. зав. кафедрой АТ

 / Южаков А.А. /

Члены комиссии:

д.т.н., проф. кафедры АТ

 / Фрейман В.И. /

к.т.н., доц. кафедры АТ

 / Гончаровский О.В. /

« 23 » 09 2024 г.



УТВЕРЖДАЮ

Директор ФИЦ ИУ РАН


 М.А. Посыпкин

«27» 09 2024 г.

Акт о внедрении

результатов диссертационных исследований

Советова Станислава Игоревича

Настоящим актом подтверждается, что в научно-исследовательской работе Федерального исследовательского центра «Информатика и управление» Российской академии наук (ФИЦ ИУ РАН) по теме государственного задания «Информационные, управляющие и телекоммуникационные системы 2024-2028», шифр FFNG-2024-0010 использовались следующие научные результаты, полученные в кандидатской диссертации аспиранта кафедры «Автоматика и телемеханика» Пермского национального исследовательского политехнического университета Советова Станислава Игоревича, раздел 5.3 отчета «Разработка логики самосинхронных ПЛИС», № госрегистрации 124040200035-3:

1. Модели многофункционального логического элемента LUT, реализующих вычисление нескольких функций одновременно, и логического элемента LUT, реализующего одновременно вычисление логической функции и дешифрацию набора переменных;
2. Оценки сложности логических элементов LUT, реализующих несколько функций одновременно, а также реализация логической функции совместно с дешифрацией входных переменных.

Кроме этого, в рамках отчета за 2024 год по гранту РФФИ № 22-19-00237 «Разработка и апробация элементной базы сбоеустойчивых робототехнических систем на базе самосинхронной парадигмы», в разделе «Исследование возможностей современных отечественных и зарубежных программируемых логических интегральных схем для реализации на них самосинхронных схем», № госрегистрации 122052300078-4 использовались следующие результаты Советова Станислава Игоревича:

1. Методы синтеза многофункционального логического элемента LUT, реализующего вычисление нескольких функций одновременно и логического элемента LUT, реализующего одновременно вычисление логической функции и дешифрацию набора переменных;
2. Алгоритмы подключения дополнительных транзисторов в многофункциональном логическом элементе LUT, который реализует 2^N логических функций при одной заданной конфигурации.

Разработанные модели, методы, алгоритм и оценки сложности используются при проектировании программируемых логических устройств на основе отечественной элементной базы, при этом обеспечивается снижение аппаратных затрат в количестве транзисторов и площади кристалла на реализацию логических функций более 15%.

Кандидат технических наук,
ведущий научный сотрудник,
руководитель отдела «Архитектура и
схемотехника инновационных
вычислительных систем»



Ю.А. Степачков

«27» сентября 2024 г.