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

На правах рукописи

Васенин Иван Андреевич

ЭЛЕМЕНТЫ ПЛИС С ИСПОЛЬЗОВАНИЕМ КОМБИНИРОВАННОГО КОДИРОВАНИЯ

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 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы исследования. Программируемые логические интегральные схемы (ПЛИС или программируемые пользователем вентильные матрицы – ППВМ, Field-Programmable Gate Array – FPGA или сложные программируемые логические устройства CPLD – Complex Programmable Logic Device) широко используются во многих областях применения. За последние четыре года объем производства ПЛИС увеличился более чем на 50%. На период 2025-2029 гг. ожидается среднегодовой темп роста глобального рынка высокопроизводительных ПЛИС более 11%. Актуальность практического применения подтверждается также ростом использования ПЛИС в областях критического применения – медицина, авионика, космическая техника, аппаратура управления АЭС, ГЭС и ТЭС, вооружение и военная техника, и др. В настоящее время ПЛИС (FPGA) передовых производителей содержат уже десятки миллионов логических элементов (коммутаторов межсоединений в глобальных и локальных матрицах связей на несколько порядков больше) и десятки миллиардов транзисторов. В ПЛИС используются логические элементы, называемые «таблицами просмотра» или «таблицами истинности» LUT (Look Up Table), в которых используется бинарное (позиционное) кодирование наборов переменных, они же могут реализовывать коммутаторы межсоединений, тогда бинарным кодом кодируется требуемая связь. Коммутаторы межсоединений могут строится и как унитарные мультиплексоры из передающих транзисторов, использующие унитарное кодирование связей (активен только один бит), что экспоненциально увеличивает объем памяти настройки. Но в отличие от LUT на n переменных, в которых путь сигнала содержит не менее чем n транзисторов, в таких коммутаторах этот путь включает, как правило, всего один транзистор. Однако, ПЛИС, как универсальное устройство, проигрывает заказным микросхемам (ASIC – application-specific integrated circuit) в быстродействии, выигрывая в стоимости. Количество переменных, реализуемых логических функций в одном адаптивном элементе, может составлять до восьми переменных.

При увеличении числа переменных в одном элементе резко снижается быстродействие. Таким образом, актуальным является проведение исследований по снижению временной задержки элементов ПЛИС при ограничении объема памяти настройки или числа связей переменных.

Степень разработанности темы исследования. Вопросы синтеза и анализа логических элементов ПЛИС [92] ранее были предложены и исследованы в работах отечественных авторов: А.В. Строгонова [2–3, 14–16], С.А. Цыбина, А.Н. Денисова [18], Г.П. Аксеновой, Д.Е. Иванова [96], Ю.А. Скобцова [21], А.А. Баркалова [47]. Зарубежные ученые в области ПЛИС: В.И. Хаханов [22], А.В. Дрозд [23, 96], В.С. Харченко [24], Е. Зорян [25], Н. Мехта, Раджитх К. Шрикант, Мандар Д. Чафекар, Цзяо Вей, Ли Чжун [9], Мартин Шмитц, Лаурент Фандо, Джон Оустерхаут, Питер Юнг и др [30-32, 91. 93]. Новые логические элементы ПЛИС предложены и исследованы в работах научной группы кафедры АТ ПНИПУ: С.Ф. Тюрина [33, 34, 42, 79, 87, 88, 89, 90], А.В. Грекова [35, 87], О.А. Громова [36, 87], А.Ю. Городилова [37, 89], А.Н. Каменских [38], Р.В. Вихорева [38–42, 44], А.Ю. Скорняковой [43, 44], С.И. Советова [45–46, 79] и др. Известные варианты комбинирования унитарного кодирования путем разбиения на группы (А.В. Строгонов) не используют возможность комбинирования с позиционным кодом. Кроме того в известных работах не используются варианты унитарного кода для вычисления логических функций с использованием разложения Шеннона. Противоречие в науке заключается в следующем: созданы предпосылки комбинирования двух подходов в реализации логических функций и коммутации связей, но они еще не объединены единым методом. Поэтому целесообразно исследовать различные варианты комбинирования позиционного и унитарного кода для реализации элементов, как для вычисления логических функций, так и для коммутации связей.

Объектом исследования являются ПЛИС с элементами LUT, использующими позиционное (бинарное) кодирование и с элементами коммутирования межсоединений, использующими унитарное кодирование.

Предметом исследования является научно-методический аппарат синтеза элементов ПЛИС, использующих комбинированное кодирование и обладающих повышенным быстродействием.

Цель диссертационного исследования заключается в решении научной задачи разработки модели и метода синтеза элементов ПЛИС, использующих комбинированное кодирование.

Для достижения поставленной цели в диссертационной работе поставлены и решены следующие **задачи исследования:**

1. Аналитический обзор, анализ, исследование и сравнение существующих моделей и методов синтеза базовых элементов ПЛИС для вычисления логических функций и коммутации сигналов в матрицах межсоединений.
2. Разработка математической модели элемента с комбинированным и универсальным кодированием (Паспорт специальности: п.2).
3. Разработка метода синтеза элементов с комбинированным и универсальным кодированием (Паспорт специальности: п.2).
4. Разработка алгоритма синтеза элемента с заданным вариантом кодирования (Паспорт специальности: п.2).
5. Разработка схем электрических функциональных и принципиальных предлагаемых элементов.
6. Схемотехническое моделирование разработанных элементов с комбинированным кодированием.
7. Топологическое моделирование разработанных элементов с комбинированным кодированием.

8. Получение оценок сложности и эффективности различных вариантов комбинированного кодирования, позволяющих осуществлять выбор оптимального варианта (вариантов).

9. Апробация разработанной модели, метода, алгоритма и оценок сложности, внедрение в ФИЦ ИУ РАН.

Положения, выносимые на защиту и обладающие научной новизной:

1. Разработана новая **математическая модель** элемента, отличающаяся тем, что описывает комбинированные варианты, использующие как позиционное, так и унитарное кодирование в одном устройстве, а также универсальный элемент с настраиваемым типом кодирования (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 специальности 2.3.2).

2. **Создан метод** синтеза элементов с комбинированным и универсальным кодированием, отличающийся тем, что позволяет создавать новые устройства с лучшими характеристиками по быстродействию при допустимом увеличении сложности (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 специальности 2.3.2).

3. Получены математические выражения **оценок сложности** новых элементов с комбинированным кодированием, позволяющие выбирать требуемый вариант комбинирования (п. 6 «Разработка научных подходов и методов, архитектурных и структурных решений, обеспечивающих эффективную техническую реализацию аппаратно-программных систем и комплексов за счет оптимизации применяемой электронной компонентной

базы, элементов вычислительных систем и встраиваемого программного обеспечения» паспорта специальности 2.3.2).

4. Разработан **алгоритм** синтеза элемента с комбинированным кодированием, отличающийся тем, что обеспечивает по заданным параметрам требуемые соединения, используя предложенный элемент с конфигурируемым кодированием (п. 6 «Разработка научных подходов и методов, архитектурных и структурных решений, обеспечивающих эффективную техническую реализацию аппаратно-программных систем и комплексов за счет оптимизации применяемой электронной компонентной базы, элементов вычислительных систем и встраиваемого программного обеспечения» паспорта специальности 2.3.2).

Теоретическая значимость результатов диссертационной работы состоит в развитии научно-методического аппарата синтеза элементов ПЛИС путем разработки модели, метода, алгоритма и оценок сложности элемента с комбинированным кодированием переменных.

Практическая значимость результатов диссертационной работы состоит в разработке нового, запатентованного элемента ПЛИС с комбинированным кодированием переменных, обладающего лучшими характеристиками, чем существующие [79, 100]. Разработаны схемы электрические функциональные и принципиальные, а также топологии новых элементов и программа их синтеза. Временная задержка снижается более, чем на 15%, а также в ряде случаев снижаются и аппаратурные затраты более, чем на 20% в зависимости от разрядности унитарной и позиционной части.

Результаты работы внедрены в учебный процесс кафедры «Автоматика и телемеханика» ФГАОУ ВО «Пермский национальный исследовательский политехнический университета» 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». Также результаты внедрены в ООО «Динамика роста» (г. Пермь), которое приобрело лицензию у Пермского Национального Исследовательского Политехнического Университета (ПНИПУ) на право использования изобретения «Программируемое логическое устройство» (патент РФ № 2 811 404, приоритет от 02.08.2023 г., дата государственной регистрации 11.01. 2024 г). Кроме того, результаты исследования внедрены в научно-исследовательской работе отдела 52 Федерального исследовательского центра «Информатика и управление» Российской академии наук (ФИЦ ИУ РАН) «Архитектура и схемотехника инновационных вычислительных систем» по теме государственного задания «Информационные, управляющие и телекоммуникационные системы 2024-2028».

Методология и методы исследования. В диссертационной работе используются методы, модели, алгоритма и программное обеспечение, позволяющие произвести схемотехническое и топологическое моделирования для проведения анализа и синтеза схем, а также расчеты показателей сложности. Применяемые и используемые методы, средства и инструменты имеют научное обоснование. Основу этих обоснований составляют положения дискретной математики, математической логики, комбинаторики, принципы МОП-схемотехники, теории булевых функций и автоматов.

Достоверность и обоснованность результатов, полученных в ходе исследования, не противоречат теоретическим положениям и выводам, ранее опубликованным результатам отечественных и зарубежных ученых. Подтверждение результатов было получено в двух различных системах моделирования (Multisim, Microwind), апробированием и внедрением методов, моделей и алгоритма подключения, предложенных в диссертации. Расчеты проводились с помощью системы компьютерной алгебры Mathcad. Основные теоретические и практические результаты работы докладывались на научно-технических конференциях: «Международная конференция молодых

исследователей в области электротехники и электроники 2023 ElConRus», «ElCon» 2025, «Инновационные технологии: теория, инструменты, практика» (InnoTech-2022, 2023, 2024), «Научно-техническая конференция 2023 IEEE 24th International Conference of Young Professionals in Electron Devices and Materials (EDM) (республика Алтай, НГТУ)», Всероссийская научно-техническая конференция «Автоматизированные системы управления и информационные технологии» (АСУИТ-2024, 2023, г. Пермь), Школа молодых учёных в рамках Российского форума «Микроэлектроника 2024».

Публикации. Основные результаты диссертационной работы опубликованы в 11 печатных работах, из них 4 публикации в ведущих рецензируемых научных изданиях, 2 публикации в изданиях, индексированных в международной базе цитирования Scopus, 2 патента на изобретения.

Объем и структура работы. Диссертация состоит из введения, пяти глав, заключения, списка литературы из 103 наименований и трех приложений. Полный объем диссертации составляет 220 страниц, из которых 194 страниц занимает основной текст диссертации, включающий 110 рисунков и 10 таблиц.

ГЛАВА 1. АКТУАЛЬНОСТЬ И АНАЛИЗ ОБЪЕКТА И ПРЕДМЕТА ИССЛЕДОВАНИЯ. ПОСТАНОВКА ЗАДАЧИ ИССЛЕДОВАНИЯ

1.1 Обзор и анализ объекта исследования – элементов ПЛИС. Противоречие в практике

1.1.1 Анализ современных ПЛИС

История создания и развития ПЛИС включает около 40 лет, хотя сама программируемая логика (универсальные логические модули), начиная с программируемой памяти, программируемых логических матриц (ПЛМ, PLA, ПМЛ, PAL) – насчитывает около 60 лет [4–8]. Доступ к последней информации по ПЛИС в настоящее время сопряжен с определенными известными трудностями. Рынок современных ПЛИС, представлен, например, в источнике [17]. В плане импортозамещения электронной компонентной базы в современных реалиях крайне актуальны исследования по совершенствованию отечественных ПЛИС [14]. Примером могут быть впечатляющие темпы совершенствования китайских производителей ПЛИС [15, 16]. Прогноз глобального рынка ПЛИС до 2031 г. по данным [17] показан на рисунке 1.1.

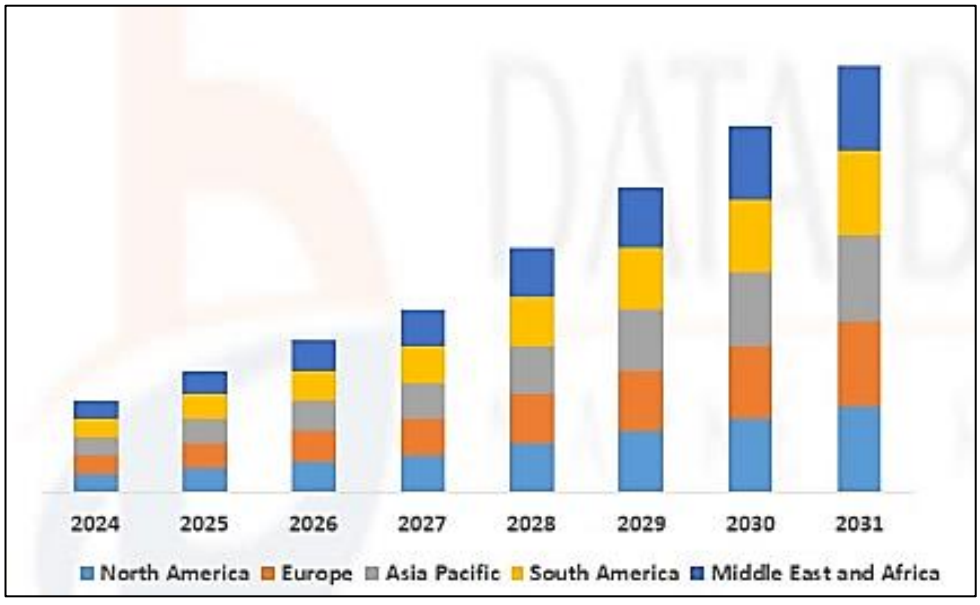


Рисунок 1.1 – Прогноз глобального рынка ПЛИС (Global Low-End Field-Programmable Gate Array (FPGA) Market – Industry Trends and Forecast to 2031)

Показатель CAGR (Compound Annual Growth Rate) – совокупный среднегодовой темп роста на этот период оценивается более 11%, размер рынка на прогнозируемый период – более 14 млрд. долларов. В настоящее время на рынке электронной техники ПЛИС делят на три сегмента по производительности: высококачественные ПЛИС (высокопроизводительные, High End FPGA), ПЛИС среднего класса (Mid-End FPGA,ПЛИС средней производительности), ПЛИС для бытовых устройств, начального уровня (низкопроизводительные ПЛИС, Low-End FPGA). Высокопроизводительные ПЛИС используются, например, в центрах обработки данных.

По особенностям хранения конфигурационных настроек выделяются три основных варианта.

Первый использует оперативную память конфигураций (ОЗУ, SRAM), при этом конфигурация (настройка функций и связей) теряется при выключении источника питания [86], а при его включении необходимо определенное время для загрузки конфигурации последовательным кодом, например, из внешнего компьютера или внешней энергонезависимой (non-volatile) памяти. Большинство ПЛИС используют ОЗУ (SRAM).

Второй вариант предполагает использование постоянной перепрограммируемой памяти (ППЗУ, EEPROM), например вида Flash, что требует совмещения нескольких технологий и, соответственно, дороже. Кроме того, число перепрограммирований ограничено (например, около 10 000 раз), в отличие от технологии (ОЗУ, SRAM), зато ПЛИС сразу готова к работе после включения электропитания.

Вариант Antifuse («Антипредохранитель», fuse – «предохранитель») характеризуется однократным и относительно медленным программированием, подобно так называемым полузаказным базовым матричным кристаллам (БМК). Если при программировании fuse «плавкая вставка» пережигается либо не пережигается, то Antifuse либо «наплавляется», либо не наплавляется [48].

Этот вариант тоже требует совмещение разных технологий, но зато такие ПЛИС меньше подвержены радиационным воздействиям и обладают большим быстродействием.

Другие особенности современных ПЛИС отражены, например, в публикациях [49–53]. Количество логических элементов, которые объединяются в целые «фабрики логики», достигает десятков миллионов в High End FPGA, а общее количество транзисторов – десятки миллиардов. Количество переменных в одном элементе в ранних ПЛИС было всего 3-4. В настоящее время используются так называемые адаптивные логические модули, в которых можно создать элемент на 5,6 и 7 переменных. В ряде случаев конфигурируется элемент на 8 входов, но реализуются лишь некоторые функции.

1.1.2 Анализ логического элемента FPGA и элемента – коммутатора межсоединений

Реализация логических функций в ПЛИС типа FPGA основана на использовании мультиплексора в режиме коммутации констант настройки [1–3], представляющих собой значения из таблицы истинности соответствующей функции. Такое устройство обычно называют LUT (Look Up Table). LUT представляет собой бинарное дерево передающих транзисторов [2, 3], но может содержать и $2n$ ветвей по n транзисторов [29]. Классические логические элементы LUT реализуют (вычисляют) логические функции одной, двух, трёх и четырёх переменных. Это связано с ограничениями на количество последовательно соединенных транзисторов (не более четырех) [54]. Имеются более сложные элементы, называемые адаптивными логическими модулями (АЛМ), которые реализуют любые функции до семи аргументов и некоторые функции восьми аргументов путем конфигурируемого соединения классических LUT [49 –51, 85]. Несмотря на впечатляющие достижения в области ПЛИС, они остаются универсальными устройствами со всеми

соответствующими преимуществами и недостатками, которые требуют особого внимания в области так называемых критических приложений [10–13].

Логический элемент LUT [2, 49] на четыре переменные показан на рисунке 1.2.

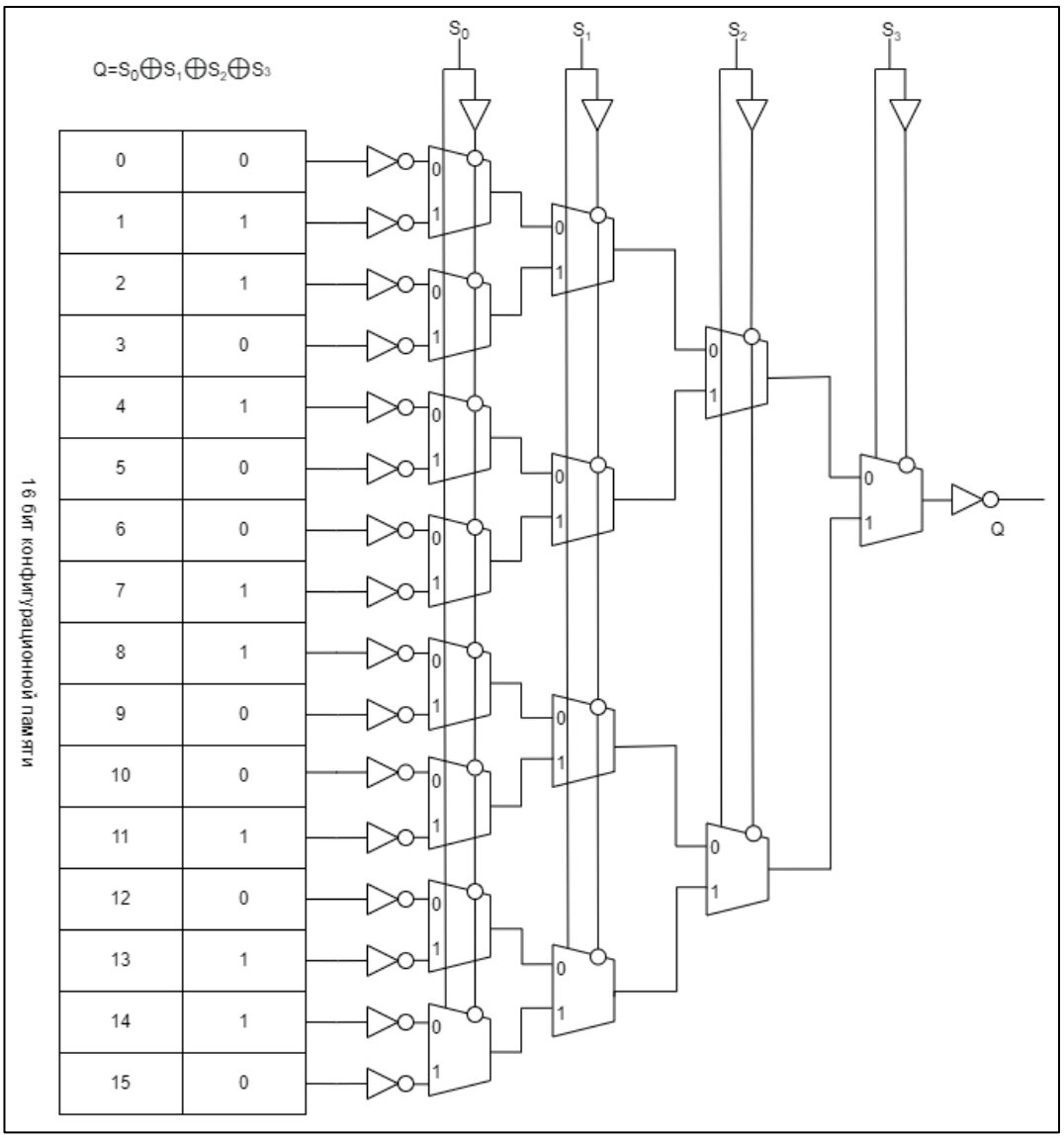


Рисунок 1.2 – Логический элемент LUT на четыре переменные *S*.0, *S*.1, *S*.2, *S*.3; биты конфигурационной памяти – *d*.0-*d*.15

Логические элементы соединяются между собой и с контактами ПЛИС с помощью коммутаторов в матрицах межсоединений. На рисунке 1.3 показан пример реализации логической функции трёх внешних переменных $f = x_1 x_2 \vee \bar{x}_2 x_3$, использующей три LUT на две переменные.

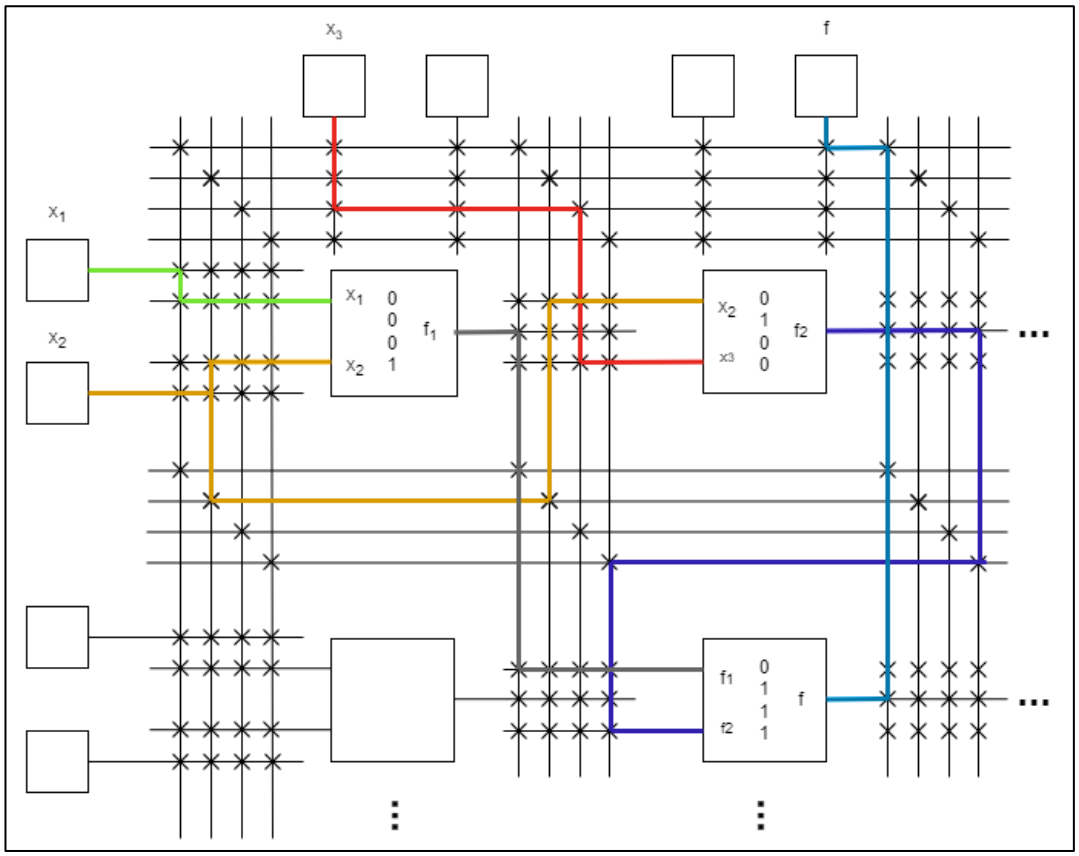


Рисунок 1.3 – Реализация логической функции трёх внешних переменных

Коммутаторы сигналов показаны синими крестиками на требуемом пересечении линий связи в матрицах межсоединений (черные – неактивные пересечения). Первый LUT реализует конъюнкцию двух переменных x_1x_2 (таблица истинности 0001). В развернутой форме это выглядит так – таблица 1.1:

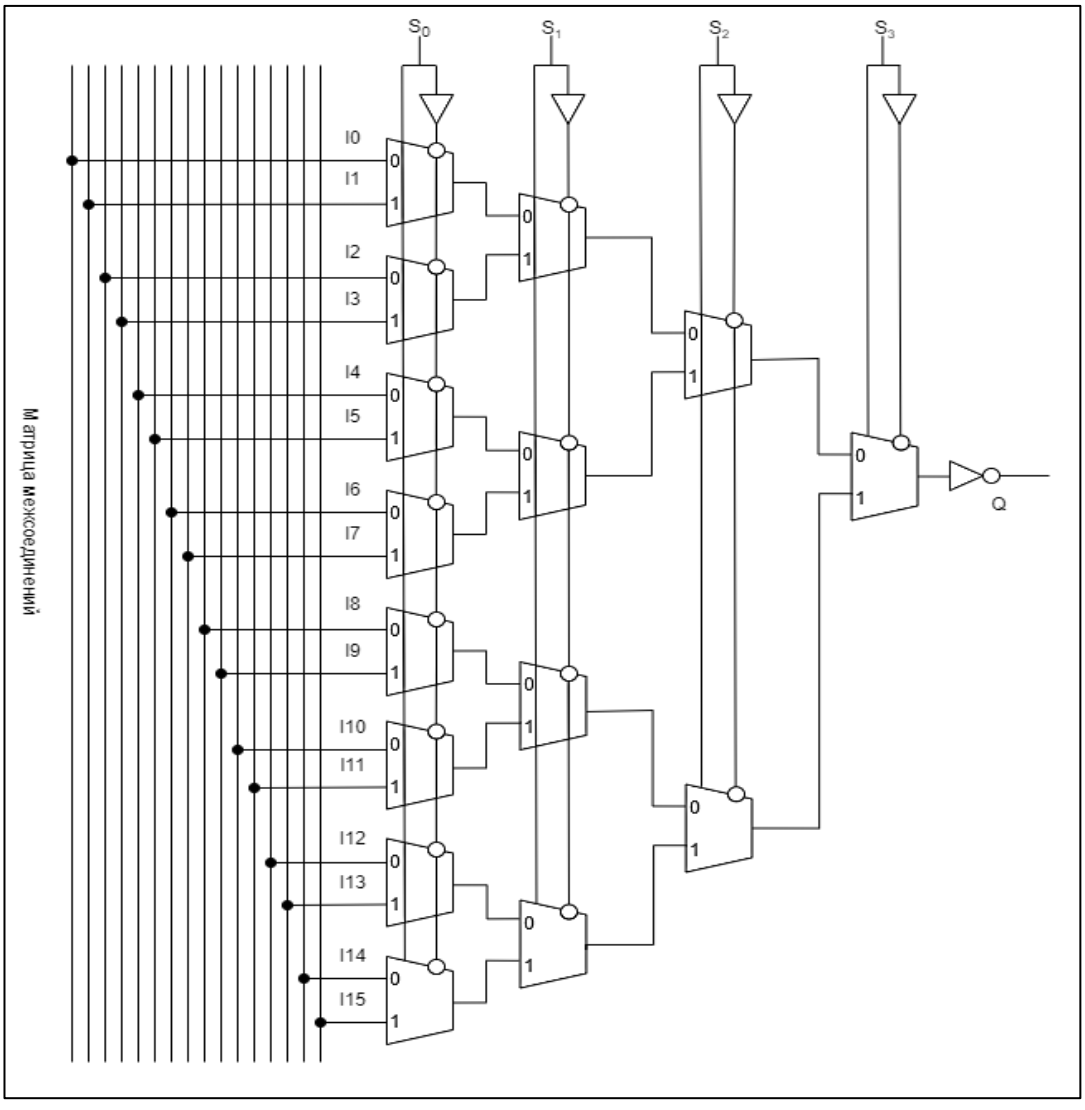
Таблица 1.1 – Пример таблицы истинности – конъюнкция

Старшая переменная x_1	Младшая переменная x_2	Функция f_1
0	0	0
0	1	0
1	0	0
1	1	1

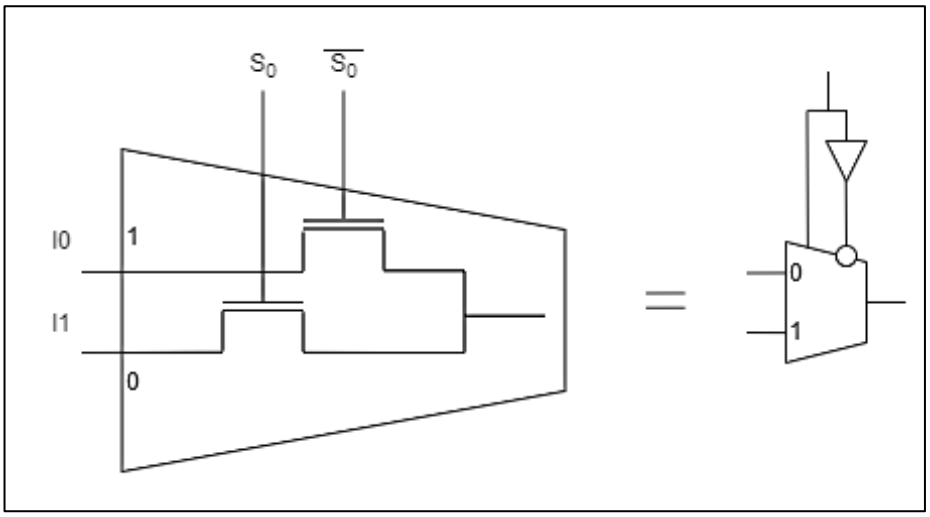
Второй LUT реализует функцию запрета второй переменной $\bar{x}_2x_3$ (таблица истинности 0100). Третий LUT реализует функцию дизъюнкции результатов вычислений первых двух LUT (таблица истинности 0111). Выход третьего LUT подключен к контакту *f* ПЛИС.

Анализ показывает, что производители для минимизации площади кристалла выбирают оптимальную разрядность LUT от 3 до 4 переменных. Для достижения максимального быстродействия целесообразен LUT от 4 до 6 переменных. В ПЛИС с АЛМ может быть конфигурирован LUT до 7 переменных. LUT на 8 переменных реализует не все возможные логические функции.

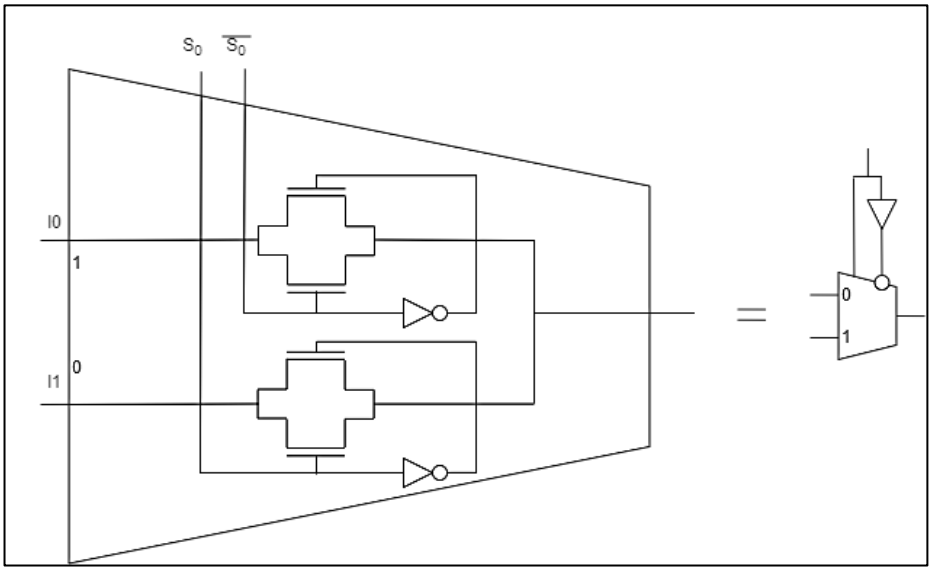
Строго говоря, LUT – это такой мультиплексор с настройкой (рисунок 1.2), при отсутствии настройки, это не будет являться LUT. Для простоты мы будем использовать обозначение LUT и для коммутаторов, то есть в случае подключения вместо настройки – связей, а сама настройка теперь – по входам переменных (рисунок 1.4):



а)



б)



в)

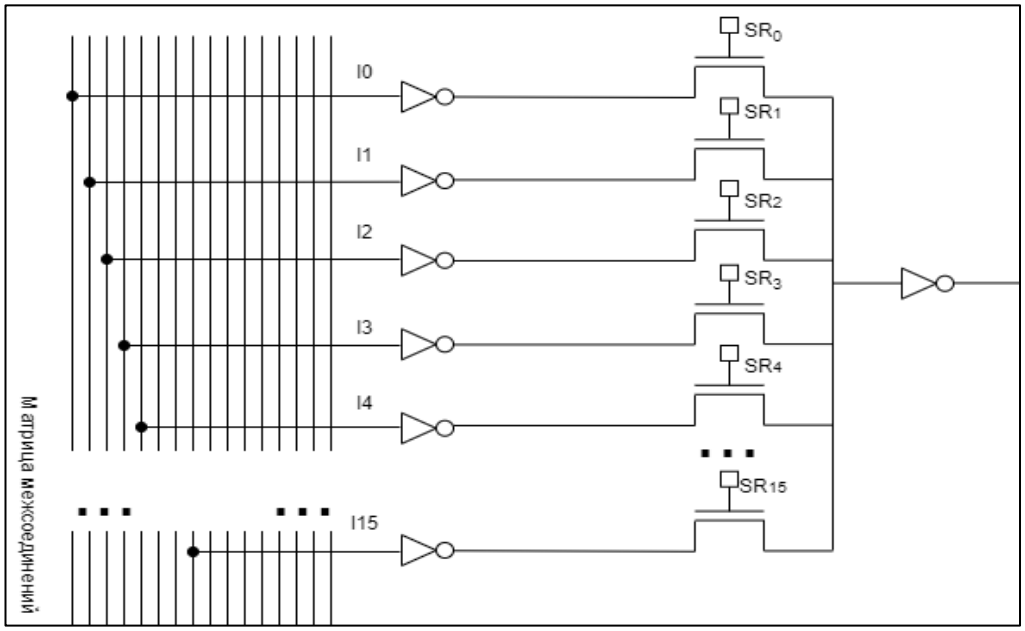
Рисунок 1.4 – а) Коммутатор сигналов с локальных межсоединений на вход конфигурируемого логического блока с использованием мультиплексоров (S0–S3 — конфигурационные биты памяти); б) мультиплексор на *n*-МОПТ ключах; в) мультиплексор на КМОП-ключах

В этом случае путь сигнала проходит через *n* транзисторов. Настройка бинарным позиционным кодом выбирает одну из 2^n связей.

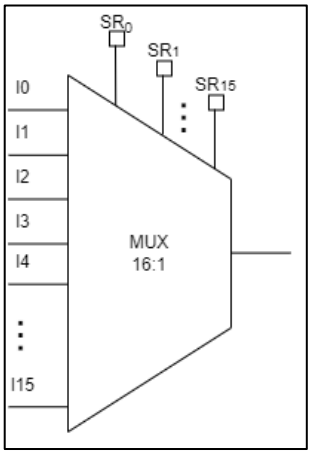
В случае как на рисунке 1.4 – четыре транзистора, что является критичным в силу ограничений Мида-Конвей [54, 94] на число последовательно соединенных транзисторов (это обусловлено падением напряжения на транзисторах на величину порогового напряжения). Обычно

допускают три подряд соединенных транзистора, после которых устанавливается восстановитель уровня сигнала (буфер).

Использование унитарного кода для коммутации сигналов (например, в сети многоканальный соединений MultiTrack с использованием технологии Direct Drive [2]) показано на рисунке 1.5.



а)



б)

Рисунок 1.5 – Коммутатор сигналов 16 в 1 (мультиплексор 16-1) с матрицы межсоединений на вход конфигурируемого логического блока (КЛБ) с одним транзистором в цепочке: а) с использованием *n*-МОПТ ключей (передающих n-МОП транзисторов); б) условное обозначение; SR – ячейки конфигурационной памяти, объём памяти – 16 бит

В этом случае путь сигнала проходит через один транзистор. Всего транзисторов с учетом инверторов 50.

Известный комбинированный вариант кодирования («унитарный+ унитарный») приведен в работе [2] и показан на рисунке 1.6.

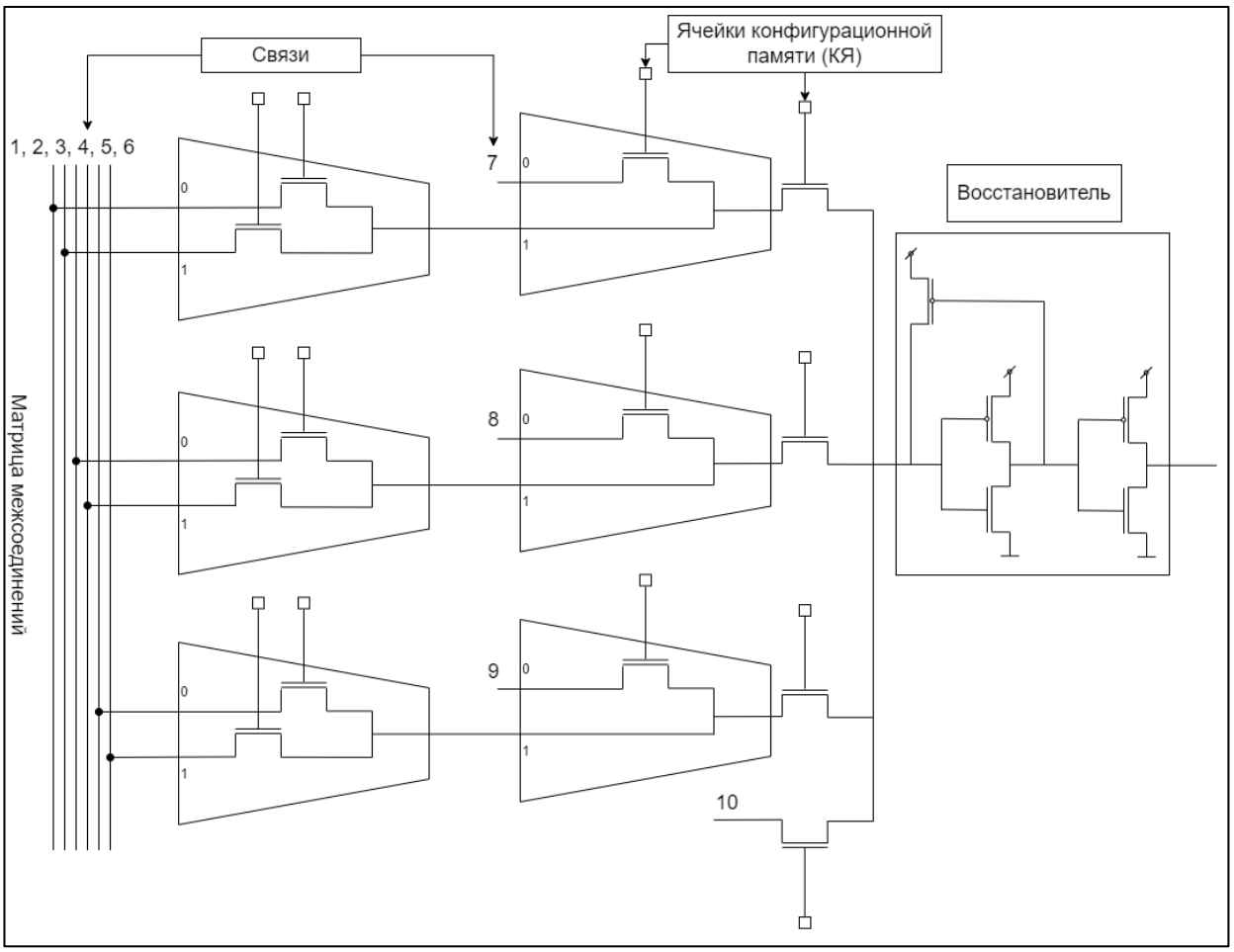


Рисунок 1.6 – Существующий комбинированный вариант коммутатора

Унитарный код используется также в САПР Quartus для ПЛИС фирм Intel (Altera). Кодирование конечного автомата, построенного по графу переходов (State Machine File), по умолчанию (Auto) принимается унитарным (One-Hot) [55].

1.2 Анализ предмета исследования – научно-методического аппарата синтеза – элементов ПЛИС. Противоречие в науке

1.2.1 Анализ публикаций по научно-методическому аппарату синтеза элементов ПЛИС

На ресурсе IEEE ieeexplore (https://ieeexplore.ieee.org) за 1989-2025 гг. поиск по запросу “FPGA” отображает более 47000 материалов научных конференций, более 6500 статей в журналах, 106 книг, 10 стандартов.

За период 2020-2025 – более 10500 материалов научных конференций, более 3000 статей в журналах, 53 книги, 2 стандарта (рисунок 1.7).

Поиск по запросу “LUT FPGA” за период 2020-2025 – около 600 материалов научных конференций, более 180 статей в журналах, 3 книги.

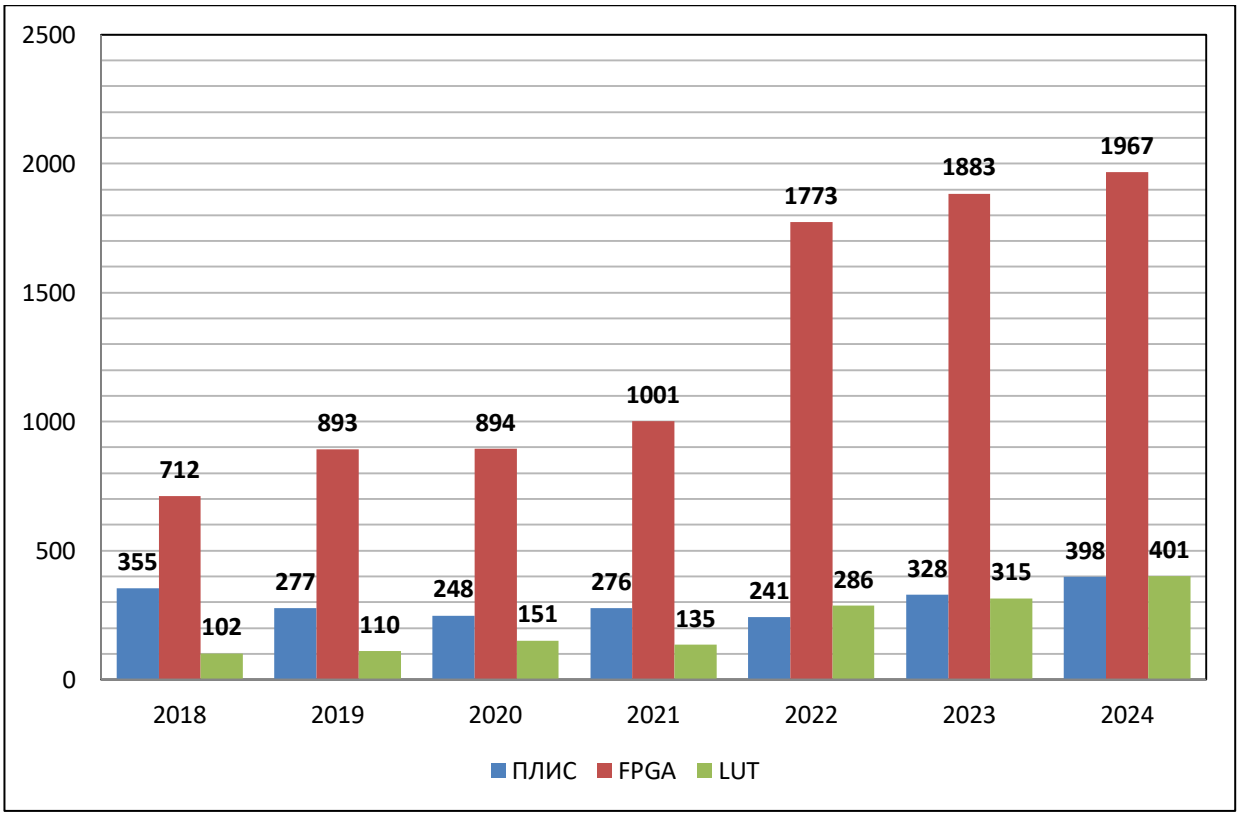


Рисунок 1.7 – График роста числа публикаций

На ресурсе Researchgate (https://www.researchgate.net) по запросу «FPGA» отображается более 1000 публикаций. По запросу «LUT FPGA» за 2024 г.

отображается более 480 публикаций. Запрос «LUT FPGA encoding» за 2025 год выдает 68 публикаций на 11.01.25.

На сайте https://www.elibrary.ru по запросу «FPGA» выдается более 18000 публикаций, из них более 10000 за 2020-2025 гг. По запросу «LUT FPGA» выдаются более 400 публикаций. Запрос «LUT FPGA encoding» формирует 20 публикаций.

Известный международный издательский ресурс https://link.springer.com по запросу «FPGA» выдает более 12500 статей и примерно столько же материалов конференций, более 300 статей за последние три месяца, три книги, более 100 материалов конференций.

Ресурс https://scholar.google.com по запросу «FPGA» выдает более миллиона результатов, за период с 2021 года – более 50000, с 2025 – более 1200. По запросу «LUT FPGA» с 2025 года – 178 результатов.

Одна из ключевых проблем публикаций последних лет – сокращение разрыва между гибкой и заказной логикой, поскольку по мере усложнения логических элементов остро встает проблема повышения быстродействия, частично решаемая, например, путем специальных способов проектирования (HyperFlex, Hyper-Registers, Hyper-Retiming и др.), в том числе, снижающих задержки в цепях обратных связей через матрицы коммутаций [2].

В частности, используется разложение Шеннона (Shannon decomposition or Boolean factorization) [49, 56] для создания, например, двух вариантов логической функции для двух значений состояния триггера, которое коммутирует выходы логических элементов, реализующих эти два варианта функций.

При этом используется «рядом стоящий» LUT на одну переменную, а переменная состояния не проходит матрицы коммутации, за счет чего достигается повышение быстродействия, хотя и увеличиваются затраты в количестве логических элементов. По сути, используется заранее вычисленные значения логических функций для двух разных значений состояний.

Дальнейшее развитие этого подхода приводит к необходимости использования унитарного кода (в англоязычной литературе unitary code, one-hot, one-cold) для более чем одного триггера и более, чем два варианта логических функций от входных переменных.

Однако, это приводит к увеличению аппаратных затрат, что вызывает необходимость нахождения «золотой» середины между унитарным и позиционным бинарным кодом.

В то же время, несмотря на то, что имеется достаточно много публикаций в области ПЛИС и созданы все предпосылки для объединения этих двух подходов, советующих публикаций обнаружено не было.

1.2.2 Исследование направления создания элементов с комбинированным кодированием переменных или настройки

Метод, использующий комбинированный код, приведенный в публикации [2] (рисунок 1.6) не использует соединения унитарного и позиционного кодов. Кодирование такого дерева – не стандартное, активны два бита: один – выбор ветви дерева, второй выбор листа дерева (рисунок 1.5). Показана коммутация сигнала *I*5, при коде [(0)(0000)] [(1)(0100)] [(0)(0000)] [(0)(0000)]. Первый элемент кода в квадратных скобках – код ветви (1), второй – код листа (0100). То есть имеется двухуровневое кодирование.

Однако, на рисунке 1.8 всего 10 связей, а на рисунке 1.5 – 16. Если предположить, что ветвь к связи №10 аналогична остальным, то получим 12 связей.

Далее в публикации такой вариант сравнивается с унитарным (рисунок 1.4). Но тогда и связей должно быть 16. Это можно сделать, например, так, как показано на рисунке 1.9.

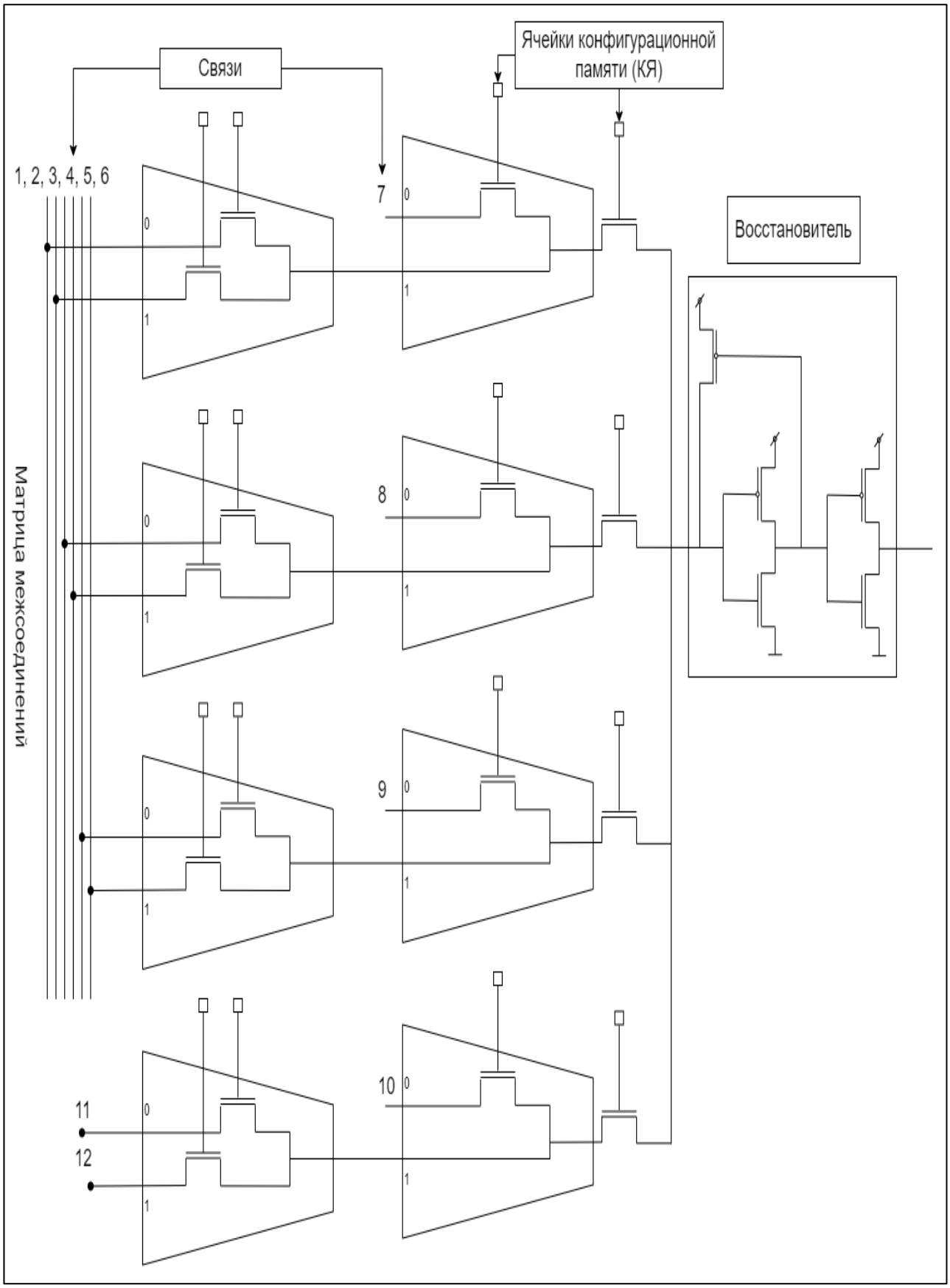


Рисунок 1.8 – Существующий комбинированный вариант коммутатора, доработанный до 12 связей

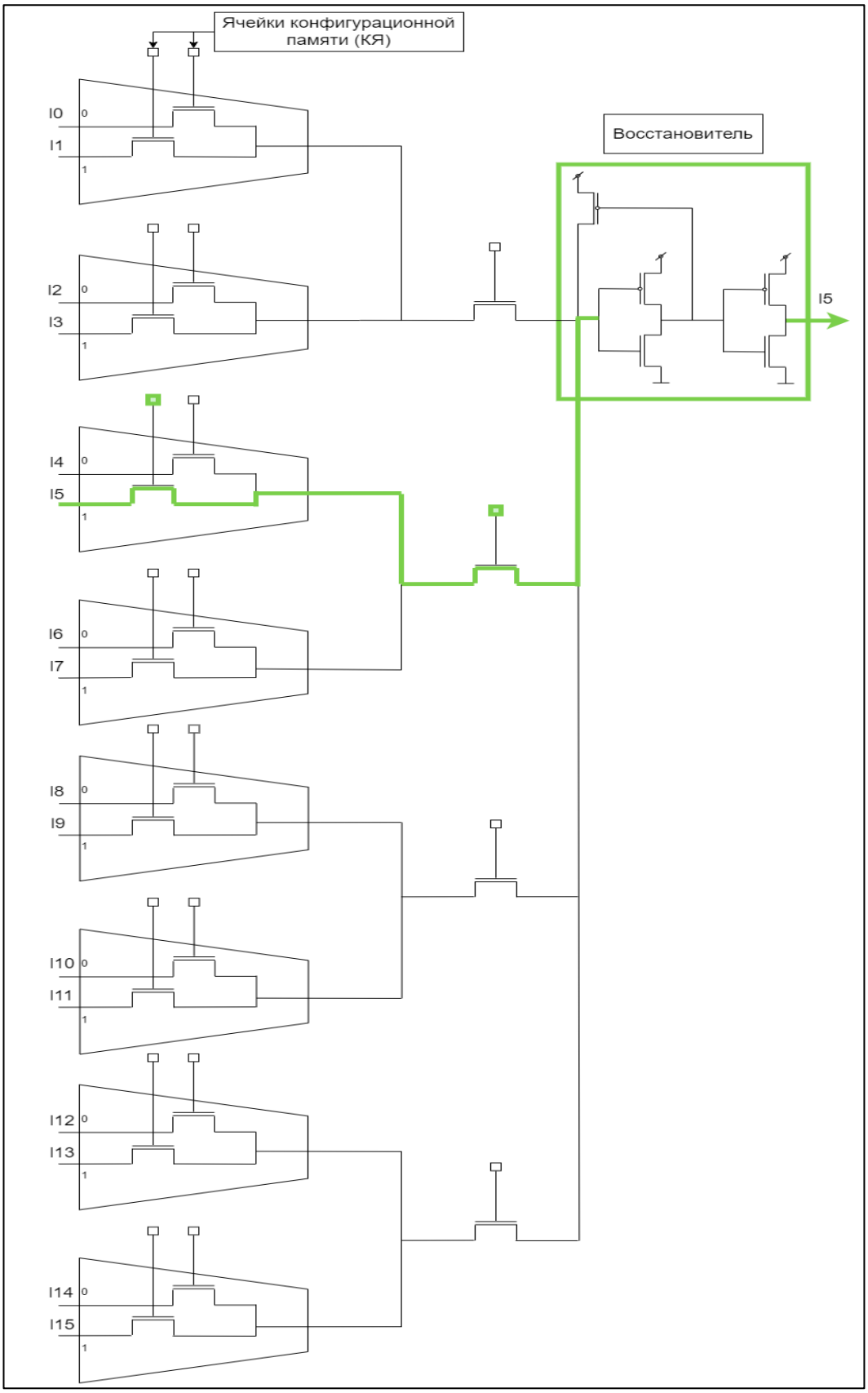


Рисунок 1.9 – Комбинированный вариант коммутатора на 16 связей (Мультиплексор 16-1 с двумя транзисторам в цепочке); объём памяти – 20 бит (и 20 передающих транзисторов соответственно) против 16-ти

Здесь сигнал проходит через два транзистора, но все равно используется восстановитель уровня сигнала. Хотя, по сути, восстановитель в виде инвертора есть и на рисунке 1.5. По сравнению с рисунком 1.5 (50 транзисторов) число транзисторов уменьшено до 25 (20+5 в восстановителе), но увеличен объем памяти = 20 против 16 на рисунке 1.4.

Однако с учетом конфигурационной памяти получаем для рисунка.1.5 (унитарное кодирование) 50+16·6=146 транзисторов. Для рисунка 2.2: 25+20·6 =145 транзисторов. Таким образом, получаем по всем показателям хуже унитарного кодирования (рисунок.1.5). Сведем результаты анализа в таблицу 1.2:

Таблица 1.2 – Сравнение трех вариантов кодирования коммутатора межсоединений

№ Варианта	Название	Количество передающих транзисторов	Объем Памяти (бит)	Общее количество транзисторов	Задержка в количестве транзисторов	Примечание
1	LUT-коммутатор	**30**	**4**	**62**	**4**	Рисунок 1.4
2	Унитарный коммутатор	**16**	**16**	**146**	**1**	Рисунок 1.5
3	Комбинированный	**20**	**20**	**145**	**2**	Рисунок 1.8

Видим, что варианты 1 и 2 несравнимы: LUT – коммутатор выигрывает по числу транзисторов, но проигрывает унитарному коммутатору по задержке. вариант 3 по всем параметрам (один транзистор выигрыша – не в счет) хуже второго.

Если же учесть инверторы по входам связей, как на рисунке 1.5, то комбинированный вариант 3 (рисунок 1.9) будет еще сложнее. Возможно, площадь кристалла комбинированного варианта меньше унитарного.

В унитарном варианте (рисунок 1.5) сразу выбирается лист дерева (0000010000000000). То есть имеется одноуровневое кодирование.

Кроме того, унитарное кодирование используется только в коммутаторах межсоединений и не рассматривается для реализации логических функций.

Другое дело, если комбинируются мультиплексоры на одну переменную и передающие транзисторы, на которые подается унитарный код (рисунок 1.10).

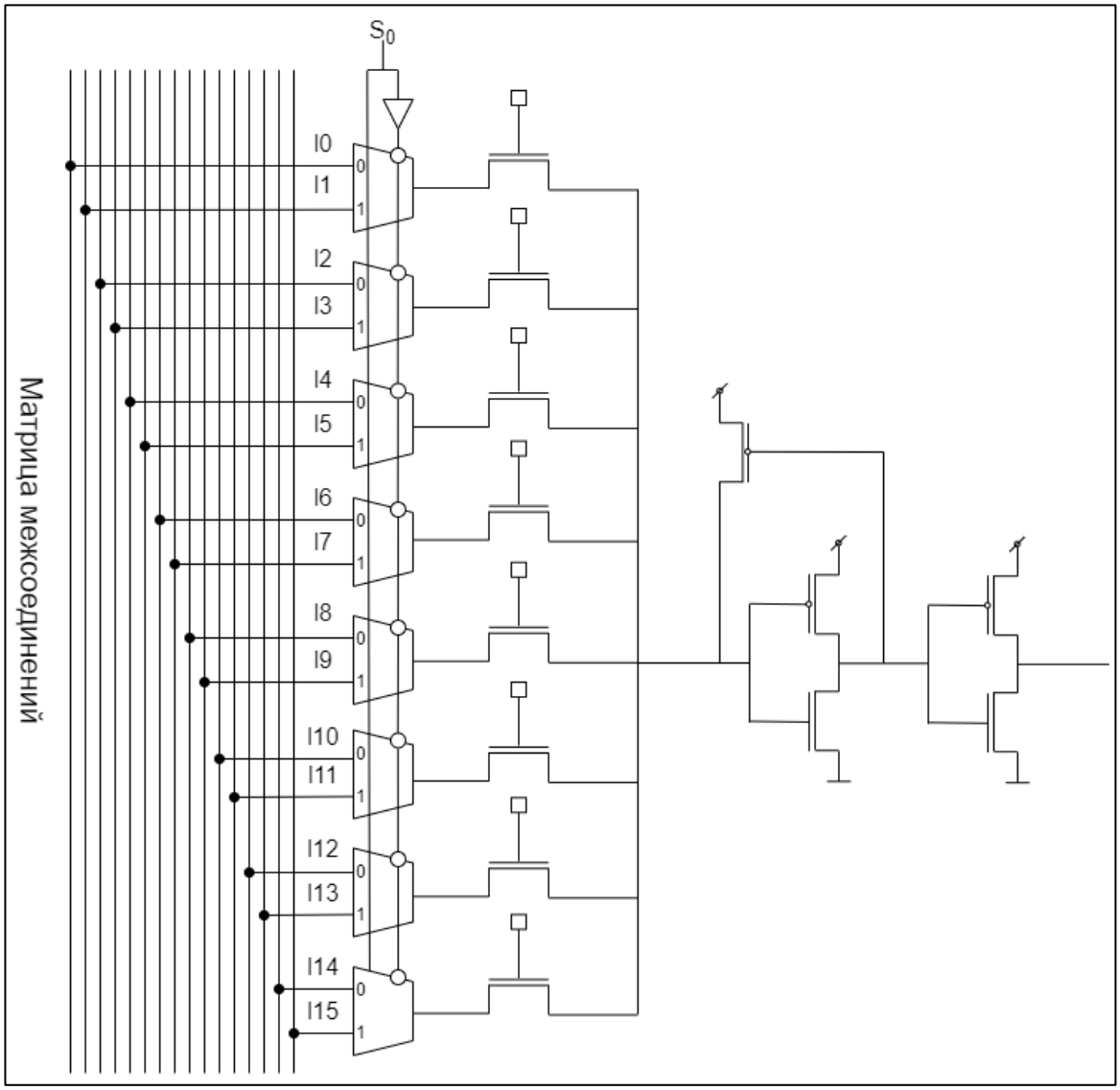


Рисунок 1.10 – Другой возможный комбинированный вариант коммутатора на 16 связей (Мультиплексор 2-1 и унитарный байт); объём памяти – 9 бит, 31 транзистор с учетом инвертора и восстановителей

Но такой вариант соединения позиционного кода (на рисунке – 1.10 один бит *S*0) и унитарного не рассматривается, так же, как вариант, изображенный на рисунке 1.11.

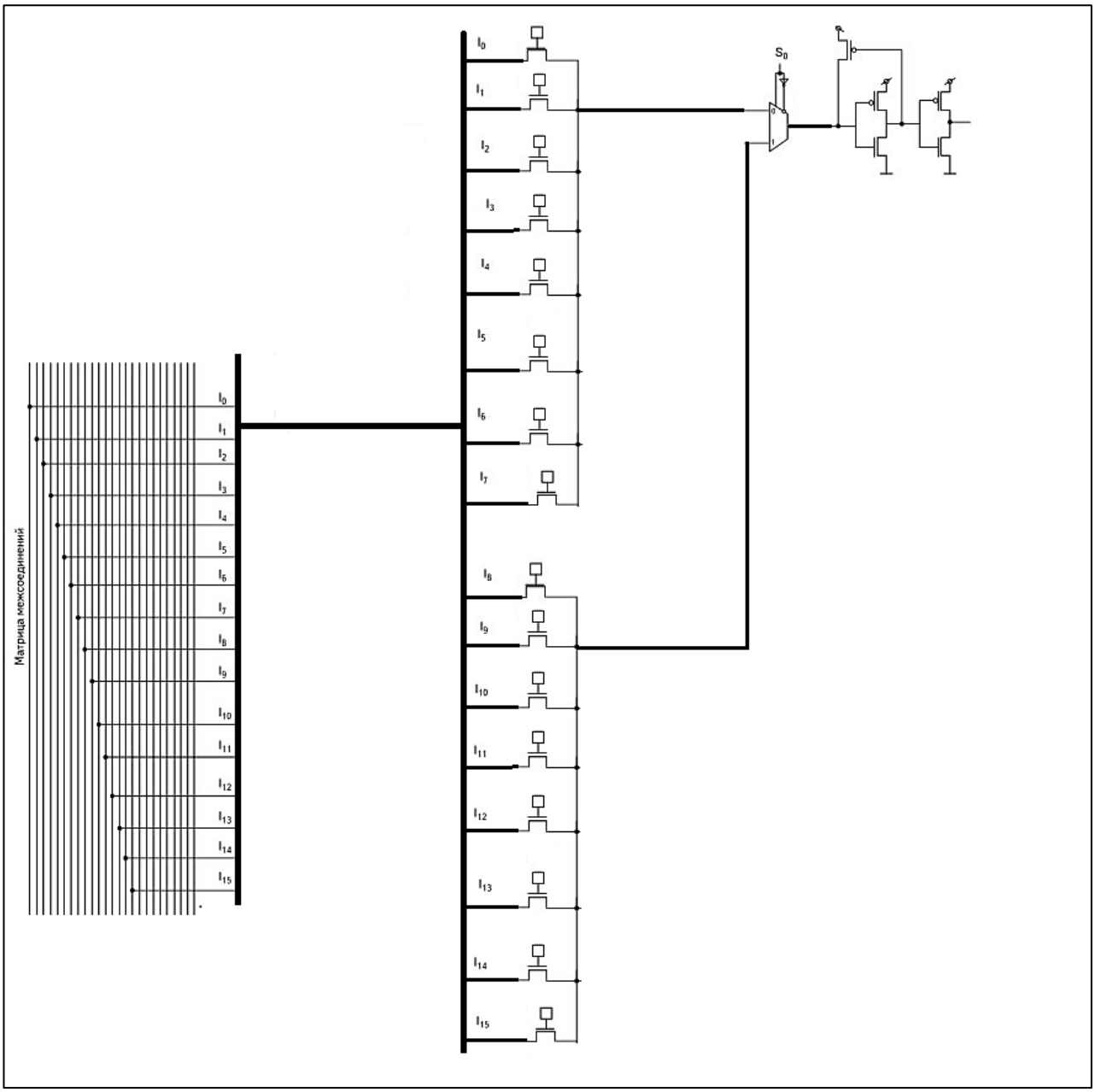


Рисунок 1.11 – Второй возможный комбинированный вариант коммутатора на 16 связей (Мультиплексор 2-1 и два унитарных коммутатора); объём памяти – 9 бит, но 25 транзисторов с учетом инвертора и восстановителей

Таким образом, по результатам анализа источников это направление исследования является новым [57].

В настоящее время имеются сведения об исследованиях применения в области применения в FPGA и других кодов, например, кода Хэмминга [58], БЧХ кода [59–61], Рида-Соломона [62, 63] и др. Однако речь идет о повышении надежности памяти или передачи информации. Код Грея и его модификации

(«соседние коды», отличающиеся только в одном разряде) [64, 65] применяются, например, для устранения состязаний (гонок) входных сигналов, для снижения ошибок преобразователей «вал-код» и пр. Код Грея использует простой алгоритм обеспечения «соседства» наборов: 000, 001, 011, 010, 110, 111, 101, 100,000. В то же время «не соседний» Natural Binary Codes (Sequential Code) использует обычный порядок наборов: 000, 001, 010, 011, 100, 101, 110, 111, 000. В этом случае, в отличие от кода Грея могут изменяться сразу все разряды, когда, например, 111 меняется на 000.

Код Джонсона [66] используется для упрощения схем счетчиков, он тоже «соседний»: 000, 001, 011, 111, 110, 100, 000. Код Айкена [67], как вариант двоично-десятичного (BCD, код 2421) используется для сложения и вычитания. Относительно LUT код Грея ничем принципиально не отличается от обычного Natural Binary Code, просто наборы меняются в другом порядке, но все равно это все возможные коды булеана (множества всех подмножеств) универсального множества из n элементов, что необходимо для выбора только одной цепочки в дереве передающих транзисторов – рисунок 1.3. Это нельзя сказать о кодах Джонсона и Айкена.

Но такой принцип справедлив для унитарного кода (unitary code, one-hot, если одна единица 0001, либо one-cold, если один ноль 0111), который представляет собой, например, код на выходе дешифратора, то есть фиксирует истинность одной из $2n$ конституент некоторой логической функции [55], используется также для кодирования состояний автомата или графа переходов автомата, когда активно (равно единице или нулю) только одно состояние, одна из вершин графа, что снижает сложность логического преобразователя. Однако при этом усложняется блок памяти. Обычный позиционный двоичный код при кодировании памяти автомата в САПР QUARTUS называется «Minimal Bits». Используются также экзотические системы счисления на основе двоичного код Фибоначчи [68], что также повышает помехоустойчивость. Коды для

шифрования информации от злоумышленников в данной работе не рассматриваются.

Таким образом, в исследуемой области, как правило, применяются двоичный позиционный и унитарный коды.

Модификаций этих кодов для элементов логики и коммутаций ПЛИС в доступных источниках найдено не было. Имеются публикации о снижении временной задержки в коммутаторах (мультиплексорах маршрутизации – routing multiplexers) на основе новых физических принципах, например, на резистивной оперативной памяти Resistive Random Access Memory (RRAM) [69].

Достигнут новый уровень быстродействия или энергоэффективности элементов ПЛИС со снижением токов утечки (Leakage current), которые резко возрастают при снижении технологических норм, основе новых типов и технологий транзисторов, использующих химические вещества, не раскрываемые производителями [70 – 73].

Это так называемые «плавниковые» FinFET (fin field-effect transistor), то есть не планарные, а имеющие "3D" затвор, «tri-gate» транзисторы, имеющие три "3D" затвора (triple-gate transistor), с круговым затвором «Gate-all-around FETs» (GAAFETs), которые могут работать при техпроцессах менее 7 nm, «ленточные» RibbonFET транзисторы, ferroelectric FET (CFeFET) [73] и др.

Но это направление (своего рода «физико-химическое») выходит за рамки данного исследования и научной специальности, как и использование, например, тернарной логики [74–76], снижение энергопотребления коммутаторов за счет отключения не используемых участков ПЛИС [27, 28] и пр. «Короткий» унитарный код, помимо указанного выше использования в коммутаторах, используется в «гипер-оптимизации» [49, 77], обеспечивающей короткую обратную связь, где задержка определяется только задержкой последнего АЛМ, а все возможные результаты уже просчитаны заранее, рисунке 1.12.

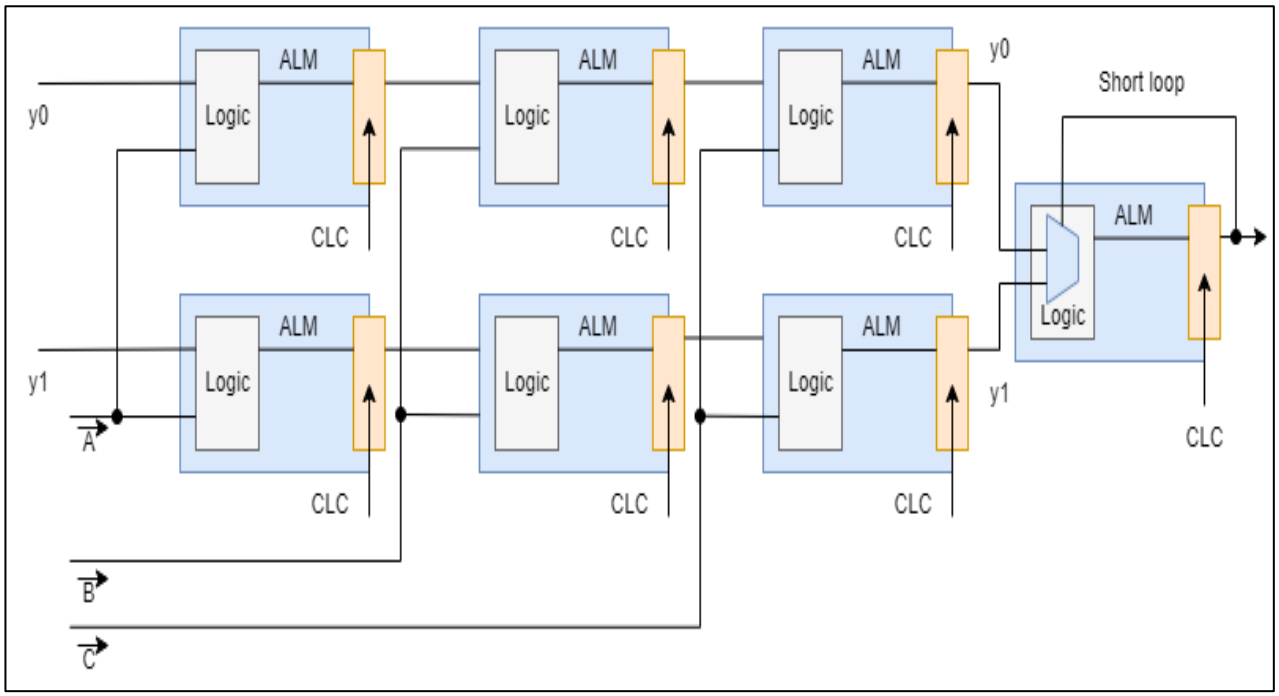


Рисунок 1.12 – Унитарный код ($y1$ $y0$) по значению переменной y

Пусть имеется некоторая условная функция переходов, выражение (1.1):

$$y\ (t+1) = d(t) = [a_1 a_2 a_3 \overline{y}\ (t) \vee \overline{a}_1 \overline{a}_2 \overline{a}_3 y\ (t)] \vee b_1 b_2 b_3 \vee c_1 c_2 c_3. \qquad (1.1)$$

Разложение Шеннона позволяет получить две функции, выражение (1.2):

$$\begin{aligned} y0 &: a_1 a_2 a_3 \vee b_1 b_2 b_3 \vee c_1 c_2 c_3; \\ y1 &: \overline{a}_1 \overline{a}_2 \overline{a}_3 \vee b_1 b_2 b_3 \vee c_1 c_2 c_3. \end{aligned} \qquad (1.2)$$

Дальнейшее развитие этого подхода может привести к разложению Шеннона по большему числу переменных, хотя аппаратные затраты возрастают, но они считаются не существенными по сравнению с выигрышем в задержке. Следует отметить, что такой подход применялся еще в прежних поколениях ПЛИС FPGA [51] для реализации арифметического режима, рисунок – 1.13.

Поэтому имеет смысл в качестве дальнейшего развития этого подхода исследовать различные варианты возможного предложенного выше комбинированного кодирования для вычисления логический функций (с целью снижения временной задержки) и для коммутаторов межсоединений (с целью

снижения объема конфигурационной памяти), а также определить наиболее предпочтительные варианты.

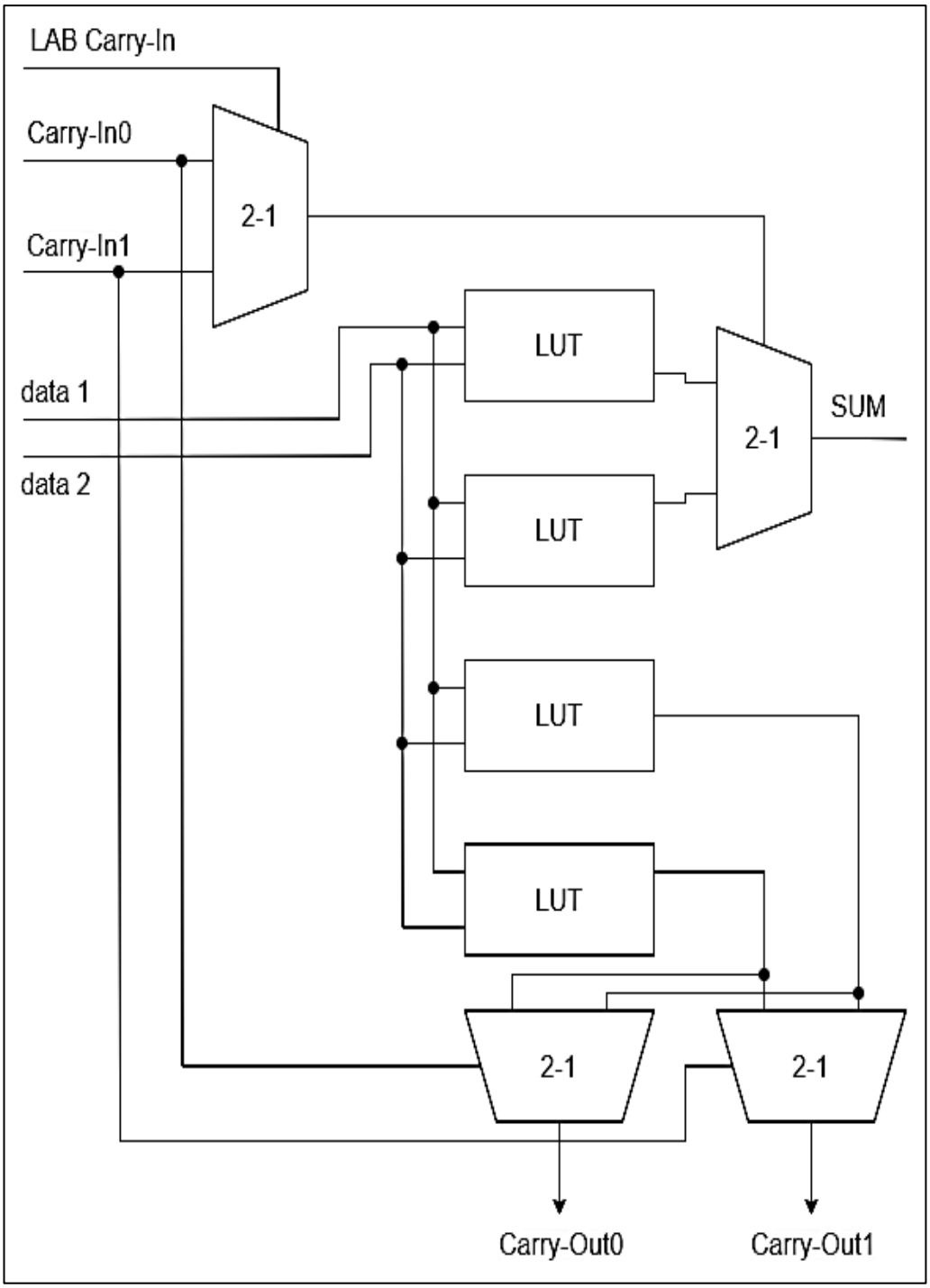


Рисунок 1.13 – Пример унитарного кода по переменной Carry-In

Для этого необходима разработка соответствующего научно-методического аппарата комбинирования унитарного и позиционного кода в элементах ПЛИС.

1.3 Математическая постановка задачи и частных задач исследования

С учетом направлений, задач и целей исследования, установленных выше, математическая постановка задачи выглядит следующим образом.

Дано:

1) Элемент с позиционным кодированием: либо для вычисления логической функции z от n переменных x с таблицей истинности $d_0 \ldots d_{2n-1}$, выражение (1.3):

$$z(d_0 ... d_{2^n-1} x_n \ldots x_1) = \bigvee_{j=0}^{2^n-1} \left(\&_{i=1}^{n} x_i^{\sigma(i,j)} d_j \right), \quad (1.3)$$

где $B(1)$ $\sigma(i,j)$ – 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.

Либо для коммутации 2^n сигналов (связей) q_j, выражение (1.4):

$$z(d_n \ldots d_1, q) = \bigvee_{j=0}^{2^n-1} \left(\&_{i=1}^{n} d_i^{\sigma(i,j)} q_j \right), \quad (1.4)$$

где $d_i \in \{0,1\}$ – настройка одного из 2^n сигналов (связей) q_j.

При этом временная задержка сигнала значения логической функции z или коммутируемого сигнала q_j составляет $T_{pos} \geq n\tau$, τ – задержка на одном n-МОП транзисторе. Неравенство связано с ограничениями на количество последовательно соединенных передающих транзисторов n типа. Сложность такой реализации оценивается в количестве транзисторов L_{pos} и соответствующей площадью кристалла S_{pos}.

Кроме того, показателем качества является потребляемая мощность W_{pos}. Объём памяти конфигурации для вычисления логической функции составляет 2^n и n для коммутации сигналов. Число линий связи U для вычисления логической функции равно n. Для коммутаторов этот символ (U) означает количество ячеек конфигурационной памятью.

2) Элемент с унитарным кодированием (one hot): либо для вычисления логической функции z от n переменных x с таблицей истинности $d_0 ... d_{2^n-1}$, выражение (1.5):

$$z(d_{2^n-1}...d_0 x_{2^n-1}...x_0) = \bigvee_{i=0}^{2^n-1}\left(x_i \cdot d_i\right), \forall x(x_i \ \& \ x_j = 0, i \neq j); \exists! x_i = 1 , \qquad (1.5)$$

либо для коммутации 2^n сигналов (связей) q_j, выражение (1.6):

$$z(d_{2^n-1}.....d_0, q) = \bigvee_{i=0}^{2^n-1}\left(d_i \cdot q_i\right), \forall d(d_i \ \& \ d_j = 0, i \neq j). \qquad (1.6)$$

При этом временная задержка сигнала значения логической функции z или коммутируемого сигнала q_j составляет $T_{oh} = l \cdot \tau$. Сложность такой реализации оценивается в количестве транзисторов L_{oh} и соответствующей площадью кристалла S_{oh}. Кроме того, показателем качества является потребляемая мощность W_{oh}. Объём памяти конфигурации для вычисления логической функции одинаков с первым элементом и составляет 2^n и n для коммутации n сигналов. Число линий связи U_{oh}. для вычисления логической функции равно 2^n (для коммутатора это объём конфигурационной памяти). Унитарное кодирование для реализации логических функций использует разложение Шеннона [49, 77] некоторой функции f по переменной x, причем имеются и другие некоторые переменные – y, z, w, выражение (1.7):

$$\begin{aligned} f\left[(x,\bar{x}),\tilde{y},\tilde{z}...\tilde{w}\right] &= x \wedge f\left[(1,0),\tilde{y},\tilde{z}...\tilde{w}\right] \vee \bar{x} \wedge f\left[(0,1),\tilde{y},\tilde{z}...\tilde{w}\right]; \\ f\left[(x,\bar{x}),\tilde{y},\tilde{z}...\tilde{w}\right] &= \left\{x \vee f\left[(0,1),\tilde{y},\tilde{z}...\tilde{w}\right]\right\}\left\{\bar{x} \vee f\left[(1,0),\tilde{y},\tilde{z}...\tilde{w}\right]\right\}. \end{aligned} \qquad (1.7)$$

В частности, возможен вариант с унитарным кодирование памяти конечного автомата, тем более что он имеется в САПР ПЛИС, например в [63].

3) Коммутатор с двумя группами унитарного кодирования (1.8):

$$\begin{aligned} z\left[\left(x_0.d_0\right)\left(y_{0.0}d_{0.0}...y_{0.\psi}d_{0.\psi}\right)\right] \ ... \ \left[\left(x_\zeta d_\zeta\right)\left(y_{0.\zeta}d_{0.\zeta}...y_{\zeta.\psi}d_{\zeta.\psi}\right)\right] = \\ = [\bigvee_{i=0}^{\zeta}\left(x_i \cdot d_i\right)], \forall x(x_i \ \& \ x_j = 0, i \neq j); \forall y(y_i \ \& \ y_j = 0, i \neq j) \\ \exists! x_i = 1; \exists! y_{i.j} = 1, U_{oh.\zeta} = \zeta \cdot \psi, \end{aligned} \qquad (1.8)$$

где $y_{i.j}$ – вторая группа унитарного кода для соответствующего x.

Требуется получить: Элемент Comb, в котором комбинируются и/или модифицируются (в дальнейшем как правило используется упрощённый термин «комбинируются») выражения (1.1) – (1.5) с целью минимизации временной задержки при заданных ограничениях (restriction) для вычисления логической функции либо для минимизации объёма конфигурационной памяти и аппаратные затраты на коммутацию межсоединений (выражение 1.9):

$$\begin{array}{rll} \textit{Временная задержка}: T_{comb} \to \min; & & U_{comb} \to \min; \\ \textit{Число транзисторов}: L_{comb} \le L_{restrict}; & & T_{comb} \le T_{restrict}; \\ \textit{Площадь кристалла}: S_{comb} \le S_{restrict}; & \text{или} & L_{comb} \le L_{restrict}; \\ \textit{Потребляемая мощность}: W_{comb} \le W_{restrict}; & & S_{comb} \le S_{restrict}; \\ \textit{Число линий связи – переменных или конфиг. бит}: U_{comb} \le U_{restrict}. & & W_{comb} \le W_{restrict}. \end{array} \quad (1.9)$$

Кроме того, необходимо разработать вариант универсального элемента с выбором способа кодирования. Таким образом, задача заключается в разработке научно-методического аппарата синтеза такого нового элемента, в котором комбинируется позиционное и унитарное кодирование переменных (связей) и оценке эффективности по сравнению с существующими.

Декомпозиция (1.9) позволяет получить следующие частные задачи:

1. Разработка математической модели элемента с комбинированным и универсальным кодированием.

2. Разработка метода реализации элемента с комбинированным и универсальным кодированием.

3. Разработка схем электрических функциональных и принципиальных предлагаемых элементов.

4. Схемотехническое моделирование разработанных элементов с комбинированным и универсальным кодированием.

5. Топологическое моделирование разработанных элементов с комбинированным и универсальным кодированием.

6. Получение оценок сложности и эффективности различных вариантов комбинированного кодирования;

7. Апробация и внедрение.

1.4 Выводы по главе 1

1. Актуальность практической задачи подтверждается увеличением объема использования ПЛИС в областях, требующих повышенного быстродействия, в которых ранее применялись только заказные микросхемы.

2. Актуальность научной задачи подтверждается ростом числа публикаций в области методов синтеза логических элементов ПЛИС.

3. Существующие логические элементы ПЛИС, количество переменных которых достигает восьми, могут быть улучшены путем принятия мер по снижению временной задержки, такая потребность имеется, особенно в критических приложениях, в том числе в военной технике.

4. Современный научно-методический аппарат синтеза логических элементов ПЛИС и коммутаторов межсоединений не в полной мере использует возможные варианты комбинирования известных подходов, в том числе группирования унитарных кодов, комбинирования позиционного и унитарного кодирования.

5. Для разработки новых вариантов кодирования необходимо создание соответствующих модели, метода и алгоритма синтеза комбинированного элемента, в том числе и для вычисления логических функций.

ГЛАВА 2. РАЗРАБОТКА МАТЕМАТИЧЕСКОЙ МОДЕЛИ, МЕТОДА И АЛГОРИТМА СИНТЕЗА ЭЛЕМЕНТОВ ПЛИС ТИПА FPGA С ИСПОЛЬЗОВАНИЕМ КОМБИНИРОВАННОГО КОДИРОВАНИЯ

2.1 Модель элемента LUT_{pos} с бинарным (позиционным) кодированием входных переменных или настройки

Реализация логических функций в ПЛИС типа FPGA в LUT [1, 2] основана на вычислении значения одной функции z от n аргументов x_i, заданной в совершенной дизъюнктивной нормальной форме (СДНФ) выражение (1.1) – линейное представление логической функции, которое может быть реализовано, например, стандартными логическими элементами в некотором технологическом базисе. Более компактное представление основано на использовании деревьев из передающих (ключевых) n-МОП транзисторов.

В таком случае (1.1) реализуется либо параллельным соединением $2n$ цепочек из n передающих транзисторов (2.1) (одноуровневым $2n$ деревом, с $2n$ ветвями по n транзисторов, корень дерева, реализующий операцию монтажного ИЛИ (wired OR), d – настройка на заданную функцию, выражение (2.1):

$$z(d_0 \ldots d_{2^n-1} x_n \ldots x_1) = \left.\begin{array}{c} \underline{d_{2^n-1} x_1 \ldots x_n} \\ \underline{d_{2^n-2} x_1 \ldots \bar{x}_n} \\ \ldots. \\ \underline{d_1 x_1 \ldots \bar{x}_n} \\ \underline{d_0 \bar{x}_1 \ldots \bar{x}_n} \end{array}\right\} (\vee \bullet), \tag{2.1}$$

либо, с целью уменьшения количества транзисторов, n-уровневым бинарным деревом транзисторов, выражение (2.2):

$$z(d_0 \ldots d_{2^n-1} x_n \ldots x_1) = \left.\begin{array}{cc} \underline{d_0 \bar{x}_1} & \\ \underline{d_1 x_1} & \\ \cdot & \\ \cdot & \ldots \underline{\bar{x}_{n-1}}\,\underline{\bar{x}_n} \\ \cdot & \ldots \underline{x_{n-1}}\,\underline{x_n} \\ \underline{d_{2^n-2} \bar{x}_1} & \\ \underline{d_{2^n-1} x_1} & \end{array}\right\}(\vee\bullet). \qquad (2.2)$$

Старшей переменной в (2.2) n – соответствуют два транзистора $\underline{\bar{x}_n}$, $\underline{x_n}$

младшей –$2n$ транзисторов ($2n/2$ для x и $2n/2$ для НЕ x):

$$\begin{array}{c} \underline{d_0 \bar{x}_1} \\ \underline{d_1 x_1} \\ \cdot \\ \cdot \\ \cdot \\ \underline{d_{2^n-2} \bar{x}_1} \\ \underline{d_{2^n-1} x_1} \end{array}.$$

В связи с ограничениями Мида-Конвей [26, 54] на число последовательно соединенных передающих p-МОП транзисторов (<=4, как правило, используют <=3, что связано со снижением высокого логического уровня после каждого транзистора на величину порогового напряжения) используют, как базовые, деревья для n=1,2,3, выражение (2.3):

$$z(d_0 d_1 x_1) = \left.\begin{array}{c} \underline{d_0 \bar{x}_1} \\ \underline{d_1 x_1} \end{array}\right\}(\vee\bullet);\quad z(d_0 \ldots d_3 x_2 x_1) = \left.\begin{array}{cc} \left.\begin{array}{c} \underline{d_0 \bar{x}_1} \\ \underline{d_1 x_1} \end{array}\right\} & \underline{\bar{x}_2} \\ \left.\begin{array}{c} \underline{d_2 \bar{x}_1} \\ \underline{d_3 x_1} \end{array}\right\} & \underline{x_2} \end{array}\right\}(\vee\bullet);$$

$$z(d_0 \ldots d_7 x_3 x_2 x_1) = \left.\begin{array}{cc} \underline{d_0 \bar{x}_1} & \\ \underline{d_1 x_1} & \underline{\bar{x}_2} \\ \underline{d_2 \bar{x}_1} & \underline{x_2} \\ \underline{d_3 x_1} & \\ \underline{d_4 \bar{x}_1} & \underline{\bar{x}_3} \\ \underline{d_5 x_1} & \underline{x_3} \\ \underline{d_6 \bar{x}_1} & \underline{\bar{x}_2} \\ \underline{d_7 x_1} & \underline{x_2} \end{array}\right\}(\vee\bullet). \qquad (2.3)$$

LUT на одну переменную используется и для объединения, например, LUT на две и три переменные, а также и как инвертор, и как повторитель, в случае необходимости трассировки через него некоторой связи.

Из этих деревьев строят деревья для реализации функций для n=4, а в современных ПЛИС и для n=5,6,7,8… в так называемых адаптивных логических модулях (АЛМ) [5, 6, 8, 49, 51, 52].

В этом случае в промежуточных деревьях вместо настройки d_j могут указываться связи c_j. Кроме того, с целью восстановления уровня сигнала используют так называемые восстановители v (в простейших случаях это инверторы или пары инверторов, NOT Gates) по входам настройки или коммутации и по выходам функции, например, LUT на четыре переменные будет представлять собой выражение (2.4):

$$\left.\begin{array}{l}\left.\begin{array}{l}\underline{\bar{d}_0 v_{4.0} \bar{x}_1}\\ \underline{\bar{d}_1 v_{4.1} x_1}\end{array}\right\} \underline{\bar{x}_2}\\ \left.\begin{array}{l}\underline{\bar{d}_2 v_{4.2} \bar{x}_1}\\ \underline{\bar{d}_3 v_{4.3} x_1}\end{array}\right\} \underline{x_2}\\ \left.\begin{array}{l}\underline{\bar{d}_4 v_{4.4} \bar{x}_1}\\ \underline{\bar{d}_5 v_{4.5} x_1}\end{array}\right\} \underline{\bar{x}_2}\\ \left.\begin{array}{l}\underline{\bar{d}_6 v_{4.6} \bar{x}_1}\\ \underline{\bar{d}_7 v_{4.7} x_1}\end{array}\right\} \underline{x_2}\end{array}\right\}(\vee\bullet)v_2 \quad \left.\begin{array}{l}\underline{\bar{x}_4}\\ \underline{x_4}\end{array}\right\}(\vee\bullet)v_1.$$

$$\left.\begin{array}{l}\left.\begin{array}{l}\underline{\bar{d}_8 v_{4.8} \bar{x}_1}\\ \underline{\bar{d}_9 v_{4.9} x_1}\end{array}\right\} \underline{\bar{x}_2}\\ \left.\begin{array}{l}\underline{\bar{d}_{10} v_{4.10} \bar{x}_1}\\ \underline{\bar{d}_{11} v_{4.11} x_1}\end{array}\right\} \underline{x_2}\\ \left.\begin{array}{l}\underline{\bar{d}_{12} v_{4.12} \bar{x}_1}\\ \underline{\bar{d}_{13} v_{4.13} x_1}\end{array}\right\} \underline{\bar{x}_2}\\ \left.\begin{array}{l}\underline{\bar{d}_{14} v_{4.14} \bar{x}_1}\\ \underline{\bar{d}_{15} v_{4.15} x_1}\end{array}\right\} \underline{x_2}\end{array}\right\}(\vee\bullet)v_3 \qquad (2.4)$$

В (2.4) v – это функции повторения, восстановители, поскольку их нечетное число в цепочках (три), поэтому настройка функции инверсная, настройка LUT на одну переменную не используется. Такое устройство называется просто мультиплексором 2-1. Например, в [49, 51] описана так называемая декомпозиция полного двоичного сумматора, использующего двухвходовые LUT и мультиплексоры 2-1, в этом случае получаем для одной из функций, выражение (2.5):

$$\left.\begin{array}{l}\left.\begin{array}{l}\underline{d_0}\,\underline{\bar{x}_1}\\ \underline{d_1}\,\underline{x_1}\end{array}\right\}(\vee\bullet)\;\underline{\bar{x}_2}\\ \left.\begin{array}{l}\underline{d_2}\,\underline{\bar{x}_1}\\ \underline{d_3}\,\underline{x_1}\end{array}\right\}(\vee\bullet)\;\underline{x_2}\end{array}\right\}\underline{\bar{x}_3}\\ \left.\begin{array}{l}\left.\begin{array}{l}\underline{d_4}\,\underline{\bar{x}_1}\\ \underline{d_5}\,\underline{x_1}\end{array}\right\}(\vee\bullet)\;\underline{\bar{x}_2}\\ \left.\begin{array}{l}\underline{d_6}\,\underline{\bar{x}_1}\\ \underline{d_7}\,\underline{x_1}\end{array}\right\}(\vee\bullet)\;\underline{x_2}\end{array}\right\}\underline{x_3}\end{array}\right\}(\vee\bullet). \qquad (2.5)$$

В (2.5) и далее для упрощения выражений в ряде случаев не указаны функции восстановителей, но они подразумеваются. В элементе LUT входным сигналом является набор двоичных переменных, например, для (2.5) при значении переменных 000 функция $z(x_3x_2x_1) = d_0$ при значении переменных 111 функция $z(x_3x_2x_1) = d_7$.

2.2 Модель элемента LUT_{oh}, использующего унитарное кодирование входных переменных или настройки

Унитарное кодирование используется для реализации коммутаторов межсоединений (рисунок 1.4). При использовании унитарного кодирования входных переменных для вычисления логических функций получаем выражение (2.6). В этом случае дерево (2.2) будет иметь только одну переменную в ветви:

$$z(d_{2^n-1}....d_0 x_{2^n-1}\dots x_0) = \left.\begin{matrix} \underline{d_{2^n-1}x_{2^n-1}} \\ \underline{d_{2^n-2}x_{2^n-2}} \\ .... \\ \underline{d_1 x_1} \\ \underline{d_0 x_0} \end{matrix}\right\}(\vee\bullet), \qquad (2.6)$$

где $x_{2^n-1} \dots x_0$ – например, «прямой» унитарный код (всегда одна единица в наборе, one-hot), что уменьшает путь сигнала в количестве транзисторов (временную задержку) от конфигурационной памяти, хранящей константы, на выход в n раз, но и увеличивает длину набора переменных $x_{2^n-1} \dots x_0$ в $\frac{2^n}{n}$ раз по сравнению с набором $x_n \dots x_0$.

Так, для реализации функции трех аргументов в унитарном коде, получим выражение (2.7):

$$z(d_7....d_0 x_7...x_1 x_0) = \left.\begin{matrix} \underline{d_7 x_7} \\ \underline{d_6 x_6} \\ .... \\ \underline{d_1 x_1} \\ \underline{d_0 x_0} \end{matrix}\right\}(\vee\bullet). \qquad (2.7)$$

Например, для (2.7) при значении унитарного кода переменных 00000001 функция $z(00000001)=d_0$, при значении переменных 10000000 функция $z(10000000)=d_7$. Здесь (2.7) выходная функция z не закодирована унитарным кодом.

В настоящее время унитарный код используется в коммутаторах связей [2, 78], в таком случае в выражениях (2.7) d_i – соответствующая связь в матрице глобальной или локальной коммутации. Настройка же, наоборот, осуществляется по входам переменных унитарным кодом.

Для коммутации связей с целью уменьшения конфигурационной памяти также может использоваться и позиционный код (2.2) с тем же уточнением.

Выражение (2.7) описывает вычисление одной функции в позиционном коде. Для получения унитарного кода функции $z(d_{2^n-1}\dots d_0 x_{2^n-1}\dots x_0)$ можно, конечно, просто выполнить отрицание и получить $\bar{z}(d_{2^n-1}\dots d_0 x_{2^n-1}\dots x_0)$, но дополнительное выражение (2.8)

$$\begin{aligned} z_0(d_{0.2^n-1}\dots d_{0.0}x_{2^n-1}\dots x_0) &= \left.\begin{matrix} \underline{d_{0.2^n-1}x_{0.2^n-1}} \\ \underline{d_{0.2^n-2}x_{0.2^n-2}} \\ \dots \\ \underline{d_{0.1}x_{0.1}} \\ \underline{d_{0.0}x_{0.0}} \end{matrix}\right\}(\vee\bullet) \\ z_1(\bar{d}_{1.2^n-1}\dots\bar{d}_{1.0}x_{2^n-1}\dots x_0) &= \left.\begin{matrix} \underline{d_{1.2^n-1}x_{1.2^n-1}} \\ \underline{d_{1.2^n-2}x_{1.2^n-2}} \\ \dots \\ \underline{d_{1.1}x_{1.1}} \\ \underline{d_{1.0}x_{1.0}} \end{matrix}\right\}(\vee\bullet) \end{aligned} \qquad (2.8)$$

позволяет осуществлять контроль достоверности вычислений значения функции, например, путем сложения по модулю 2 двух её значений, что важно при диагностике ПЛИС, например, с использованием подходов, описанных в [18, 19, 20, 37]. Переменные дублируются, поэтому указаны индексы.

Аналогично можно вычислить систему m различных функций, допустим f от одних и тех же переменных $x_{2^n-1}\dots x_0$ в позиционном коде, используя m копий входных переменных – выражение (2.9).

Уточним выражение (2.9) для случая представления системы m функций F в унитарном коде следующим образом, получим выражение (2.10).

Для снижения длины унитарного кода (объема конфигурационной памяти) возможна двух координатная коммутация [2, 21, 49], что предлагается применить и для вычисления системы m функций F и h функций G в унитарном коде по двум позиционным входным векторам x, y, размерности соответственно, например q и r – выражение (2.11).

$$f_1(d_{f_1.2^n-1}....d_{f_1.0}x_{2^n-1}\dots x_0)=\left.\begin{array}{c}\underline{d_{f_1.2^n-1}x_{1.2^n-1}}\\ \underline{d_{f_1.2^n-2}x_{1.2^n-2}}\\ \dots.\\ \underline{d_{f_1.1}x_{1.1}}\\ \underline{d_{f_1.0}x_{1.0}}\end{array}\right\}(\vee\bullet);$$

$$f_m(d_{f_m.2^n-1}....d_{f_m.0}x_{2^n-1}\dots x_0)=\left.\begin{array}{c}\underline{d_{f_m.2^n-1}x_{m.2^n-1}}\\ \underline{d_{f_m.2^n-2}x_{m.2^n-2}}\\ \dots.\\ \underline{d_{f_m.1}x_{m.1}}\\ \underline{d_{f_m.0}x_{m.0}}\end{array}\right\}(\vee\bullet).\qquad(2.9)$$

$$F_0(d_{F_0.2^n-1}....d_{F0.0}x_{2^n-1}\dots x_0)=\left.\begin{array}{c}\underline{d_{F_0.2^n-1}x_{1.2^n-1}}\\ \underline{d_{F_0.2^n-2}x_{1.2^n-2}}\\ \dots.\\ \underline{d_{F_0.1}x_{1.1}}\\ \underline{d_{F_0.0}x_{1.0}}\end{array}\right\}(\vee\bullet);$$

.....

$$F_{2^m-1}(d_{f_m.2^n-1}....d_{f_m.0}x_{2^n-1}\dots x_0)=\left.\begin{array}{c}\underline{d_{F_{2^m-1}.2^n-1}x_{m.2^n-1}}\\ \underline{d_{F_{2^m-1}.2^n-2}x_{m.2^n-2}}\\ \dots.\\ \underline{d_{F_{2^m-1}.1}x_{m.1}}\\ \underline{d_{F_{2^m-1}.0}x_{m.0}}\end{array}\right\}(\vee\bullet).\qquad(2.10)$$

Аналогично можно предложить и трех координатное унитарное кодирование, в общем же случае d координат имеется путь из d p-МОП транзисторов, то есть временная задержка увеличивается, но длина кода (и, соответственно, объем конфигурационной памяти) уменьшается с 2^n до $d{\cdot}o$, где o – размерность одной координаты, $n=\lceil\log(d\cdot o,2)\rceil$, где $\lceil\ \rceil$ ближайшее большее натуральное число, $\log(d\cdot o,2)$ – двоичный логарифм.

При многокоординатном позиционном кодировании длина кода не уменьшается, выигрыша во временной задержке также не происходит.

$$F_0(d_{F_0.r\cdot q}....d_{F_{0.1}}y_{2^r-1}\dots y_0x_{2^q-1}\dots x_0)=\left.\begin{array}{c}\underline{d_{F_{0.r\cdot q}}.y_{0.2^r-1}x_{0.2^q-1}}\\ \underline{d_{F_{0.r\cdot q-1}}.y_{0.2^r-2}x_{0.2^q-1}}\\ ....\\ \underline{d_{F_0.2}y_{0.1}x_{0.0}}\\ \underline{d_{F_0.1}y_{0.0}x_{0.0}}\end{array}\right\}(\vee\bullet);$$

.....

$$F_{2^m-1}(d_{F_{2^m-1}.r\cdot q}....d_{F_{2^m-1.1}}y_{2^r-1}\dots y_0x_{2^q-1}\dots x_0)=\left.\begin{array}{c}\underline{d_{F_{2^m-1r\cdot q}}.y_{0.2^r-1}x_{0.2^q-1}}\\ \underline{d_{.F_{2^m-1r\cdot q-1}}y_{0.2^r-2}x_{0.2^q-1}}\\ ....\\ \underline{d_{F_{2^m-1.1}}.y_{0.1}x_{0.0}}\\ \underline{d_{F_{2^m-1}.0}y_{0.0}x_{0.0}}\end{array}\right\}(\vee\bullet);$$

.....

$$G_0(d_{G_0.r\cdot q}....d_{G_{0.1}}y_{2^r-1}\dots y_0x_{2^q-1}\dots x_0)=\left.\begin{array}{c}\underline{d_{G_{0.r\cdot q}}.y_{0.2^r-1}x_{0.2^q-1}}\\ \underline{d_{G_{0.r\cdot q-1}}.y_{0.2^r-2}x_{0.2^q-1}}\\ ....\\ \underline{d_{G_{0.1}}y_{0.1}x_{0.0}}\\ \underline{d_{G_{0.0}}y_{0.0}x_{0.0}}\end{array}\right\}(\vee\bullet);$$

.....

$$G_{2^h-1}(d_{G_0.r\cdot q}....d_{G_{0.1}}y_{2^r-1}\dots y_0x_{2^q-1}\dots x_0)=\left.\begin{array}{c}\underline{d_{G_{2^h-1.r\cdot q}}.y_{0.2^r-1}x_{0.2^q-1}}\\ \underline{d_{G_{2^h-1.r\cdot q-1}}.y_{0.2^r-2}x_{0.2^q-1}}\\ ....\\ \underline{d_{G_{2^h-1.1}}y_{0.1}x_{0.0}}\\ \underline{d_{G2^h-1_0}y_{0.0}x_{0.0}}\end{array}\right\}(\vee\bullet).\qquad(2.11)$$

Получаем $q\cdot r$ бит вместо 2^{q+r} бит, однако это несколько увеличивает задержку за счет использования двух переменных x,y.

Предлагается комплексирование выражений по унитарной переменной, допустим, y, которая будет подключать k разрядный LUT с настроечными векторами D для реализации некоторой функции z – выражение (2.12):

$$z(\overrightarrow{D}_{2^n-1}....\overrightarrow{D}_0 y_{2^n-1}\dots y_0 x_1\dots x_k)=\left.\begin{array}{c}\underline{\overrightarrow{D}_{2^n-1}x_1\dots x_k y_{2^n-1}}\\ \underline{D_{2^n-2}x_1\dots\bar{x}_k y_{2^n-2}}\\ \dots.\\ \underline{D_1\bar{x}_1\dots x_k y_1}\\ \underline{D_0\bar{x}_1\dots\bar{x}_k y_0}\end{array}\right\}(\vee\bullet).\qquad(2.12)$$

Предложенное выражение (2.12) используется для получения системы функций Z в унитарном – выражение (2.13):

$$\overrightarrow{Z}_\xi(\overrightarrow{D}_{\xi.2^n-1}....\overrightarrow{D}_{\xi.0} y_{2^n-1}\dots y_0 x_1\dots x_k)=\left.\begin{array}{c}\underline{\overrightarrow{D}_{\xi.2^n-1}x_1\dots x_k y_{2^n-1}}\\ \underline{D_{\xi.2^n-2}x_1\dots\bar{x}_k y_{2^n-2}}\\ \dots.\\ \underline{D_{\xi.1}\bar{x}_1\dots x_k y_1}\\ \underline{D_{\xi.0}\bar{x}_1\dots\bar{x}_k y_0}\end{array}\right\}(\vee\bullet),\xi=0...2^m,\qquad(2.13)$$

либо в позиционном коде – выражение (2.14):

$$\overrightarrow{Z}_\zeta(\overrightarrow{D}_{\zeta.2^n-1}....\overrightarrow{D}_{\zeta.0} y_{2^n-1}\dots y_0 x_1\dots x_k)=\left.\begin{array}{c}\underline{\overrightarrow{D}_{\zeta.2^n-1}x_1\dots x_k y_{2^n-1}}\\ \underline{D_{\zeta.2^n-2}x_1\dots\bar{x}_k y_{2^n-2}}\\ \dots.\\ \underline{D_{\zeta.1}\bar{x}_1\dots x_k y_1}\\ \underline{D_{\zeta.0}\bar{x}_1\dots\bar{x}_k y_0}\end{array}\right\}(\vee\bullet),\zeta=1...m,\qquad(2.14)$$

либо и в том, и другом для разных функций, например, Z (позиционный) и Y (унитарный) – выражение (2.15):

$$\vec{Y}_{\xi}(\vec{D}_{\xi.2^n-1}....\vec{D}_{\xi.0}y_{2^n-1}\dots y_0x_1\dots x_k)=\left.\begin{array}{l}\underline{\vec{D}_{\xi.2^n-1}x_1\dots x_k y_{2^n-1}}\\ \underline{D_{\xi.2^n-2}x_1\dots \bar{x}_k y_{2^n-2}}\\ \dots.\\ \underline{D_{\xi.1}\bar{x}_1\dots x_k y_1}\\ \underline{D_{\xi.0}\bar{x}_1\dots \bar{x}_k y_0}\end{array}\right\}(\vee\bullet),\xi=0...2^m,$$

$$\vec{Z}_{\zeta}(\vec{D}_{\zeta.2^n-1}....\vec{D}_{\zeta.0}y_{2^n-1}\dots y_0x_1\dots x_k)=\left.\begin{array}{l}\underline{\vec{D}_{\zeta.2^n-1}x_1\dots x_k y_{2^n-1}}\\ \underline{D_{\zeta.2^n-2}x_1\dots \bar{x}_k y_{2^n-2}}\\ \dots.\\ \underline{D_{\zeta.1}\bar{x}_1\dots x_k y_1}\\ \underline{D_{\zeta.0}\bar{x}_1\dots \bar{x}_k y_0}\end{array}\right\}(\vee\bullet),\zeta=1...m. \quad (2.15)$$

Таким образом, получены выражения (2.6–2.15) для вычисления логической функции с использованием только унитарного кода, причем функция или системы функций также представляются в унитарном коде. Это позволяет уменьшить временную задержку за счет увеличения длины кода.

2.3 Разработка модели предлагаемого комбинированного элемента

Обычный (бинарный или позиционный) LUT (1,1), (1.2), (2.2), (2.3) также использует унитарные коды, но отдельно по каждой переменной. Расширяя такое представление, мы получили унитарный LUT по всем переменным (2.6)…(2.15). Но, можно «сделать и шаг назад», рассматривая LUT с унитарными кодами по группам переменных, например, получим две группы по две переменные. Получаем выражение (2.16).

Тогда (2.16) мы уменьшаем задержку в два раза, но вместо четырех разрядов переменных имеем восемь. Подсчитаем сложность такого LUT для вычисления любой функции четырех переменных. Очевидно, что сложность настройки будет такая же, как и в обычном (позиционном) LUT: 16·(6+2)=128 транзисторов.

$$z(\{d_0 \ldots d_{15}\}[x_{2.3}x_{2.2}x_{2.1}x_{2.0}][x_{1.3}x_{1.2}x_{1.1}x_{1.0}]) = \left.\begin{array}{l} \left.\begin{array}{l} d_0 x_{2.0} \\ d_1 x_{2.1} \\ d_2 x_{2.2} \\ d_3 x_{2.3} \end{array}\right\} \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \left.\begin{array}{l} d_{12} x_{2.0} \\ d_{13} x_{2.1} \\ d_{14} x_{2.2} \\ d_{15} x_{2.2} \end{array}\right\} \end{array}\right. \begin{array}{l} \\ \\ \\ \\ \underline{x_{1.0}} \\ \underline{x_{1.1}} \\ \underline{x_{1.2}} \\ \underline{x_{1.3}} \\ \\ \\ \\ \\ \end{array} \left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\right\} (\vee\bullet). \qquad (2.16)$$

Транзисторов в дереве имеем 4·4+4=20. В обычном (бинарном или позиционном) LUT их 30. Восстановителей входных переменных будет 4·4=16. Получаем 36=20+16. Сложность 2-LUT равна 6+2+8=16 (6 транзисторов в дереве + 5 инверторов=10). Итого 52 против 48 транзисторов в обычном LUT. Линий связи надо в два раза больше – восемь, вместо четырёх. Но выигрыш в задержке может того стоить.

Предлагаемый подход позволяет снизить задержку вычисления функций большого числа переменных. Например, можно использовать унитарный код по группам трёх переменных и вычислять логическую функцию девяти переменных на унитарном дереве с тремя уровнями, получаем выражение (2.17).

Комбинирование унитарного и обычного LUT рассмотрим на примере двух переменных *ba* позиционного кода – выражение 2.18. Выражение (2.18) может быть в разных подвариантах в зависимости от того, что слева, что справа: сначала унитарный, потом бинарный (позиционный) или наоборот.

$$z(\{d_0 \ldots d_{511}\}[x_{3.7} \ldots x_{3.2} x_{3.1} x_{3.0}] \ldots [x_{1.7} \ldots . x_{1.2} x_{1.1} x_{1.0}]) = \left\{ \begin{array}{ll} d_0 x_{3.0} & \\ d_1 x_{3.1} & \\ d_2 x_{3.2} & \\ \vdots & \\ d_7 x_{3.7} & \left\{ \begin{array}{l} x_{1.0} \\ x_{1.1} \\ \vdots \\ x_{1.7} \end{array} \right. \\ d_{505} x_{3.0} & \\ \vdots & \\ d_{509} x_{3.5} & \\ d_{510} x_{3.6} & \\ d_{511} x_{3.7} & \end{array} \right\} (\vee \bullet). \tag{2.17}$$

$$z(\{d_0 \ldots d_{15}\}[x_{2.3} x_{2.2} x_{2.1} x_{2.0}][ba]) = \left\{ \begin{array}{ll} d_0 x_{2.0} & \\ d_1 x_{2.1} & \\ d_2 x_{2.2} & \\ d_3 x_{2.3} & \\ \vdots & \\ d_{12} x_{2.0} & \\ d_{13} x_{2.1} & \\ d_{14} x_{2.2} & \\ d_{15} x_{2.2} & \end{array} \right\} \left\{ \begin{array}{l} \\ \\ \\ \\ \overline{b} \\ b \\ \overline{a} \\ a \\ \overline{b} \\ b \\ \\ \\ \\ \\ \end{array} \right. (\vee \bullet). \tag{2.18}$$

Таким образом, получаем модель комбинированного элемента, в которой модифицируется унитарная модель путем разбиения на группы, либо комбинируется путем соединения унитарной и позиционной моделей.

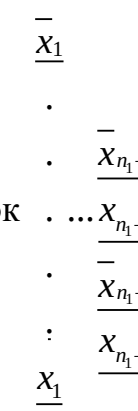
В этом случае n переменных разбиваются на две группы, а настройка логической функции осуществляется по нескольким унитарным блокам размерностью n_2, количество которых равно количеству конституент единицы в единственном позиционном блоке размерностью n_1.

В линейной форме получим выражение (2.19):

$$z\{(d_{2^{n_1}-1.0}d_{2^{n_2}-2.0}\ldots d_{1.0}d_{0.0})\ldots(d_{2^{n_2}-1.2^{n_1}-1}d_{2^{n_2}-1.2^{n_1}-1}\ldots d_{1.2^{n_1}-1}d_{0.2^{n_1}-1})$$
$$(x_{2.2^{n_2}-1}\ldots x_{2.0})\quad [x_{1.n_1}\ldots x_{1.1}]\}= \tag{2.19}$$
$$=\bigvee_{\delta=0}^{2^{n_2}-1}\left(x_{2.\delta}\cdot d_{2.\delta}\right)\&\{\bigvee_{j=0}^{2^{n_1}-1}\left(\&_{i=1}^{n_1}x_i^{\sigma(i,j)}\right)\},$$

где $\sigma(i,j)$ – показатель инверсирования соответствующей СДНФ, позиционного блока, 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.

Получаем выражение (2.20) в виде комбинированного дерева:

В (2.20) позиционный блок  расположен справа.

Рамка зеленого цвета обозначает унитарную часть, а синяя позиционную.

В дальнейшем целесообразно исследовать предлагаемые варианты и оценить их эффективность с помощью моделирования и получения показателей сложности.

Другой вариант комбинирования представляет собой выражение (2.21):

$$z\{(d_{2^{n_1}-1.0}d_{2^{n_2}-2.0}\ldots d_{1.0}d_{0.0})\ldots(d_{2^{n_2}-1.2^{n_1}-1}d_{2^{n_2}-1.2^{n_1}-1}\ldots d_{1.2^{n_1}-1}d_{0.2^{n_1}-1})$$

$$(x_{2.2^{n_2}-1}\ldots x_{2.0})\quad [x_{1.n_1}\ldots x_{1.1}]\}=$$

$$=\bigvee_{\delta=0}^{2^{n_2}-1}\left(x_{2.\delta}\cdot d_{2.\delta}\right)\&\{\bigvee_{j=0}^{2^{n_1}-1}\left(\&_{i=1}^{n_1}x_i^{\sigma(i,j)}\right)\}$$

$$z\{(d_{2^{n_1}-1.0}d_{2^{n_2}-2.0}\ldots d_{1.0}d_{0.0})\ldots(d_{2^{n_2}-1.2^{n_1}-1}d_{2^{n_2}-1.2^{n_1}-1}\ldots d_{1.2^{n_1}-1}d_{0.2^{n_1}-1})(x_{2.2^{n_2}-1}\ldots x_{2.0})[x_{1.n_1}\ldots x_{1.1}]\}=$$

унитарная часть

позиционная часть

$$=\begin{array}{l}\left.\begin{array}{l}\underline{d_{2^{n_2}-1.0}x_{2.2^{n_2}-1}}\\ \underline{d_{2^{n_2}-2.0}x_{2.2^{n_2}-2}}\\ \ldots.\\ \underline{d_{1.0}x_{2.1}}\\ \underline{d_{0.0}x_{2.0}}\end{array}\right\}(\vee\bullet)\bar{x}_1\\ \vdots\\ \left.\begin{array}{l}\underline{d_{2^{n_2}-1.2^{n_1}-1}x_{2.2^{n_2}-1}}\\ \underline{d_{2^{n_2}-2.2^{n_1}-1}x_{2.2^{n_2}-2}}\\ \ldots.\\ \underline{d_{1.2^{n_1}-1}x_{2.1}}\\ \underline{d_{0.2^{n_1}-1}x_{2.0}}\end{array}\right\}(\vee\bullet)x_1\end{array}\quad\left.\begin{array}{l}\underline{\bar{x}_{n_1-1}}\,\underline{\bar{x}_{n_1}}\\ \ldots\underline{x_{n_1-1}}\,\underline{x_{n_1}}\\ \underline{\bar{x}_{n_1-1}}\\ \underline{x_{n_1-1}}\end{array}\right\}(\vee\bullet).\qquad(2.20)$$

$$z\{(d_{2^{n_1}-1.0}d_{2^{n_2}-2.0}\ldots d_{1.0}d_{0.0})\ldots(d_{2^{n_2}-1.2^{n_1}-1}d_{2^{n_2}-1.2^{n_1}-1}\ldots d_{1.2^{n_1}-1}d_{0.2^{n_1}-1})(x_{2.2^{n_2}-1}\ldots x_{2.0})[x_{1.n_1}\ldots x_{1.1}]\}=$$

$$=\begin{array}{l}\left.\begin{array}{l}\underline{\bar{x}_{n_1-1}}\,\underline{\bar{x}_{n_1}}\\ \underline{x_{n_1-1}}\,\underline{x_{n_1}}\,x_{2.0}\\ \underline{\bar{x}_{n_1-1}}\\ \underline{x_{n_1-1}}\end{array}\right\}(\vee\bullet)\\ \vdots\\ \left.\begin{array}{l}\underline{\bar{x}_{n_1-1}}\,\underline{\bar{x}_{n_1}}\\ \underline{x_{n_1-1}}\,\underline{x_{n_1}}\,x_{2.2^{n_2}-1}\\ \underline{\bar{x}_{n_1-1}}\\ \underline{x_{n_1-1}}\end{array}\right\}(\vee\bullet)\end{array}\quad\left.\begin{array}{l}\underline{x_{2.0}}\\ \vdots\\ \ldots\\ \vdots\\ \underline{x_{2.2^{n_2}-1}}\end{array}\right\}(\vee\bullet).\qquad(2.21):$$

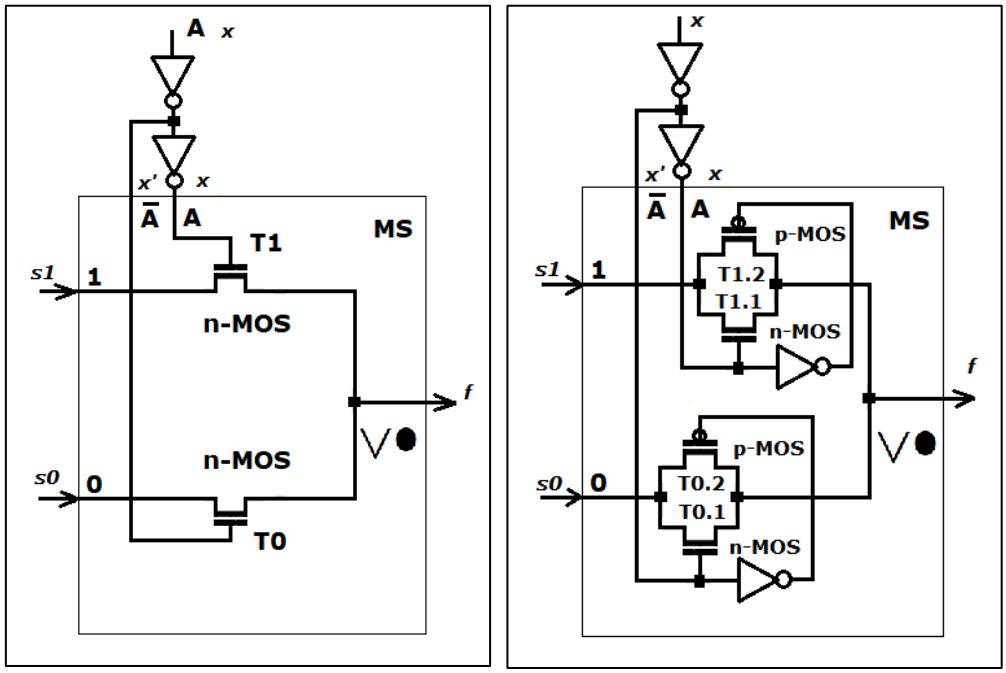
2.4 Разработка метода синтеза элементов ПЛИС типа FPGA с использованием комбинированного кодирования

Сущность предлагаемого метода, позволяющего строить элементы в соответствие моделями (2.20), (2.21), заключается в следующем.

Дано выражение (2.22) элементарных мультиплексоров 2-1 из двух передающих транзисторов *n*-МОП (или двух КМОП ключей [1, 2]) *j.k.x, j.k.x'* *k*-уровень дерева от *n* до 1 (старшая переменная – уровень 1, но переменная *n*):

$$\begin{matrix} j.k.s_0 - j.k.x' \\ ](\vee\bullet)\, j.k.f. \\ j.k.s_1 - j.k.x \end{matrix} \qquad (2.22)$$

В (2.22) «–» означает соответствующую связь, «](∨•)» означает корень дерева, реализующий операцию «Монтажное ИЛИ». В этом случае сигналы *j.k.x, j.k.x'*инверсны (ортогональны), то есть на выходе *i.f* всегда имеется некоторый логический уровень. На рисунке.2.1 показана реализация (2.22) в виде мультиплексора 2-1 (1-LUT).



а) б)

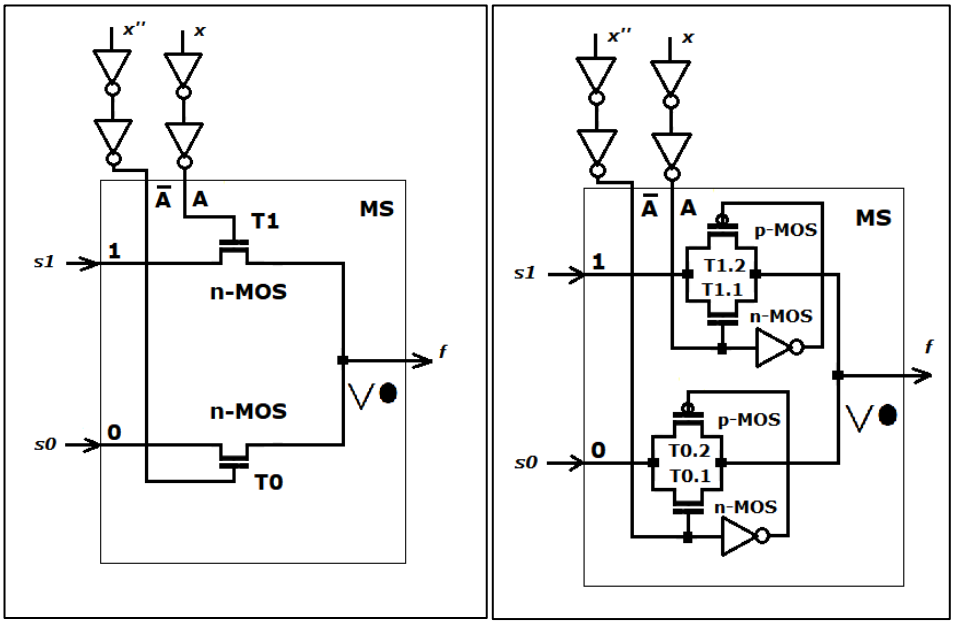
Рисунок 2.1 – Мультиплексор 2-1 (1-LUT): а) на базе *n*-МОП транзисторов; б) на базе КМОП транзисторов

Транзистор *n*-МОП Т0 (рисунок 2.1 а) получает на исток некоторый сигнал *i.s*.0, который в случае *i.x*'=1 передает на выход *i.f* сигнал *i.s*.1. Транзистор *n*-МОП Т1 получает на исток некоторый сигнал *i.s*.1, который в случае *i.x*=1 передает на выход *i.f* сигнал *i.s*.1. Аналогично работают КМОП-ключи, показанные на рисунке 2.1,б, более предпочтительные при передаче сигналов уровня логического нуля, но и более сложные.

Также имеется выражение (2.23) элементарных коммутаторов 2-1 из двух передающих транзисторов *n*-МОП *i.x, i.x*'', где *i*- номер мультиплексора (или итерации):

$$\begin{matrix} i.s_0 - i.x'' \\ ](\vee\bullet) i.f. \\ i.s_1 - i.x \end{matrix} \qquad (2.23)$$

Сигналы *i.x, i.x*'' таковы, что если один равен единице, то другой равен нулю, иначе оба нуля. То есть на выходе *i.f* может и не быть логического уровня. На рисунке 2.2 показана реализация (2) в виде коммутатора 2-1 (1-LUT).



а) б)

Рисунок 2.2 – Коммутатор (унитарный 1-LUT) 2-1: а) на базе *n*-МОП транзисторов; б) на базе КМОП транзисторов

Требуется синтезировать комбинированный элемент LUT с использованием комбинированного кодирования n переменных, $n=n_2+n_1$, где n_1 – реализуется обычным (позиционным) LUT, а n_2 – унитарным LUT. При этом n_1 и n_2 могут быть в двух вариантах, выражение (2.24):

$$1)\ (n_1, n_2),\ 2)\ (n_2, n_1). \qquad (2.24).$$

Для этого предлагается выполнить следующие этапы.

2.4.1 Синтез унитарных блоков

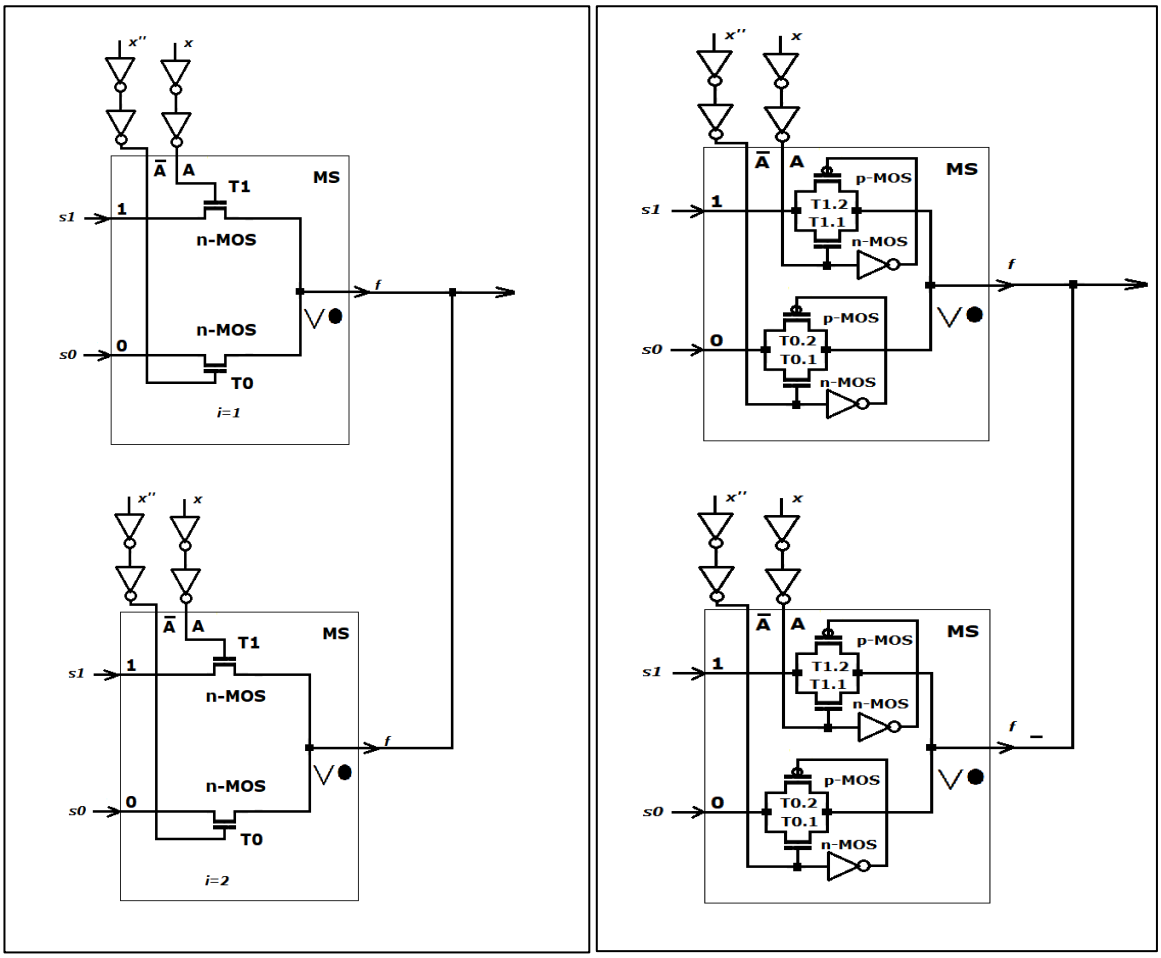
Синтез одного унитарного блока: 1.1. i=1. Соединить $i.f$ *с* $i+1.f$ – выражение (2.25), рисунок 2.3:

$$\left.\begin{matrix} i.s_0 - i.x'' \\ ](\vee\bullet) i.f \\ i.s_1 - i.x \\ i+1.s_0 - i+1.x'' \\ ](\vee\bullet) i.f \\ i+1.s_1 - i+1.x \end{matrix}\right](\vee\bullet). \qquad (2.25)$$

Повторить п. 2^{n_2-1} , где 1– раз, n_2 – размерность унитарного блока.

В результате получается выражение (2.26):

$$\left.\begin{matrix} i.s_0 - i.x'' \\ ](\vee\bullet) i.f \\ i.s_1 - i.x \\ i+1.s_0 - i+1.x'' \\ ](\vee\bullet) i.f \\ i+1.s_1 - i+1.x \end{matrix}\right](\vee\bullet)\ldots \left.\begin{matrix} \\ \\ \\ \\ \\ \\ \cdot \\ \cdot \\ i+2^{n_2}.s_0 - i+2^{n_2}.x'' \\ ](\vee\bullet) i.f \\ i+2^{n_2}.s_1 - i+2^{n_2}.x \end{matrix}\right](\vee\bullet). \qquad (2.26)$$



а) б)

Рисунок 2.3 – Коммутатор (унитарный 2-LUT) 4-1: а) на базе *n*-МОП транзисторов; б) на базе КМОП транзисторов

Далее, если производится вычисление логической функции *f*(*dx*), выполняется пункт 1.3. Подключаются *i.s* к соответствующим выходам конфигурационной памяти *d*. 1.4. Подключить *i.x*, *i.x*'' к соответствующим входам переменных унитарного блока (унитарной части кода переменных) для реализации логической функции *f*(*dx*), выражение (2.27):

$$\left[\begin{array}{l} d_0(i.s_0)-i.x'' \\ \qquad](\vee\bullet)i.f(dx) \\ d_1(i.s_1)-i.x \\ d_2(i+1.s_0)-i+1.x'' \\ \qquad](\vee\bullet)i.f(dx) \\ d_3(i+1.s_1)-i+1.x \end{array}\right](\vee\bullet)\ldots \left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \end{array}\right] \left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \end{array}\right](\vee\bullet). \quad (2.27)$$

$$\begin{array}{l} \cdot \\ \cdot \\ d_{2^{n_2}-2}(i+2^{n_2}-2.s_0)-i+2^{n_2}-2.x'' \\ \qquad](\vee\bullet)i.f(dx) \\ d_{2^{n_2}-1}(i+2^{n_2}-1.s_1)-i+2^{n_2}-1.x \end{array}$$

Если реализуется коммутатор, то необходимо подключить *i.s* к входам матрицы межсоединений *q* для реализации *f*(*dq*), по входам *i.x*, *i.x*'' осуществляется настройка соответствующими выходами конфигурационной памяти *d*, выражение (2.28):

$$\left[\begin{array}{l} q_0(i.s_0)-i.x''(d_0) \\ \qquad](\vee\bullet)i.f(qd) \\ q_1(i.s_1)-i.x(d_1) \\ q_2(i+1.s_0)-i+1.x''(d_2) \\ \qquad](\vee\bullet)i.f(qd) \\ q_3(i+1.s_1)-i+1.x(d_3) \end{array}\right](\vee\bullet)\ldots \left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \end{array}\right] \left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \end{array}\right](\vee\bullet). \quad (2.28)$$

$$\begin{array}{l} \cdot \\ \cdot \\ q_{2^{n_2}-2}(i+2^{n_2}-2.s_0)-i+2^{n_2}-2.x''(d_{2^{n_2}-2}) \\ \qquad](\vee\bullet)i.f(qd) \\ q_{2^{n_2}-1}(i+2^{n_2}-1.s_1)-i+2^{n_2}-1.x(d_{2^{n_2}-1}) \end{array}$$

Выражения (2.27), (2.28) содержат один транзистор в пути сигнала от *d* до *f*(*dx*) или от *q* до *f*(*qd*), поэтому своего рода «внутренние» восстановители не требуются, однако, они устанавливаются на входах и выходах. По входам настройки (или переменных) это пара инверторов, по входам связей – это инверторы, а по выходу – тоже инвертор NOT или восстановитель уровня.

2.4.2 Синтез *j* позиционных блоков

Синтез многоуровневого (k=1…n_1) дерева j-го блока необходимо соединить один блок $j.1.f_1$ с выходным инвертором NOT (восстановителем), $j.1.x$, , $j.1.x$’ с соответствующими входами x_{n1}, $x’_{n1}$, выражение (2.29):

$$\begin{matrix} j.1.s_0-(j.1.x')x_{n_1} \\ ](\vee\bullet)\, j.1.f_1(NOT). \\ j.1.s_1-(j.1.x)x_{n_1} \end{matrix} \quad (2.29)$$

Далее соединяются два блока (j+1.2.f_2, j+1.2.f_2), (j+2.2.f_2, j+2.2.f_2), с соответствующими входами $j.n_1.s1$; $j.n_1.s2$; $j.1.x$, $j.1.x$’ и соответствующими входами переменных $x_{n1\text{-}1}$, $x’_{n1\text{-}1}$, выражение (2.30):

$$\begin{matrix} j+3.2.s_0-(j+3.2.x')x_{n_1-1} \\ ](\vee\bullet)\, j+1.2.f_2\bullet j.1.s_1-(j.1.x')x_{n_1} \\ j+3.2.s_1-(j+3.2.x)x_{n_1-1} \\ ](\vee\bullet)\, j.1.f_1(NOT). \\ j+4.2.s_0-(j+2.2.x')x_{n_1-1} \\ ](\vee\bullet)\, j+2.2.f_3\bullet j.1.s_2-(j.1.x)x_{n_1} \\ j+4.2.s1-(j+2.2.x)x_{n_1-1} \end{matrix} \quad (2.30)$$

Выполняются вышеуказанные пункты: 2^{n_1} – 2 раз, n_1 – размерность позиционного блока до подключения всех позиционных переменных, если не задано ограничение Мида-Конвей [24]. Получаем одно дерево.

В случае задания ограничения Мида-Конвей [24] необходимо строить несколько деревьев, соединяя их для получения общего дерева.

Так, для ограничения (<=3) требуемое n дерево строится, как композиция деревьев n=1, n=2, n=3. Такие деревья генерируются, исходя из требуемого n, а затем соединяются в одно дерево.

Так, для n=5 получают (3+3+1=4), (3+3+1=4), (4+4+1=5), здесь знак «+» означает соединение деревьев. Но возможна и другая декомпозиция: (3+3+3+3)+2=5.

2.4.3 Соединение позиционных и унитарных блоков

Соединение требуемых типов позиционных блоков и унитарных в двух вариантах. Первый вариант: соединить $i.f$ всех i унитарных блоков к соответствующим входам $j.n_2.s1$; $j.n_2.s2$ всех $j.n_2$ позиционных блоков (рассматриваем многоуровневое дерево для вычисления логической функции). Так, для унитарного на 2^{n2}=4 переменных и позиционного на одну n_1=1 получаем выражение (2.31). Здесь позиционный блок справа, выражение (2.31):

$$\left[\begin{matrix} i.s_0 - i.x'' \\ ](\vee\bullet) i.f \\ [\begin{matrix} i.s_1 - i.x \\ i+1.s_0 - i+1.x'' \end{matrix} \\ ](\vee\bullet) i.f \\ i+1.s_1 - i+1.x \end{matrix}\right](\vee\bullet)][(\vee\bullet)\cdot j.k.s_0 - (j.k.x')x_{n_1}$$

$$](\vee\bullet)\, j.k.f\,(dx_{n_1+n_2})(NOT).$$

$$\left[\begin{matrix} i.s_0 - i.x'' \\ ](\vee\bullet) i.f \\ [\begin{matrix} i.s_1 - i.x \\ i+1.s_0 - i+1.x'' \end{matrix} \\ ](\vee\bullet) i.f \\ i+1.s_1 - i+1.x \end{matrix}\right](\vee\bullet)][(\vee\bullet)\cdot j.k.s_1 - (j.k.x)x_{n_1} \qquad (2.31)$$

Второй вариант: соединить $i.f$ всех i позиционных блоков к соответствующим входам унитарных блоков, получаем выражение (2.32), здесь унитарный блок на 2^{n2} переменных справа, позиционный на одну переменную n_1=1.

Графически этапы метода синтеза комбинированного элемента показаны на рисунке 2.4. Сущность метода заключается в выполнении различных вариантов комбинирования унитарных и бинарных деревьев для заданного числа переменных как для создания элемента для вычисления логической функции, так и для коммутации связей.

$$
\begin{array}{l}
\left[\begin{array}{l}
\left[\begin{array}{l}
(i.s_0)-(i.x'')x_0 \\
\qquad\qquad](\vee\bullet)i.f(dx) \\
...(i.s_1)-(i.x)x_1
\end{array}\right](\vee\bullet)... \\
...(i+1.s_0)-(i+1.x'')x_2 \\
\qquad\qquad](\vee\bullet)i.f(dx) \\
....(i+1.s_1)-(i+1.x)x_3
\end{array}\right](\vee\bullet). \\
\bullet \\
\bullet \\
...(i+2^{n_2}-2.s_0)-(i+2^{n_2}-2.x'')x_{2^{n_2}-2} \\
\qquad\qquad](\vee\bullet)i.f(dx) \\
(i+2^{n_2}-1.s_1)-(i+2^{n_2}-1.x)x_{2^{n_2}-1}
\end{array}
\qquad (2.32)
$$

Сначала производится синтез унитарных деревьев, далее – синтез бинарных (позиционных) деревьев. При комбинировании этих двух видов деревьев рассматриваются два варианта расположения позиционных и унитарных элементов относительно друг друга: унитарные блоки слева (на них поступают настройки функции или подключаются связи, соответственно, один бинарный блок справа) или бинарные блоки слева (на них поступают настройки функции или подключаются связи, соответственно, один унитарный блок справа). То есть, вводятся своего рода два «слоя»: унитарный и бинарный, большее число слоёв не рассматривается.

Для синтеза комбинированного элемента при конкретном количестве части переменных, отведенных для унитарного или бинарного дерева, предлагается следующий алгоритм [102].

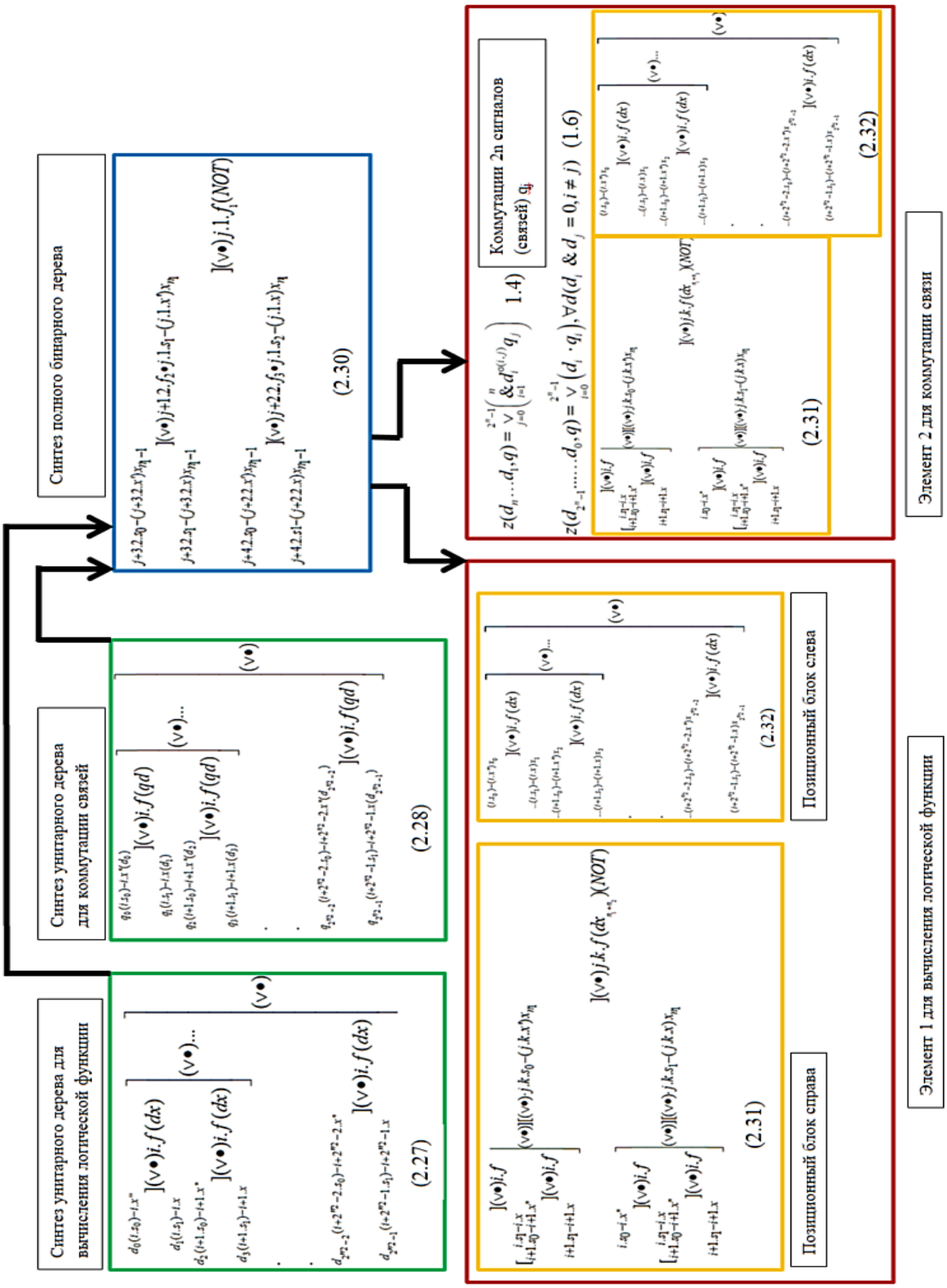
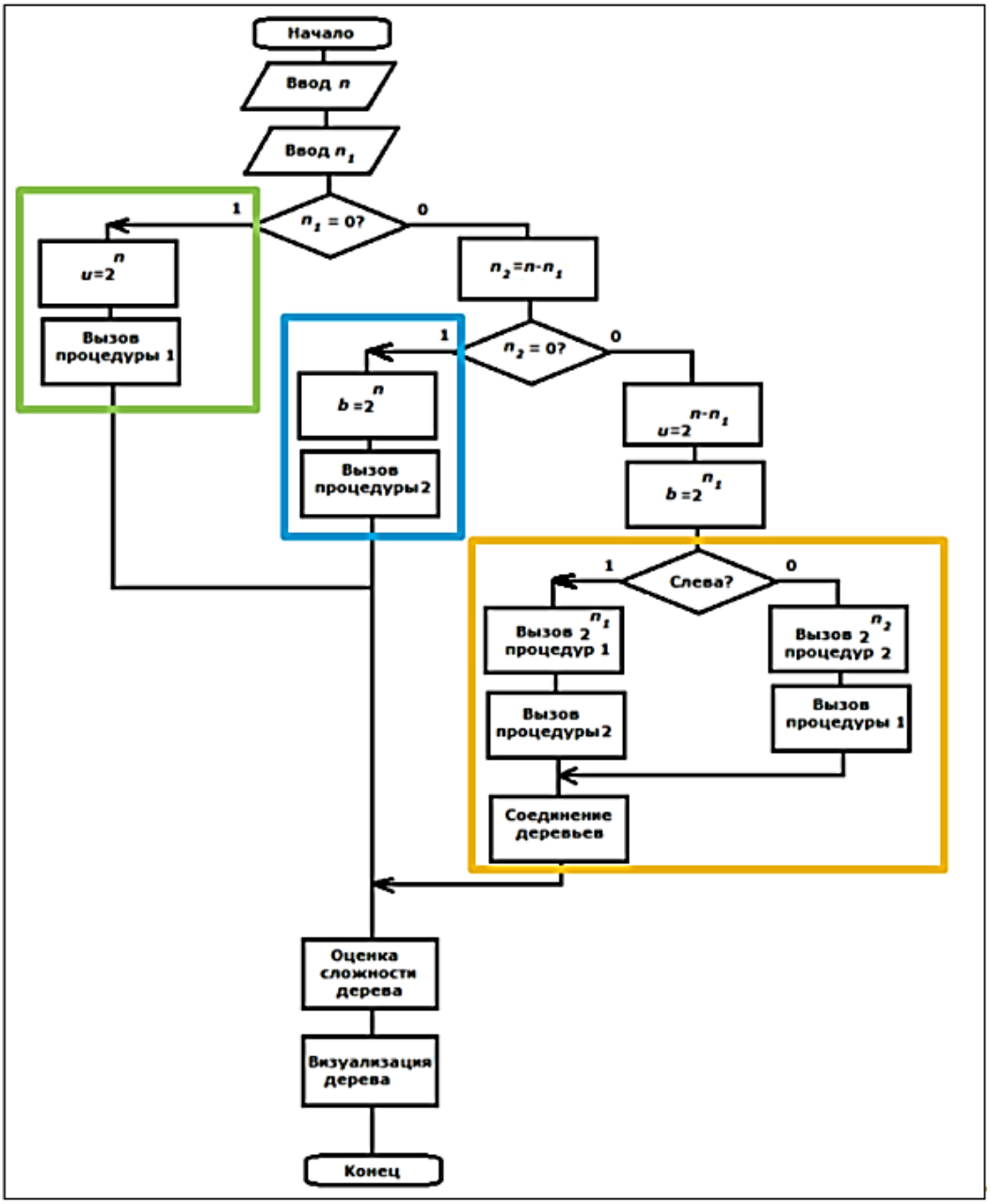


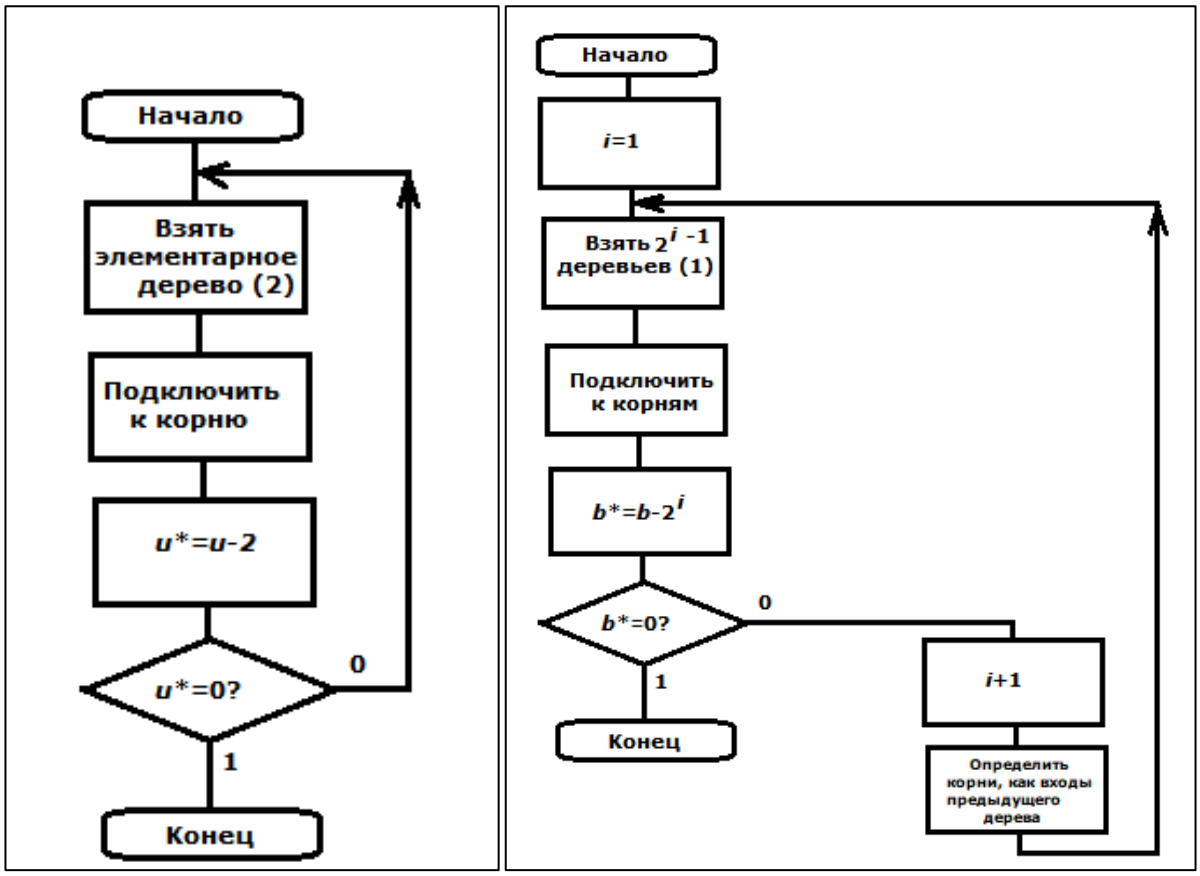
Рисунок 2.4 – Сущность метода синтеза в виде построения комбинированного дерева

2.5 Алгоритм и программа синтеза элемента с комбинированным кодированием

Разработанная схема алгоритма синтеза элемента с комбинированным кодированием показана на рисунке 2.5,а. В синей области – построение блоков схемы с позиционным кодированием, в зеленой области – построение блоков схемы с унитарным кодированием, в оранжевой области – построение блоков схемы с комбинированным кодированием.



а)



б) в)

Рисунок 2.5 – Схема алгоритма синтеза элемента с комбинированным кодированием: а) основная процедура; б) процедура №1 синтеза унитарного дерева; в) процедура №2 синтеза полного бинарного дерева

Особенностью является то, что ветвей унитарного дерева 2^{n_2} в отличие произвольного случая. Процедура 1 соответствует выражению (2.26). Процедура (2.23) соответствует выражению (2.30). Соединение деревьев – выражения (2.31) (условие «слева») или (2.32). Если переменная n_1=0, то определяется n_2 для унитарного дерева и присваивается значение u. Иначе $n_1 \neq 0$, то определяется n_2 и присваивается значение b для построения позиционной части.

По предложенному алгоритму разработана программа синтеза на языке C++. Описание программы находится в приложении А.

2.6 Разработка универсального элемента с конфигурируемым кодированием переменной

Вышеописанные модели, метод и алгоритм предполагают наличие в ПЛИС двух разных элементов для вычисления логических функций и коммутирования связей, а также синтез из них комбинированного элемента.

С целью унификации архитектуры ПЛИС предлагается универсальная реализация, которая конфигурируется на заданный пользователем элемент. Такой подход является дальнейшим развитием направления программируемой логики, которая начиналась с настройки функций, дополненной затем настройкой связей.

Дополняем эти три этапа настройкой варианта кодирования переменных или связей.

Элементарный элемент 1-LUT [1–3] (конфигурируемый логический элемент на одну переменную) можно представить в виде выражения (2.33):

$$f_{1-LUT}(d_0, d_1, x) = d_0 \bar{x} \vee d_1 x, \tag{2.33}$$

где x – переменная, d_0, d_1 – конфигурационные биты для настройки логической функции $f(x)$ одной переменной x.

Представление элементарного мультиплексора маршрутизации на две связи 2-RMu (Routing Multiplexer) [1], использующего унитарный код «Одна единица» (One-Hot encoding) показывает выражение (2.34):

$$f_{2-RM_u}(d_0, d_1, x_2, x_1) = d_0 x_1 \vee d_1 x_2, \tag{2.34}$$

где d_0, d_1 – связи, x_2, x_1 – конфигурационные данные.

Представление элементарного мультиплексора маршрутизации на две связи 2-RMb (Routing Multiplexer) [1], использующего бинарный код показывает выражение (2.35):

$$f_{2-RM_b}(d_0, d_1, x) = d_0 \bar{x} \vee d_1 x, \tag{2.35}$$

где x – конфигурационные данные, d_0, d_1 – связи.

Известный элемент 1-LUT (2.33) может использоваться и для подключения одной из двух связей (connect1/connect2), выражение (2.36):

$$f_{1-LUT-2-RM}(d_0, d_1, x) = d_0 \bar{x} \vee d_1 x, \tag{2.36}$$

где x – конфигурационный бит, d_0, d_1 – связи.

Выражение (2.34) может быть использовано также для вычисления логической функции $F(x2x1)$ ($x2x1$ – унитарный код, One-Cold encoding, 01 – переменная x равна 0, 10 – переменная x равна 1), выражение (2.37):

$$f_{2-RM_u-LUT}(d_0, d_1, x_2, x_1) = d_0 x_1 \vee d_1 x_2, \tag{2.37}$$

где d_0, d_1 – конфигурационные данные, x_2, x_1 – унитарный код.

Объединяя эти подходы, предложим универсальный элемент, выражение (2.38):

$$f_{1-LUT,2-RM}(d_0, d_1, x_2, x_1, u, b), \tag{2.38}$$

где u – настройка на унитарный код, b – настройка на бинарный код.

Применяя (2.38) с учетом (2.33) – (2.37) получаем выражение (2.39):

$$\begin{gathered} f_{1-GCE(1-LUT,2-RM)}[d_0(connect1), d_1(connect2), x_2, x_1, u, b] = \\ = \{[d_0(connect1) \,\&\, \bar{x}_2 \vee d_1(connect2) \,\&\, x_2] \,\&\, b \vee \\ \vee [d_0(connect1) \,\&\, x_2 \vee d_1(connect2) \,\&\, x_1] \,\&\, u\}, \\ u, b \in \{0,1\}; u \neq b. \end{gathered} \tag{2.39}$$

В случае b=1 u=0 реализуется либо бинарный 2-RMb (2.35) либо1-LUT (2.33), при этом используется в качестве переменной вход x2, выражение (2.40):

$$\begin{gathered} f_{1-GCE(1-LUT,2-RM)}[d_0(connect1), d_1(connect2), x_2, x_1, 0, 1] = \\ = d_0(connect1) \,\&\, \bar{x}_2 \vee d_1(connect2) \,\&\, x_2. \end{gathered} \tag{2.40}$$

В случае b=0 u=1 реализуется либо унитарный 2-RMu (2.34) либо унитарный 1-LUT (2.37) получаем выражение (2.41):

$$\begin{gathered} f_{1-GCE(1-LUT,2-RM)}[d_0(connect1), d_1(connect2), x_2, x_1, 1, 0] = \\ = d_0(connect1) \,\&\, x_2 \vee d_1(connect2) \,\&\, x_1. \end{gathered} \tag{2.41}$$

В свою очередь (2.40) и (2.41) используются для коммутации связей или вычисления логической функции. Синтез более сложных элементов из описанных элементарных рассмотрен в главе 3.

2.7 Выводы по главе 2

1) Известные модели позиционного элемента (использующие позиционный бинарной код для переменных вычисляемой логической функции или кодирования связей в коммутаторе) могут быть представлены в линейном виде без детализации структуры и в древовидном с детализацией. В последнем случае возможны многоуровневое и одноуровневое деревья.

2) Известные унитарные модели элементов – коммутаторов (мультиплексоров связей) представляют собой всегда одноуровневое дерево и не используются для вычисления логических функций большого числа переменных в связи с очень большим объемом конфигурационной памяти.

3) При использовании многокоординатного унитарного кода увеличивается временная задержка, но снижается конфигурационная память, однако по длине кода выигрывает вариант с комбинированным кодированием унитарного и позиционного кода.

4) Предлагаемая модель элемента, как для вычисления заданной логической функции, так и для коммутации заданных межсоединений, в отличие от известной, комбинирует унитарные и позиционные блоки в одном устройстве. Комбинирование может быть в двух вариантах, в зависимости от того, какие блоки находятся слева и справа, считая, что выход находится справа.

5) Существо нового метода и алгоритма синтеза состоит в том, чтобы в зависимости от варианта модели, а) построить заданное количество унитарных и позиционных блоков, а затем б) соединить их для реализации или логической функции или коммутатора. После этого осуществляется в) настройка на

заданную функцию или межсоединение, используя конфигурационную память, и в заключении производится д) подключение переменных или межсоединений.

6) Новая модель и метод ориентированы на минимизацию временной задержки при вычислении логической функции при ограничениях на аппаратурные затраты либо на минимизацию объема конфигурационной памяти коммутатора при ограничениях на временную задержку.

7) Предлагаемый универсальный элемент с выбором варианта кодирования позволяет унифицировать архитектуру ПЛИС и настраивать их на реализацию либо известных элементов, либо предлагаемых элементы.

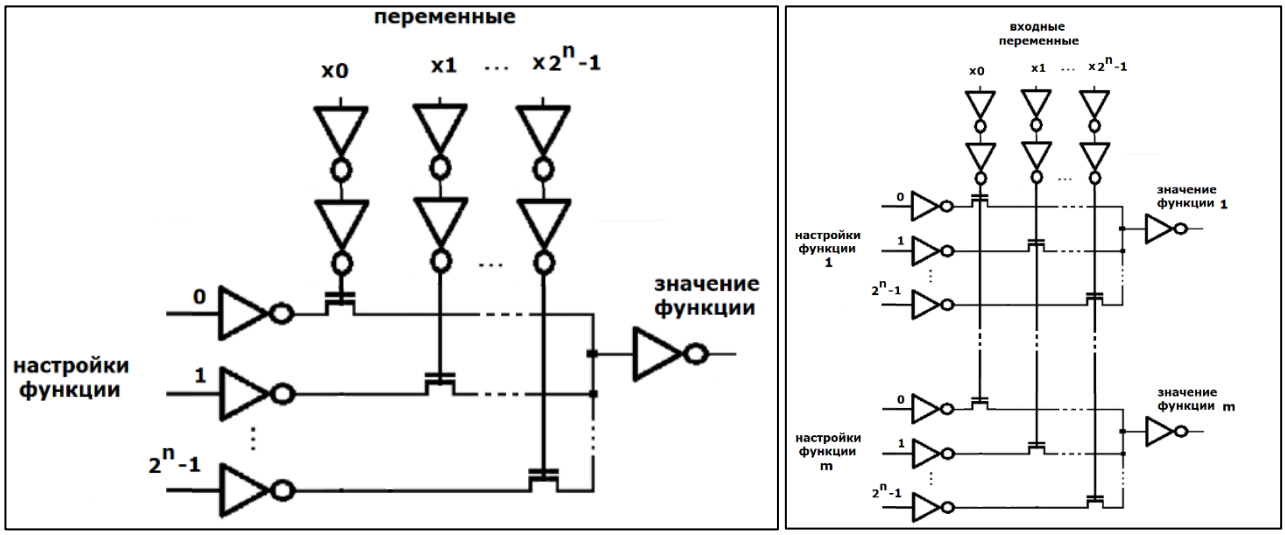
8) Для исследования работоспособности, определения свойств и характеристик, предложенных в виде математических выражений элементов, необходимо разработать их электрические схемы.

ГЛАВА 3. РАЗРАБОТКА СХЕМ ЭЛЕКТРИЧЕСКИХ ФУНКЦИОНАЛЬНЫХ ПРЕДЛАГАЕМЫХ ЭЛЕМЕНТОВ С ИСПОЛЬЗОВАНИЕМ КОМБИНИРОВАННОГО КОДИРОВАНИЯ

3.1 Разработка схем электрических функциональных различных предлагаемых вариантов логических элементов LUT с использованием унитарного кодирования

Предлагаемый 2^n-LUT на основе унитарного кода n переменных для одной функции в позиционном коде изображен на рисунке 3.1,а.

Такой элемент достаточно просто масштабируется для вычисления нескольких функций от одних и тех же переменных – рисунок 3.1,б.



а) б)

Рисунок 3.1 – LUT на основе унитарного кода: а) для вычисления одной функции; б) для вычисления m функций.

Пары инверторов по входам переменных необходимы для восстановления сигнала, уровень которого снижается после прохождения матриц коммутаций. На рисунке 3.1 функция представляется в позиционном коде. Пример вычисления логической функции в унитарном LUT на две переменные показан на рисунке 3.2.

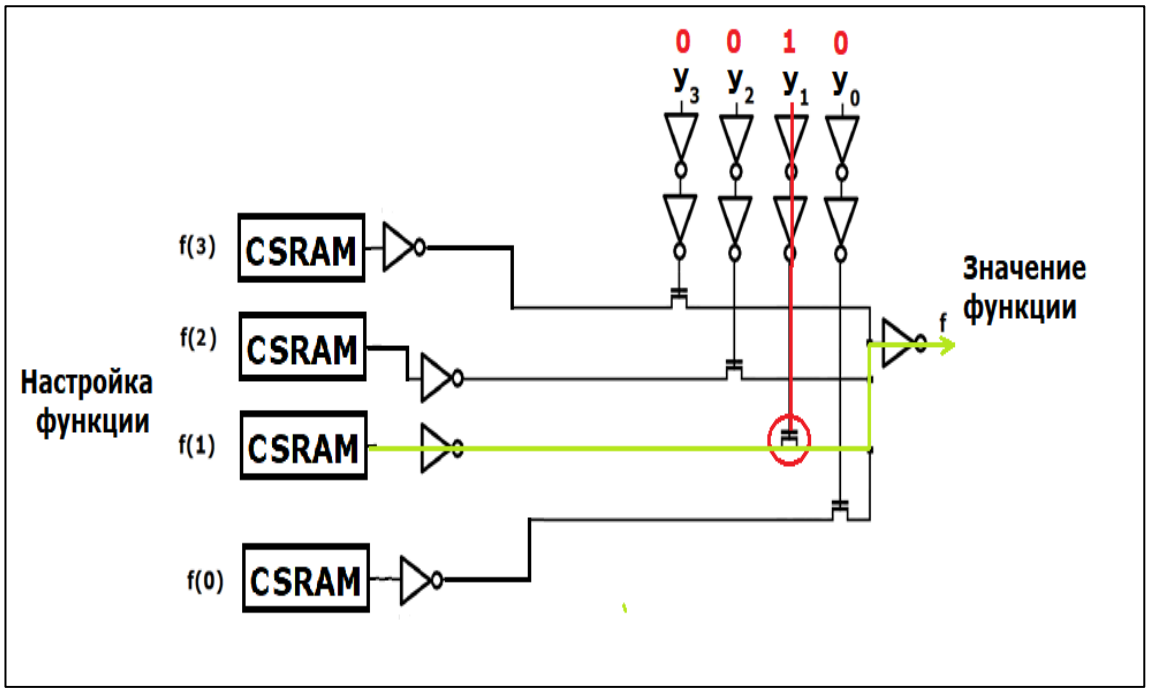


Рисунок 3.2 – Пример унитарного 2-LUT

При вычислении логической функции в обычном LUT на две переменные с позиционным кодированием схема выглядит как на рисунке 3.3.

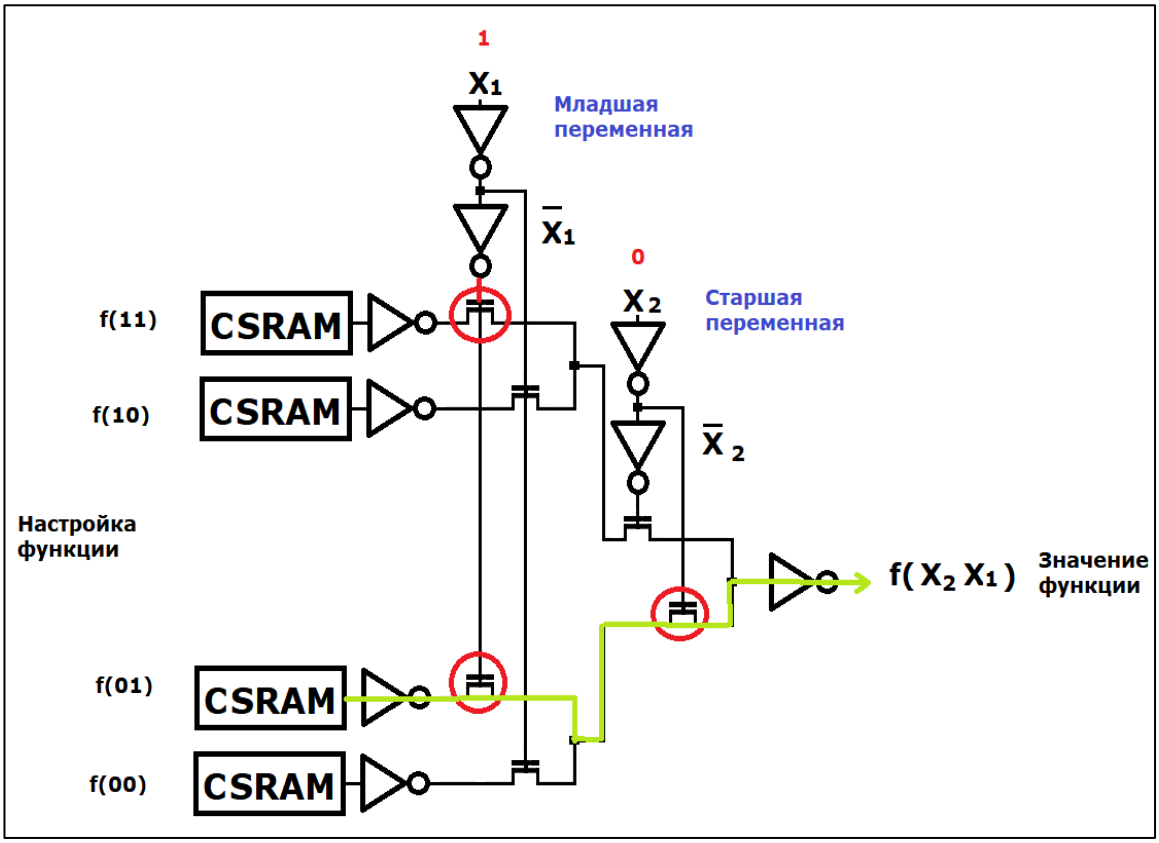
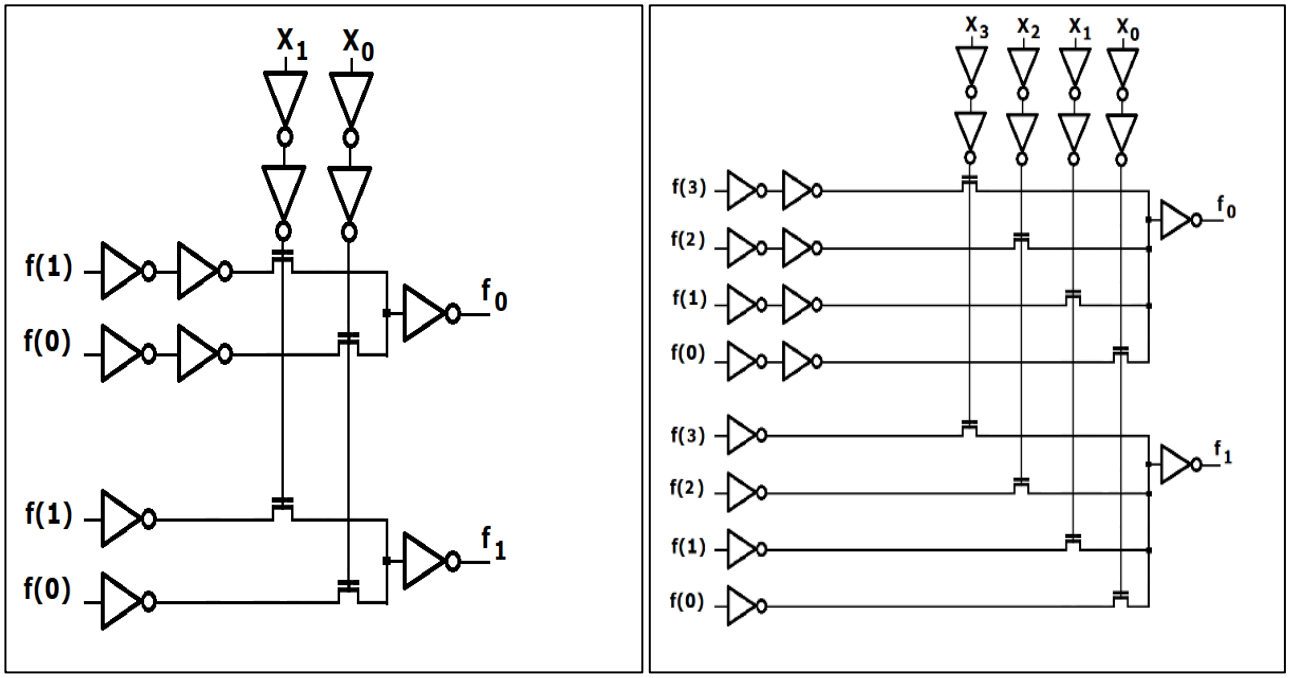


Рисунок 3.3 – Пример позиционного (бинарного) 2-LUT

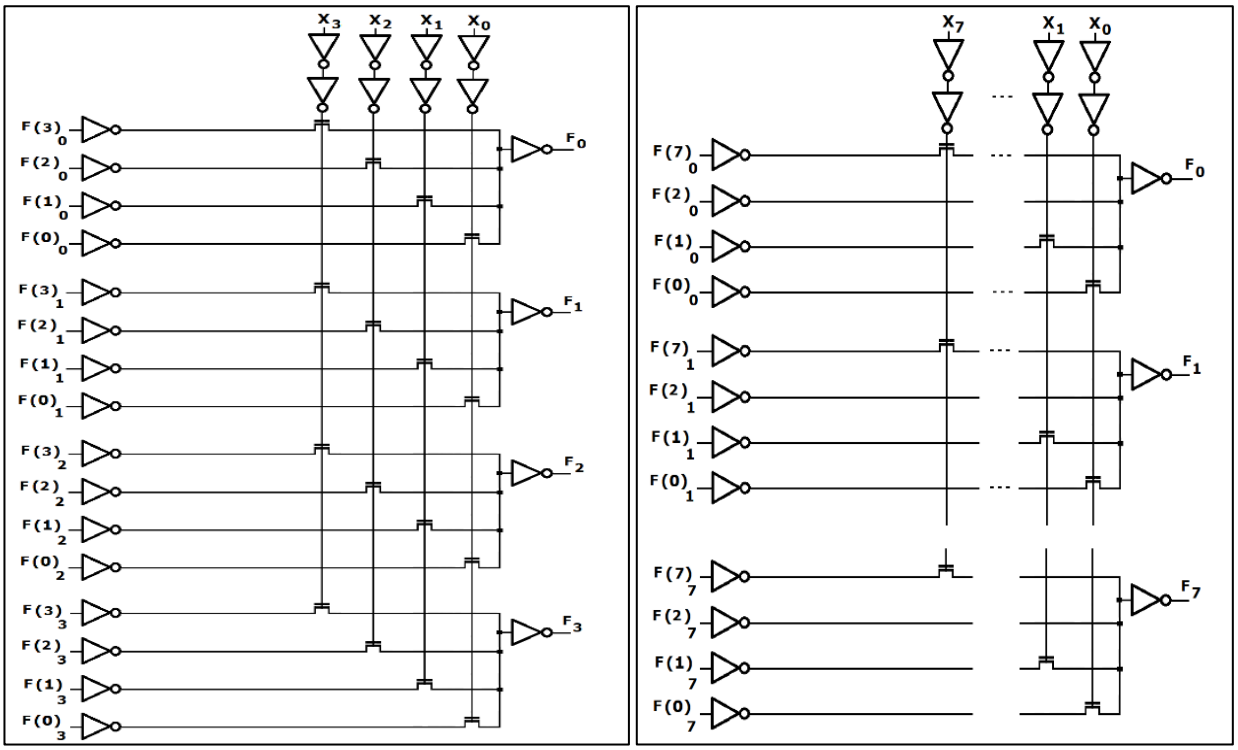
Для формирования одной функции рисунке 3.1 а) в унитарном коде предлагается использование второй подсхемы с соответствующей инверсной настройкой. Схемы с инверторами по входам конфигурации для n=1,2 изображены на рисунке 3.4, вместо настройки d_i указано значение функций на соответствующем наборе [99].



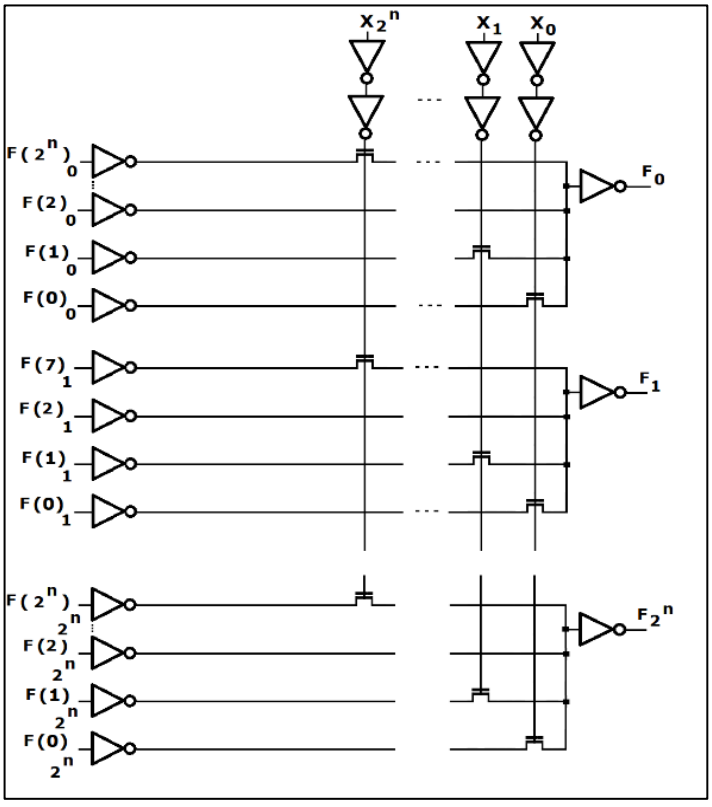
а) б)

Рисунок 3.4 – LUT с унитарным кодированием входной переменной и функции f: а) на одну переменную; б) на две переменные

Можно также удалить вторые входы конфигурации, соединяя выходы первой группы инверторов с нижним деревом. Для реализации системы функций из m функций необходимо уже $2^m - 1$ дополнительных подсхем. Предлагаемый LUT с унитарным кодированием двух входных переменных и двух функций, трёх входных переменных и трёх функций, на n входных переменных и n функций [99] представлен, на рисунке 3.5.



а) б)



в)

Рисунок 3.5 – LUT с унитарным кодированием входных переменных и функций: а) на две входных переменных и две функции; б) на три входных переменных и три функции; в) на *n* входных переменных и *n* функций

3.2 Примеры настройки предлагаемых логических элементов LUT_{uc} на две переменные при реализации системы двух функций

Рассмотрим реализацию в унитарном коде двух функций двух переменных – дизъюнкцию и сложение по модулю 2 (исключающее ИЛИ) при унитарном кодировании переменных. Вначале получаем значения системы функций в позиционном коде (столбцы *Z1*, *Z2*). Далее переводим значения системы функций в унитарный код (столбцы *F*(*X*)).

Таблица истинности двух функций от двух входных переменных при унитарном кодировании входов и выходов представлена в таблице 3.1 [83].

Таблица 3.1 – Система двух функций от двух входных переменных при унитарном кодировании входов и выходов

X_3	X_2	X_1	X_0	Z1=OR	Z2=XOR	$F_0(X)$	$F_1(X)$	$F_2(X)$	$F_3(X)$	Примечание
0	0	0	**1**	0	0	1	0	0	0	F (0)
0	0	**1**	0	1	1	0	0	0	1	F (1)
0	**1**	0	0	1	1	0	0	0	1	F (2)
1	0	0	0	1	0	0	0	1	0	F (3)

Получим соответствующее выражение (3.1) с соответствующей настройкой функций, обозначенных $F_0 \dots F_3$:

$$\begin{cases} F_0(0001x_3\dots x_0) = \dfrac{0x_{F_0(3)}}{0x_{F_0(2)}} \vee \bullet; F_1(0001x_3\dots x_0) = \dfrac{0x_{F_1(3)}}{0x_{F_1(2)}} \vee \bullet; \\ \qquad\qquad\qquad\qquad \dfrac{0x_{F_0(1)}}{1x_{F_0(0)}} \qquad\qquad\qquad\qquad \dfrac{0x_{F_1(1)}}{0x_{F_1(0)}} \\ F_2(1000x_3\dots x_0) = \dfrac{0x_{F_2(3)}}{0x_{F_2(2)}} \vee \bullet; F_3(0110x_3\dots x_0) = \dfrac{0x_{F_3(3)}}{0x_{F_3(2)}} \vee \bullet. \\ \qquad\qquad\qquad\qquad \dfrac{0x_{F_2(1)}}{1x_{F_2(0)}} \qquad\qquad\qquad\qquad \dfrac{0x_{F_3(1)}}{1x_{F_3(0)}} \end{cases} \quad (3.1)$$

В таком случае настройки устройства на рисунке 3.4,а имеет вид, указанный на рисунке 3.6.

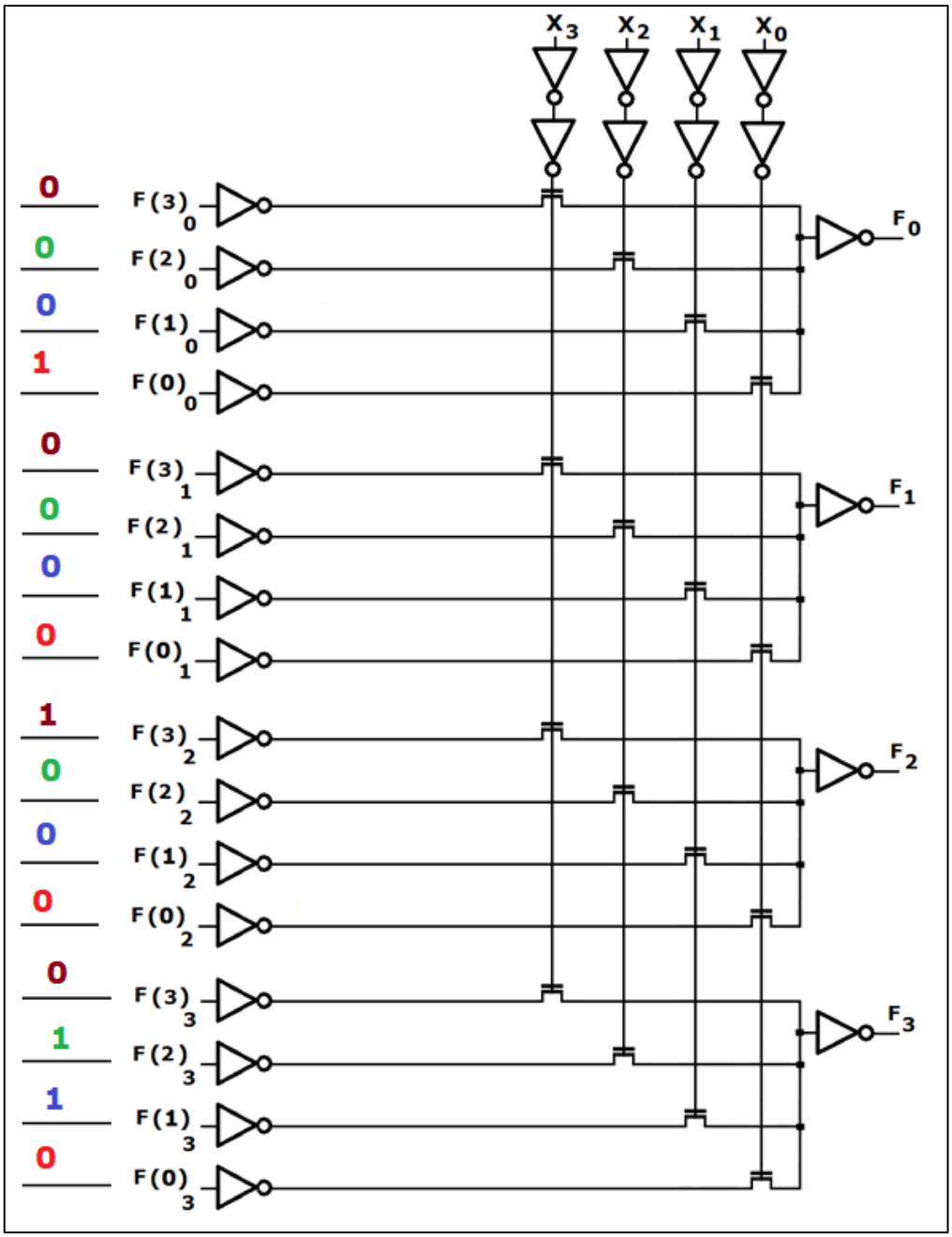


Рисунок 3.6 – Настройка предлагаемого LUT на две входных переменных и две функции при реализации системы из двух функций (OR, XOR)

Добавим еще одну функцию, увеличим длину кода. Пусть третья функция дополнительно к таблице 3.1 будет конъюнкцией, построим таблицу 3.2.

Таблица 3.2 – Система трёх функций от двух входных переменных при унитарном кодировании входов и выходов

X_3	X_2	X_1	X_0	Z_1 =OR	Z_2= XOR	Z_3= AND	$F_0(X)$	$F_1(X)$	$F_2(X)$	$F_3(X)$	$F_4(X)$	$F_5(X)$	$F_6(X)$	$F_7(X)$	
0	0	0	**1**	0	0	0	1	0	0	0	0	0	0	0	F (0)
0	0	**1**	0	1	1	0	0	0	0	0	0	0	1	0	F (1)
0	**1**	0	0	1	1	0	0	0	0	0	0	0	1	0	F (2)
1	0	0	0	1	0	1	0	0	1	0	0	1	0	0	F (3)

Таким образом, получаем соответствующее выражение (3.2):

$$\begin{cases} F_0(0001x_3\ldots x_0)=\dfrac{0x_{F_0(3)}}{0x_{F_0(2)}}\vee\bullet;\dfrac{0x_{F_0(1)}}{1x_{F_0(0)}}; F_1(0000x_3\ldots x_0)=\dfrac{0x_{F_1(3)}}{0x_{F_1(2)}}\vee\bullet;\dfrac{0x_{F_1(1)}}{0x_{F_1(0)}}; \\ F_2(0000x_3\ldots x_0)=\dfrac{0x_{F_2(3)}}{0x_{F_2(2)}}\vee\bullet;\dfrac{0x_{F_2(1)}}{1x_{F_2(0)}}; F_3(0000x_3\ldots x_0)=\dfrac{0x_{F_3(3)}}{0x_{F_3(2)}}\vee\bullet;\dfrac{0x_{F_3(1)}}{0x_{F_3(0)}}; \\ F_4(1000x_3\ldots x_0)=\dfrac{0x_{F_4(3)}}{0x_{F_4(2)}}\vee\bullet;\dfrac{0x_{F_4(1)}}{1x_{F_4(0)}}; F_5(1000x_3\ldots x_0)=\dfrac{1x_{F_5(3)}}{0x_{F_5(2)}}\vee\bullet;\dfrac{0x_{F_5(1)}}{0x_{F_5(0)}}; \\ F_6(0110x_3\ldots x_0)=\dfrac{0x_{F_6(3)}}{0x_{F_6(2)}}\vee\bullet;\dfrac{0x_{F_6(1)}}{1x_{F_6(0)}}; F_7(0000x_3\ldots x_0)=\dfrac{0x_{F_7(3)}}{0x_{F_7(2)}}\vee\bullet.\dfrac{0x_{F_7(1)}}{0x_{F_7(0)}}; \end{cases} \quad (3.2)$$

Настройки соответствующего устройства показаны на рисунке 3.7.

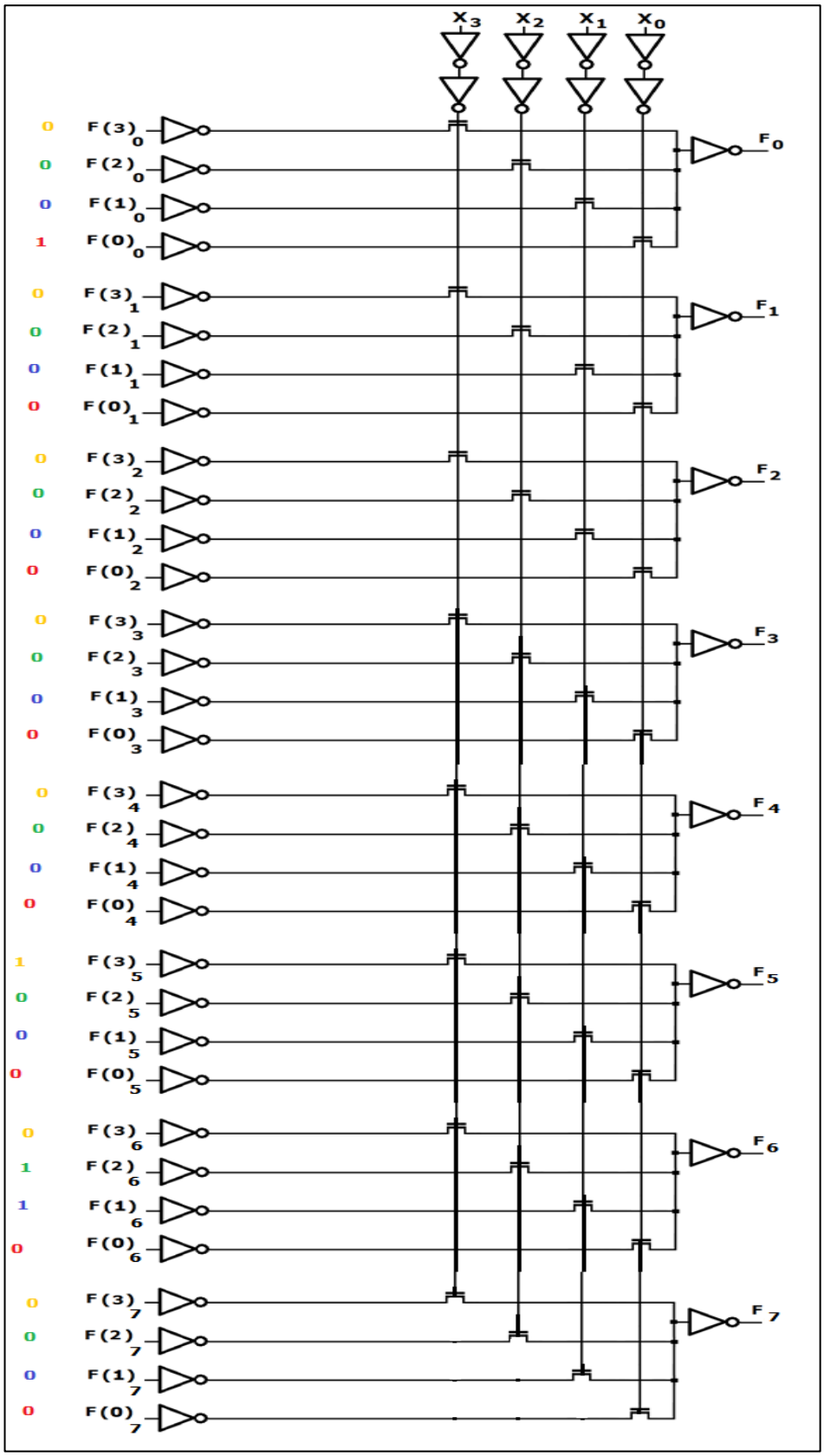


Рисунок 3.7 – LUT с унитарным кодированием двух входных переменных и трёх функций, настроенный на реализацию дизъюнкции, суммы по модулю два и конъюнкции

3.3 Пример электрических функциональных схем кодопреобразователей в унитарный код из позиционного и наоборот; построение сумматора унитарных кодов

Для реализации кодопреобразователя из унитарного в двоичный позиционный для n=2 предлагается выражение (3.3) с соответствующей настройкой функций, обозначенных a и b:

$$\begin{cases} a(1100x_3\ldots x_0) = \dfrac{1x_{a(3)}}{1x_{a(2)}} \vee \bullet, \\ \qquad\qquad\qquad \dfrac{0x_{a(1)}}{0x_{a(0)}} \\ b(1010x_3\ldots x_0) = \dfrac{1x_{b(3)}}{1x_{b(2)}} \vee \bullet. \\ \qquad\qquad\qquad \dfrac{0x_{b(1)}}{0x_{b(0)}} \end{cases} \quad (3.3)$$

Получаем схему функциональную изображенную на рисунке 3.8:

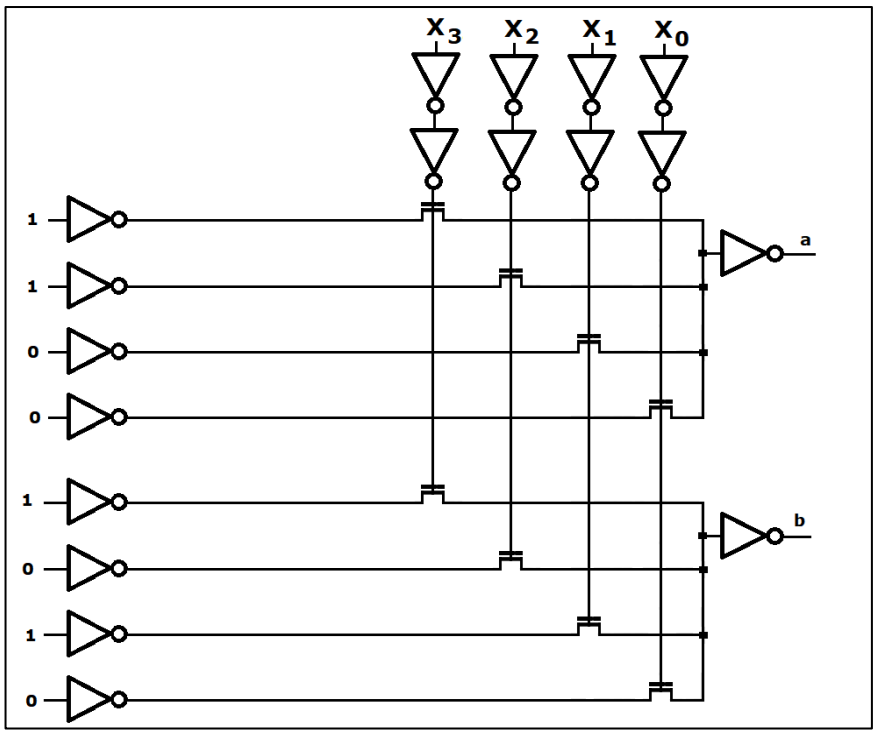


Рисунок 3.8 – Кодопреобразователь из унитарного в двоичный позиционный для n=2

Это относительно просто и быстро. Обратное преобразование (переменные обозначены *w*, функции обозначены *x*) выполняется гораздо сложнее, выражение (3.4):

$$x_j(0...0w_n...w_1) = \&_{i=1}^{n} w_{ji}^{\sigma(i,j)};\ j = 0...2^n - 1. \qquad (3.4)$$

Для трёх переменных (обозначены *a*,*b*,*c*) и трёх функций (обозначены *x*) конкретизируем выражение (3.4), получим, выражение (3.5):

$$x_j(000adc) = \&_{i=1}^{3} x_{ji}^{\sigma(i,j)};\ j = 0...7. \qquad (3.5)$$

Соответствующая схема приведена на рисунке 3.9.

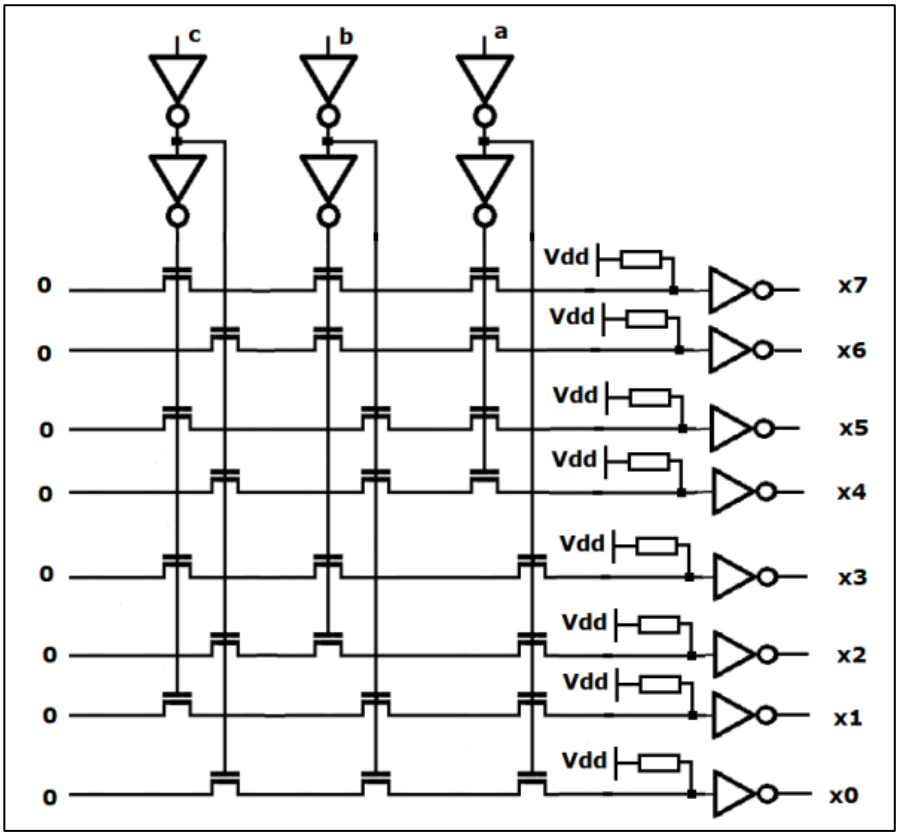


Рисунок 3.9 – Кодопреобразователь из двоичного позиционного кода в унитарный для *n*=3

Для исключения обрыва цепи коммутации нуля используются подтягивающие резисторы (Pull Up). Таким образом, это преобразование выполняется сложно (по числу транзисторов) и относительно медленно.

Поэтому вычисления лучше проводить в унитарном коде, а преобразовывать в двоичный на последнем этапе, если это необходимо.

При комбинировании разных унитарных кодов возникает задача объединения их в один общий код, при этом необходимо специальное суммирование: (0=0+0); (1=1+0; 0+1); (2=0+2; 2+0; 1+1); (3=0+3; 3+0; 1+2; 2+1); (4=1+3; 3+1;2+2); (5=2+3; 3+2); (6=3+3).

Получаем рисунок 3.10.

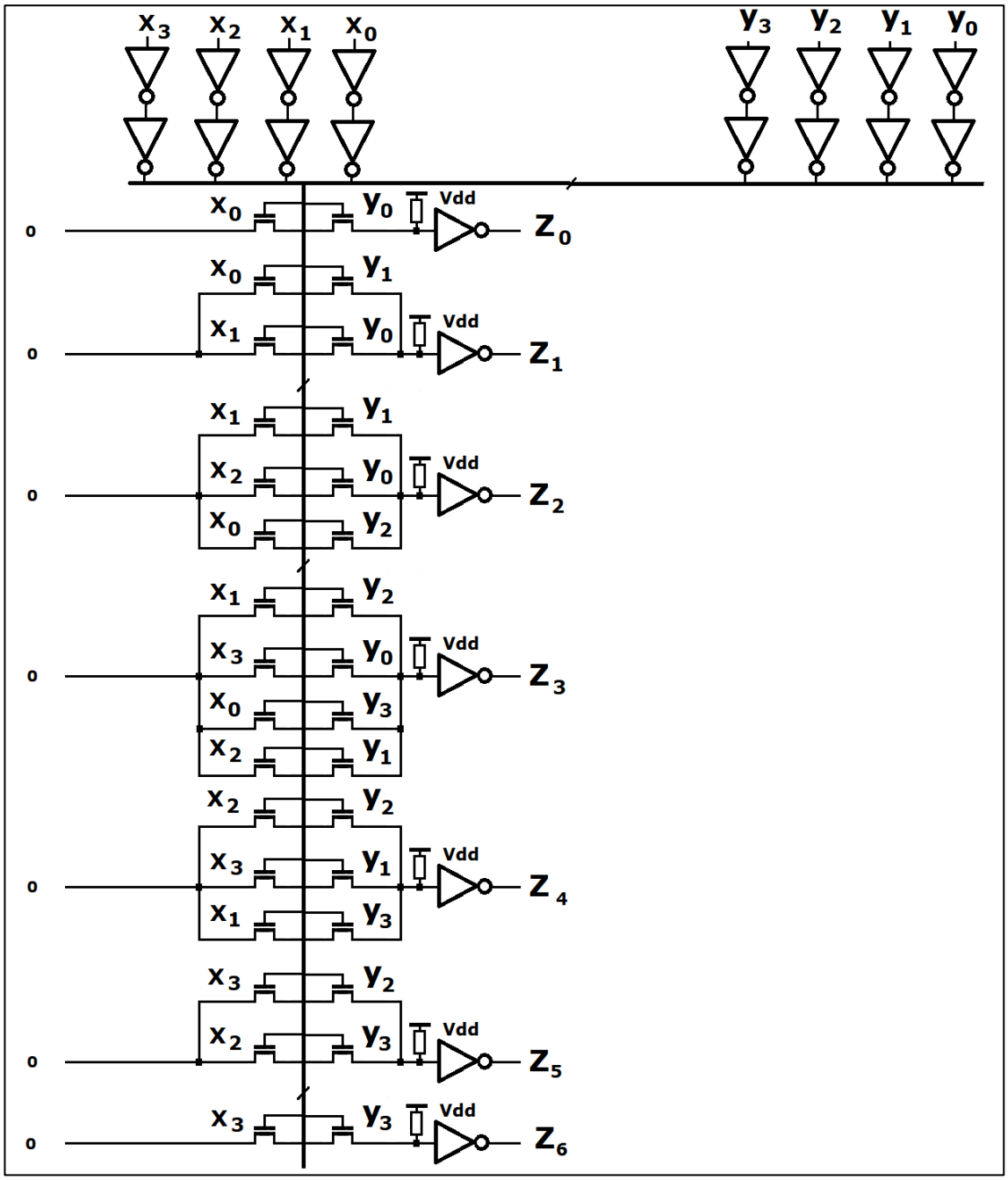


Рисунок 3.10 – Сумматор двух четырехразрядных унитарных кодов

3.4 Пример реализации логических функций автоматов в ПЛИС типа FPGA с использованием комбинированного кодирования

Предлагаемый метод предполагает разбиение входных переменных на несколько подвекторов с целью уменьшения количества разрядов. Например, при реализации конечного автомата целесообразно взять в качестве первого вектора – вектор состояния автомата, а в качестве второго – вектор входных переменных. Поскольку такой подход снижает быстродействие, то необходима оценка такого снижения, если оно допустимо, то возможно также комбинирование унитарного кодирования с позиционным с целью снижения аппаратурных затрат.

Рассмотрим пример комбинированного кодирования. Пусть задан автомат-распознаватель последовательности 0132 с учетом допустимости только соседних наборов. При получении заданной последовательности формируется z_1=1, при любом нарушении z_2=1. Функции переходов-выходов автомата приведены в таблице 3.3.

Таблица 3.3 – Таблица переходов-выходов автомата-распознавателя последовательности 0132 с позиционным бинарным кодированием состояний

Код состояния	Входной набор *ab*				Функции переходов и выходов
$y_2\,y_1\,(t)$	00	01	11	10	
00	$\frac{00}{00}$	$\frac{01}{00}$	~	$\frac{00}{10}$	
01	$\frac{11}{10}$	$\frac{01}{00}$	$\frac{11}{00}$	~	
11	$\frac{11}{10}$	$\frac{11}{10}$	$\frac{11}{00}$	$\frac{11}{01}$	$\frac{y_2\,y_1\,(t+1)}{z_2 z_1}$

При таком кодировании используются четыре классических LUT по числу функций размерностью четыре (на четыре переменные), выражение (3.6):

$$y_2(t+1)(d_{y_2(t+1).0}\ldots d_{y_2(t+1).15}y_2y_1ab)=\left.\begin{array}{r}d_{y_2(t+1).15}y_2y_1ab\\ d_{y_2(t+1).14}y_2y_1a\bar{b}\\ \ldots\ldots\ldots\ldots\ldots\\ d_{y_2(t+1).1}\bar{y}_2\bar{y}_1\bar{a}b\\ d_{y_2(t+1).0}\bar{y}_2\bar{y}_1\bar{a}\bar{b}\end{array}\right\}\vee\bullet;\ y_1(t+1)(d_{y_1(t+1).0}\ldots d_{y_1(t+1).15}y_2y_1ab)=\left.\begin{array}{r}d_{y_1(t+1).15}y_2y_1ab\\ d_{y_1(t+1).14}y_2y_1a\bar{b}\\ \ldots\ldots\ldots\ldots\ldots\\ d_{y_1(t+1).1}\bar{y}_2\bar{y}_1\bar{a}b\\ d_{y_1(t+1).0}\bar{y}_2\bar{y}_1\bar{a}\bar{b}\end{array}\right\}\vee\bullet;$$

$$z_1(d_{z_1.0}\ldots d_{z_1.15}y_2y_1ab)=\left.\begin{array}{r}d_{z_1.15}y_2y_1ab\\ d_{z_1.14}y_2y_1a\bar{b}\\ \ldots\ldots\ldots\ldots\ldots\\ d_{z_1.1}\bar{y}_2\bar{y}_1\bar{a}b\\ d_{z_1.0}\bar{y}_2\bar{y}_1\bar{a}\bar{b}\end{array}\right\}\vee\bullet;\ z_2(d_{z_2.0}\ldots d_{z_1.15}y_2y_1ab)=\left.\begin{array}{r}d_{z_2.15}y_2y_1ab\\ d_{z_2.14}y_2y_1a\bar{b}\\ \ldots\ldots\ldots\ldots\ldots\\ d_{z_2.1}\bar{y}_2\bar{y}_1\bar{a}b\\ d_{z_2.0}\bar{y}_2\bar{y}_1\bar{a}\bar{b}\end{array}\right\}\vee\bullet. \quad (3.6)$$

Унитарное кодирование четырех функций от четырех переменных требует 16 бит входного вектора и столько же бит выходного. Выполним унитарное кодирование отдельно по строкам и по столбцам. Для этого необходимо 8 бит (поскольку у нас три строки, используем 7 бит, третью строку в таблице 3.3 будем считать закодированной позиционным кодом 10). Получим таблицу 3.4.

Таблица 3.4 – Таблица переходов-выходов автомата-распознавателя последовательности 0132 с соответствующим унитарным кодированием отдельно по строкам и по столбцам

Код состояния унитарный	Входной набор унитарный $x_3\,x_2\,x_1\,x_0$				Функции переходов и выходов унитарные
$y_2\,y_1\,y_0\,(t)$	0001	0010	1000	0100	
001	$\frac{001}{001}$	$\frac{010}{001}$	~	$\frac{001}{100}$	
010	$\frac{100}{100}$	$\frac{010}{001}$	$\frac{100}{001}$	~	
100	$\frac{100}{100}$	$\frac{100}{100}$	$\frac{100}{001}$	$\frac{100}{010}$	$\frac{y_2\,y_1\,y_0\,(t+1)}{z_2z_1z_0}$

Предлагается использовать двухкоординатный унитарный вычислитель системы логических функций для реализации автомата 0132 – рисунок 3.11 и выражение (3.7).

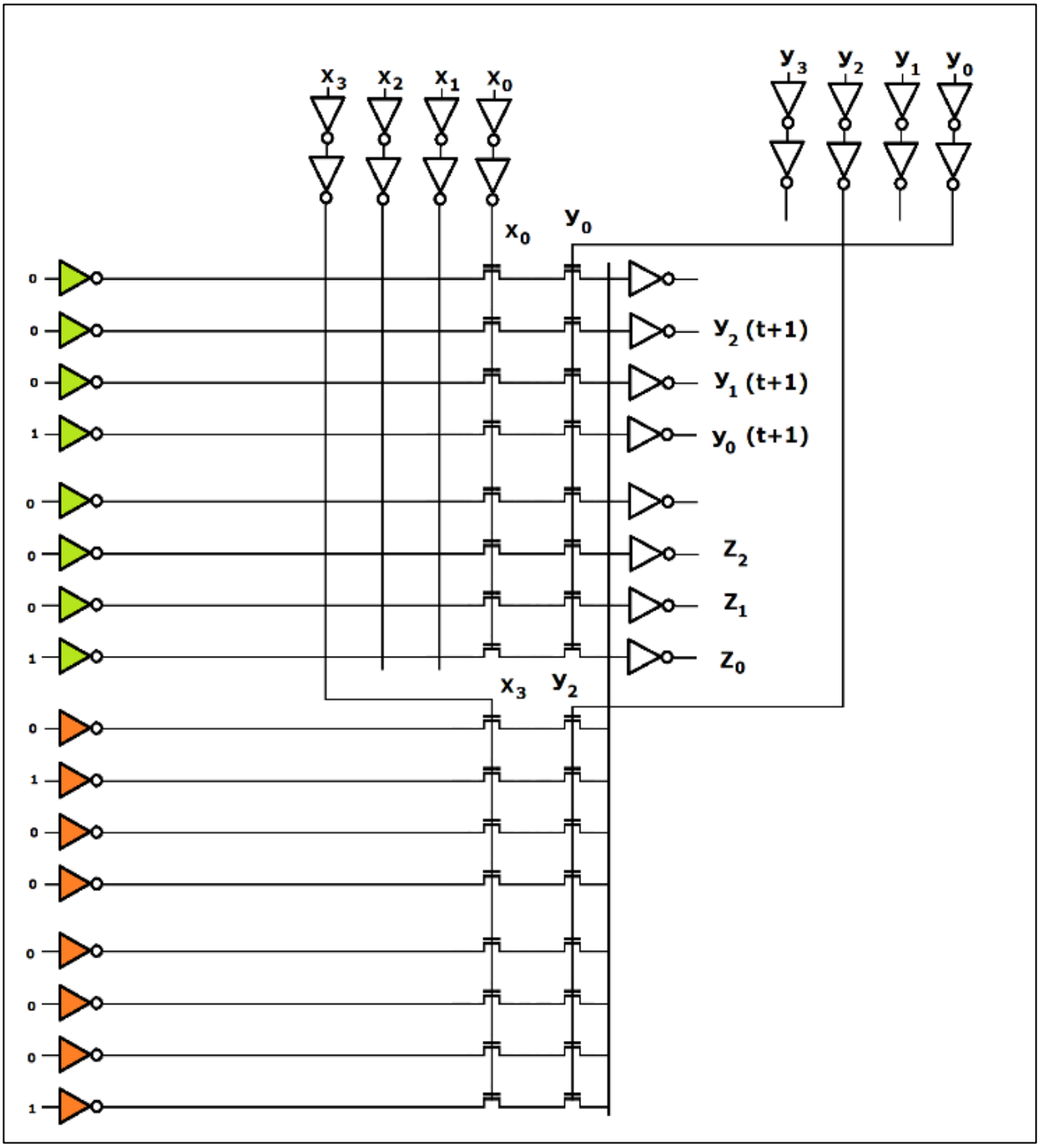


Рисунок 3.11 – Двухкоординатный унитарный вычислитель системы логических функций, соответствующих

$$y_2(t+1)(d_{y_2(t+1).0}\ldots d_{y_2(t+1).11}y_2y_1y_0x_3x_2x_1x_0)=\left.\begin{matrix}d_{y_2(t+1).11}y_2x_3\\ d_{y_2(t+1).10}y_2x_2\\ \ldots\ldots\ldots\ldots\ldots\\ d_{y_1(t+1).1}y_0x_1\\ d_{y_1(t+1).0}y_0x_0\end{matrix}\right\}\vee\bullet;$$

$$y_1(t+1)(d_{y_1(t+1).0}\ldots d_{y_1(t+1).11}y_2y_1y_0x_3x_2x_1x_0)=\left.\begin{matrix}d_{y_1(t+1).11}y_2x_3\\ d_{y_1(t+1).10}y_2x_2\\ \ldots\ldots\ldots\ldots\ldots\\ d_{y_1(t+1).1}y_0x_1\\ d_{y_1(t+1).0}y_0x_0\end{matrix}\right\}\vee\bullet;$$

$$y_0(t+1)(d_{y_0(t+1).0}\ldots d_{y_0(t+1).11}y_2y_1y_0x_3x_2x_1x_0)=\left.\begin{matrix}d_{y_0(t+1).11}y_2x_3\\ d_{y_0(t+1).10}y_2x_2\\ \ldots\ldots\ldots\ldots\ldots\\ d_{y_0(t+1).1}y_0x_1\\ d_{y_0(t+1).0}y_0x_0\end{matrix}\right\}\vee\bullet;$$

$$z_2(d_{z_2.0}\ldots d_{z_1.11}y_2y_1y_0x_3x_2x_1x_0)=\left.\begin{matrix}d_{z_2.11}y_2x_3\\ d_{z2.10}y_2x_2\\ \ldots\ldots\ldots\ldots\ldots\\ d_{z_2.1}y_0x_1\\ d_{z_2.0}y_0x_0\end{matrix}\right\}\vee\bullet;$$

$$z_1(d_{z_1.0}\ldots d_{z_1.11}y_2y_1y_0x_3x_2x_1x_0)=\left.\begin{matrix}d_{z_1.11}y_2x_3\\ d_{z_1.10}y_2x_2\\ \ldots\ldots\ldots\ldots\ldots\\ d_{z_1.1}y_0x_1\\ d_{z_1.0}y_0x_0\end{matrix}\right\}\vee\bullet;$$

$$z_0(d_{z_0.0}\ldots d_{z_0.11}y_2y_1y_0x_3x_2x_1x_0)=\left.\begin{matrix}d_{z_0.11}y_2x_3\\ d_{z_0.10}y_2x_2\\ \ldots\ldots\ldots\ldots\ldots\\ d_{z_0.1}y_0x_1\\ d_{z_0.0}y_0x_0\end{matrix}\right\}\vee\bullet. \qquad (3.7)$$

Таким образом, уменьшается длина слова с 2^{2+2} до $3+4$, но быстродействие несколько ухудшается: теперь в цепи не один коммутирующий транзистор, а два. Аналогично можно построить *s* координатное устройство, 4=<*s*>=2, что опять-таки связано с ограничениями Мида-Конвей [54].

С целью еще большего уменьшения длины слова комбинируем унитарный код по строкам (по состояниям) с позиционным по столбцам (по входным переменным). Получаем таблицу 3.5 и выражение 3.8.

Для нашего примера получим 5 бит. В этом случае для вычислений придётся использовать классические LUT, выбираемые унитарным кодом строки, при этом размерность их уменьшается до двух:

Таблица 3.5 – Таблица переходов-выходов автомата-распознавателя последовательности 0132 с унитарным кодированием только состояний

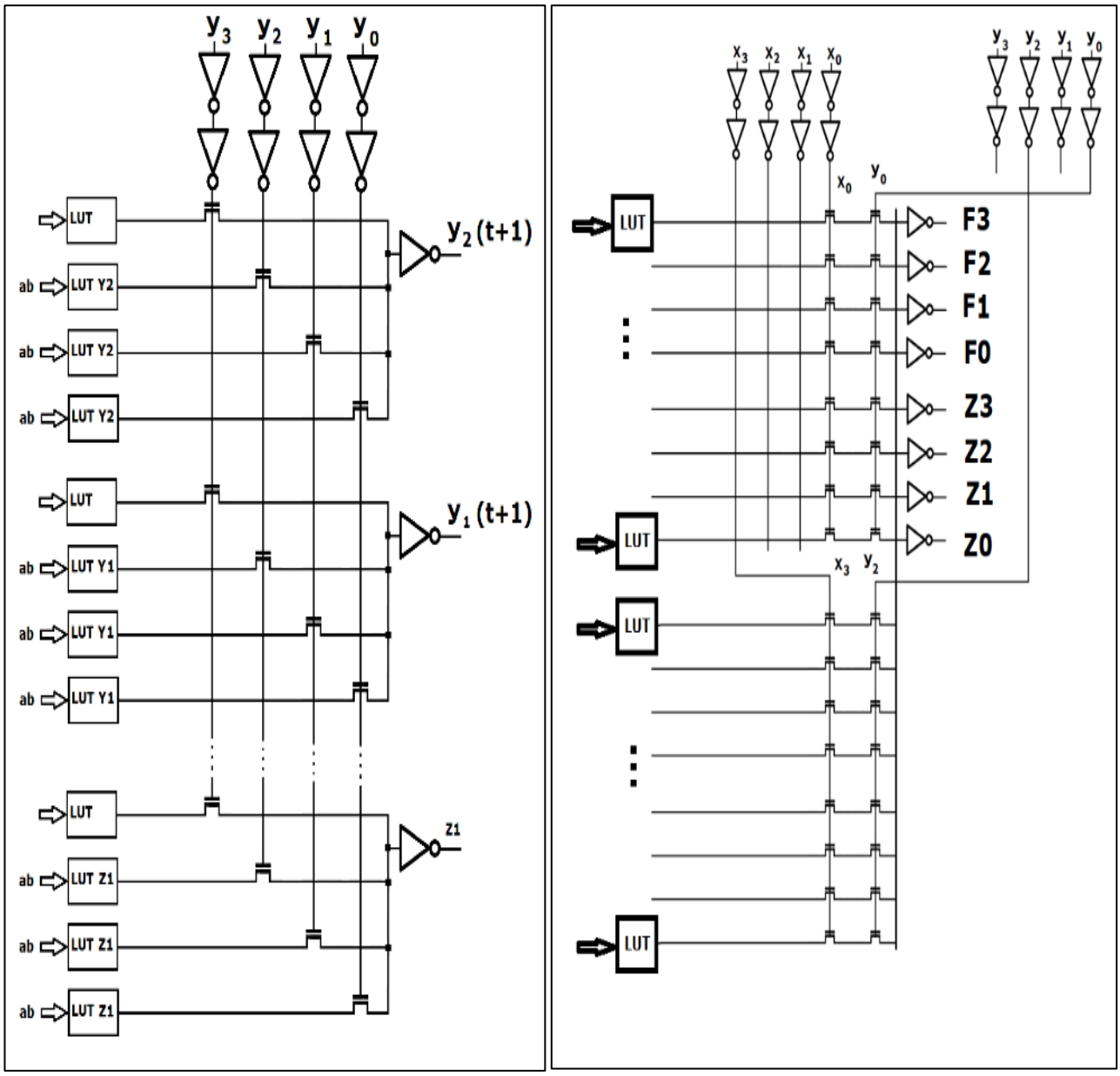
Код состояния унитарный	Входной набор *ab*				Функции переходов и выходов
$y_2\, y_1\, y_0\ (t)$	00	01	11	10	
001	$\frac{001}{00}$	$\frac{010}{00}$	~	$\frac{001}{10}$	
010	$\frac{100}{10}$	$\frac{010}{00}$	$\frac{100}{00}$	~	
100	$\frac{100}{10}$	$\frac{100}{10}$	$\frac{100}{00}$	$\frac{100}{01}$	$\frac{y_2\, y_1\, y_0}{(t+1)}$ $z_2 z_1$

Комбинированное вычисление пяти функций переходов и выходов автомата-распознавателя последовательности 0132 показано на рисунке 3.12. Верхний LUT в каждой группе не используется и зарезервирован для таблицы из четырех строк. Необходимо три LUT для каждой функции (а четвертый не использован для данного автомата) по числу строк в таблице переходов-выходов (рисунок 3.12,а). Абстрактный пример комбинирования LUT с двух координатным унитарным вычислением показан на рисунке 3.12,б.

$$y_2(t+1)(d_{y_2(t+1).y_2.0}\ldots d_{y_2(t+1).y_0.3}y_2y_1y_0ab)=\begin{Bmatrix}d_{y_2(t+1).y_2.3}y_2ab\\ d_{y_2(t+1).y_2.2}y_2a\bar{b}\\ d_{y_2(t+1).y_2.1}y_2\bar{a}b\\ d_{y_2(t+1).y_2.0}y_2\bar{a}\bar{b}\\ d_{y_2(t+1).y_1.3}y_1ab\\ d_{y_2(t+1).y_1.2}y_1a\bar{b}\\ d_{y_2(t+1).y_1.1}y_1\bar{a}b\\ d_{y_2(t+1).y_1.0}y_1\bar{a}\bar{b}\\ d_{y_2(t+1).y_0.3}y_0ab\\ d_{y_2(t+1).y_0.2}y_0a\bar{b}\\ d_{y_2(t+1).y_0.1}y_0\bar{a}b\\ d_{y_2(t+1).y_0.0}y_0\bar{a}\bar{b}\end{Bmatrix}\vee\bullet;\ y_1(t+1)(d_{y_1(t+1).y_2.0}\ldots d_{y_1(t+1).y_0.3}y_2y_1y_0ab)=\begin{Bmatrix}d_{y_1(t+1).y_2.3}y_2ab\\ d_{y_1(t+1).y_2.2}y_2a\bar{b}\\ d_{y_1(t+1).y_2.1}y_2\bar{a}b\\ d_{y_1(t+1).y_2.0}y_2\bar{a}\bar{b}\\ d_{y_1(t+1).y_1.3}y_1ab\\ d_{y_1(t+1).y_1.2}y_1a\bar{b}\\ d_{y_1(t+1).y_1.1}y_1\bar{a}b\\ d_{y_1(t+1).y_1.0}y_1\bar{a}\bar{b}\\ d_{y_1(t+1).y_0.3}y_0ab\\ d_{y_2(t+1).y_0.2}y_0a\bar{b}\\ d_{y_1(t+1).y_0.1}y_0\bar{a}b\\ d_{y_1(t+1).y_0.0}y_0\bar{a}\bar{b}\end{Bmatrix}\vee\bullet;$$

$$y_0(t+1)(d_{y_0(t+1).y_2.0}\ldots d_{y_0(t+1).y_0.3}y_2y_1y_0ab)=\begin{Bmatrix}d_{y_1(t+1).y_2.3}y_2ab\\ d_{y_1(t+1).y_2.2}y_2a\bar{b}\\ d_{y_1(t+1).y_2.1}y_2\bar{a}b\\ d_{y_1(t+1).y_2.0}y_2\bar{a}\bar{b}\\ d_{y_1(t+1).y_1.3}y_1ab\\ d_{y_1(t+1).y_1.2}y_1a\bar{b}\\ d_{y_1(t+1).y_1.1}y_1\bar{a}b\\ d_{y_1(t+1).y_1.0}y_1\bar{a}\bar{b}\\ d_{y_1(t+1).y_0.3}y_0ab\\ d_{y_2(t+1).y_0.2}y_0a\bar{b}\\ d_{y_1(t+1).y_0.1}y_0\bar{a}b\\ d_{y_1(t+1).y_0.0}y_0\bar{a}\bar{b}\end{Bmatrix}\vee\bullet;\ z_1(d_{z_1.y_2.0}\ldots d_{z_1.y_0.3}y_2y_1y_0ab)=\begin{Bmatrix}d_{z_1.y_2.3}y_2ab\\ d_{z_1.y_2.2}y_2a\bar{b}\\ d_{z_1.y_2.1}y_2\bar{a}b\\ d_{z_1.y_2.0}y_2\bar{a}\bar{b}\\ d_{z_1.y_1.3}y_1ab\\ d_{z_1.y_1.2}y_1a\bar{b}\\ d_{z_1.y_1.1}y_1\bar{a}b\\ d_{z_1.y_1.0}y_1\bar{a}\bar{b}\\ d_{z_1.y_0.3}y_0ab\\ d_{z_2.y_0.2}y_0a\bar{b}\\ d_{z_1.y_0.1}y_0\bar{a}b\\ d_{z_1.y_0.0}y_0\bar{a}\bar{b}\end{Bmatrix}\vee\bullet; \tag{3.8}$$

$$z_1(d_{z_2.y_2.0}\ldots d_{z_2.y_0.3}y_2y_1y_0ab)=\begin{Bmatrix}d_{z_2.y_2.3}y_2ab\\ d_{z_2.y_2.2}y_2a\bar{b}\\ d_{z_2.y_2.1}y_2\bar{a}b\\ d_{z_2.y_2.0}y_2\bar{a}\bar{b}\\ d_{z_2.y_1.3}y_1ab\\ d_{z_2.y_1.2}y_1a\bar{b}\\ d_{z_2.y_1.1}y_1\bar{a}b\\ d_{z_2.y_1.0}y_1\bar{a}\bar{b}\\ d_{z_2.y_0.3}y_0ab\\ d_{z_2.y_0.2}y_0a\bar{b}\\ d_{z_2.y_0.1}y_0\bar{a}b\\ d_{z_2.y_0.0}y_0\bar{a}\bar{b}\end{Bmatrix}\vee\bullet.$$



а) б)

Рисунок 3.12 – Комбинированное вычисление: а) функций автомата-распознавателя последовательности 0132 для унитарного вектора состояний из четырех бит; б) абстрактная системы функций с комбинированием LUT и двух координатного унитарного вычисления

Пример четвертого варианта комбинированного LUT с двухуровневым унитарным деревом показан на рисунке 3.13.

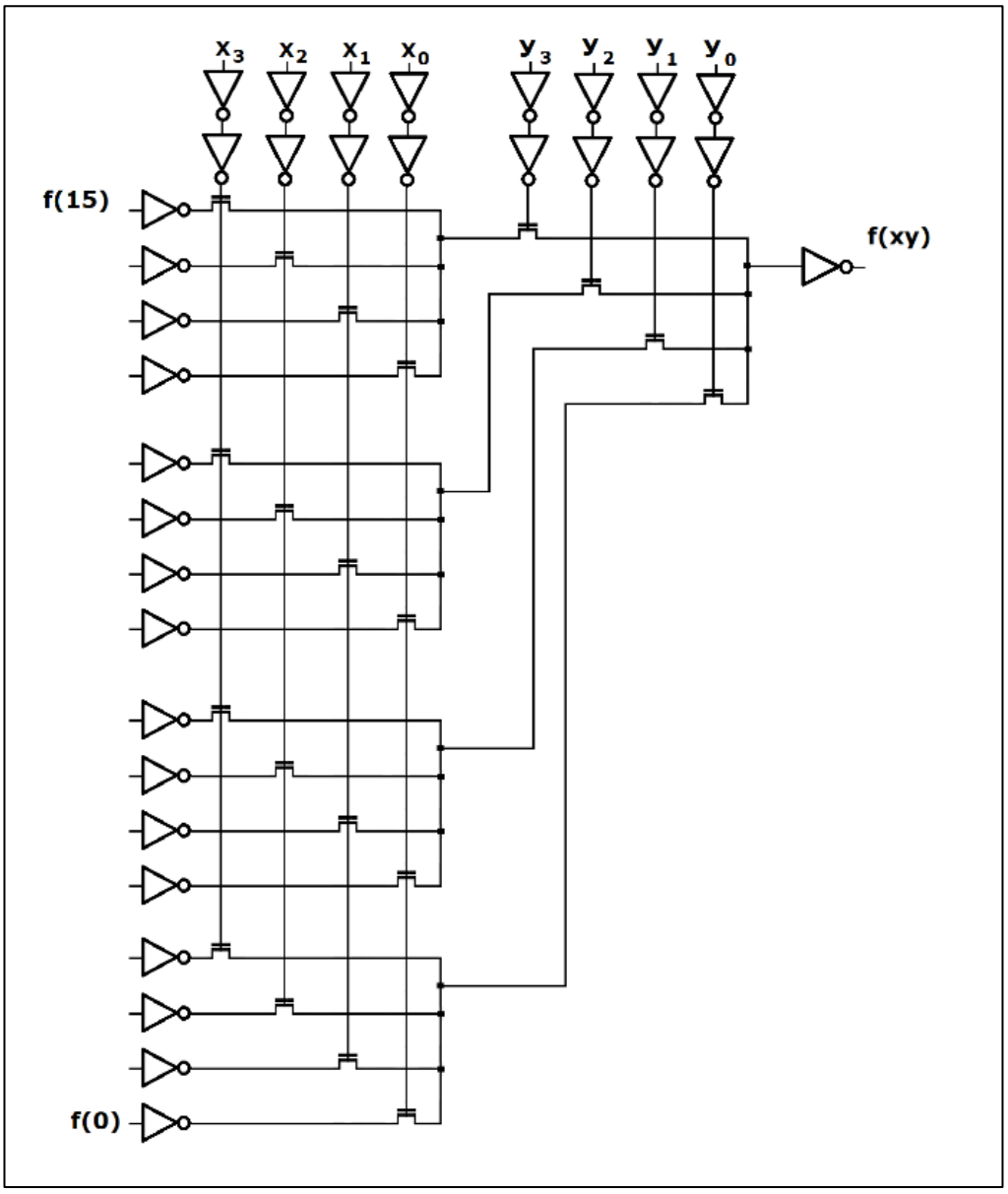
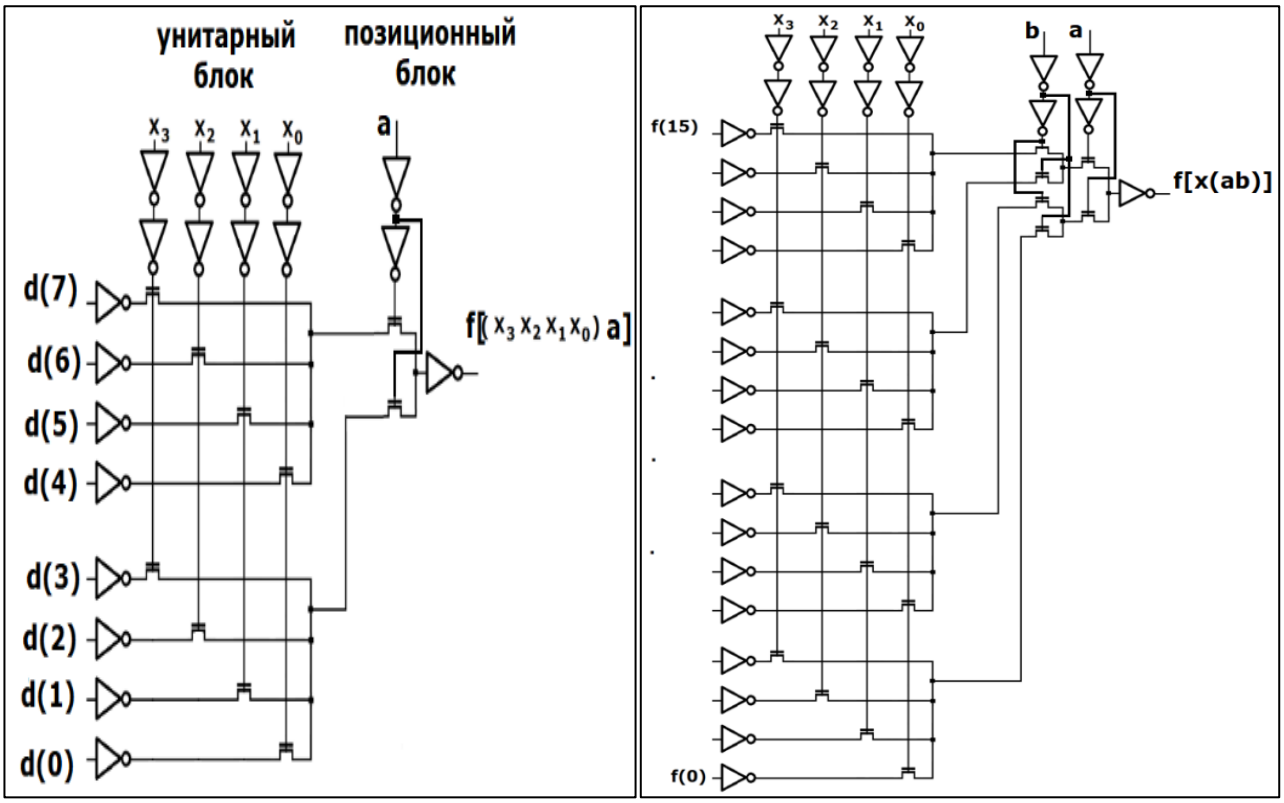


Рисунок 3.13 – Пример четвертого варианта комбинированного LUT с двухуровневым унитарным деревом

Пример пятого варианта комбинированного LUT с двухуровневым деревом, первый уровень – обычный LUT показан на рисунке 3.14.



а) б)

Рисунок 3.14 – Пример пятого варианта комбинированного LUT с двухуровневым деревом, первый уровень – обычный LUT: а) элемент на 3 переменные; б) элемент на 4 переменные

3.5 Синтез функциональных электрических схем по предлагаемому методу

Упрощенный элементарный мультиплексор 2-1 из двух передающих транзисторов, который используется в методе синтеза, показан на рисунке 3.15.

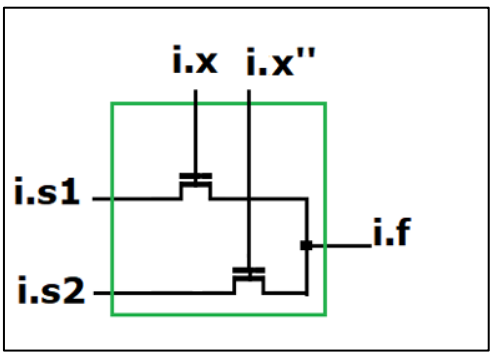


Рисунок 3.15 – Упрощенный элементарный мультиплексор 2-1 из двух передающих транзисторов без требования ортогональности *i.x*, *i.x*’’

Соединение таких упрощенных элементарных мультиплексоров 2-1 с целью синтеза известного мультиплексора-маршрутизатора на 2^{n_2} – унитарного блока связей показано на рисунке 3.16.

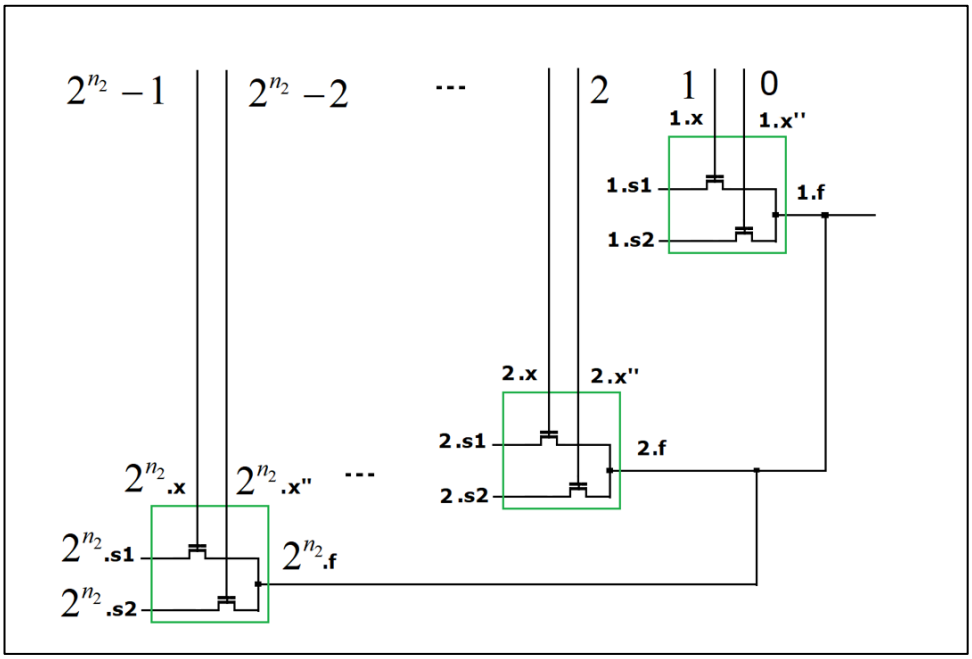


Рисунок 3.16 – Унитарный блок

Для синтеза блоков с бинарным кодированием (позиционный блок) используется элементарный мультиплексор 2-1 из двух передающих транзисторов *p*-МОП с заданной ортогональностью *j.k.x, j.k.x'* – рисунок 3.17.

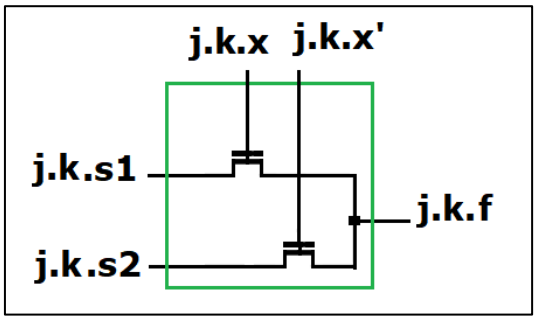


Рисунок 3.17 – Элементарный мультиплексор 2-1 из двух передающих транзисторов p-МОП с заданной ортогональностью *j.k.x, j.k.x'*

При этом возможен синтез по двум вариантам: позиционный блок многоуровневый показан на рисунке 3.18.

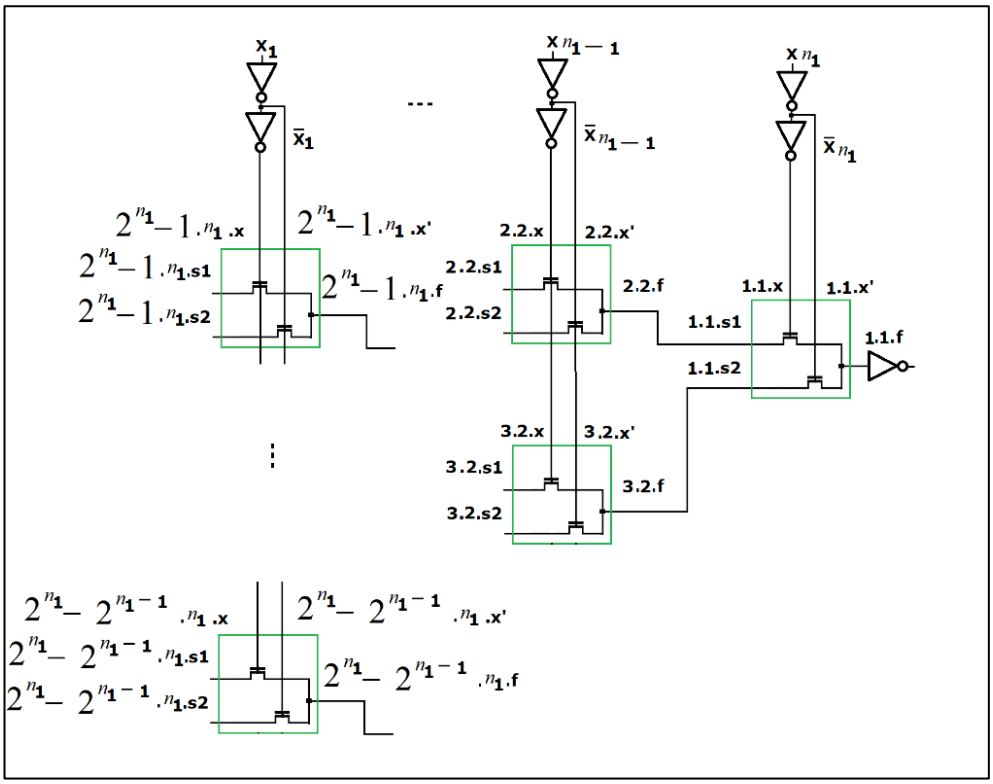


Рисунок 3.18 – Позиционный блок многоуровневый на n_1 переменных

По второму варианту строится позиционный блок одноуровневый – рисунок 3.19.

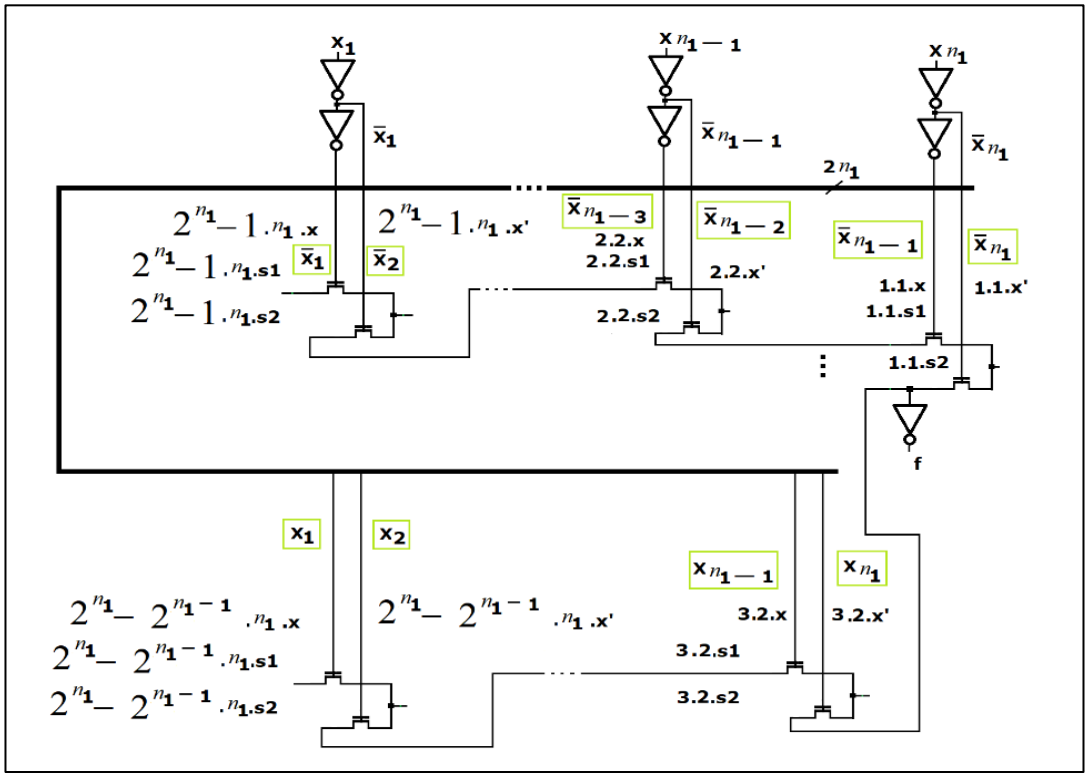


Рисунок 3.19 – Позиционный блок одноуровневый

Синтезированный комбинированный элемент показан на рисунке 3.20.

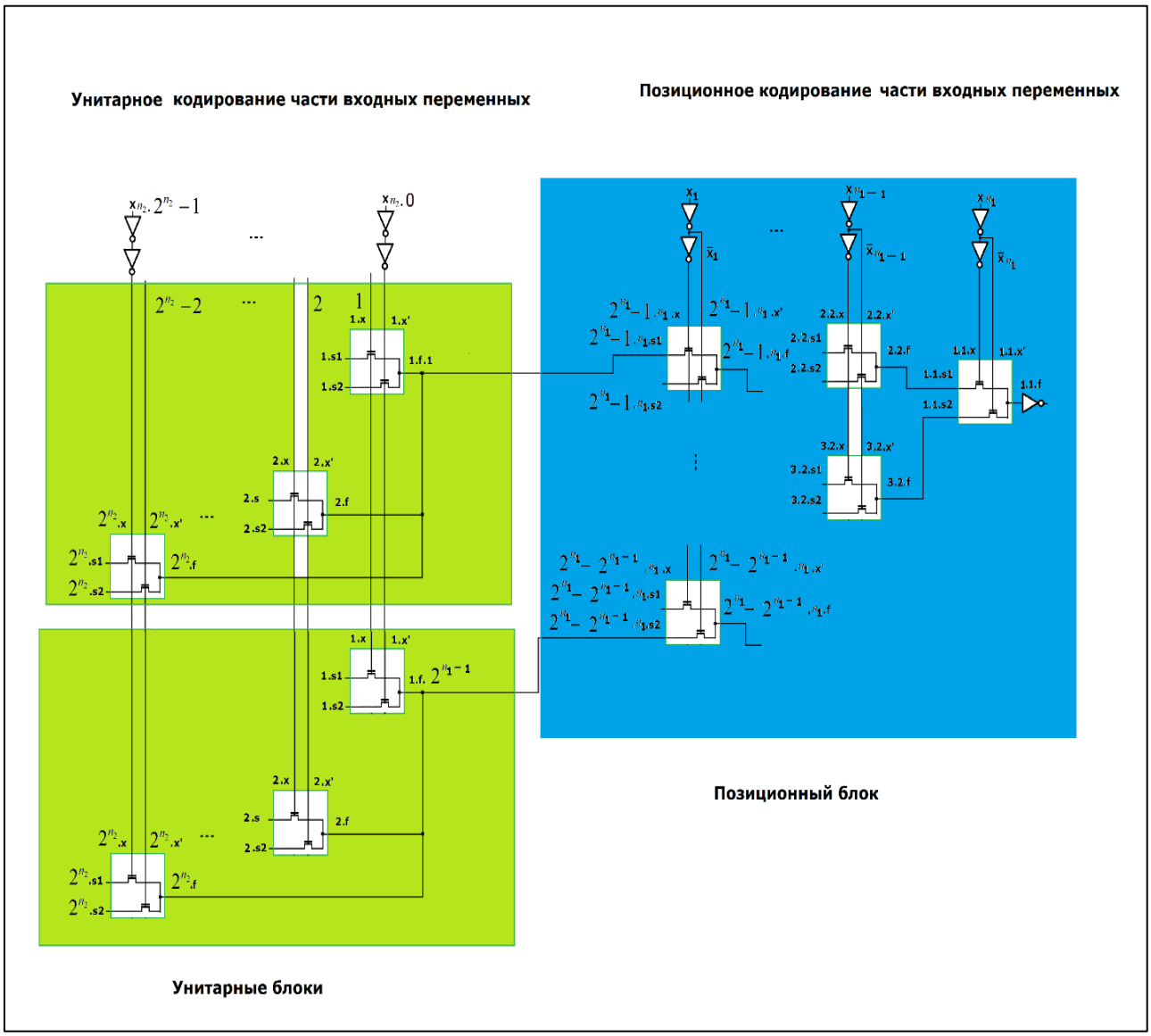


Рисунок 3.20 – Комбинированный элемент, позиционный блок справа

Синтезированные элементы представлены в общем виде, примеры элементов для конкретных параметров представлены в системах схемотехнического моделирования.

3.6 Разработка схемы электрической функциональной универсального элемента с конфигурируемым кодированием переменной

Известный элемент 1-LUT используемый для вычисления логической функции *f(x)* одной переменной *x* или для подключения одной из двух связей (connect1/connect2) показан на рисунке 3.21.

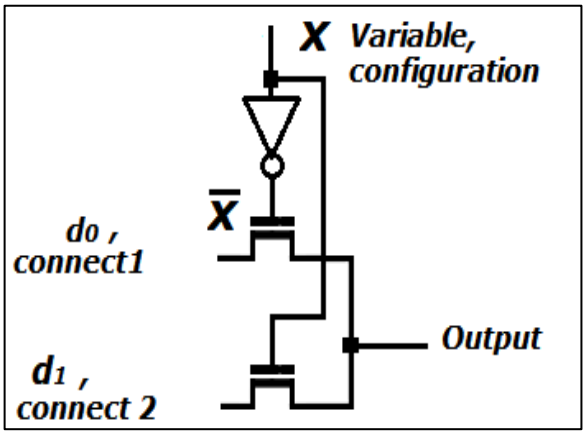


Рисунок 3.21 – Известный элемент 1-LUT для вычисления логической функции *f(x)* одной переменной *x* или для подключения одной из двух связей (connect1/connect2)

Известный элементарный мультиплексор маршрутизации на две связи 2-RM_u (Routing Multiplexer), использующий унитарный код «Одна единица» (One-Hot encoding), показан на рисунке 3.22.

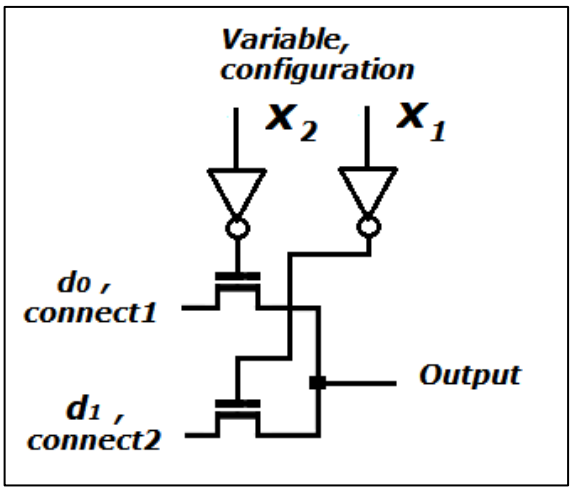


Рисунок 3.22 – Известный элементарный мультиплексор маршрутизации на две связи 2-RM_u (Routing Multiplexer)

Предлагаемый универсальный элемент с конфигурируемым кодированием переменной (u – настройка на унитарный код, b – настройка на бинарный код) показан на рисунке 3.23.

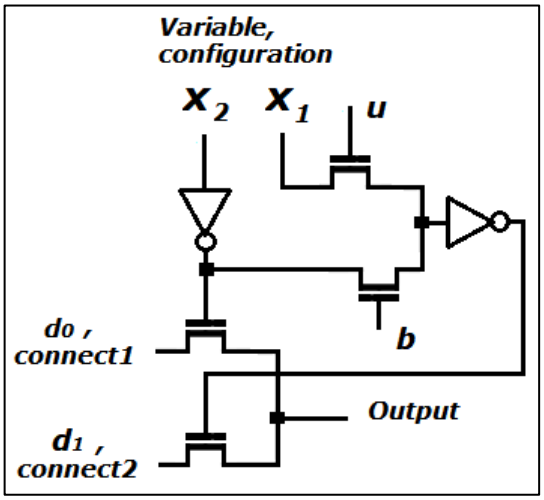


Рисунок 3.23 – Предлагаемый универсальный элемент с конфигурируемым кодированием переменной

В случае b=1, u=0 схема работает, как 1-LUT или как элементарный мультиплексор маршрутизации на две связи 2-RM_b с бинарным кодированием, в случае b=0, u=1 схема работает как элементарный мультиплексор маршрутизации на две связи 2-RM_u с унитарным кодированием или как логический элемент с унитарным кодированием переменной. Здесь используется инверсный унитарный код.

Каскадируя три схемы как на рисунке 3.23 (b=1, u=0), получаем 2-LUT или мультиплексор маршрутизации на четыре связи 4-RM_b с бинарным кодированием – рисунок 3.24.

Соединяя два элемента с рисунка 3.22, получаем мультиплексор маршрутизации на четыре связи 4-RM_u с унитарным кодированием, либо логический элемент на две переменные с унитарным кодированием – рисунок 3.25.

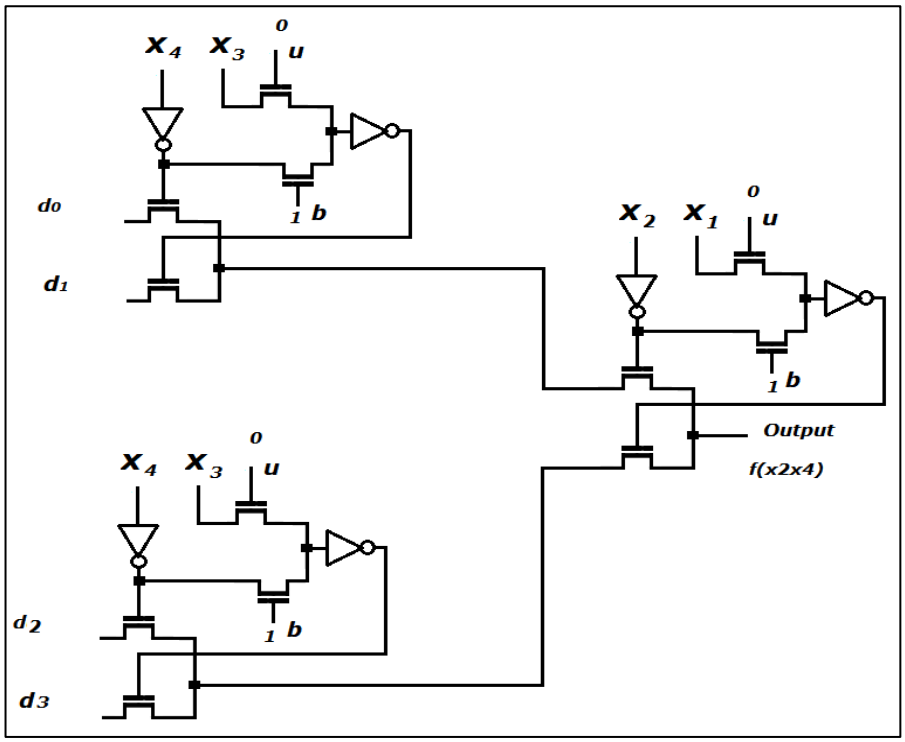


Рисунок 3.24 – Схема 2-LUT (b=1,u=0) или мультиплексор маршрутизации на четыре связи 4-RM_b с бинарным кодированием

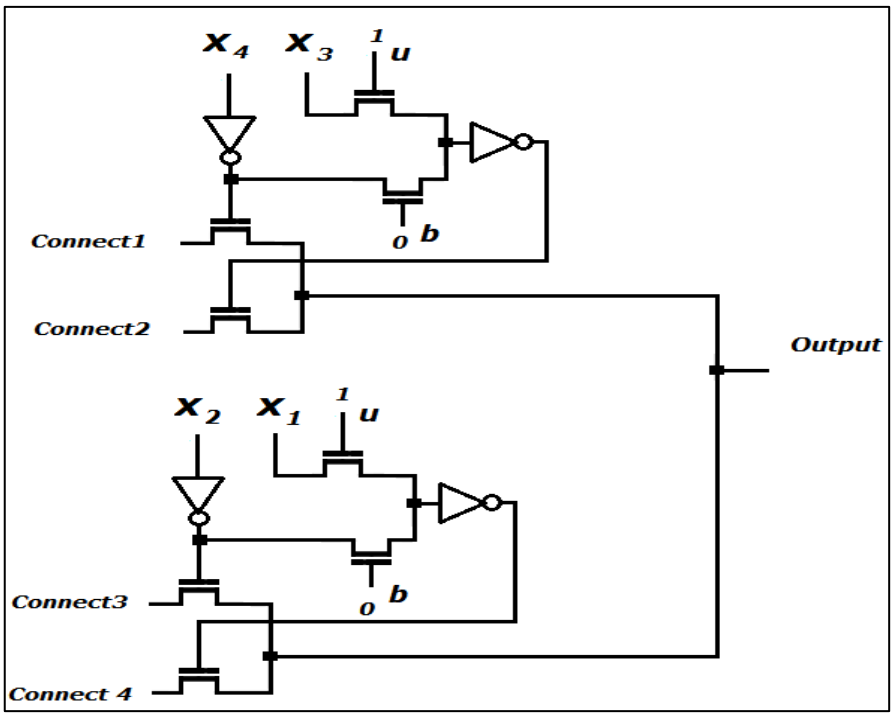


Рисунок 3.25 – Мультиплексор маршрутизации на четыре связи 4-RMu с унитарным кодированием или логический элемент на две переменные с унитарным кодированием

Например, в случае вычисления логической функции, если конфигурационные данные $d_0d_1d_2d_3$=0110, то реализуется $f(x_4x_2)=x_4$XORx_2. Если конфигурационные данные $d_0d_1d_2d_3$=0111, то $f(x_4x_2)=x_4$ORx_2, если конфигурационные данные $d_0d_1d_2d_3$=0001, то $(x_4x_2)=x_4$ANDx_2.

При маршрутизации, если конфигурационные данные $x_4x_3x_2x_1$=0111, то $f(x_4x_3x_2x_1)$ = connect1. Если конфигурационные данные $x_4x_3x_2x_1$ =1110 $f(x_4x_3x_2x_1)$ =connect4.

Соединяя семь схем с рисунка 3.22, получаем 3-LUT или мультиплексор маршрутизации на восемь связей 8-RMb с бинарным кодированием – рисунок 3.26.

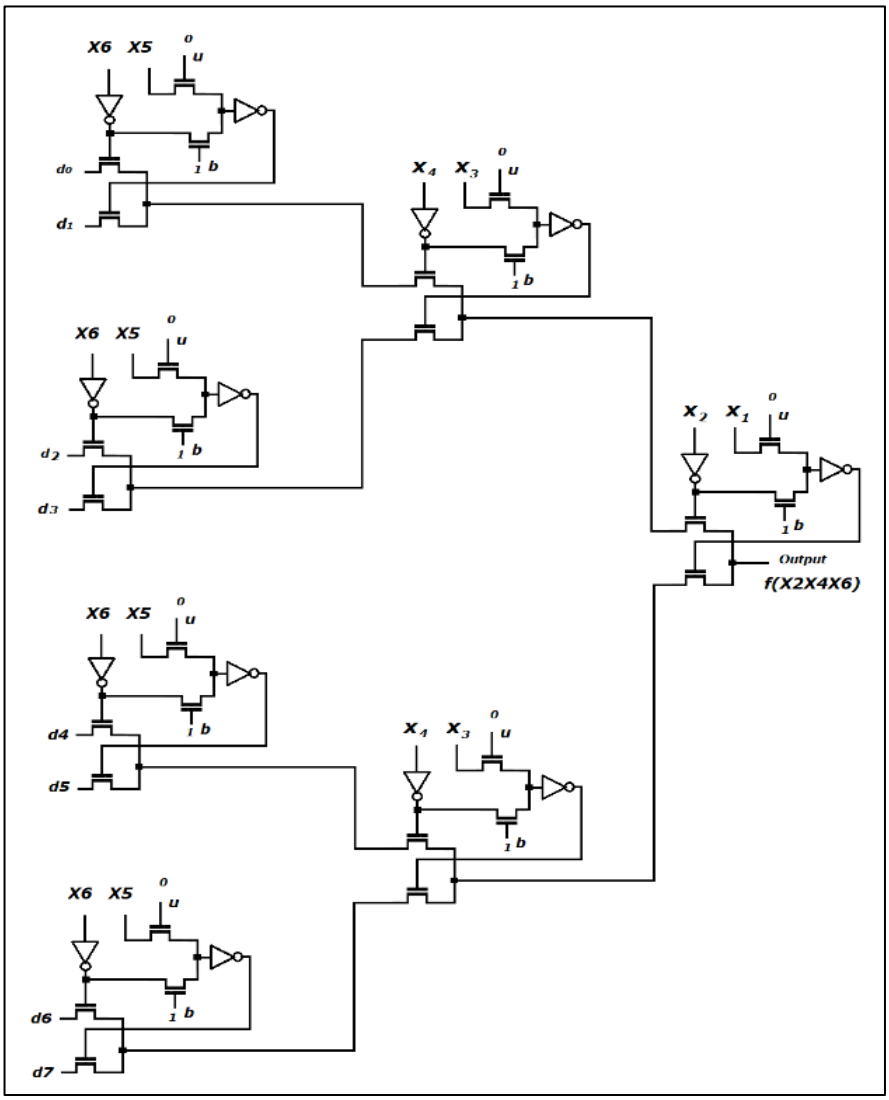


Рисунок 3.26 – 3-LUT или мультиплексор маршрутизации на восемь связей 8-RM_b с бинарным кодированием (b=1, u=0)

Для 3-LUT реализуется $f(x_2x_4x_6)$, x_2 – старшая переменная, x_6 – младшая переменная, x_4 – средняя переменная.

Например, в случае вычисления логической функции, если конфигурационные данные $d_0d_1d_2d_3d_4d_5d_6d_7$=01101001, то $f(x_2x_4x_6)$ = (x_2)XOR(x_4)XOR(x_6).

Если конфигурационные данные $d_0d_1d_2d_3d_4d_5d_6d_7$=01111111, то $f(x_2x_4x_6)$ = (x_2)OR(x_4)OR(x_6).

Если конфигурационные данные $d_0d_1d_2d_3d_4d_5d_6d_7$=00000001, то $f(x_2x_4x_6)$=(x_2)AND(x_4)AND(x_6).

Предлагаемый 8-RM_u, синтезированный из четырех схем с рисунка 3.22 (b=0, u=1) показан на рисунке 3.27.

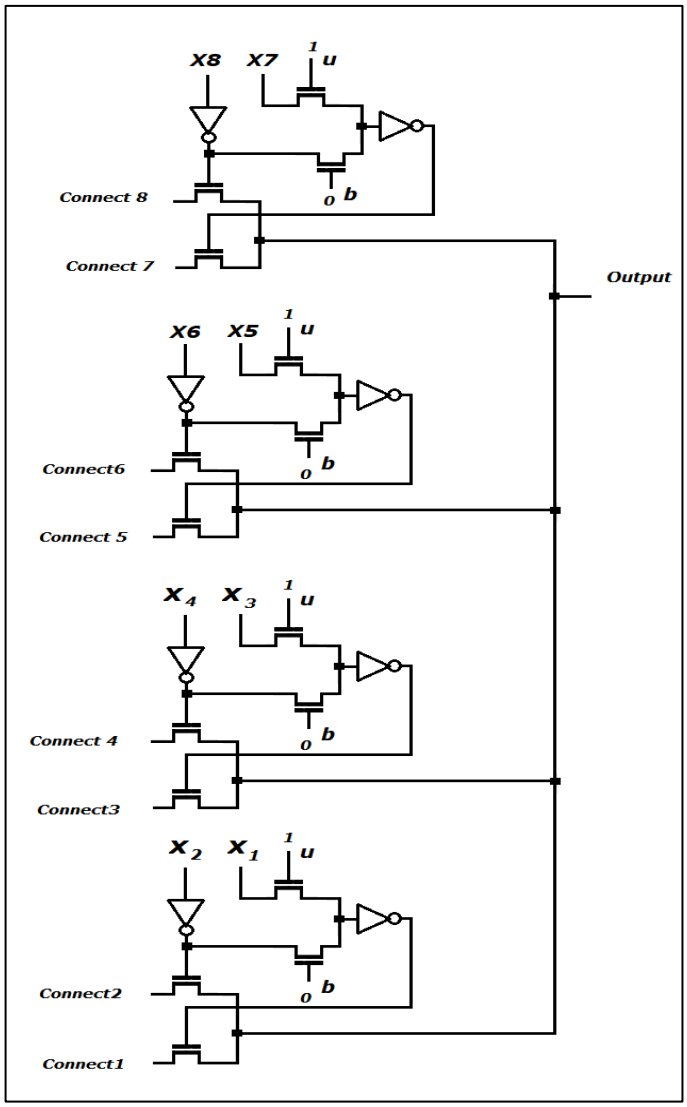


Рисунок 3.27 – Предлагаемый 8-RM_u

Если конфигурационные данные $d_0d_1d_2d_3d_4d_5d_6d_7$=01111111, то $f(x_8x_7x_6x_5x_4x_3x_2x_1)$=connect 8. Если конфигурационные данные $d_0d_1d_2d_3d_4d_5d_6d_7$=11111110, то $f(x_8x_7x_6x_5x_4x_3x_2x_1)$=connect 1 и так далее.

Предлагаемый комбинированный элемент вида: бинарный 1-LUT+ унитарный 3-LUT показан на рисунке 3.28.

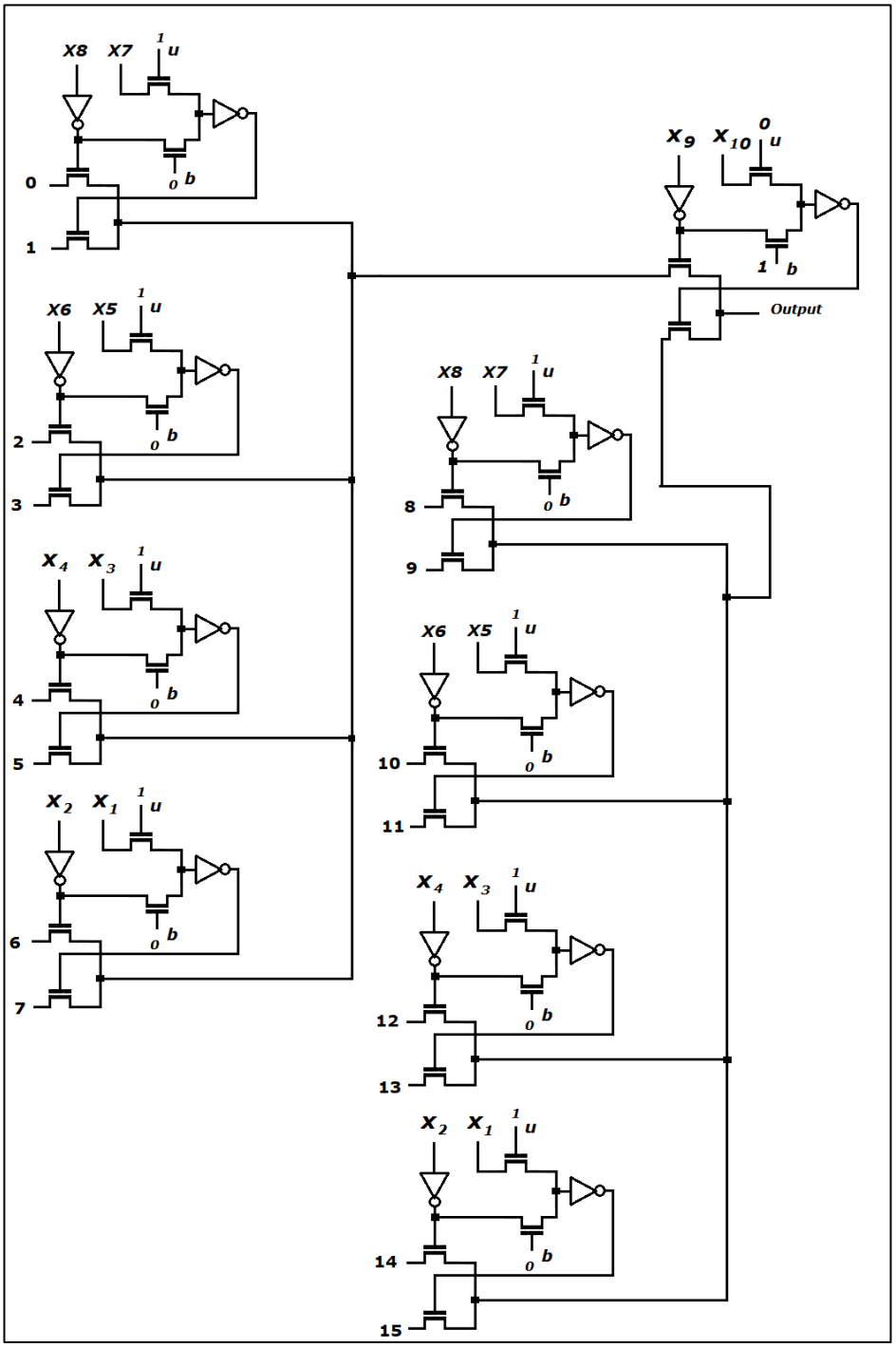


Рисунок 3.28 – Предлагаемый комбинированный элемент вида: бинарный 1-LUT(x_9)+ унитарный 3-LUT ($x_8x_7x_6x_5x_4x_3x_2x_1$)

В этом случае количество передающих транзисторов в схеме (входы настройки функции 0,1,…15 – выход) сокращается до двух в комбинированной версии против четырех в 4-LUT и обеспечивает выполнение ограничений Мида-Конвея [54].

3.7 Выводы по главе 3

1) Предложенные схемы унитарной реализации логических функций позволяет получить минимальную временную задержку в количестве задержек транзисторов, но при этом резко возрастает число линий связи для передачи кода переменных. Конфигурационная же память всегда одинакова и оценивается как $2n$.

2) Комбинированные варианты элементов могут быть получены путем разделения унитарного кода на группы (координаты), однако такой известный подход не является наилучшим.

3) Наиболее перспективными являются модель, использующая как унитарный, так и позиционный код в одном элементе и метод синтеза таких элементов, причем возможны два подварианта в зависимости от положения унитарного блока в элементе: слева или справа.

4) Схема предлагаемого универсального элемента с выбором варианта кодирования является расширением так называемых адаптивных логических модулей (АЛМ).

5) Для подтверждения правильности функционирования разработанных схем элементов необходимо схемотехническое моделирование в статическом и динамическом режимах.

ГЛАВА 4. ИССЛЕДОВАНИЕ ЭЛЕКТРИЧЕСКИХ ПРИНЦИПИАЛЬНЫХ СХЕМ ПРЕДЛОЖЕННЫХ ЭЛЕМЕНТОВ В СИСТЕМАХ СХЕМОТЕХНИЧЕСКОГО И ТОПОЛОГИЧЕСКОГО МОДЕЛИРОВАНИЯ

4.1 Моделирование электрических принципиальных схем предложенных элементов в системе схемотехнического моделирования Multisim фирмы National Instruments

С целью подтверждения работоспособности разработанных схем электрических функциональных в процессе исследования разработаны и проанализированы схемы электрические принципиальные в системе схемотехнического моделирования. Multisim фирмы National Instruments. [80] Схема – модель элемента с унитарным кодированием для вычисления логической функции одной переменной (рисунок 3.3,а), показанная на рисунке 4.1.

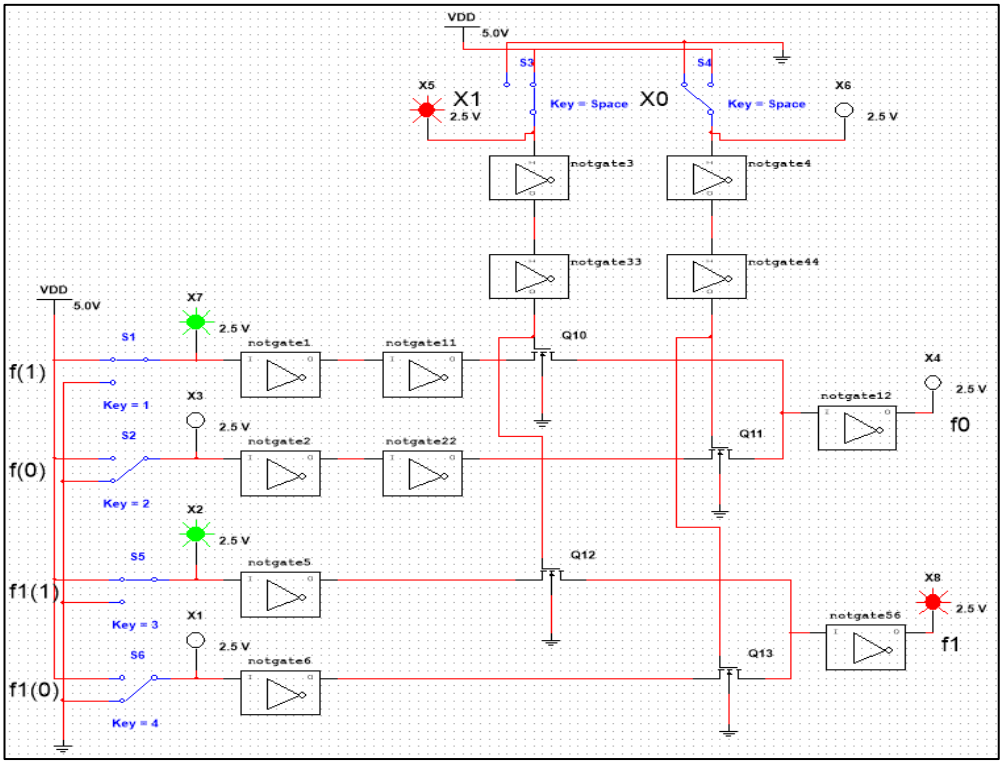


Рисунок 4.1 – Модель элемента с унитарным кодированием для вычисления логической функции одной переменной $f_0\, f_1$ =01

На рисунке 4.1 функция f представлена в унитарном коде, как f_0 f_1. Для этого помимо основной настройки f_1 (1) f_1 (0) введена дополнительная схема с настройкой f(1) f (0), инверсия обеспечивается дополнительными инверторами notgate11, notgate22. Входная переменная кодируется ключами *X1, X0, X1* – старший разряд. То есть на рисунке 4.1 показано значение входной переменной X, равное единице. Настройкой задана функция повторения, поэтому f_0 f_1 =01. Такое же значение переменной показано на рисунке 4.2, но при другой настройке (задана функция «константа нуля»), поэтому f_0 f_1 =10.

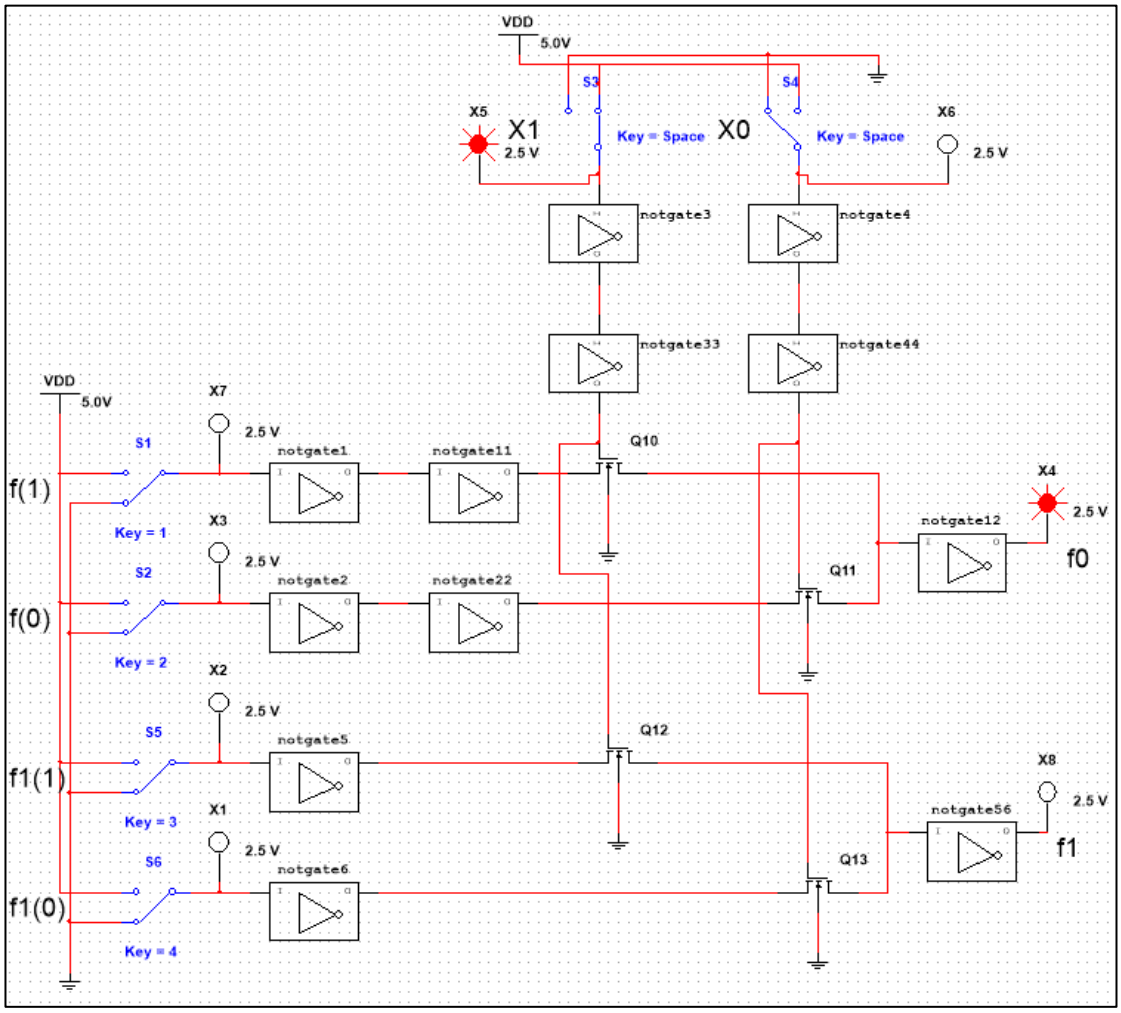


Рисунок 4.2 – Модель элемента с унитарным кодированием для вычисления логической функции одной переменной f_0 f_1 =10

Аналогично моделируются унитарные логические элементы на большее число переменных (например, рисунок 3.3,б).

Разработанная модель элемента с комбинированным кодированием (рисунок 3.13,б), где значение функции f (*X3X2X1X0BA*) представляется в бинарном коде и показана на рисунке 4.3.

Схема на рисунке 4.3 вычисляет значение функции четырех бинарных переменных, представленных комбинированным кодом *X3X2X1X0BA*, где *X3X2X1X0* – унитарная часть (четыре унитарных элемента), *BA* – бинарная часть, 2-LUT. Всего имеется 16 бит настройки f (00)…. f (15). Известный логический элемент 4-LUT содержит четыре транзистора в каждой из шестнадцати ветвей дерева передающих транзисторов, а в предлагаемой схеме таких транзисторов три.

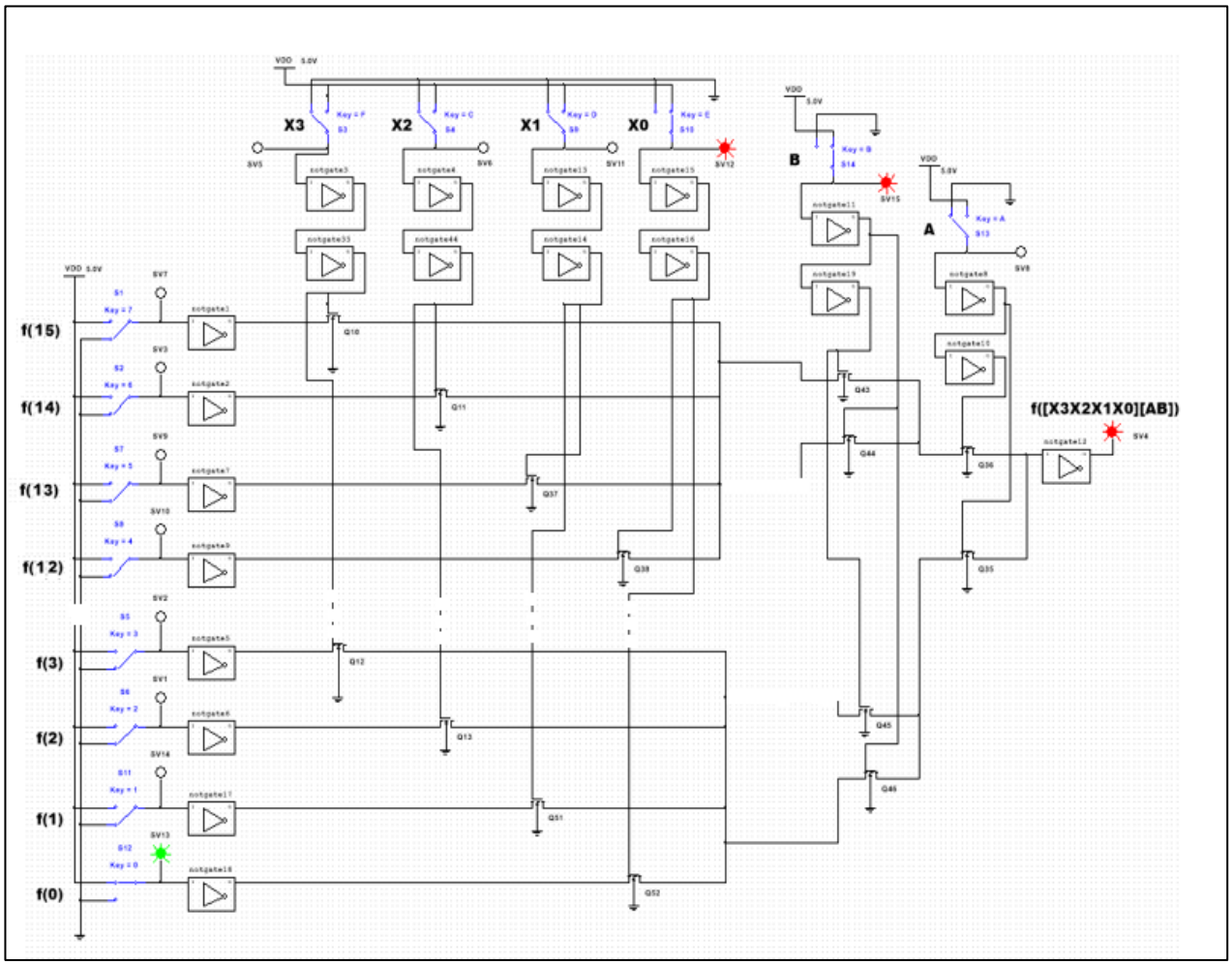


Рисунок 4.3 – Модель элемента с комбинированным кодированием для вычисления логической функции четырех бинарных переменных от набора комбинированных переменных *X3X2X1X0BA*

Конфигурационных бит в 4-LUT столько же, сколько на рисунке 4.3, но здесь больше переменных – шесть, вместо четырех.

Разработанная модель элемента с комбинированным кодированием для вычисления логической функции от набора переменных *u3u2u1u0a* (рисунок 3.13,а) показана на рисунке 4.4. Здесь имеется восемь конфигурационных бит *d*(0)…*d*(7). На рисунке 4.4. показано вычисление логической функции на наборе (*u*3=1, *u*2=0, *u*1=0, *u*0=0) (*a*=1). На этом наборе функция равна единице, то есть, очевидно, что задана функция конъюнкции трех бинарных переменных.

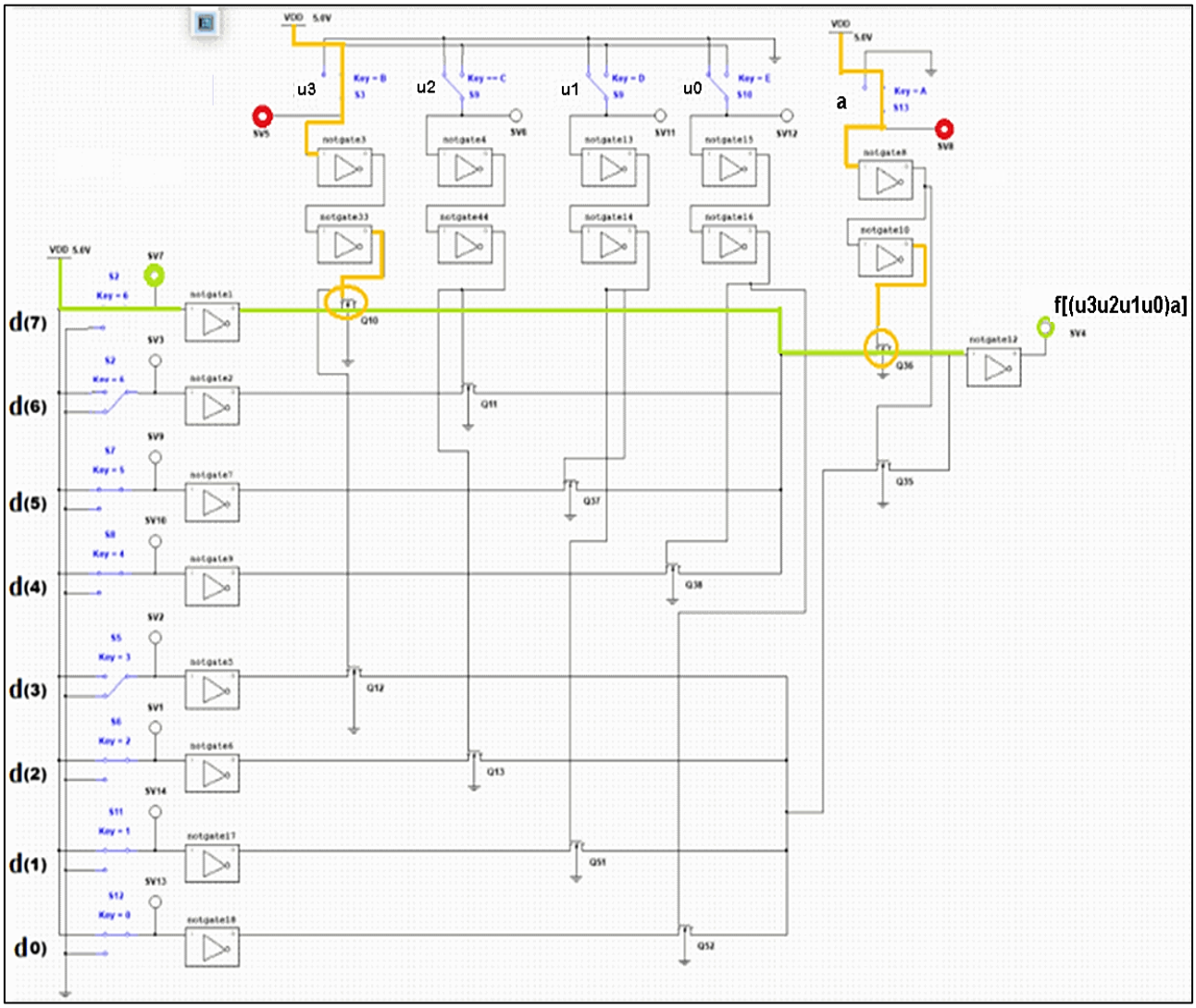


Рисунок 4.4 – Модель элемента с комбинированным кодированием для вычисления логической функции трех бинарных переменных от набора комбинированных переменных *u3u2u1u0a*

В отличие от известного 3-LUT, содержащего три передающих транзистора в каждой из восьми ветвей дерева, в схеме на рисунке 4.4 два транзистора. Однако общее число переменных увеличивается с трех до пяти.

Эти же схемы могут быть использованы для коммутации связей, при этом вместо входов настройки, например, *d*(0)…*d*(7) для рисунка 4.4 подключаются связи, например, connect(0)….connect(7), а настройка переносится на входы *u3u2u1u0a*. То есть, настройка, по сравнению с известным 3-LUT в режиме коммутации связей увеличивается с трех бит до пяти, но путь сигнала связей уменьшается с трех до двух. Элемент с комбинированным кодированием для реализации функции на 4 переменных при комбинированном кодировании показан на рисунке 4.5. Элементы в синей области – позиционные элементы схемы, в зеленой области – унитарные элементы схемы.

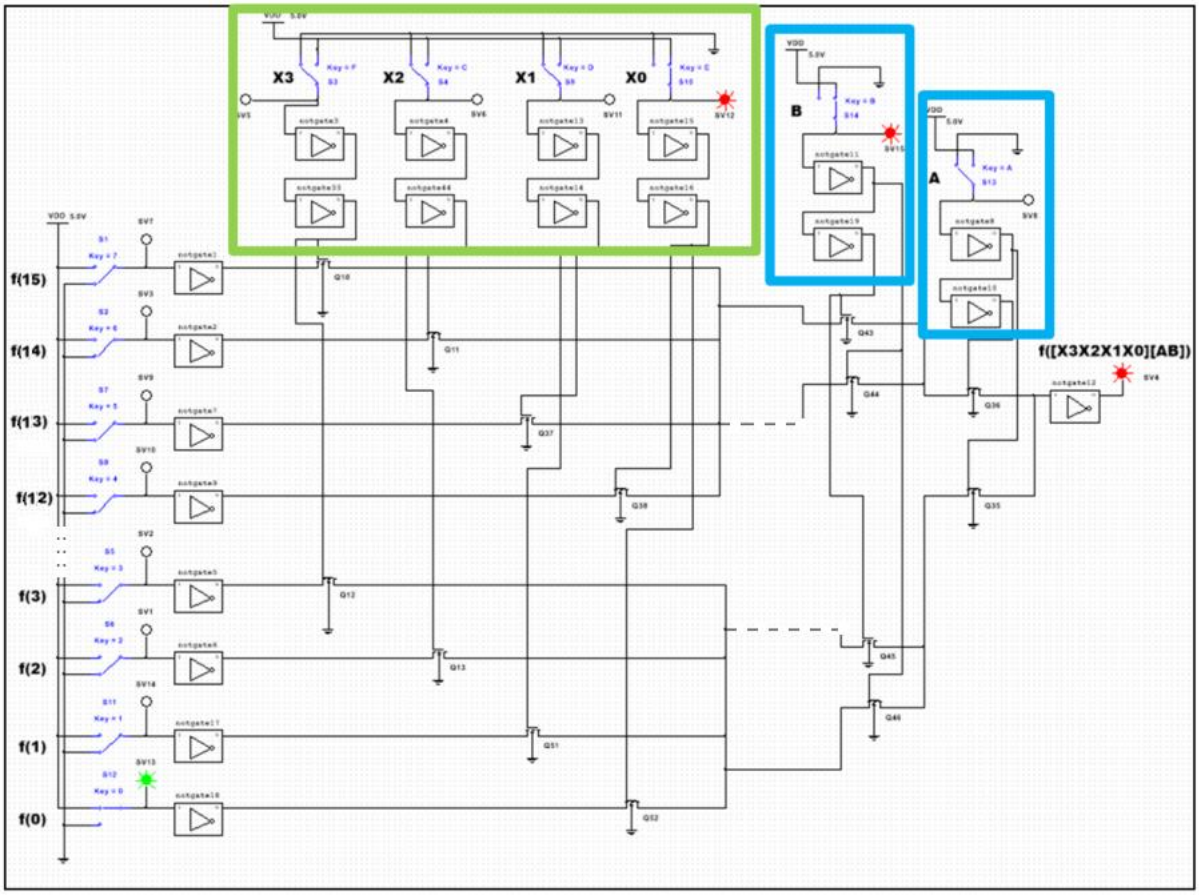
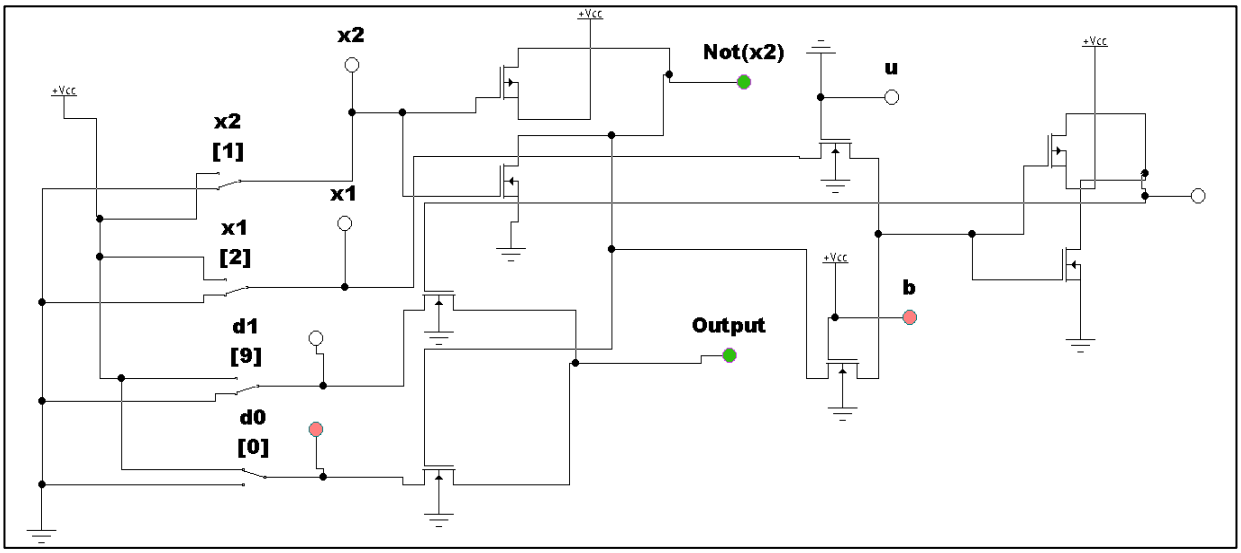


Рисунок 4.5 – Элемент с комбинированным кодированием для реализации функции на 4 переменных

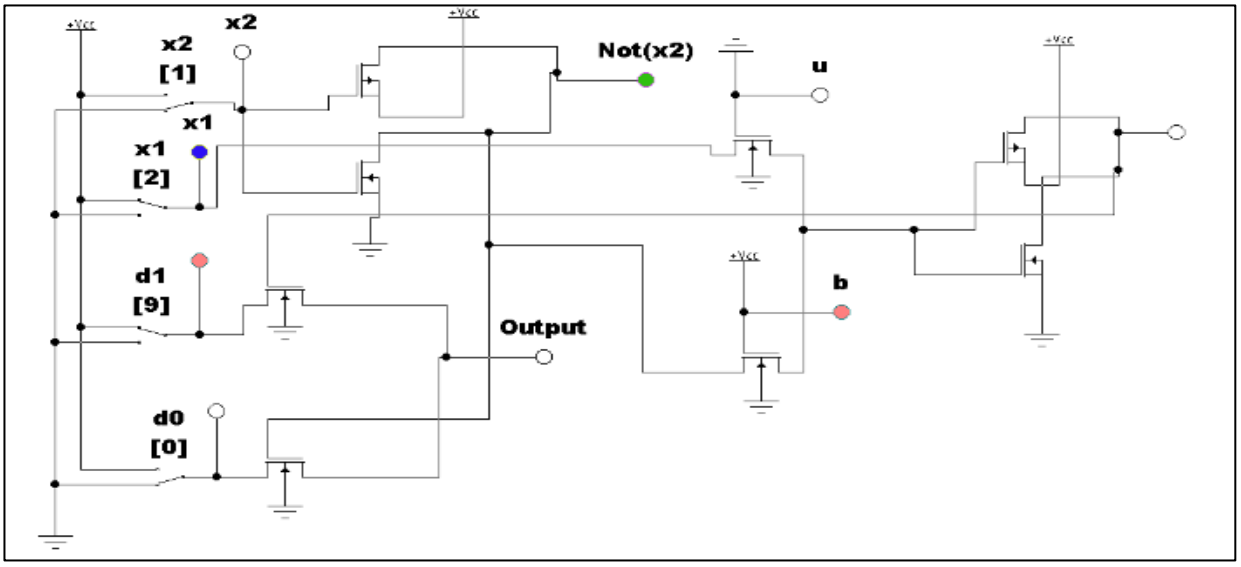
Схемотехническое моделирование подтвердило работоспособность предложенных схем.

4.2 Моделирование универсального элемента с конфигурируемым кодированием переменной в системе схемотехнического моделирования Multisim фирмы National Instruments

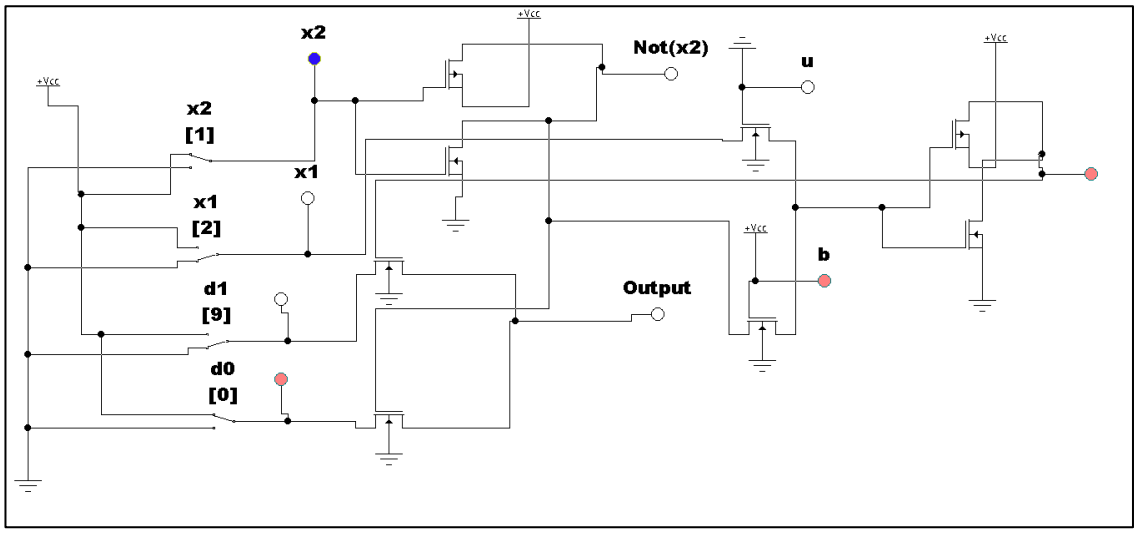
На рисунке 4.6 показана модель предлагаемого универсального элемента 1-LUT с конфигурируемым кодированием переменной в системе схемотехнического моделирования в режиме бинарного кодирования.



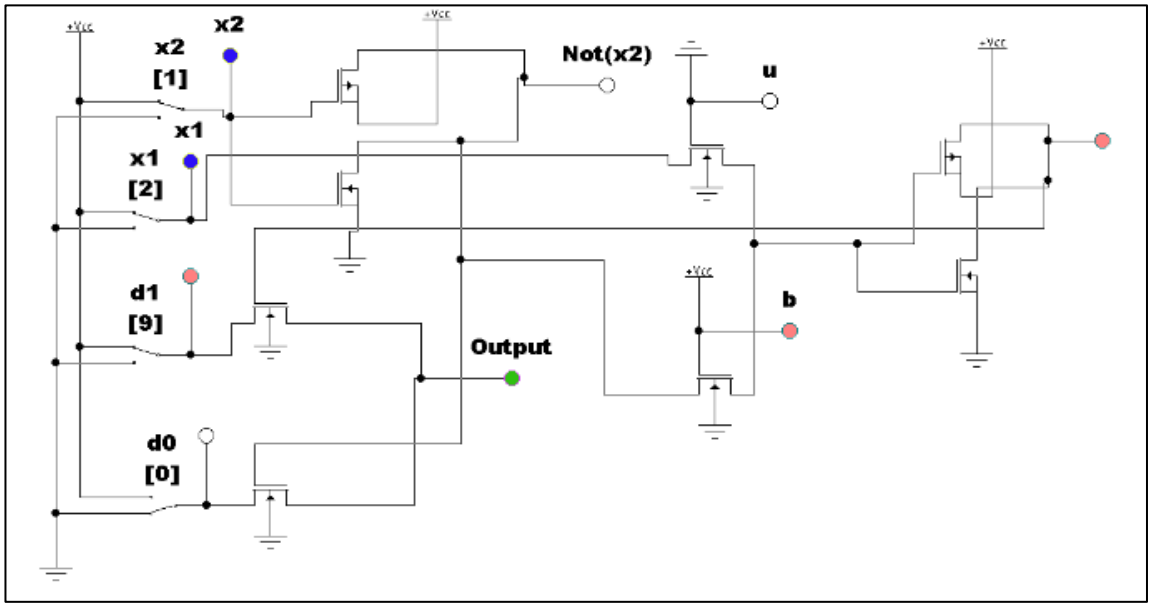
а)



б)



в)

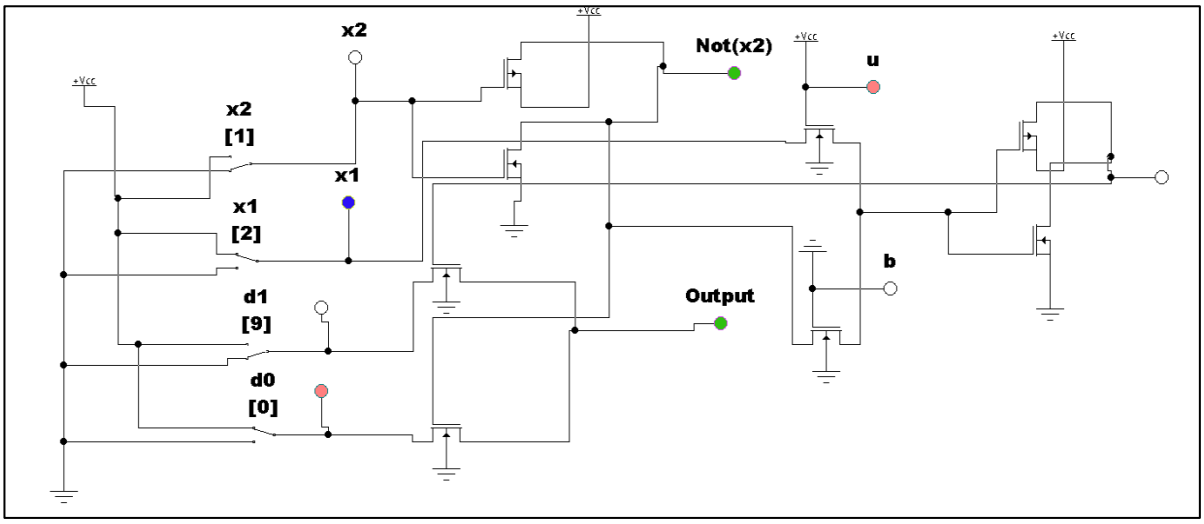


г)

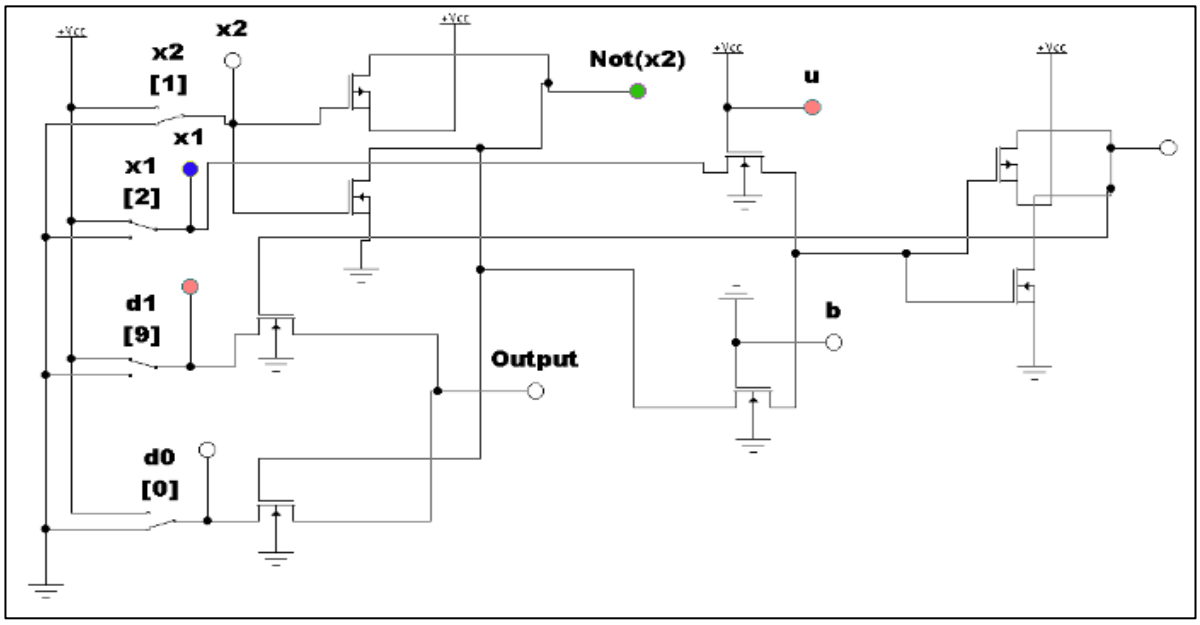
Рисунок 4.6 – 1-LUT в режиме бинарного кодирования (*b*=1, *u*=0): а) *x2*=0(коммутация входа *d0*), выход=1; б) *x2*=0(коммутация входа d0), выход=0; в) *x2*=1, выход=0 (коммутация входа *d1*=0); г) *x2*=1, выход=1 (коммутация входа *d1*=1)

При (*b*=1, *u*=0) реализуется либо при *x2*=0 коммутация входа *d0*, либо при *x2*=1 коммутация входа *d1* (рисунок 4.5,б).

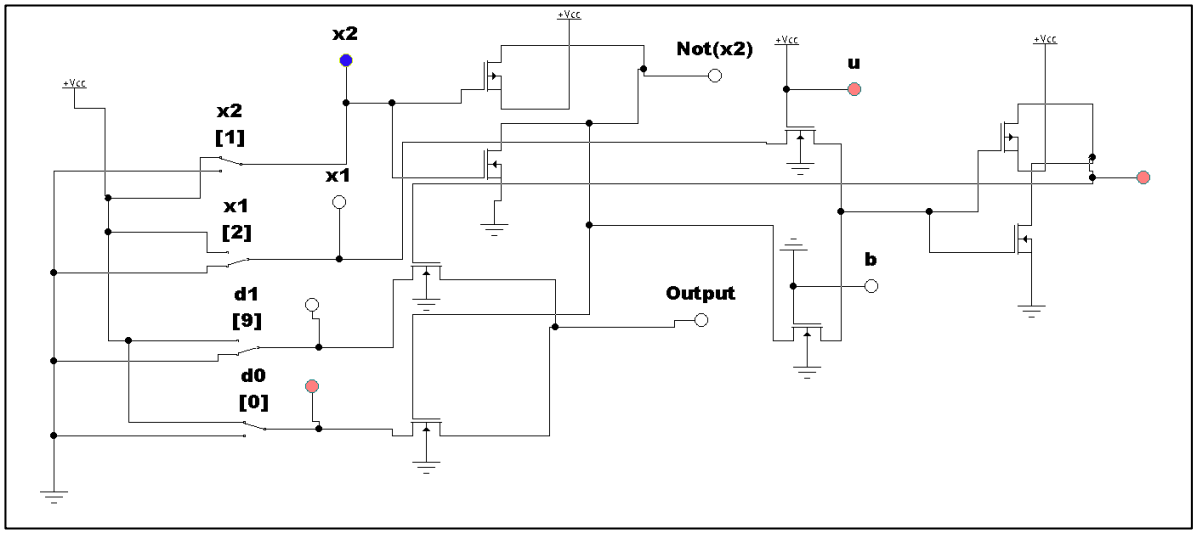
На рисунке 4.7 показана модель предлагаемого универсального элемента 1-LUT с конфигурируемым кодированием переменной в системе схемотехнического моделирования в режиме унитарного кодирования.


x2
Not(x2)
u
x2
[1]
x1
x1
[2]
b
d1
[9]
Output
d0
[0]

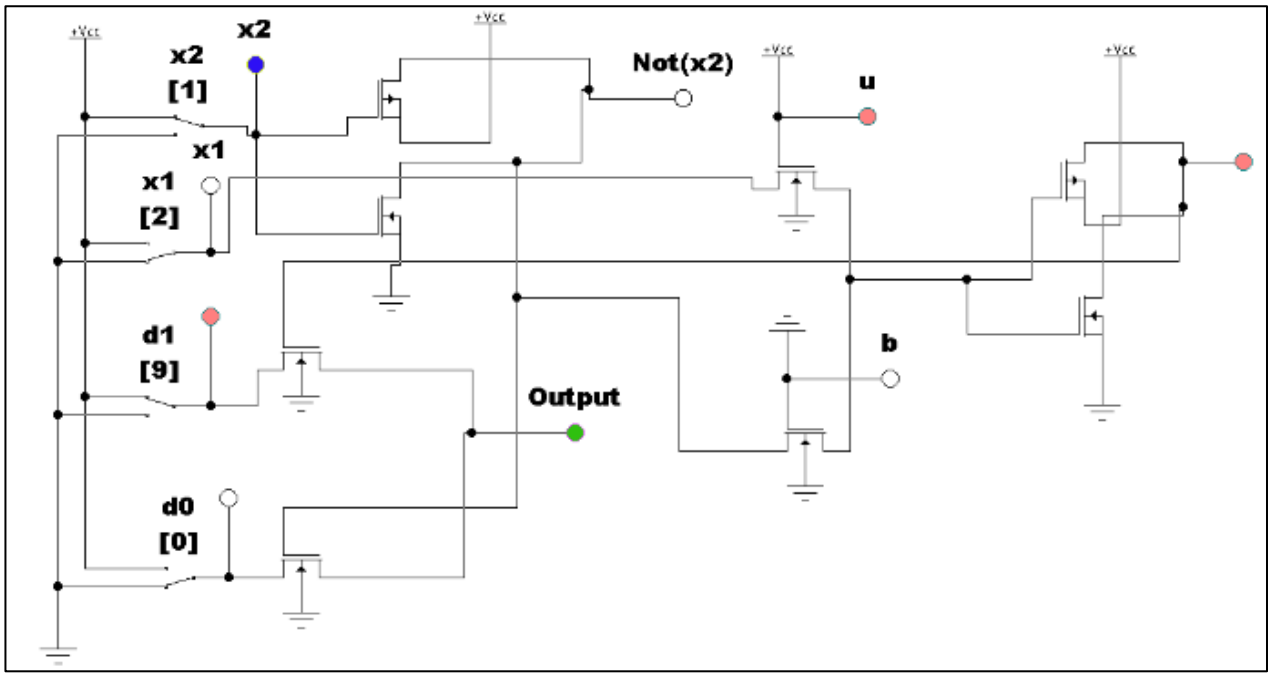
а)


x2
x2
[1]
Not(x2)
u
x1
x1
[2]
d1
[9]
b
Output
d0
[0]

б)


x2
Not(x2)
u
x2
[1]
x1
x1
[2]
b
d1
[9]
Output
d0
[0]

в)



г)

Рисунок 4.7 – 1-LUT в режиме унитарного кодирования (*b*=0, *u*=1): а) *x2*=0, *x1*=1 (коммутация входа *d0*=1), выход=1; б) *x2*=0, *x1*=1 выход=0 (коммутация входа *d0*=0); в) *x2*=1, *x1*=0 выход=0 (коммутация входа *d1*=0); г) *x2*=1, *x1*=0 выход=1 (коммутация входа *d1*=1)

Моделирование подтверждает работоспособность предложенного универсального элемента. Требуется уточнение характеристик схем с использованием топологического моделирования, например, в системе Microwind [81, 82].

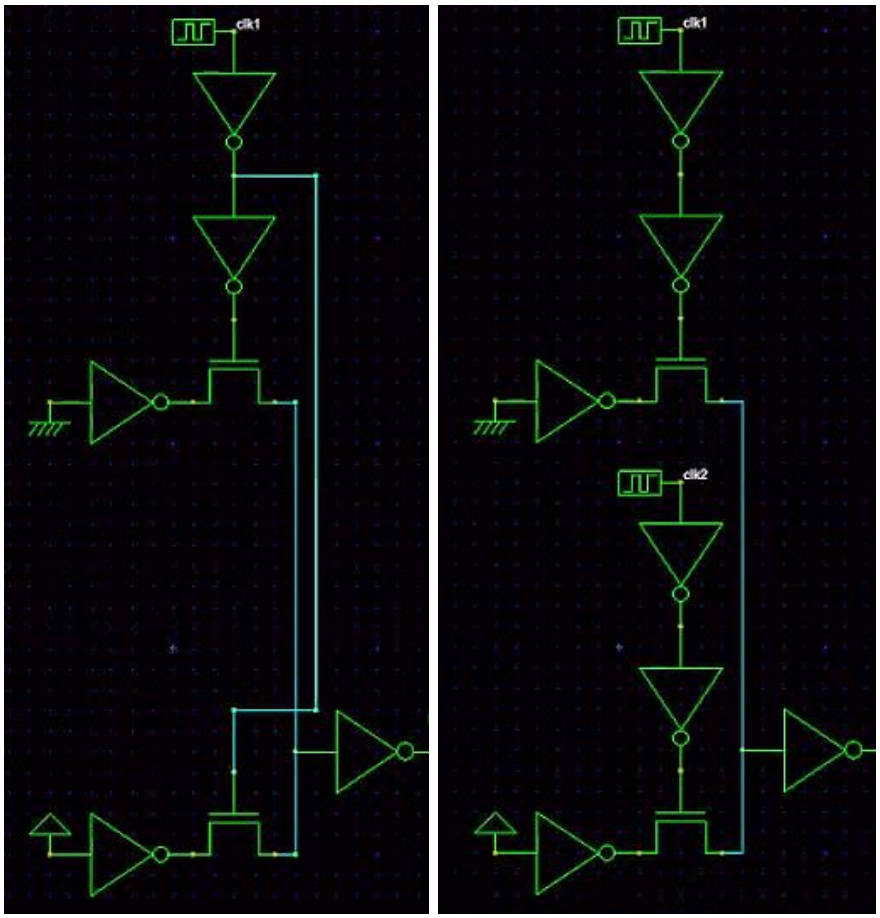
4.3 Моделирование в системе топологического моделирования Microwind элементов для вычисления заданной логической функции

4.3.1 Моделирование в Microwind известного элемента для вычисления логической функции на одну переменную

Элемент 1-LUT реализует функцию одной переменной. Будем называть коммутатор на две связи, используемый для вычисления функции, также1-LUT, но унитарным. Сначала в Microwind, с помощью специального пакета

программы DSCH, [81] строится схема электрическая функциональная – рисунок 4.8.

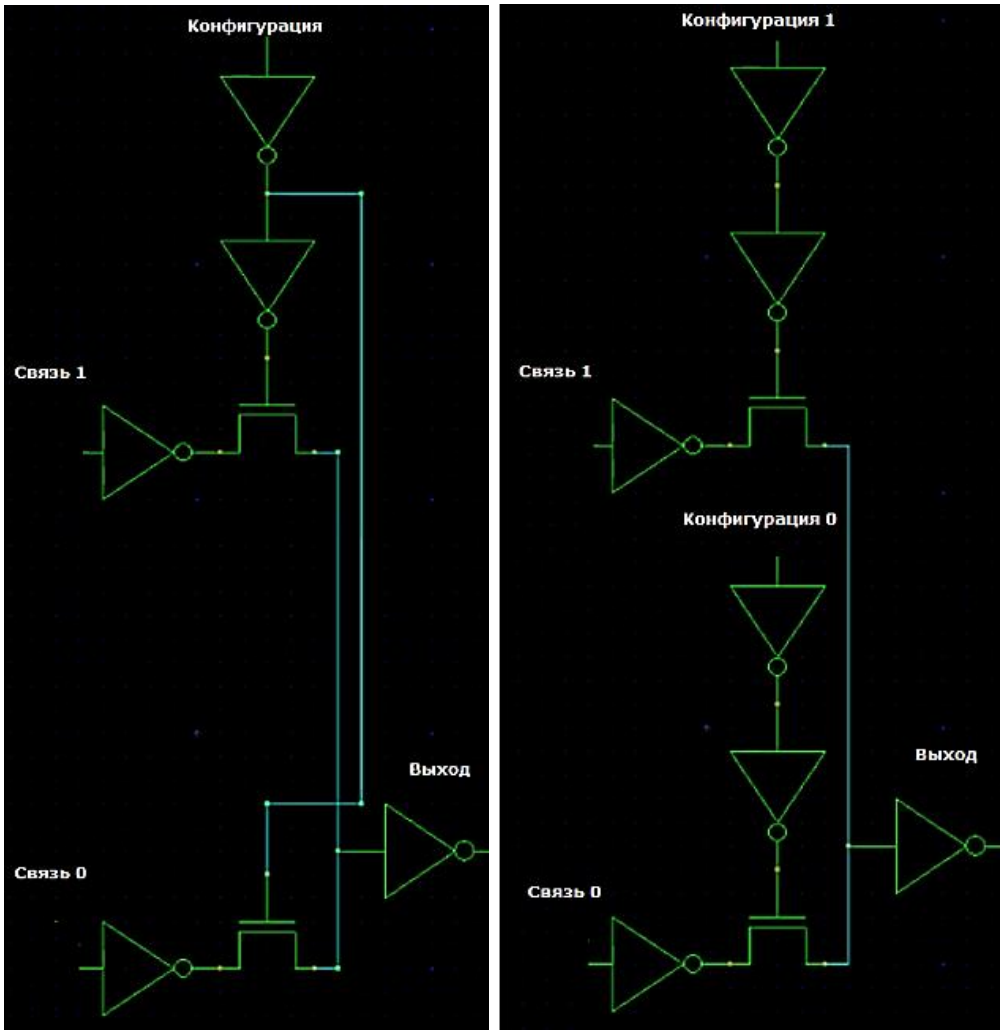
Дерево транзисторов строится из двух *n*-МОП (*n*-MOS) транзисторов; инверторы предполагают использование каждый двух КМОП транзисторов (*n*-MOS,*p*-MOS: Complementary MOS, CMOS) [101].



а) б)

Рисунок 4.8 – Схема электрическая функциональная 1-LUT, настроенная на реализацию логической функции инверсии входного сигнала Clk: а) известного; б) унитарного

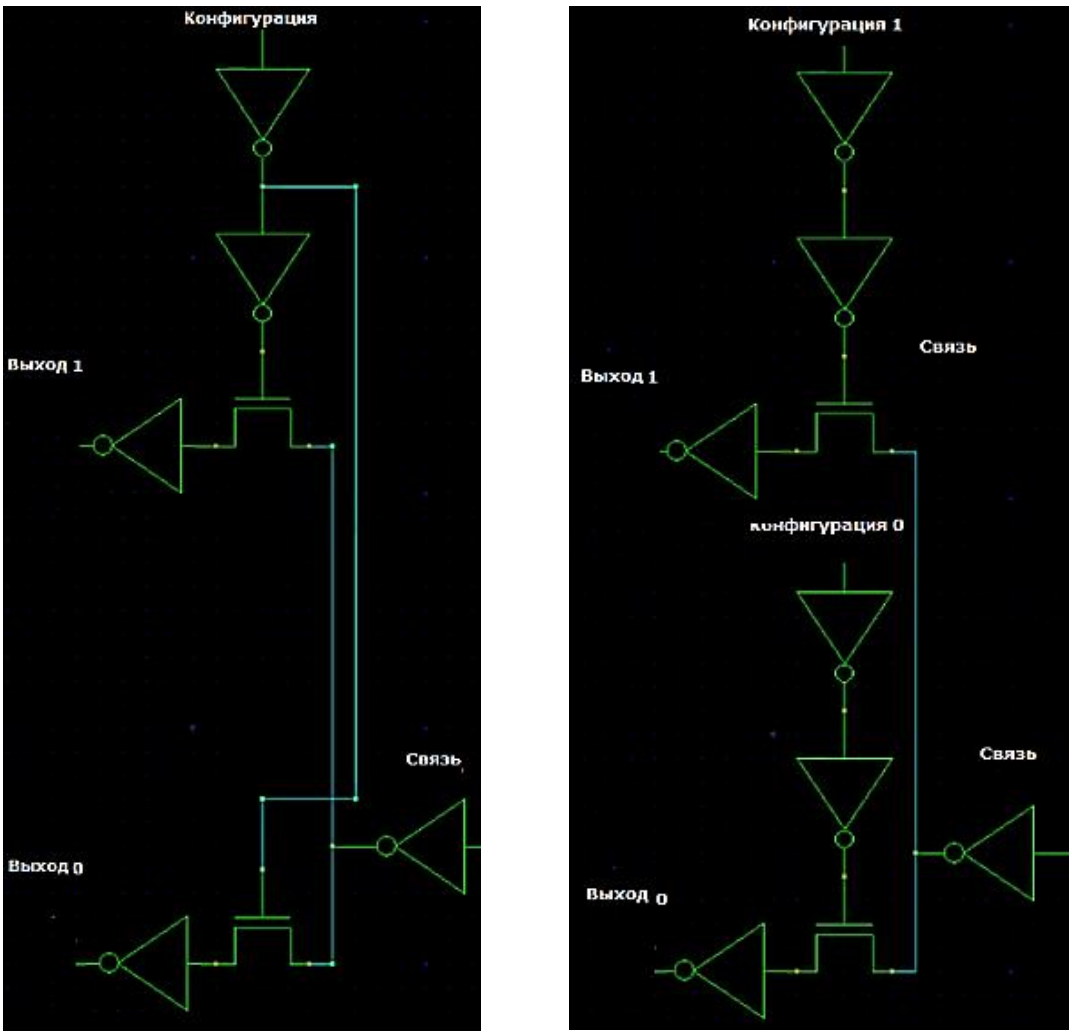
Эти схемы соответствующих элементов (рисунок 4.8) могут быть использованы не для вычисления логической функции, а для коммутации связей (мультиплексора 2-1) – рисунок.4.9.



а) б)

Рисунок 4.9 – Схема электрическая функциональная коммутатора связей – мультиплексора 2-1: а) на основе1-LUT позиционного; б) на основе 1-LUT унитарного

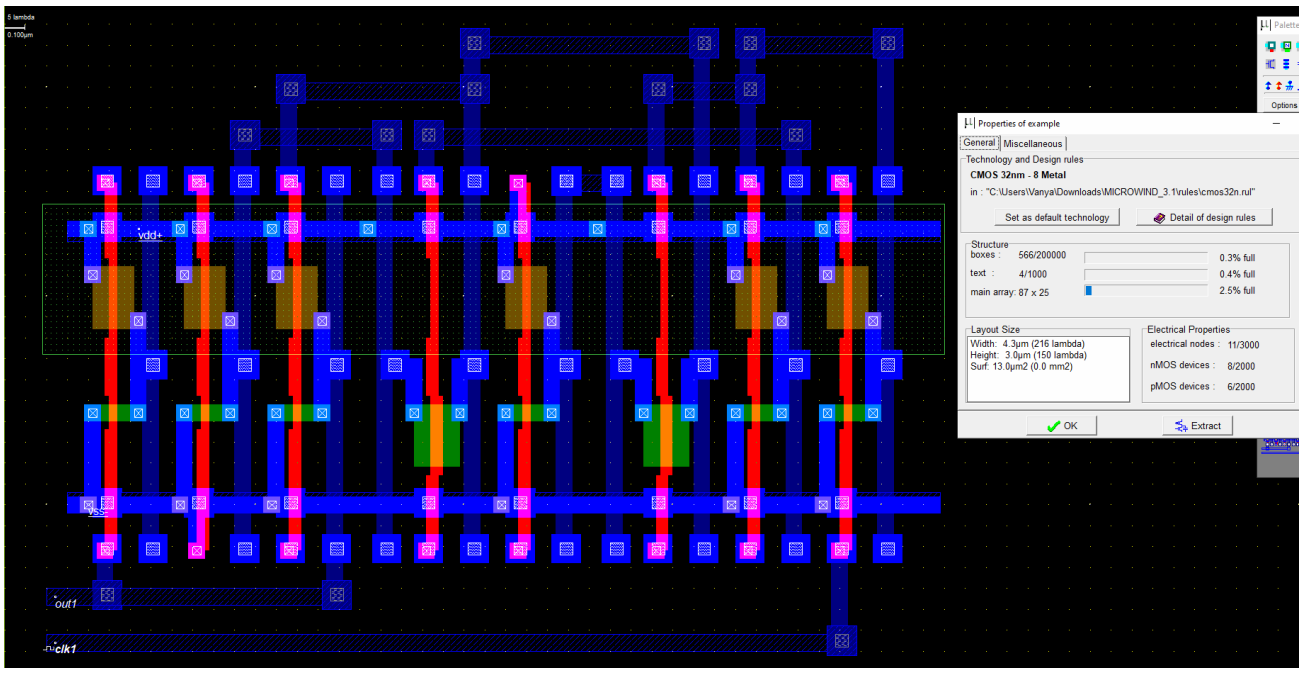
Демультиплексор 1-2 строится так, как показано на рисунке 4.10.



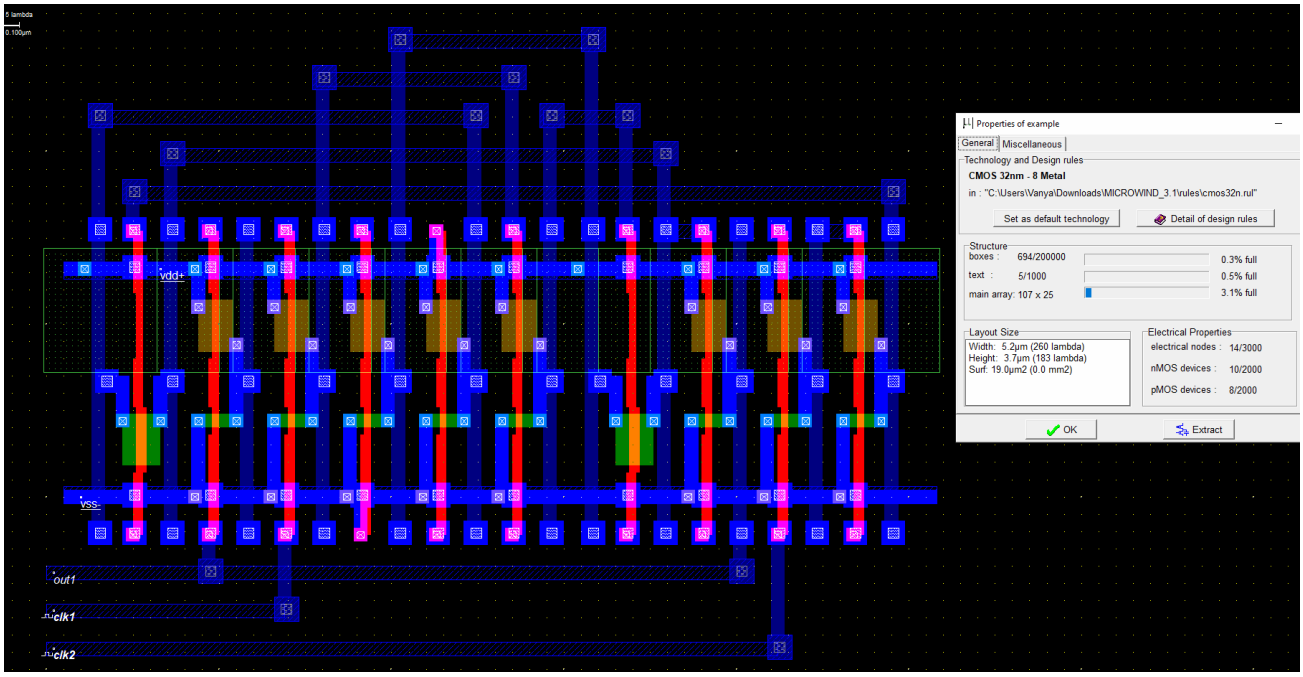
а) б)

Рисунок 4.10 – Схема электрическая функциональная коммутатора связей – демультиплексора 2-1: а) на основе1-LUT позиционного; б) на основе1-LUT унитарного

Далее был получен соответствующий VHDL код с описанием компонентов схемы (англ. VHSIC (Very high speed integrated circuits) Hardware Description Language) и по нему строилась топология по технологии CMOS 32 [95] nm, например, для схемы на рисунке 4.11.



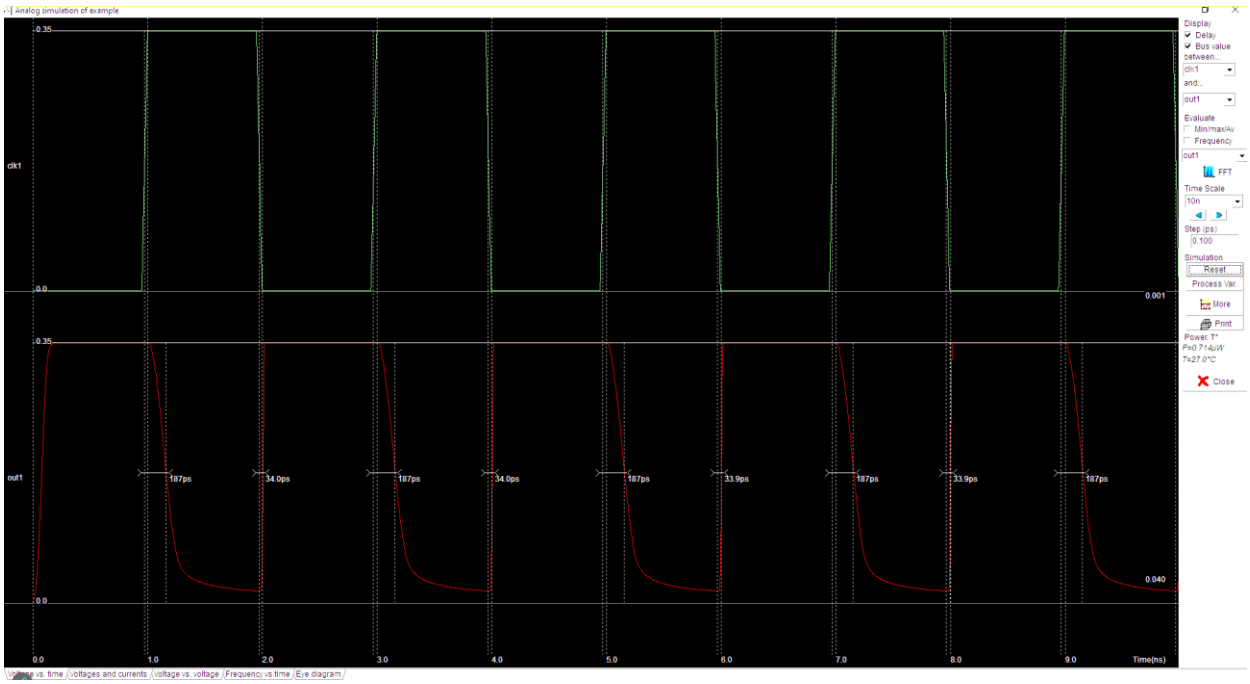
а)



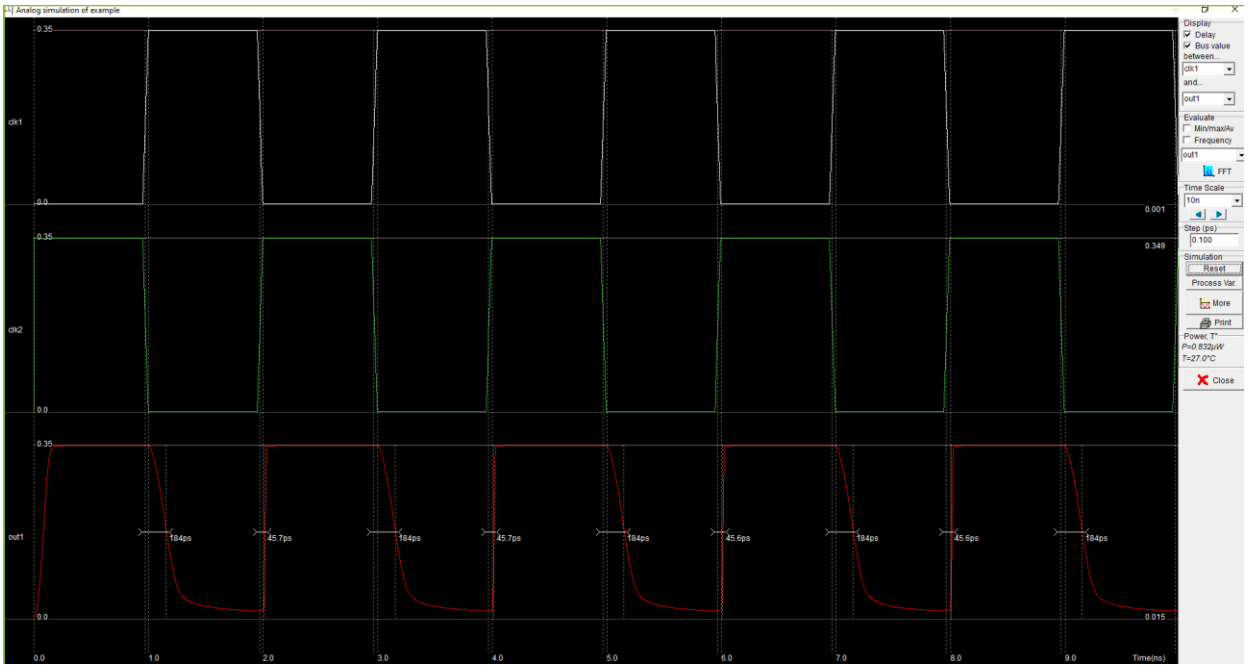
б)

Рисунок 4.11 – Топология 1-LUT: а) позиционного; б) унитарного

В дальнейшем для топологий, приведенных на рисунке 4.11, выполнялось временное моделирование с получением осциллограмм [98] – рисунок 4.12.



а)



б)

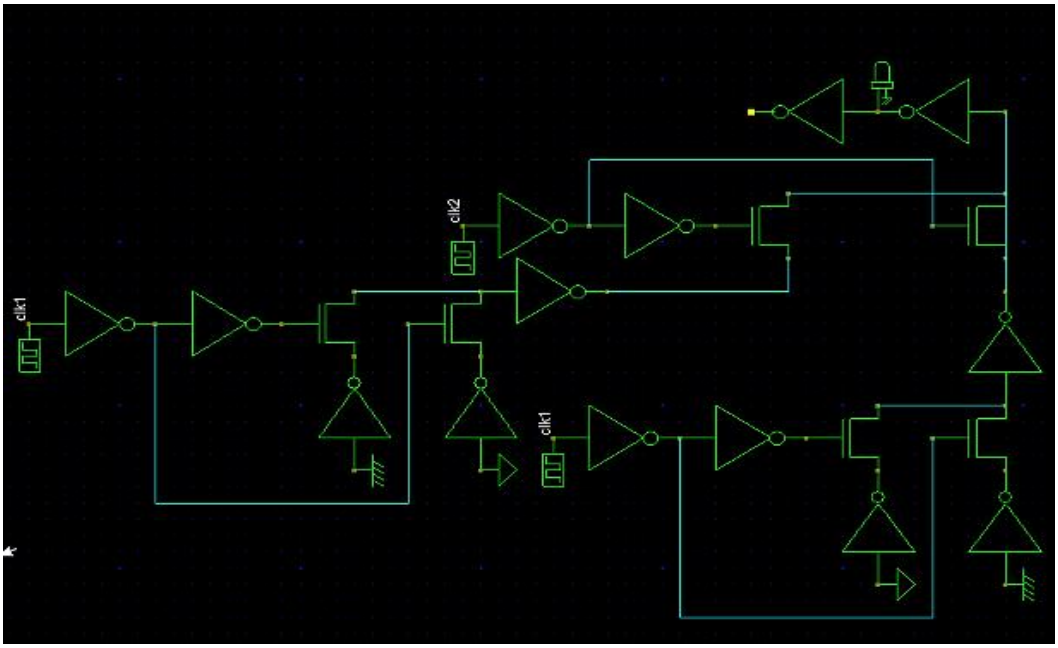
Рисунок 4.12 – Временная диаграмма работы:

а) позиционного 1-LUT CMOS 32 nm; б) унитарного 1-LUT CMOS 32 nm

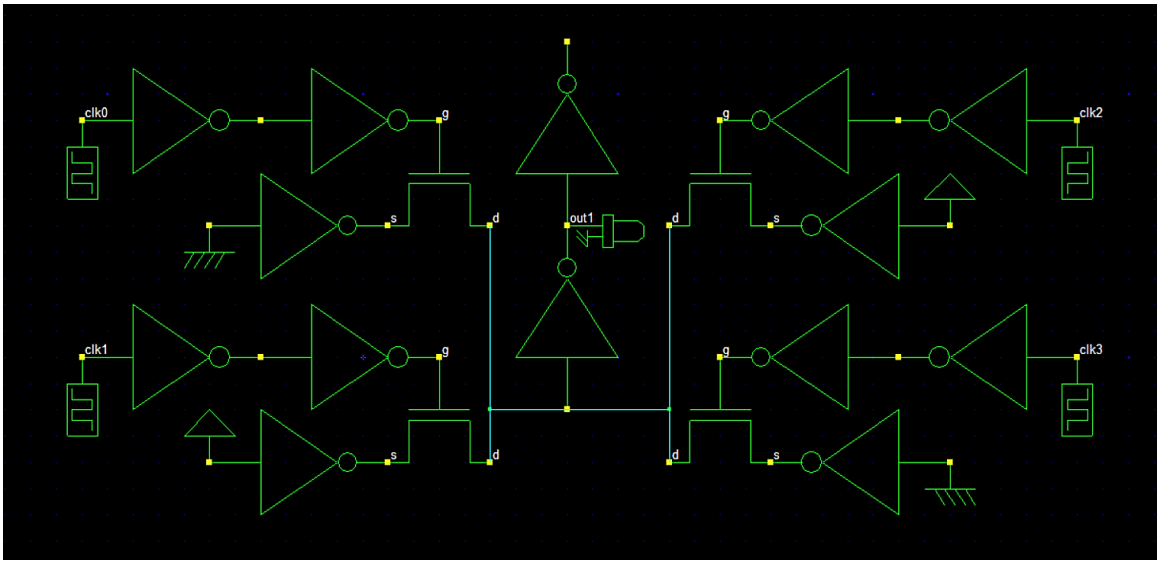
Видим, что и по форме сигнала, и по временным параметрам унитарный вариант несколько лучше существующего, но у существующего одна линия переменной, а у унитарного две и нужно еще два дополнительных инвертора, хотя по конфигурационной памяти функции они одинаковы.

4.3.2 Моделирование в Microwind известного элемента для вычисления логической функции на две переменные

Построенные схемы электрические функциональные 2-LUT, настроенные на реализацию логической функции инверсии входного сигнала показаны на рисунке 4.13.



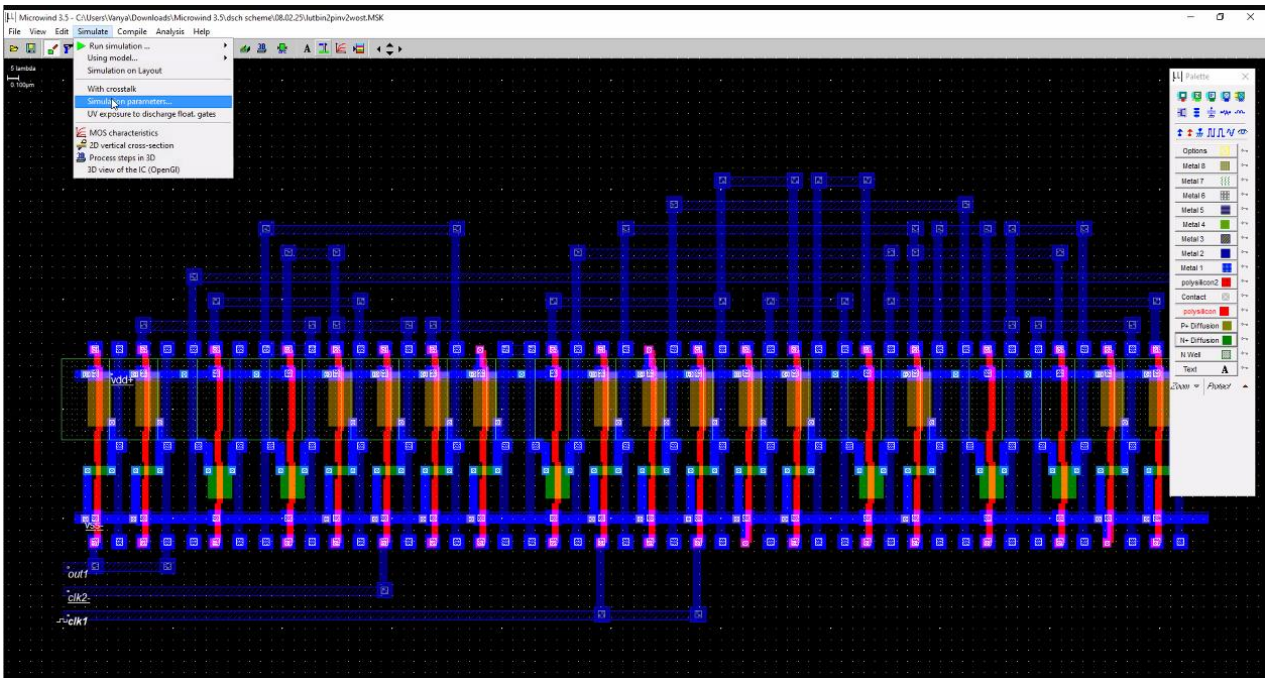
а)



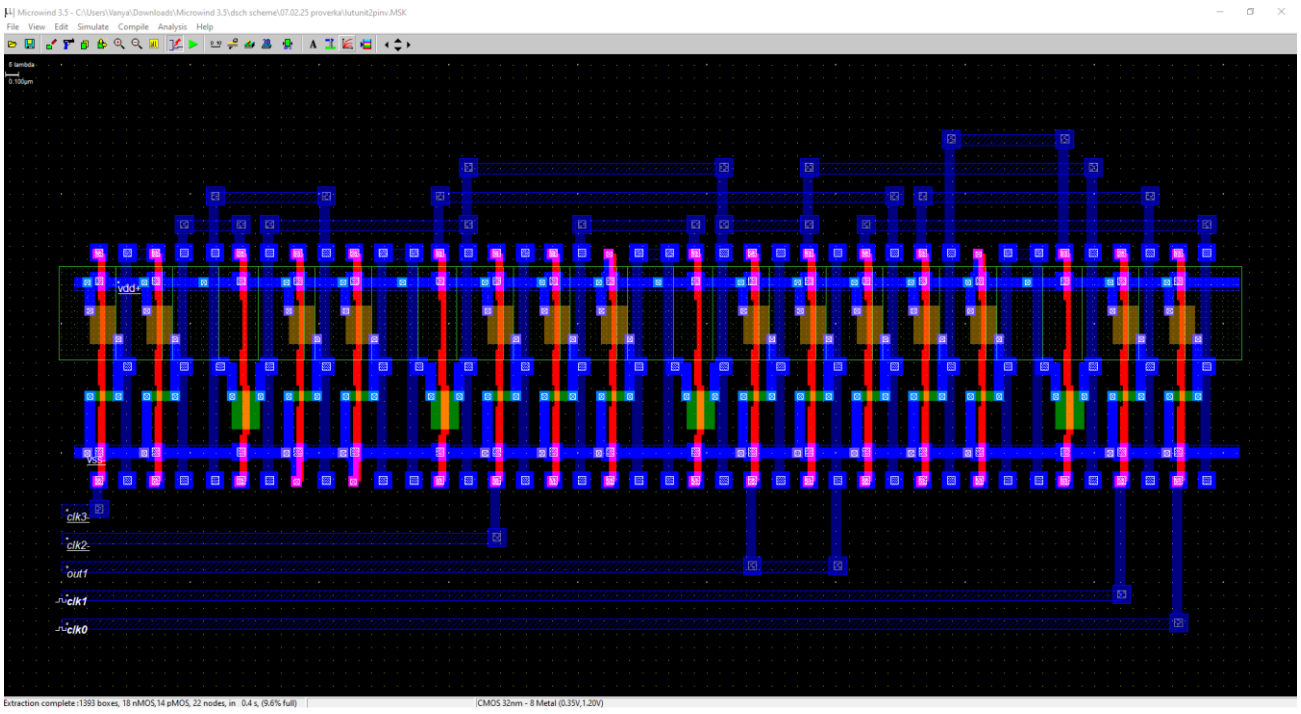
б)

Рисунок 4.13 – Схема электрическая функциональная 2-LUT, настроенная на реализацию логической функции инверсии входного сигнала Clk1- Clk2: а) позиционного; б) унитарного

Полученные соответствующие топологии показаны на рисунке 4.14.



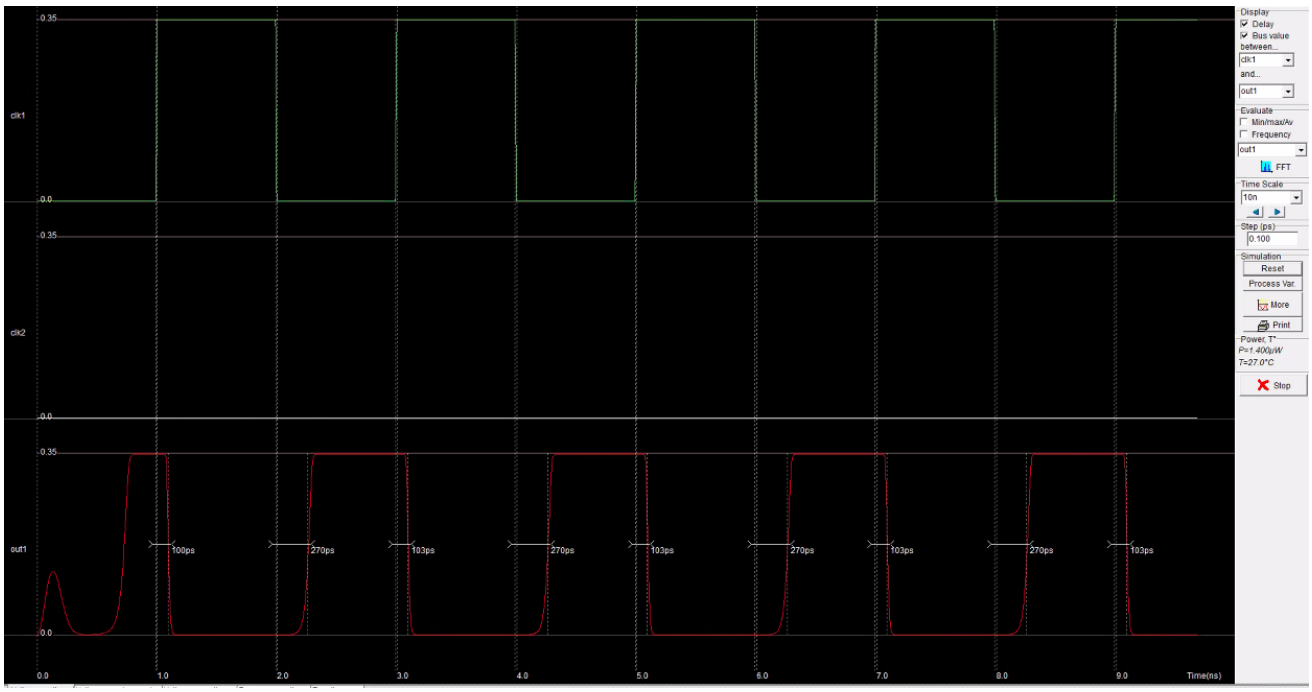
а)



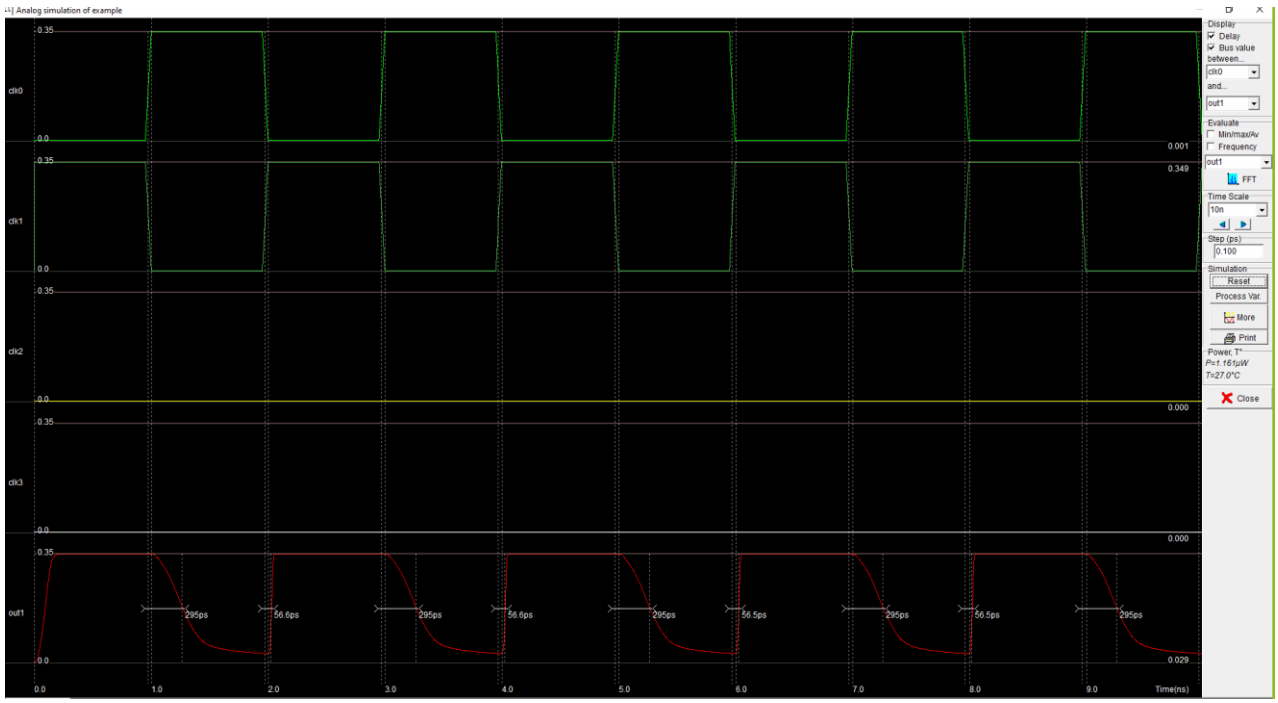
б)

Рисунок 4.14 – Топологии 2-LUT: а) позиционного; б) унитарного

Полученные соответствующие временные диаграммы показаны на рисунке 4.15.



а)

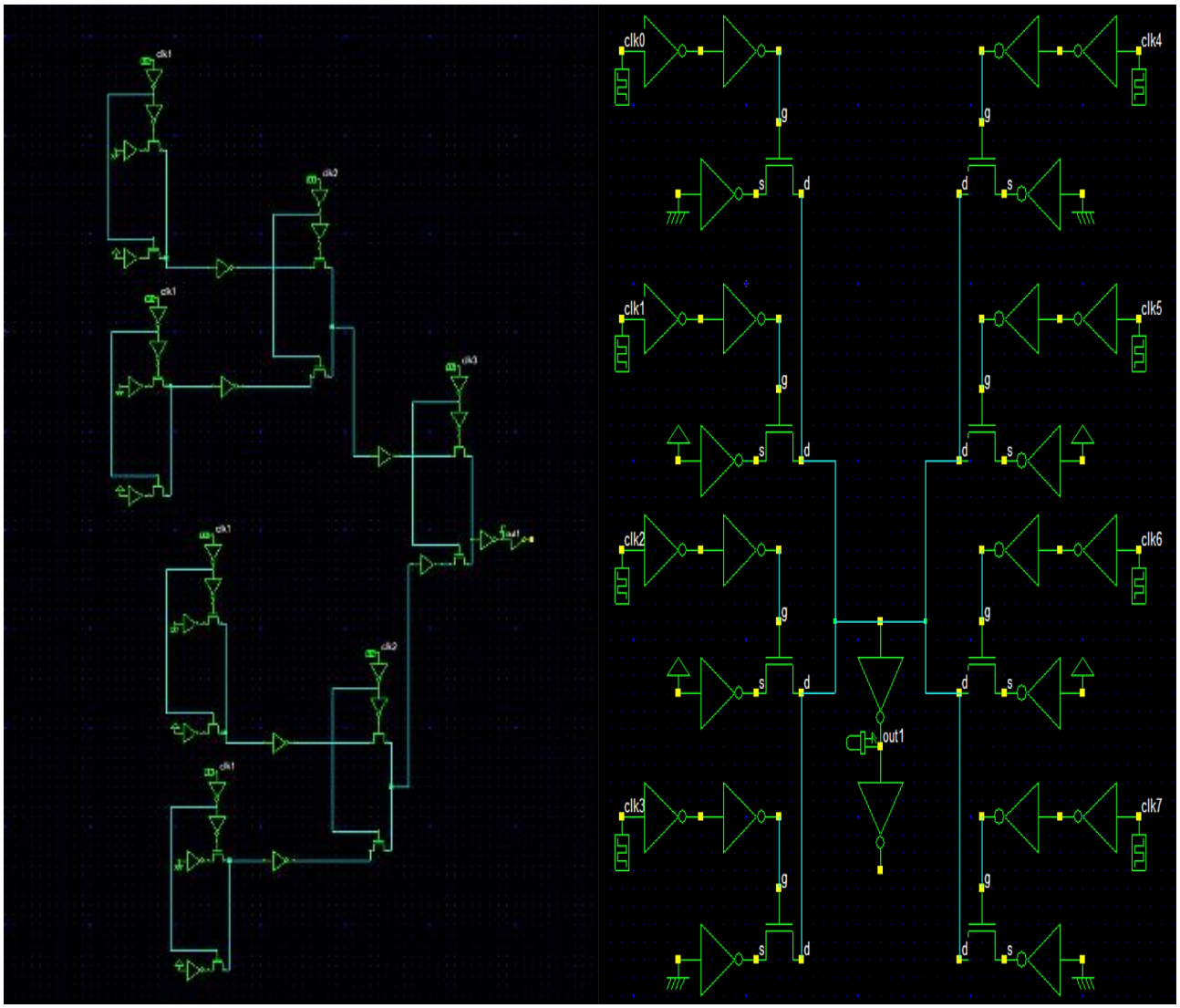


б)

Рисунок 4.15 – Временные диаграммы работы 2-LUT: а) позиционного; б) унитарного

4.3.3. Моделирование в Microwind известного элемента для вычисления логической функции на три переменные

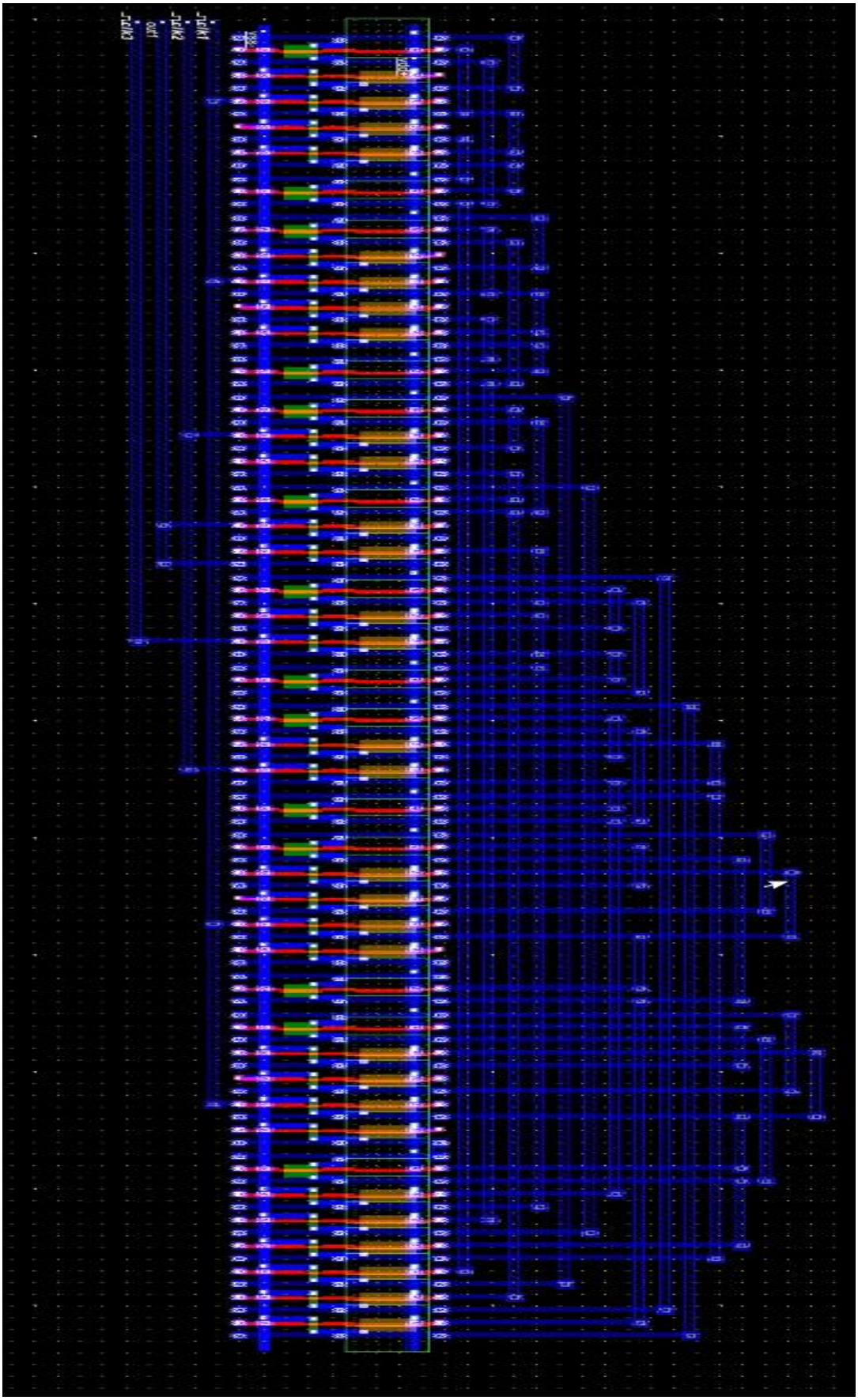
Построенные схемы электрические функциональные 3-LUT, настроенные на реализацию логической функции инверсии входного сигнала показаны на рисунке 4.16.



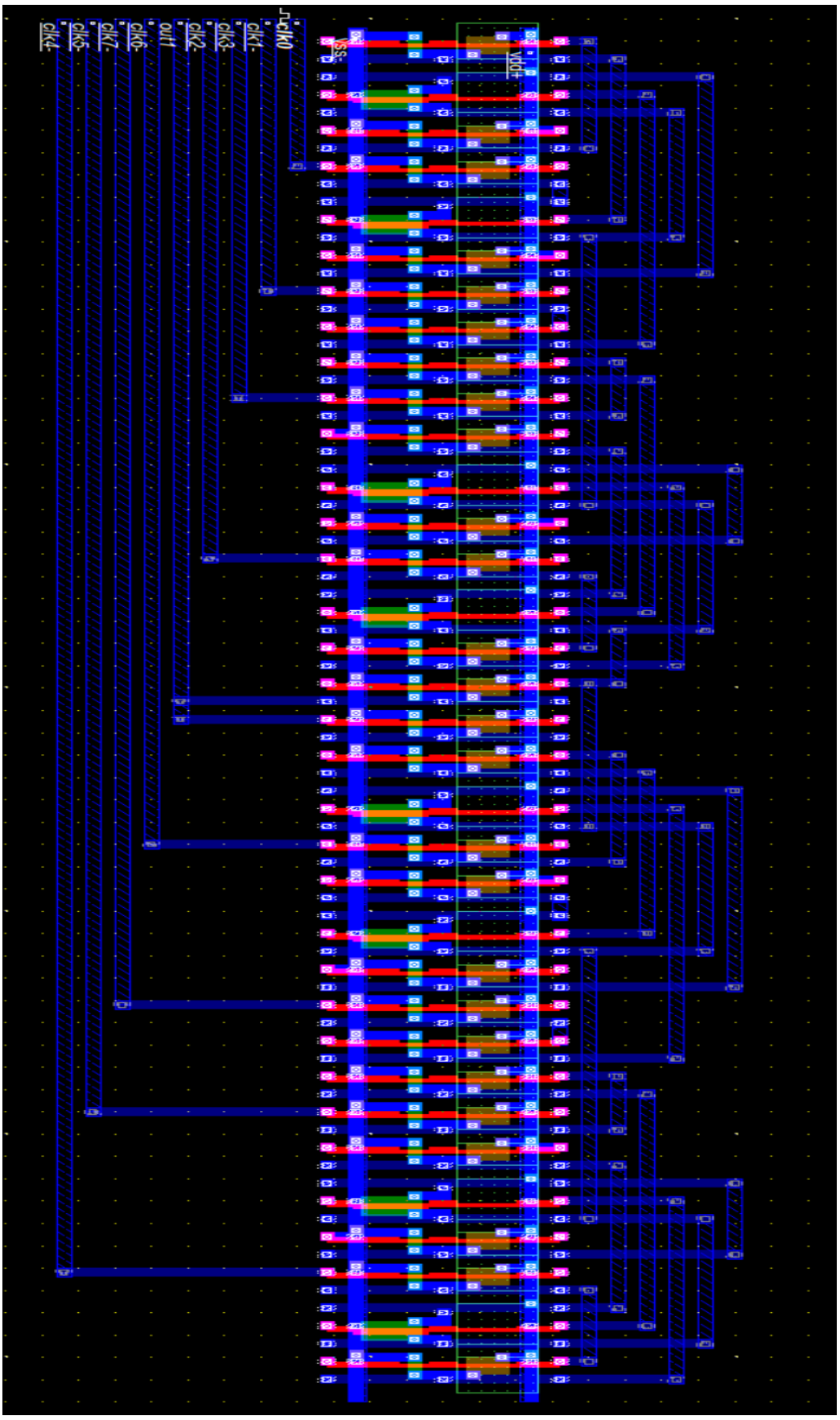
а) б)

Рисунок 4.16 – Схемы электрические функциональные 3-LUT: а) позиционного; б) унитарного

Полученные соответствующие топологии показаны на рисунке 4.17.



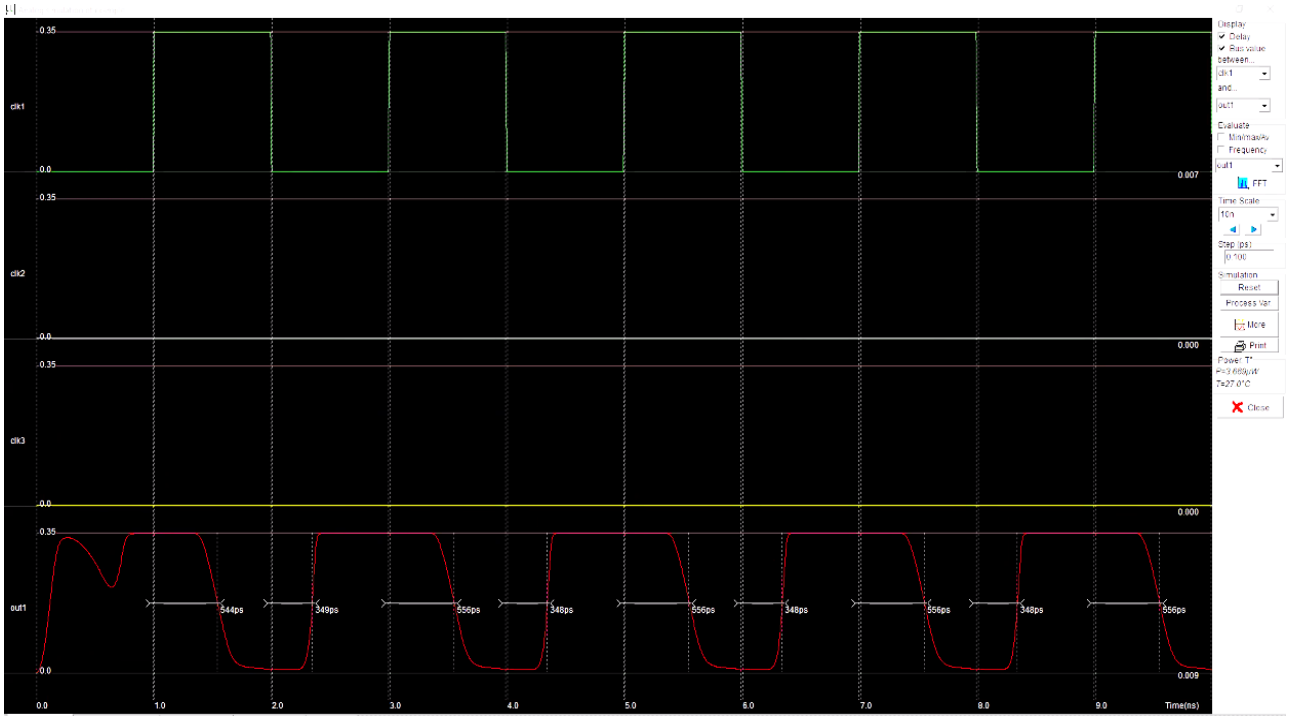
а)



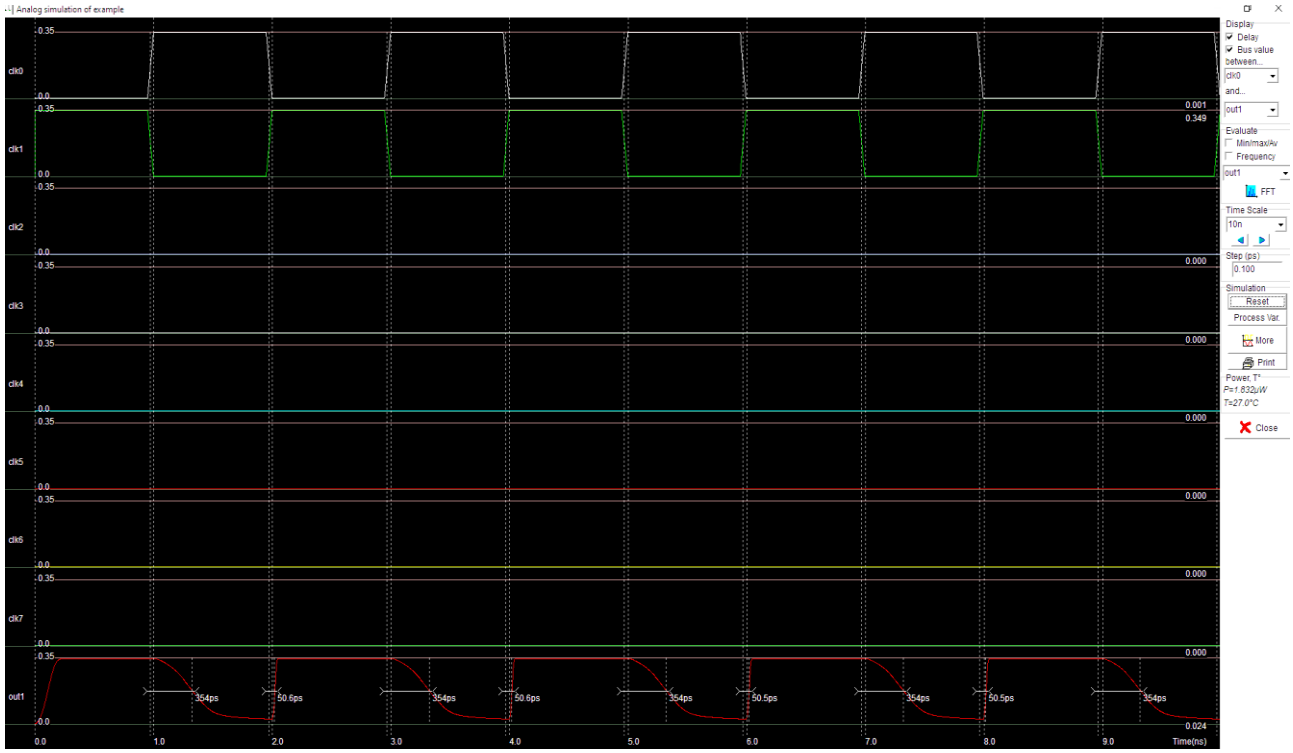
б)

Рисунок 4.17 – Топологии 3-LUT: а) позиционного; б) унитарного

Полученные соответствующие временные диаграммы показаны на рисунке 4.18.



а)



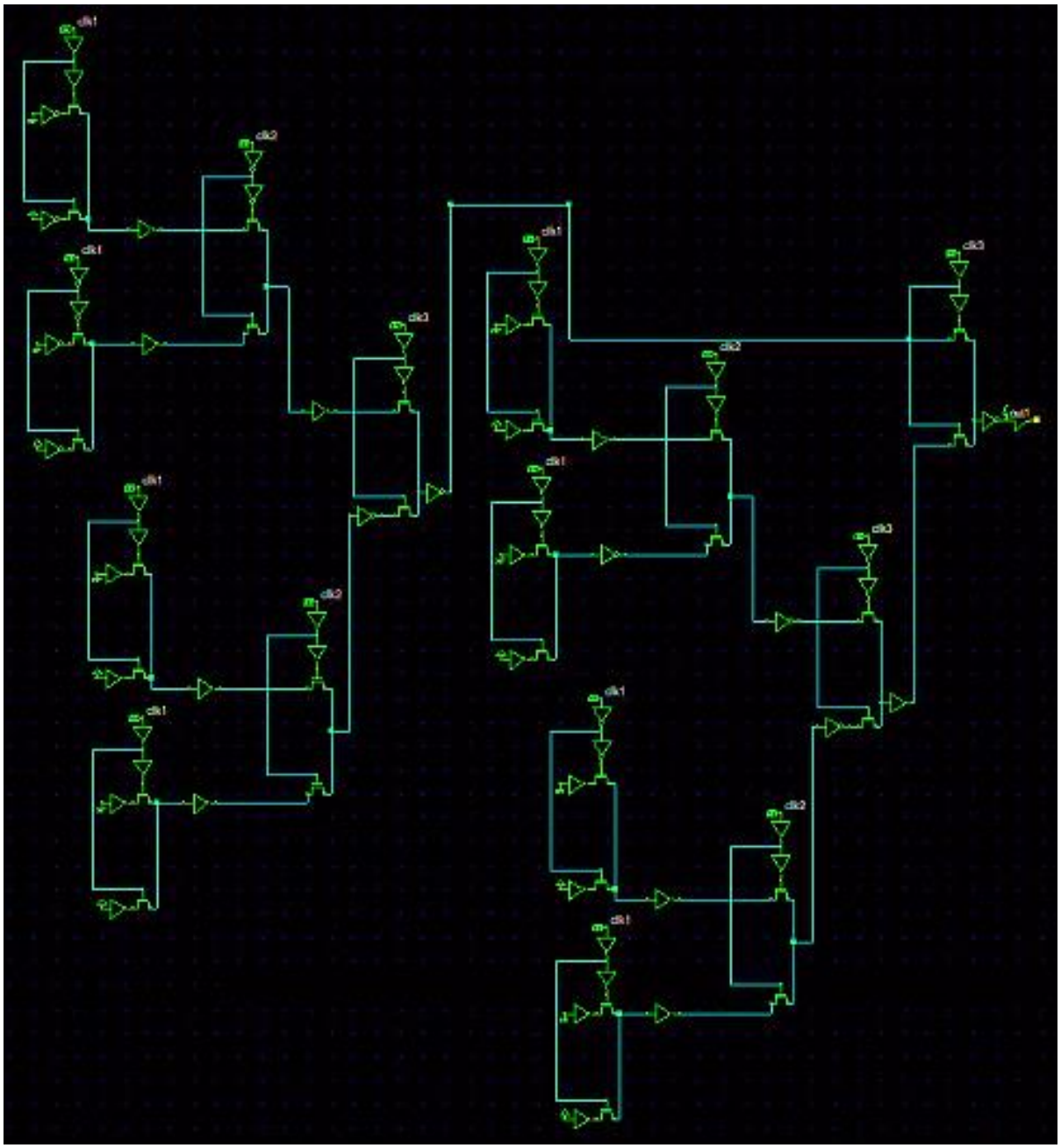
б)

Рисунок 4.18 – Временные диаграммы работы 3-LUT: а) позиционного; б) унитарного

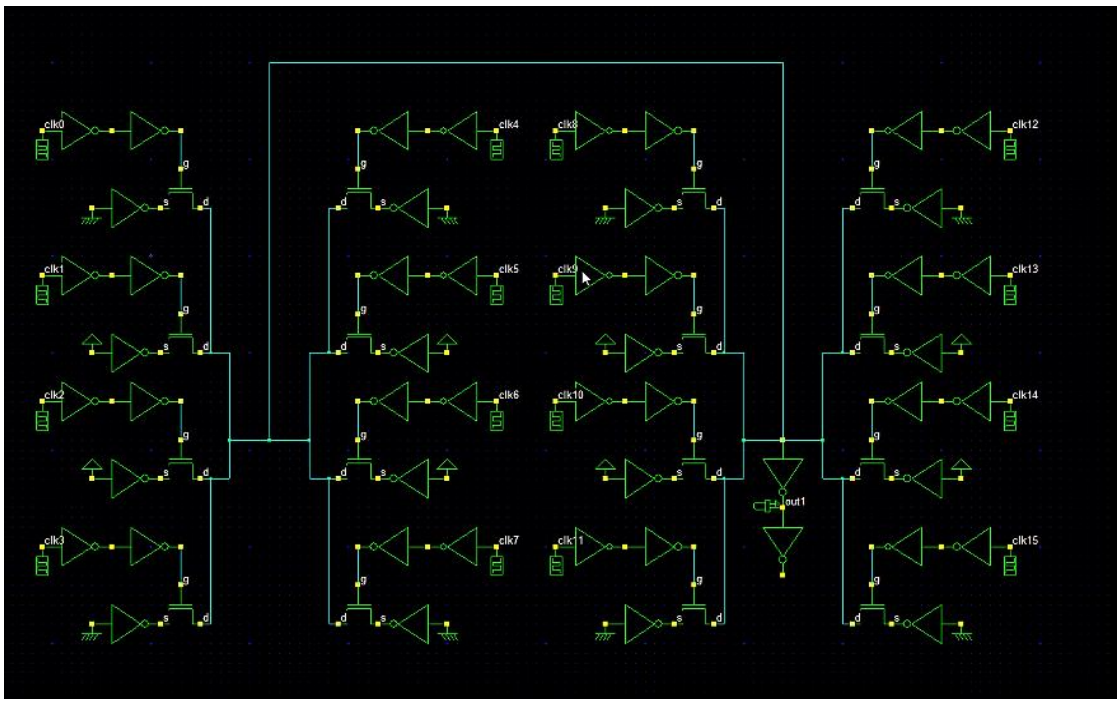
При анализе сигнала на временной диаграммы видно, что унитарный элемент имеет меньшую временную задержку, чем позиционный – 354 ps против 556 ps.

4.3.4 Моделирование в Microwind элемента для вычисления логической функции на четыре переменные

Построенные схемы электрические функциональные 4-LUT, настроенные на реализацию логической функции инверсии входного сигнала показаны на рисунке 4.19.



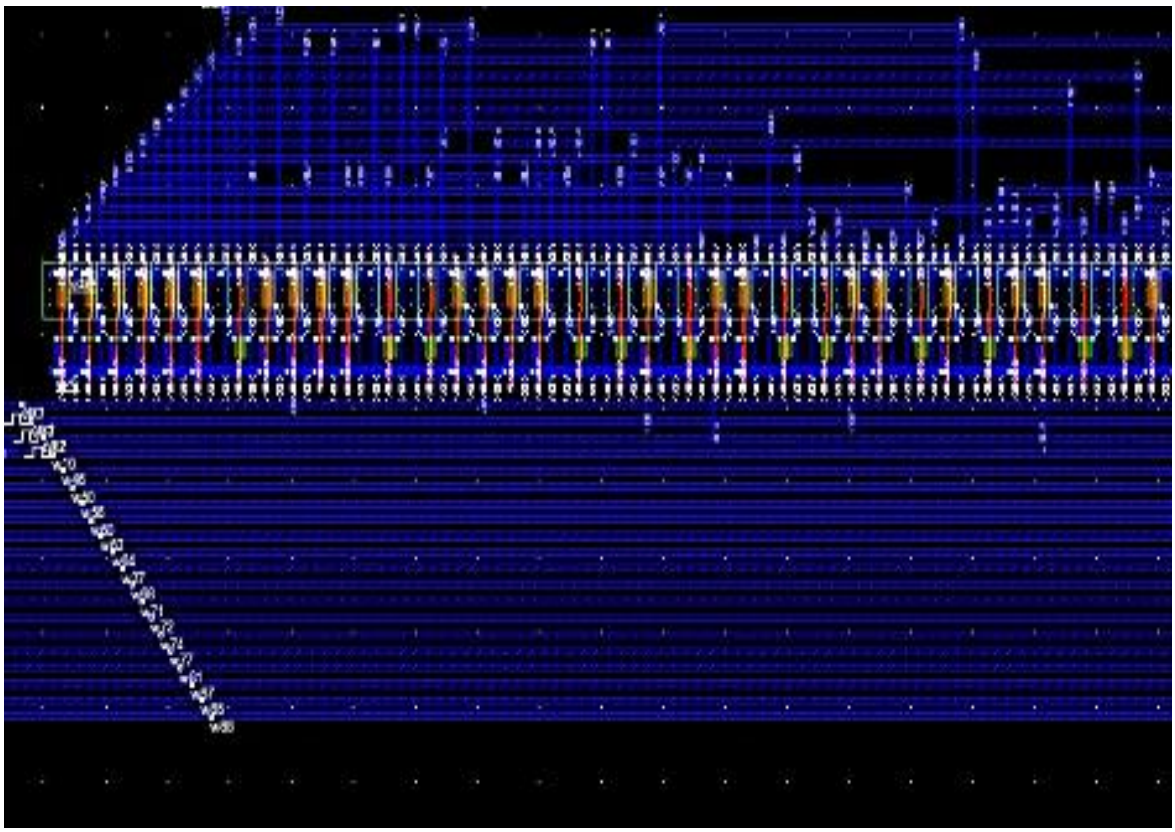
а)



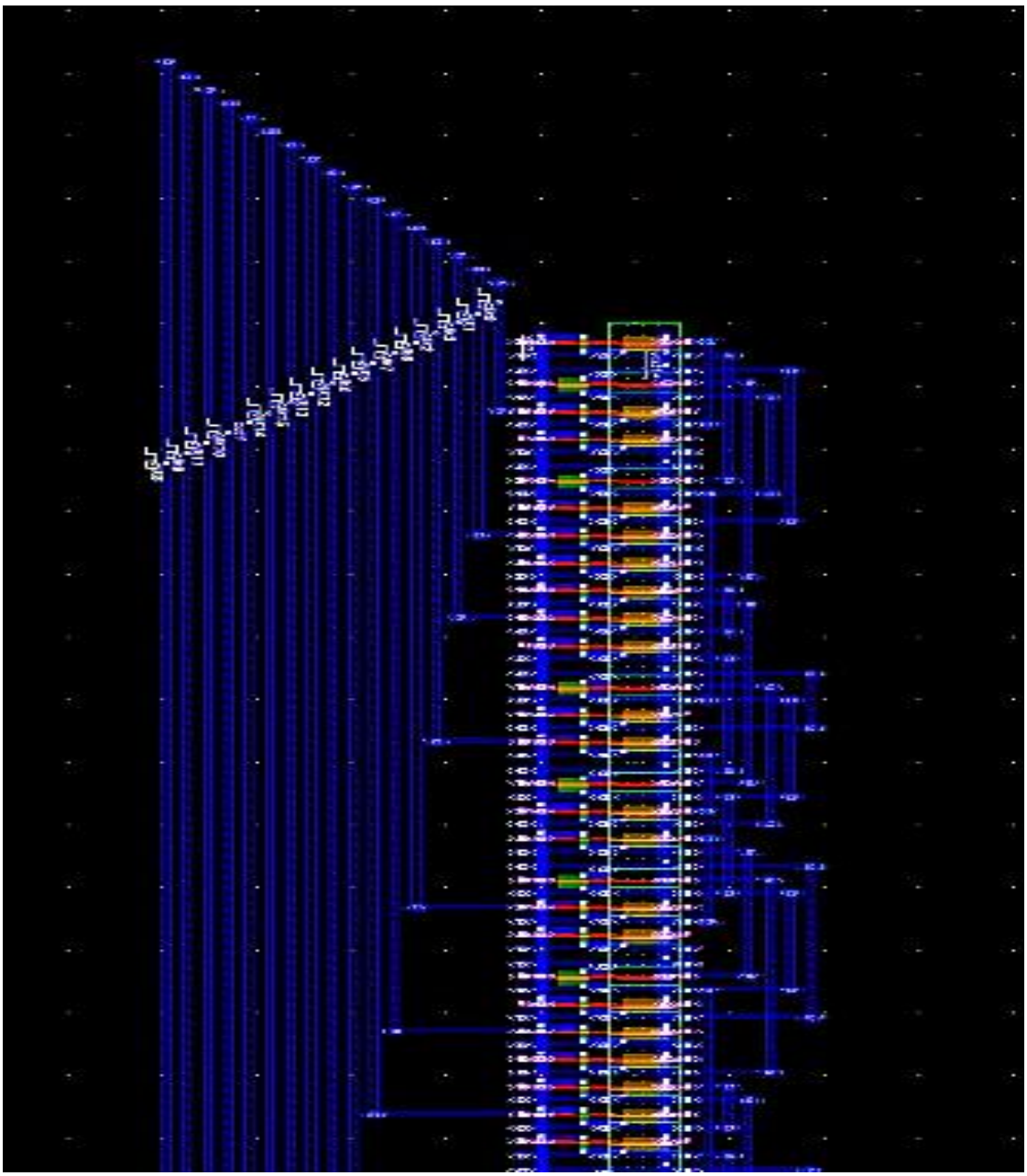
б)

Рисунок 4.19 – Схемы электрические функциональные 4-LUT: а) позиционного; б) унитарного

Полученные соответствующие топологии показаны на рисунке 4.20.



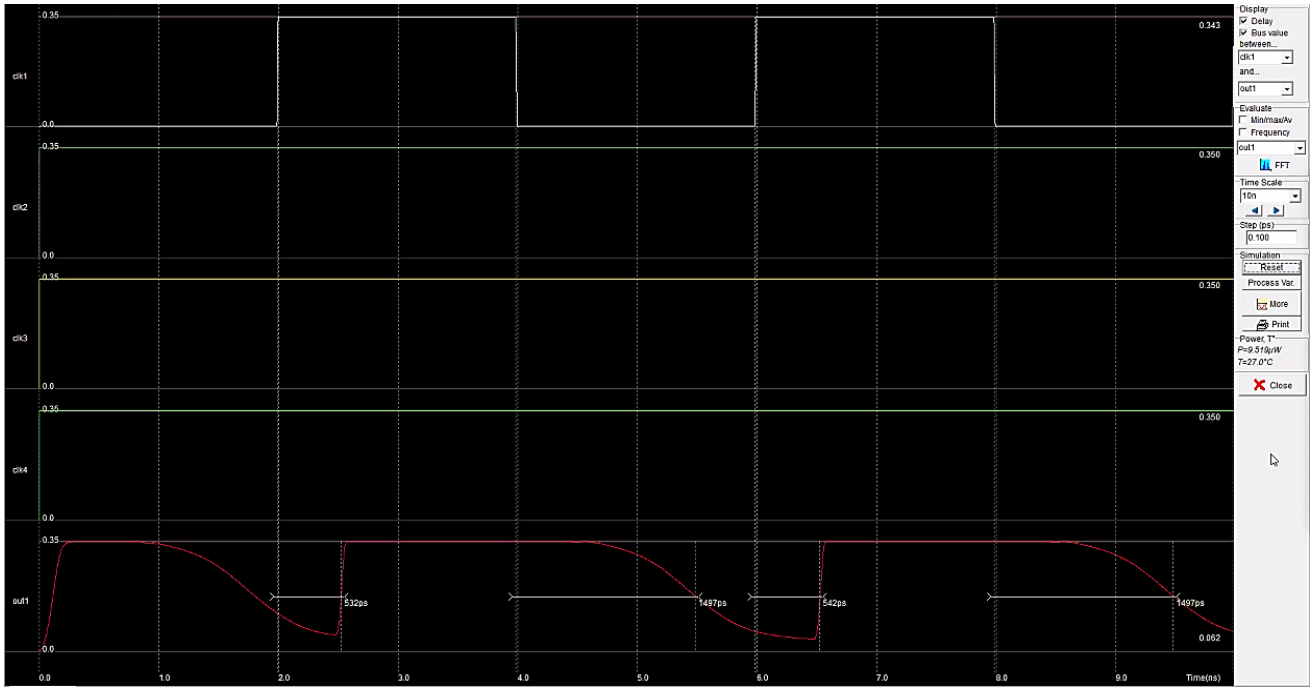
а)



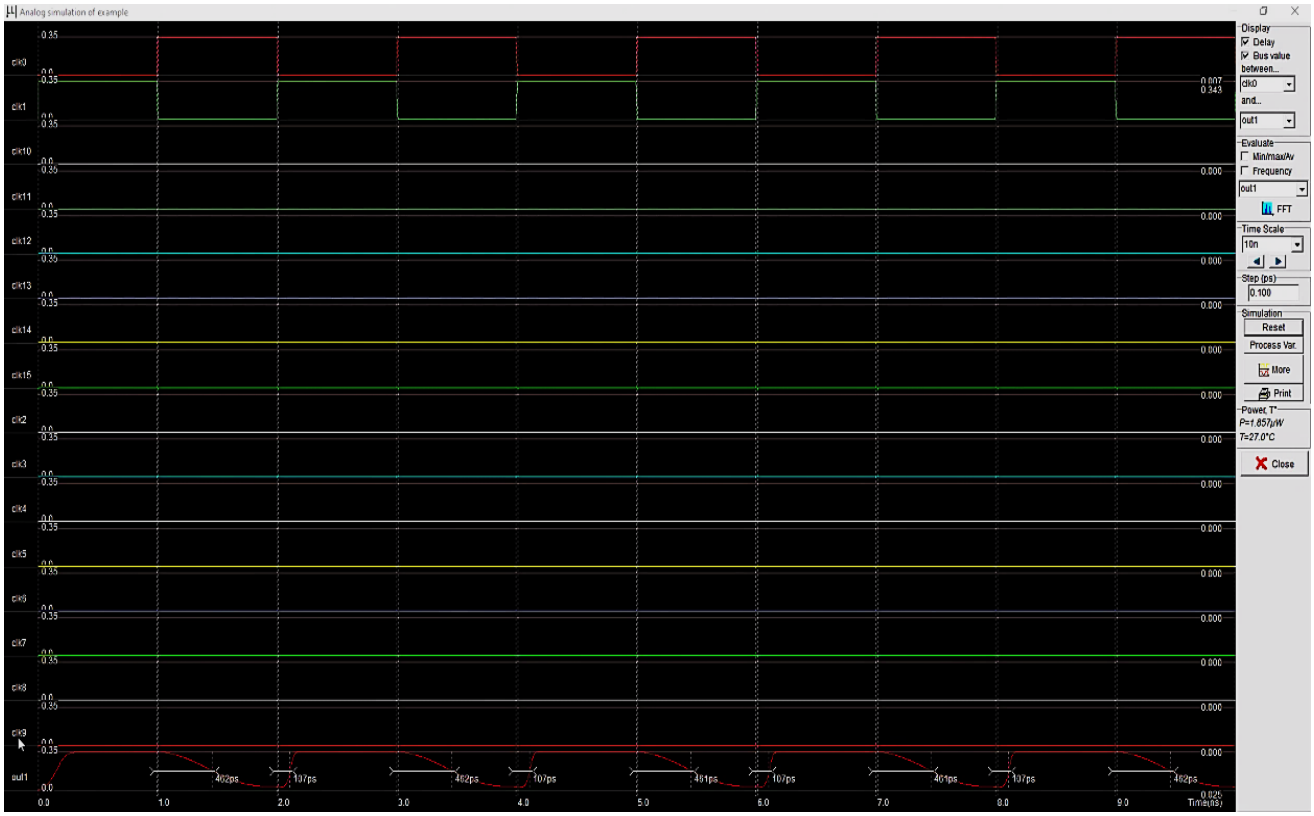
б)

Рисунок 4.20 – Топологии 4-LUT: а) фрагмент схемы позиционного элемента; б) фрагмент схемы унитарного элемента

Полученные соответствующие временные диаграммы показаны на рисунке 4.21.



а)



б)

Рисунок 4.21 – Временные диаграммы работы 3-LUT: а) известного; б) унитарного

4.3.5 Моделирование в Microwind предложенного комбинированного элемента для вычисления логической функции на три переменные

Предлагаемый комбинированный вариант имеет позиционную часть (1-LUT) и унитарную, включающую два унитарных 2-LUT – рисунок 4.22.

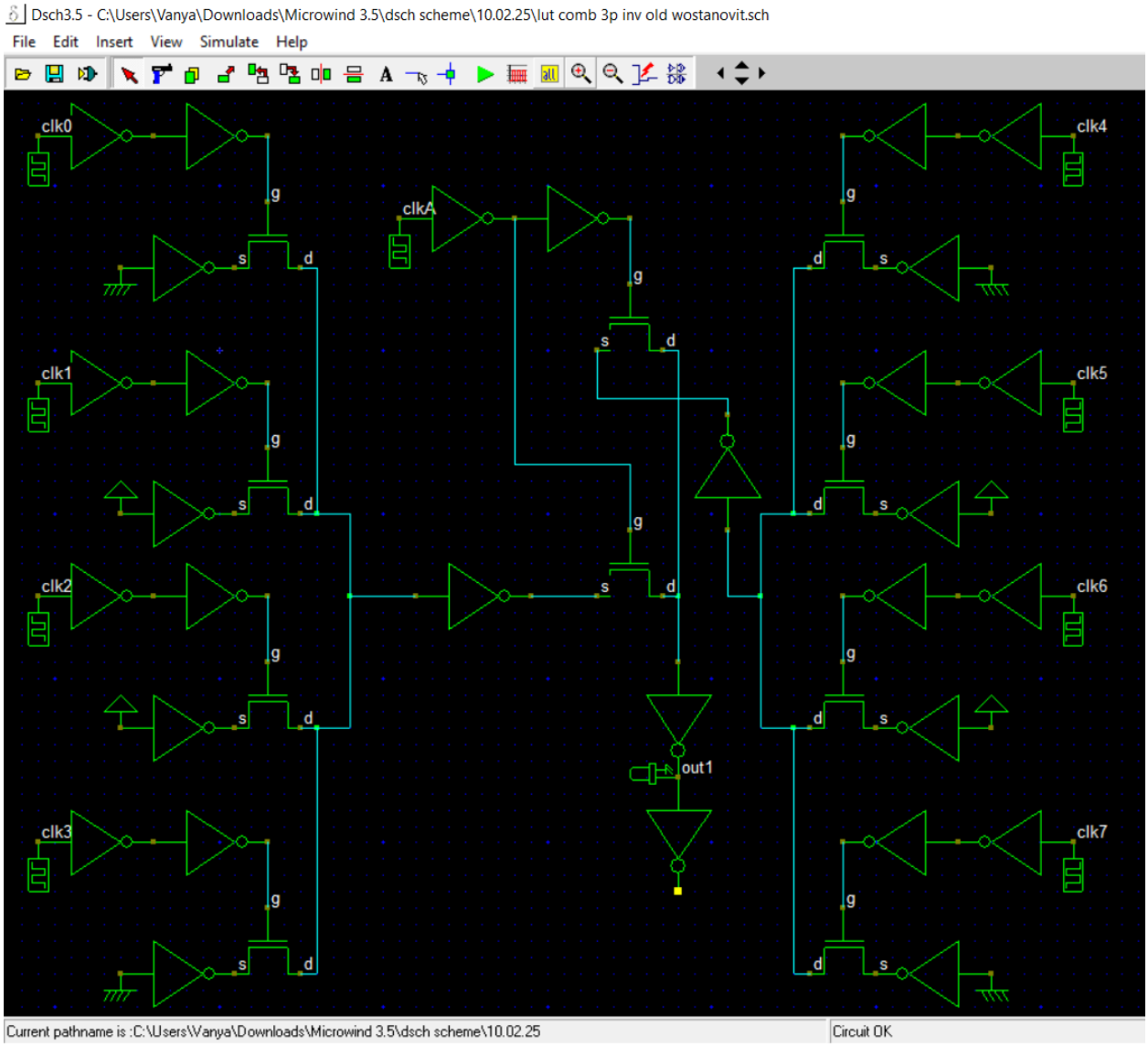


Рисунок 4.22 – Схема электрическая функциональная предлагаемого комбинированного 3-LUT

Топология предлагаемого комбинированного 3-LUT – рисунок 4.23 [84].

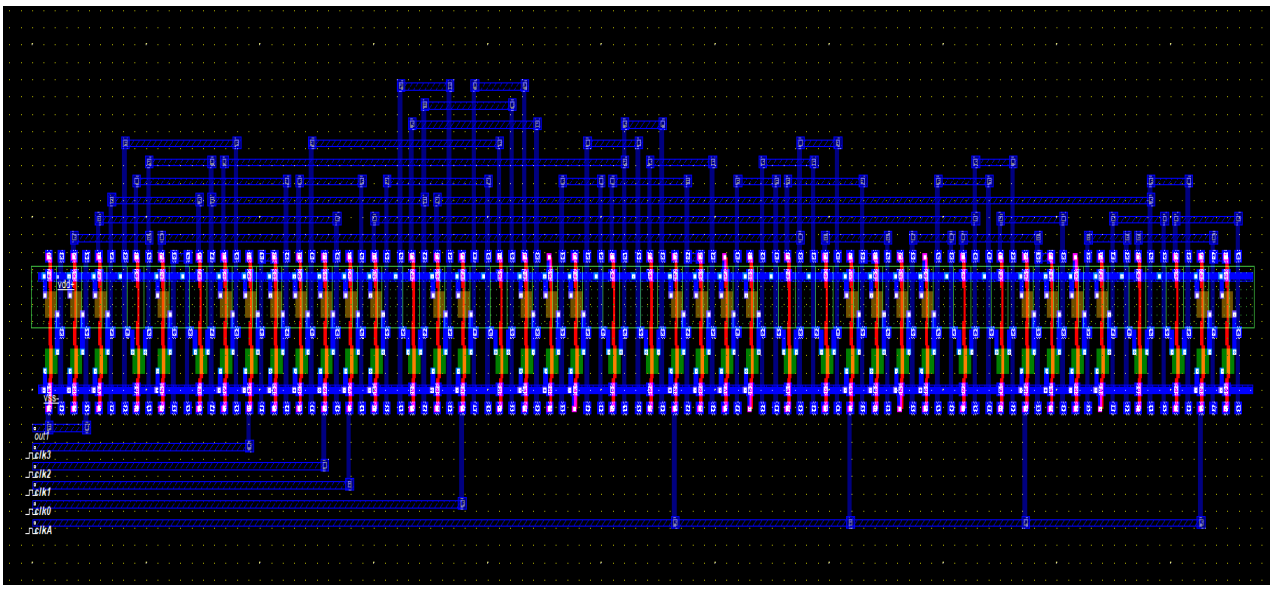


Рисунок 4.23 – Топология предлагаемого комбинированного 3-LUT

Временная диаграмма работы предлагаемого комбинированного элемента 3-LUT – рисунок 4.24.

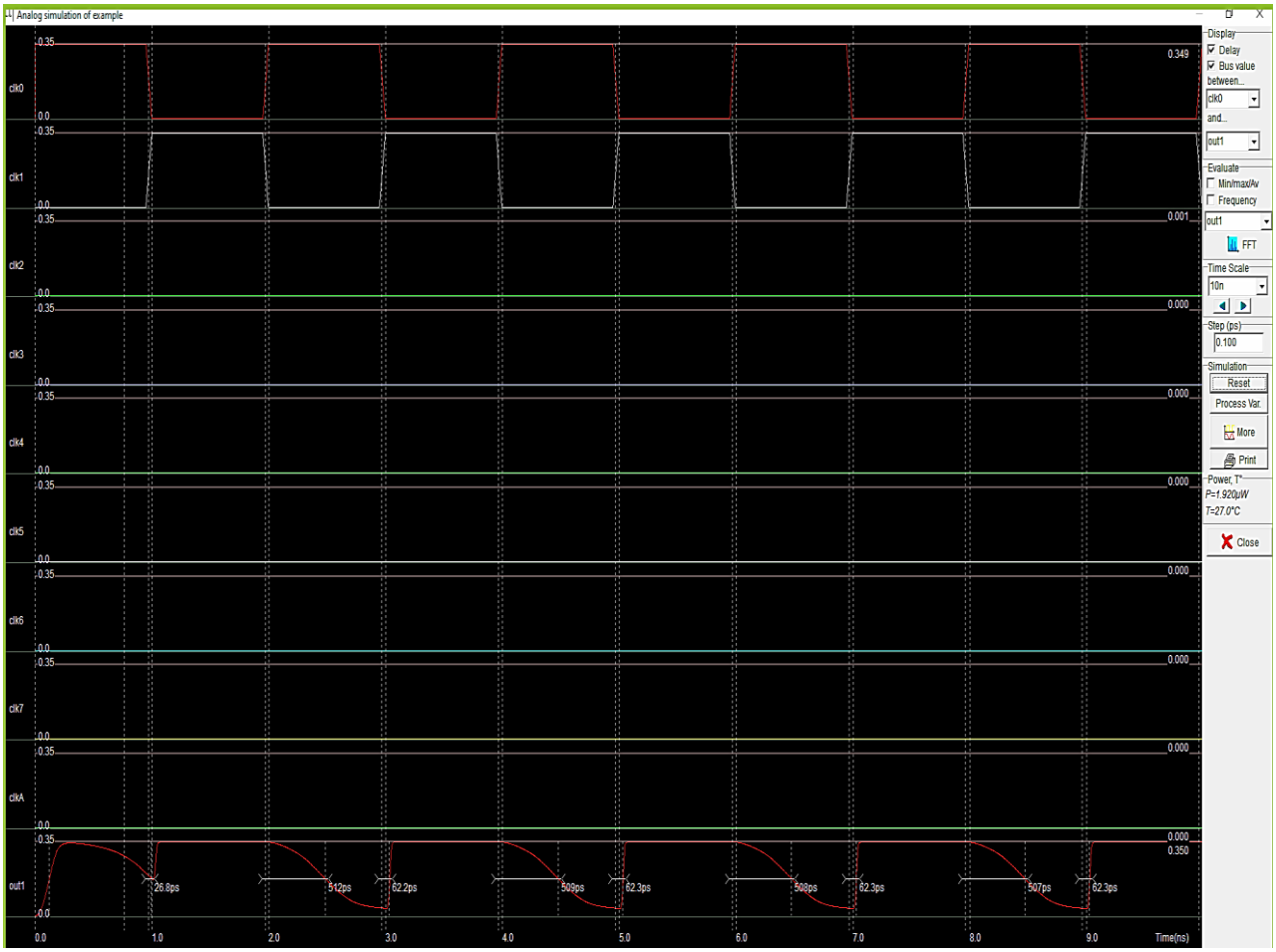


Рисунок 4.24 – Временная диаграмма работы предлагаемого комбинированного 3-LUT

Второй вариант комбинированного 3-LUT (унитарный блок справа) показан на рисунке 4.25. Топология второго варианта комбинированного 3-LUT (унитарный блок справа) показана на рисунке 4.26, а временные диаграммы – на рисунке 4.27.

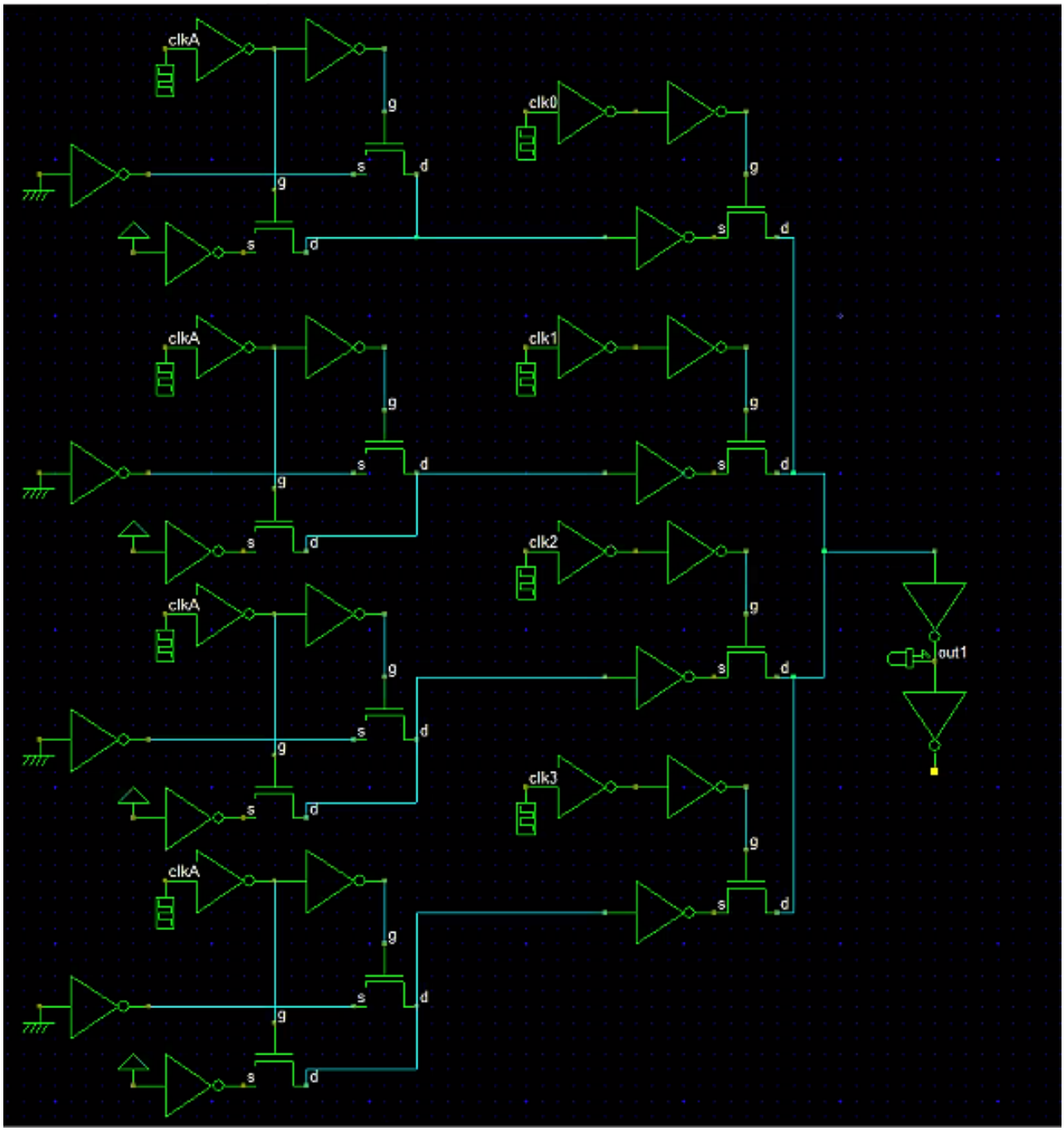


Рисунок 4.25 – Второй вариант предлагаемого комбинированного 3-LUT (унитарный блок справа)

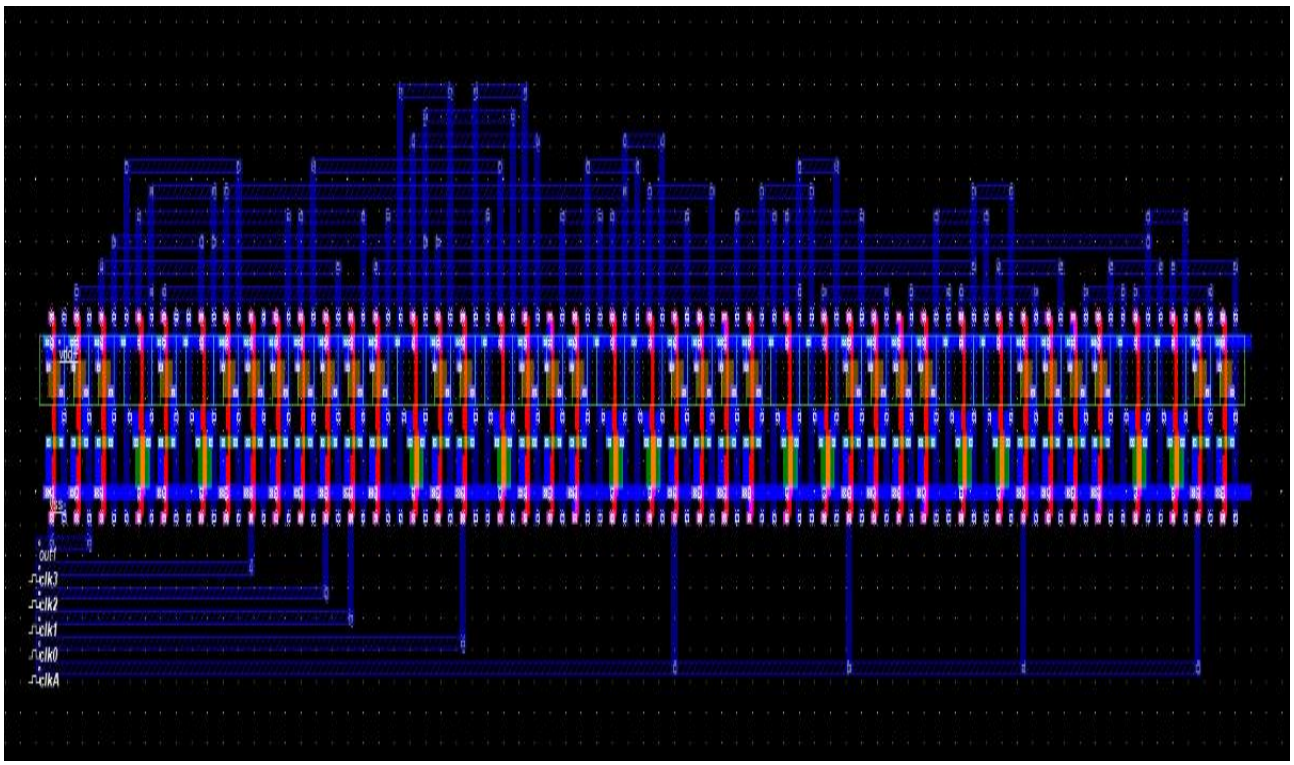
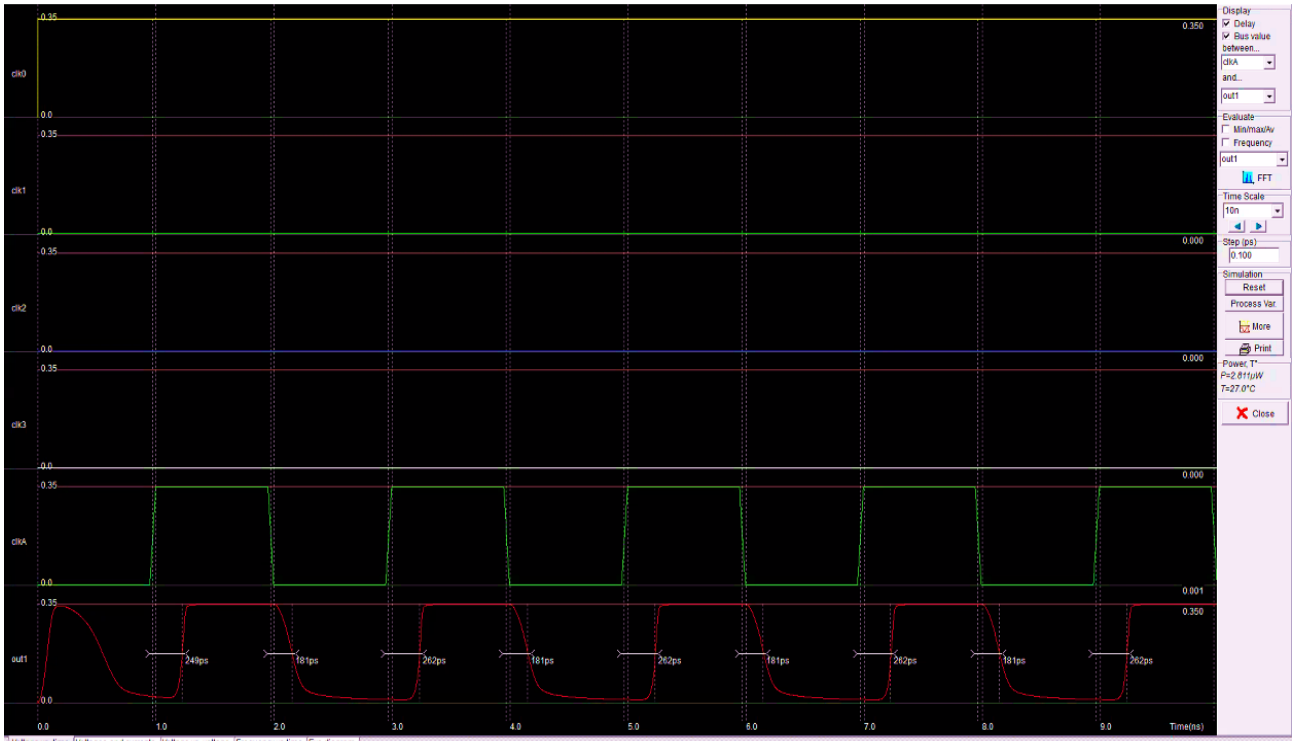
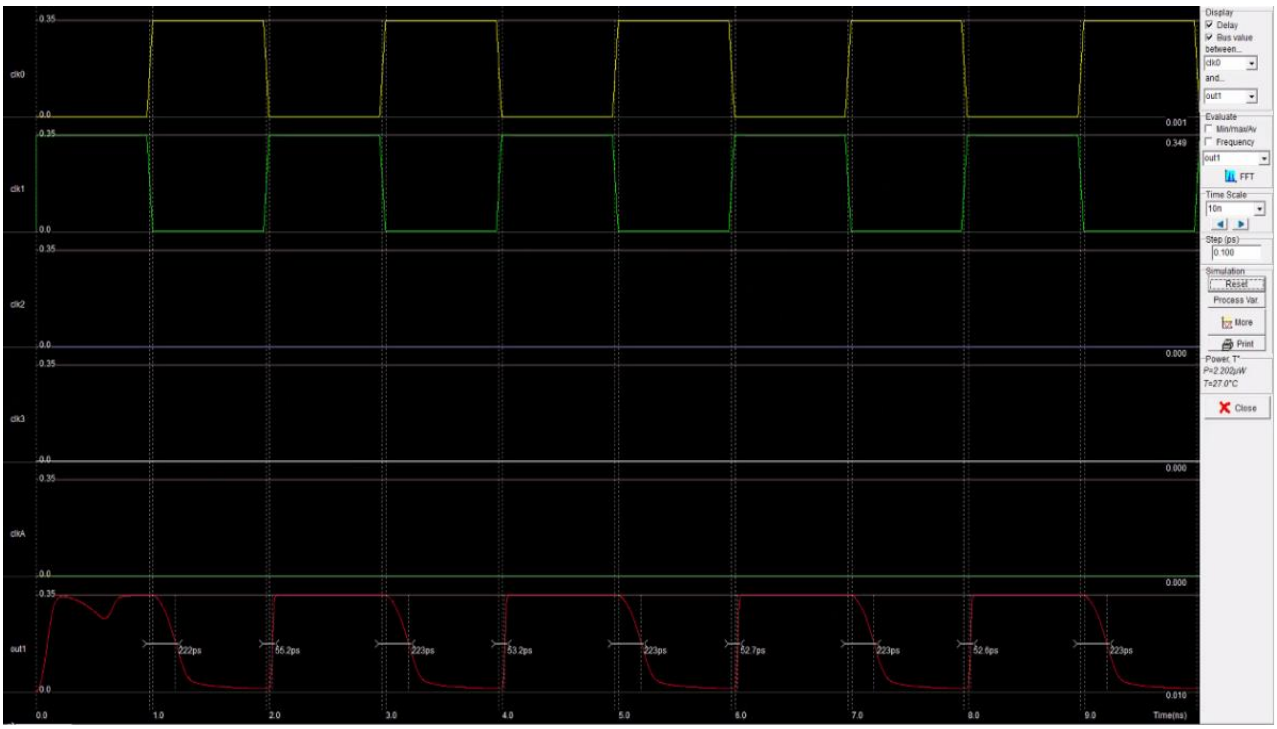


Рисунок 4.26 – Топология второго варианта предлагаемого комбинированного 3-LUT

Временные диаграммы предлагаемых комбинированных элементов представлены на рисунке 4.27.



а)



б)

Рисунок 4.27 – Временная диаграмма второго варианта предлагаемого комбинированного 3-LUT: а) при подаче импульсов на вход CLKA; б) при подаче ортогональных импульсов на входы CLK0, CLK1

4.3.6 Моделирование в Microwind предложенного комбинированного элемента для вычисления логической функции на четыре переменные

Предлагаемый комбинированный элемент на четыре переменных имеет позиционную часть (1-LUT) и унитарную, включающую два унитарных блока по 8 унитарных разрядов – рисунки 4.28, 4.29, 4.30.

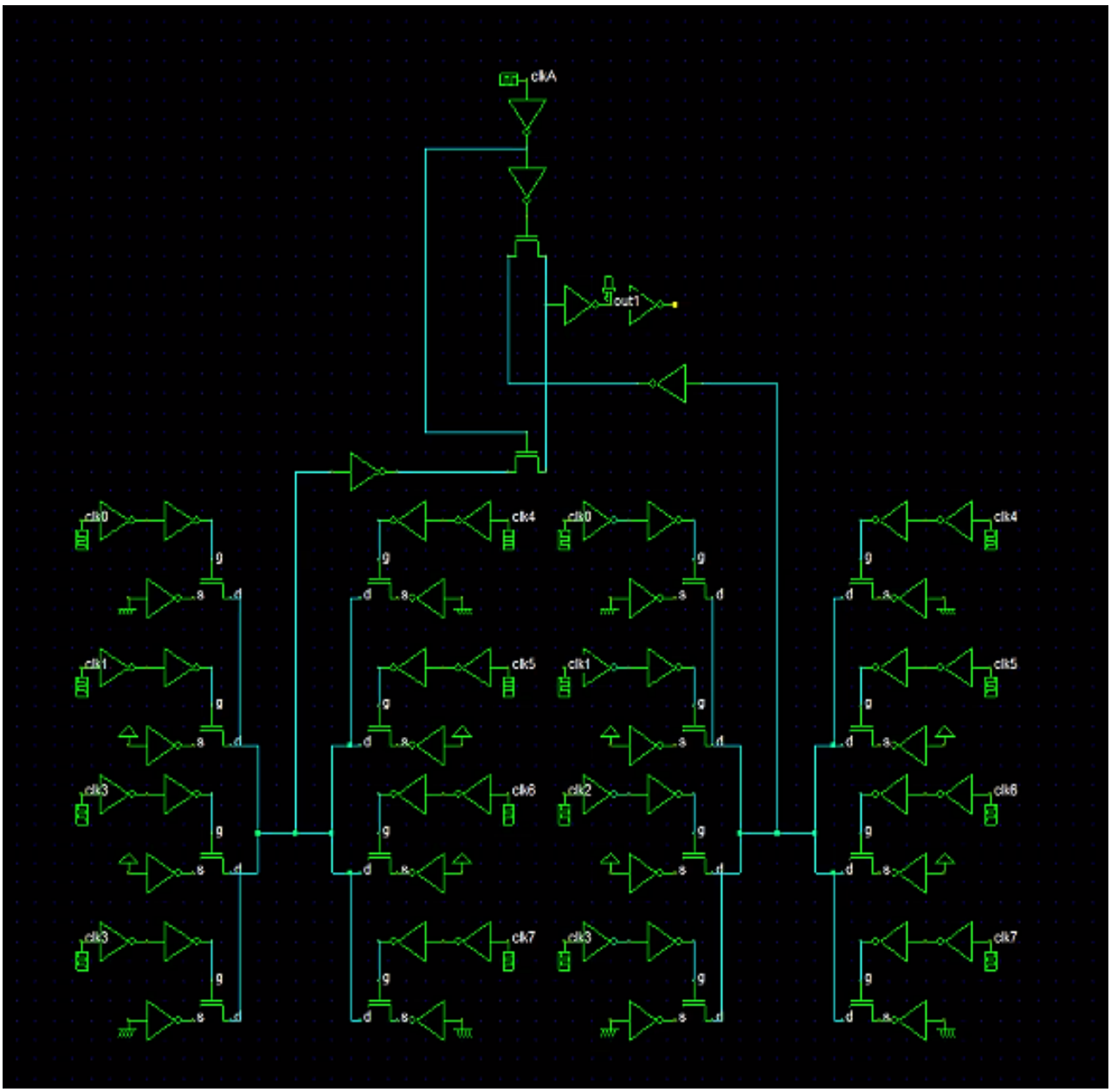


Рисунок 4.28 – Предлагаемый комбинированный 4-LUT №1 (два унитарных блока слева)

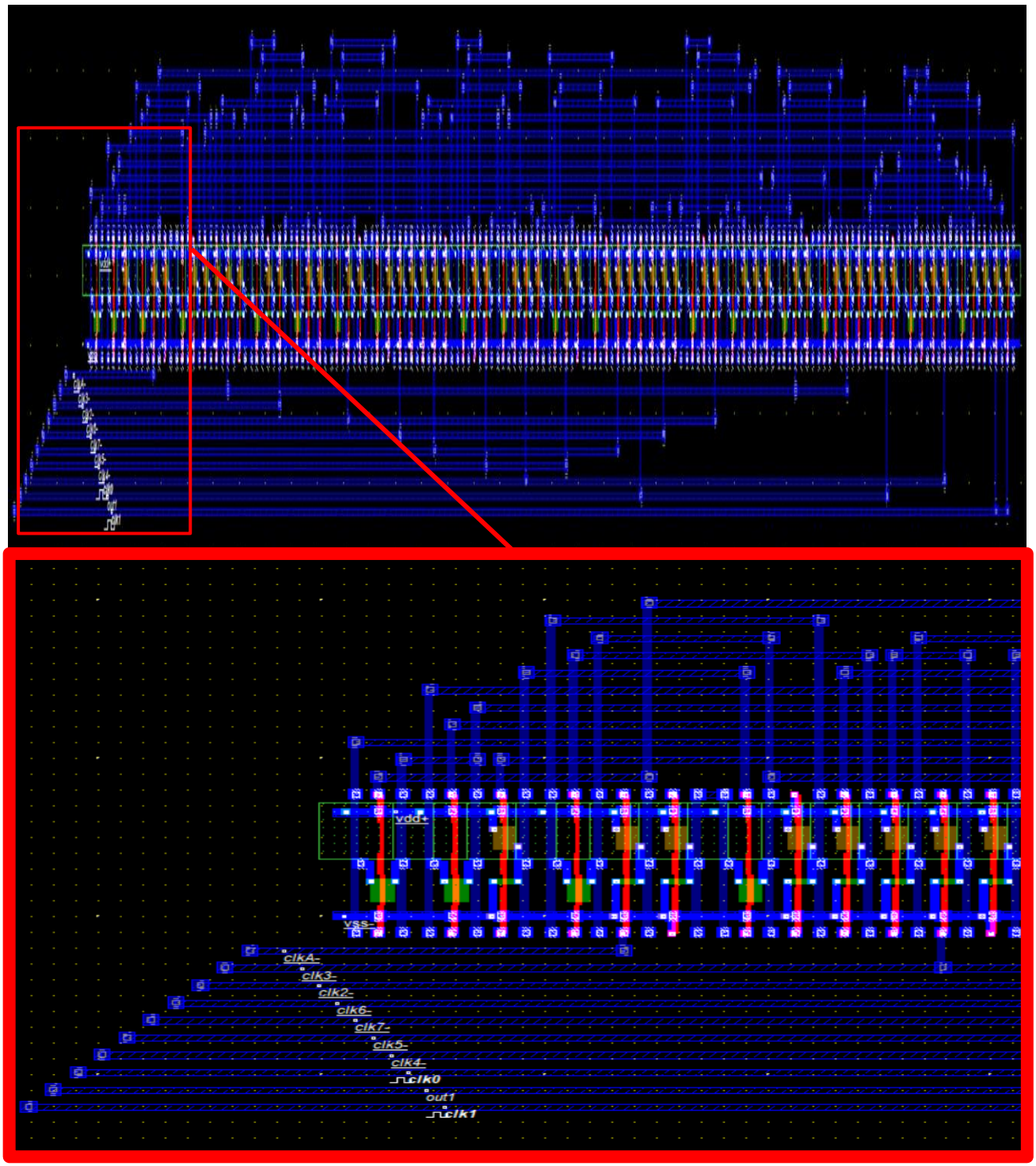


Рисунок 4.29 – Топология первого варианта предлагаемого комбинированного 4-LUT №1 и ее фрагмент

Временная диаграмма первого варианта предлагаемого комбинированного 4-LUT показана на рисунке 4.30.

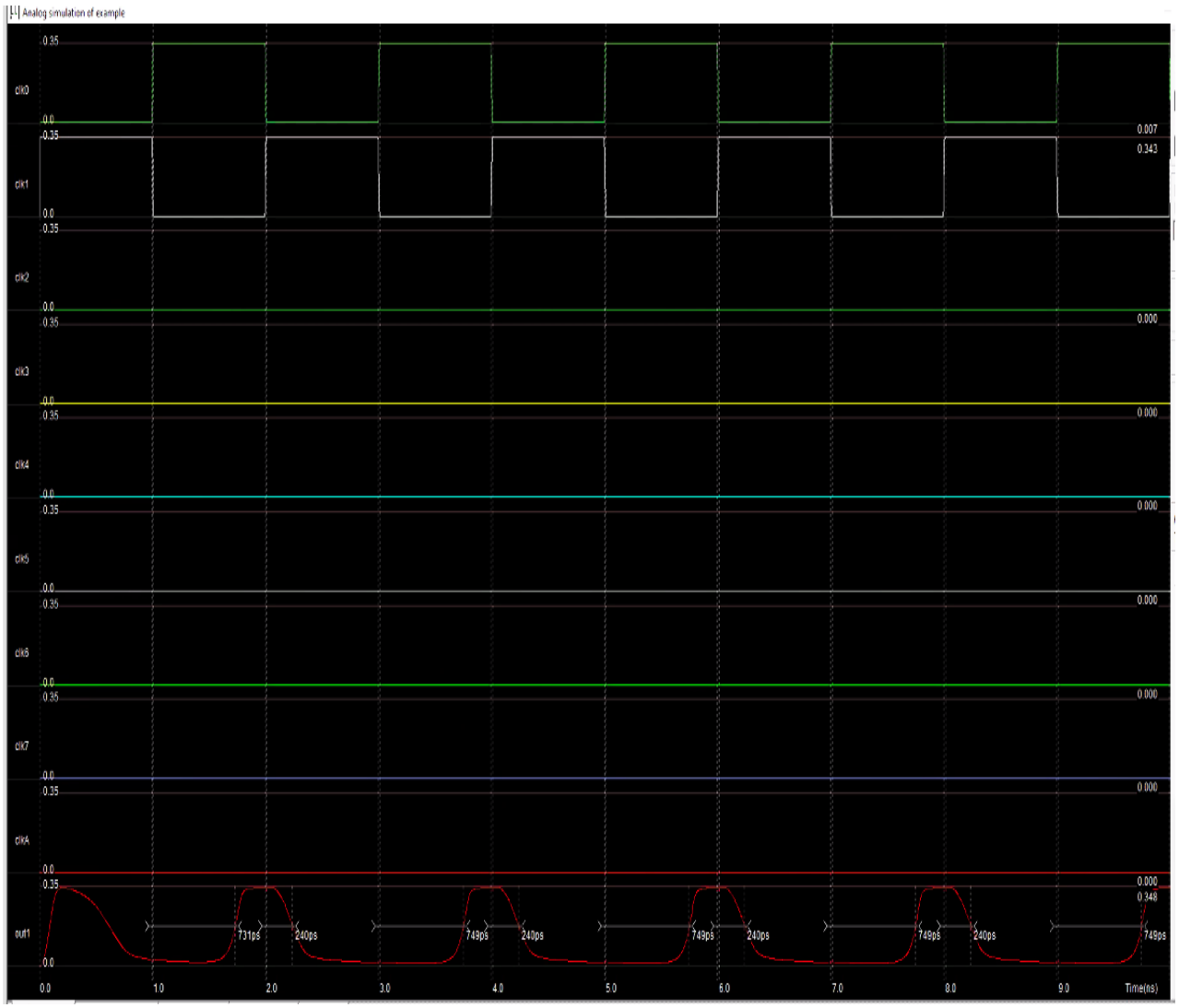


Рисунок 4.30 – Временная диаграмма первого варианта предлагаемого комбинированного 4-LUT№1

Второй вариант комбинированного элемента на четыре переменных имеет восемь бинарных частей (1-LUT) и унитарную, включающую один унитарный блок на 8 унитарных переменных – рисунки 4.31, 4.32, 4.33.

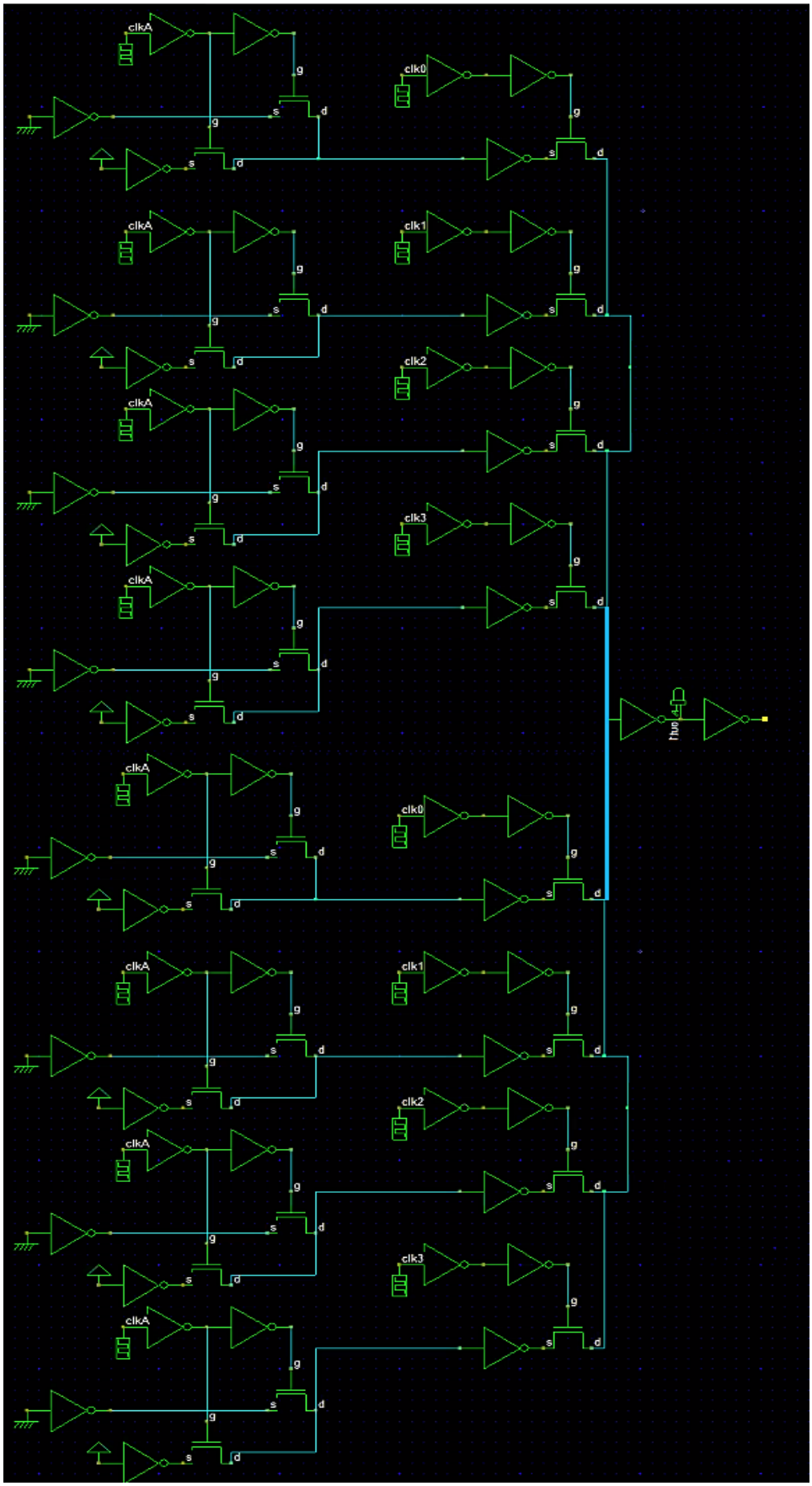


Рисунок 4.31 – Предлагаемый комбинированный 4-LUT№2 (один унитарный блок справа)

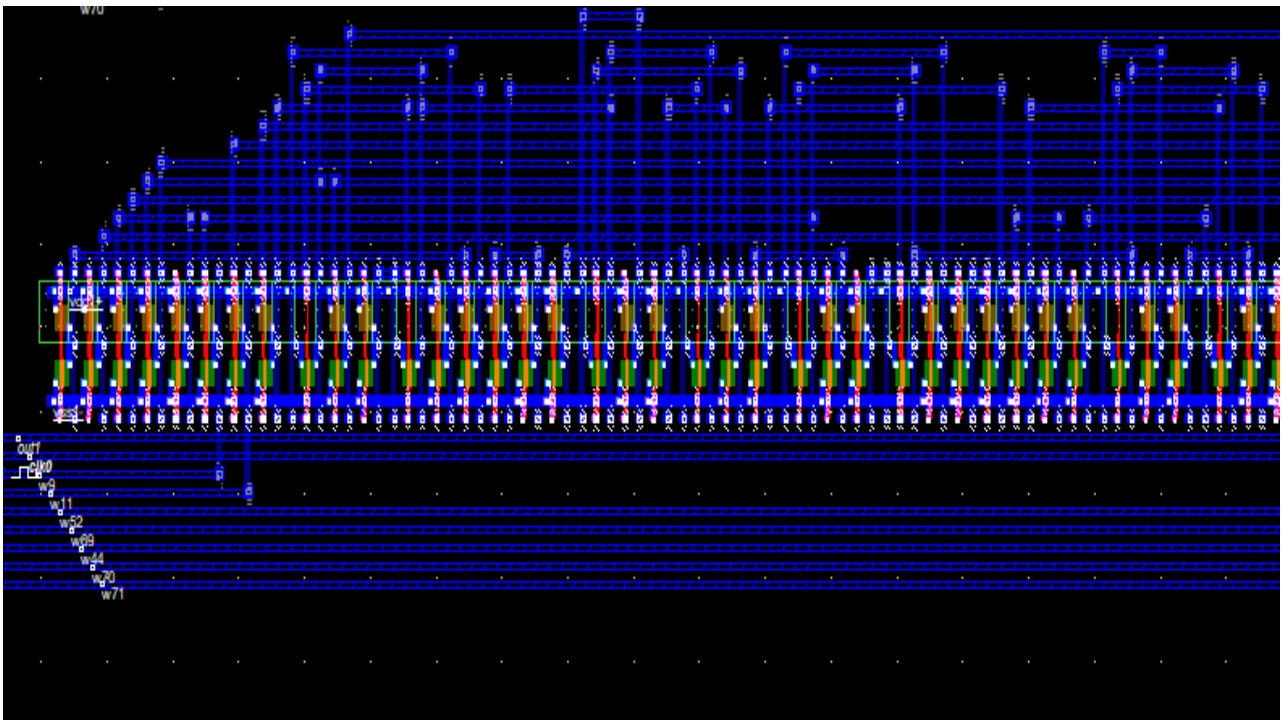


Рисунок 4.32 – Фрагмент топологии второго варианта предлагаемого комбинированного 4-LUT №2

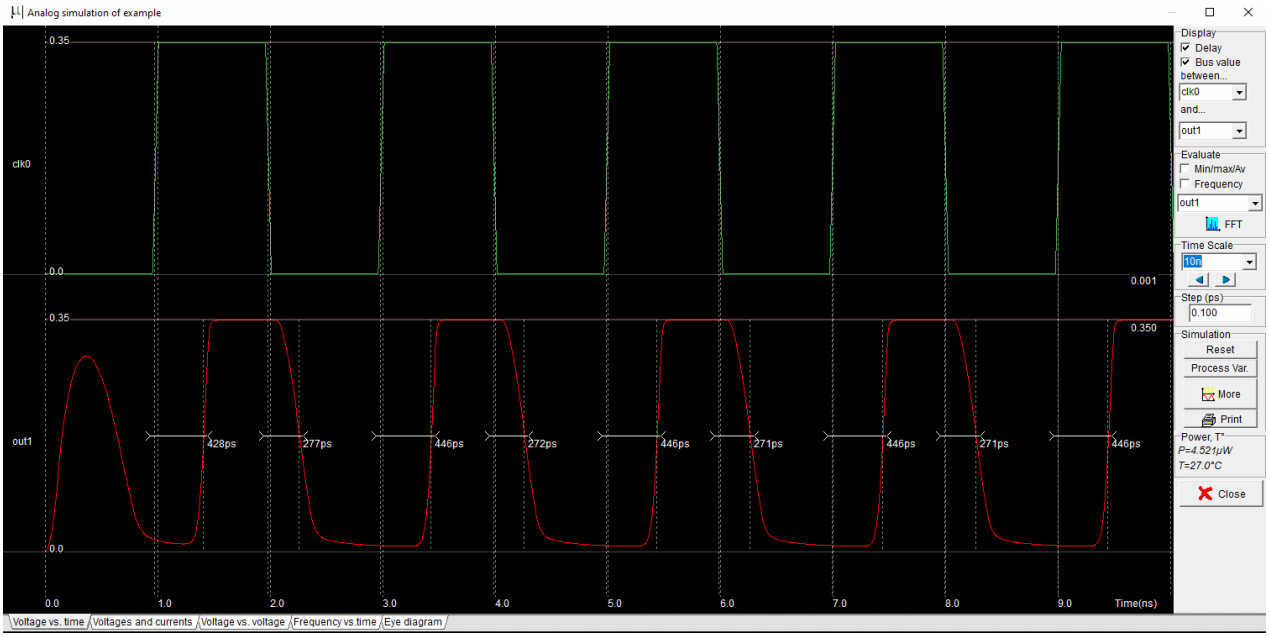


Рисунок 4.33 – Временная диаграмма второго варианта предлагаемого комбинированного 4-LUT№2

В этих двух вариантах №1, №2 по пути сигнала находится по два транзистора.

Если использовать 2-LUT, то унитарный блок уменьшается до четырех унитарных разрядов (вариант№3) – рисунки 4.34, 4.35, 4.36.

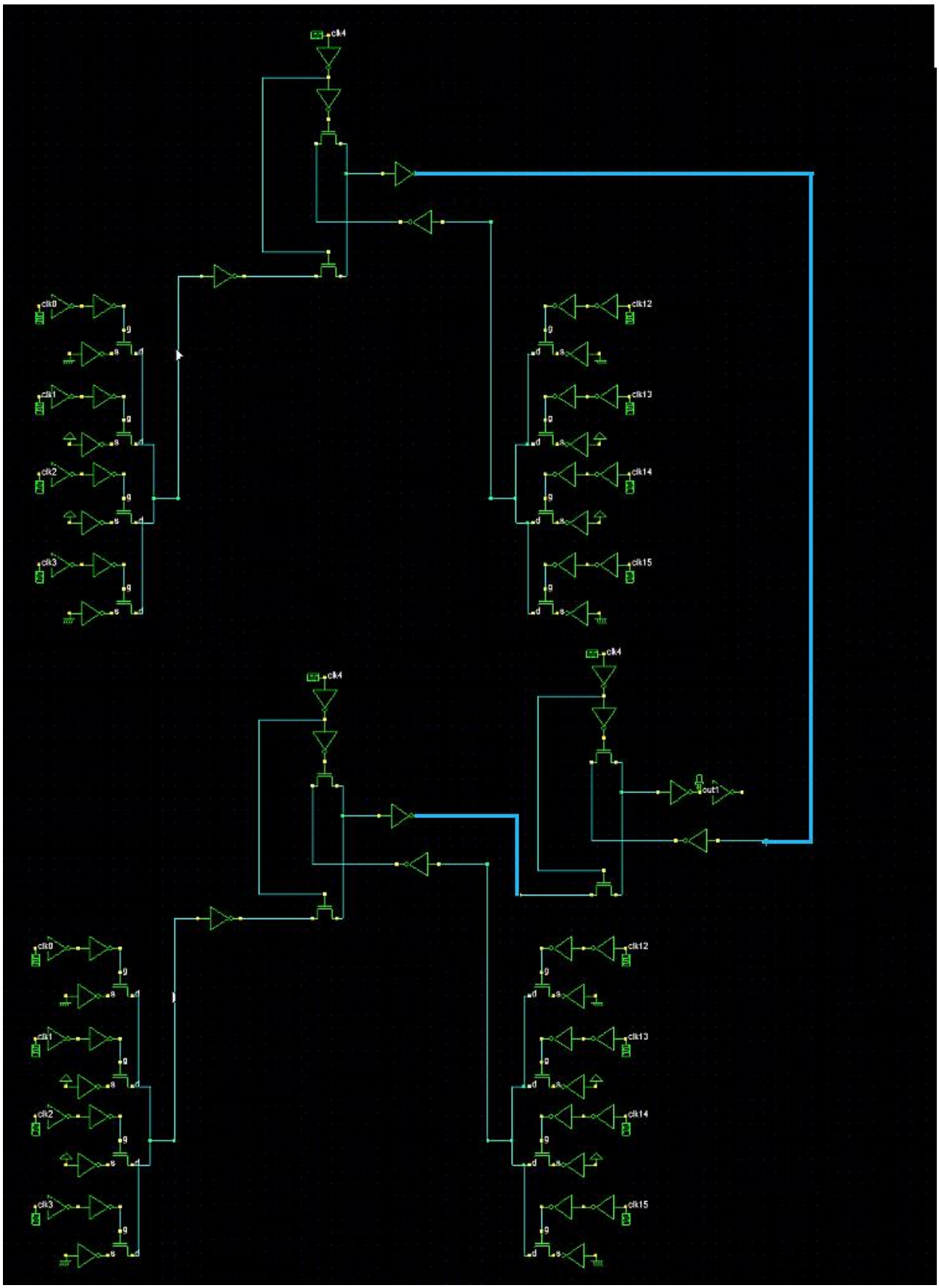


Рисунок 4.34 – Предлагаемый комбинированный 4-LUT №3 (унитарные блоки на 4 унитарных разряда слева)

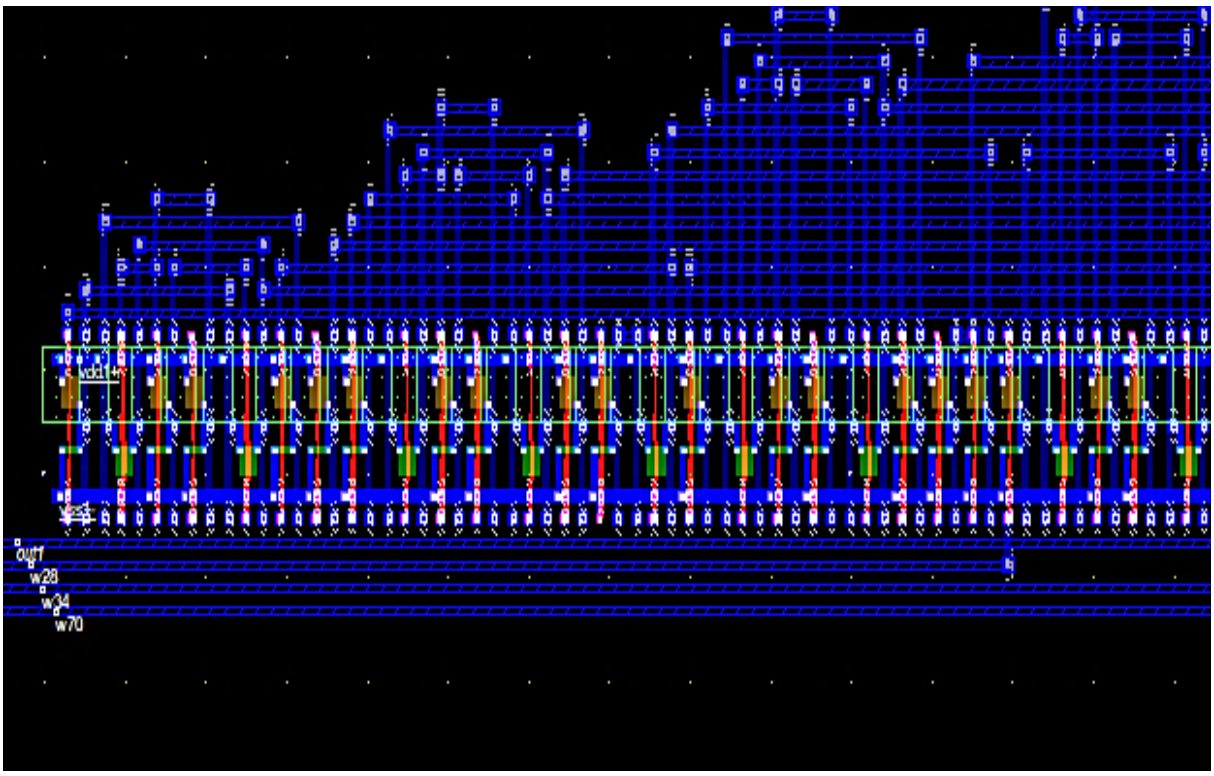


Рисунок 4.35 – Фрагмент топологии третьего варианта предлагаемого комбинированного 4-LUT №3

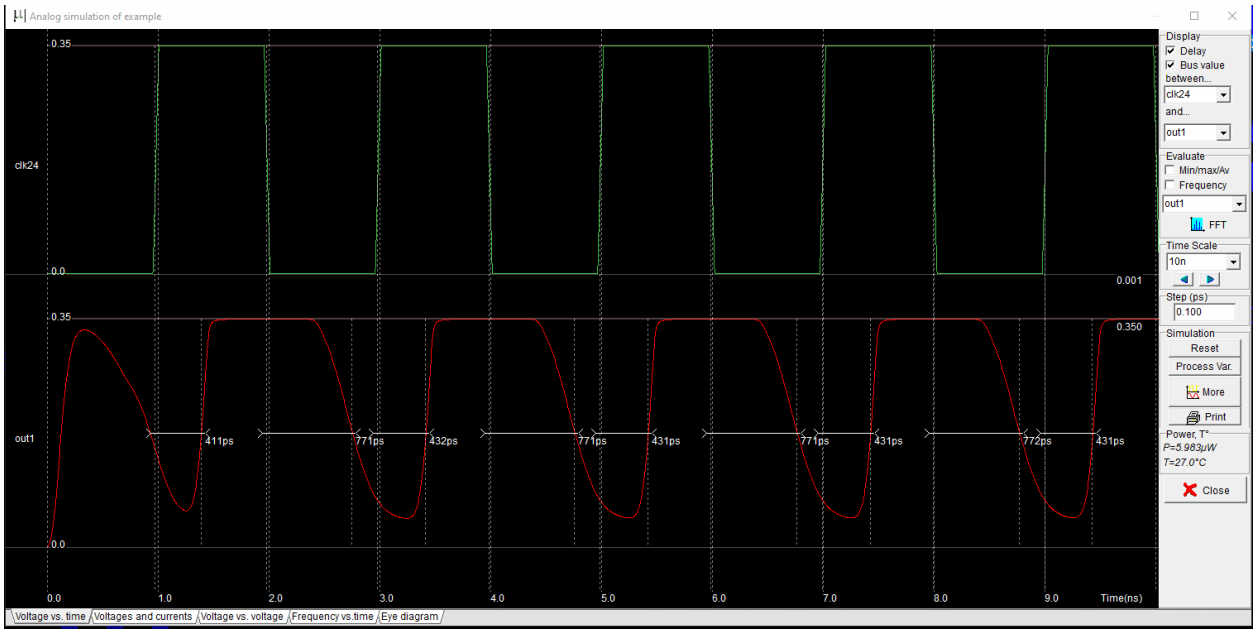


Рисунок 4.36 – Временная диаграмма третьего варианта предлагаемого комбинированного 4-LUT №3

Если использовать 2-LUT слева, то унитарный блок уменьшается до одного унитарного на 4 разряда (вариант №4) – рисунки 4.37, 4.38, 4.39.

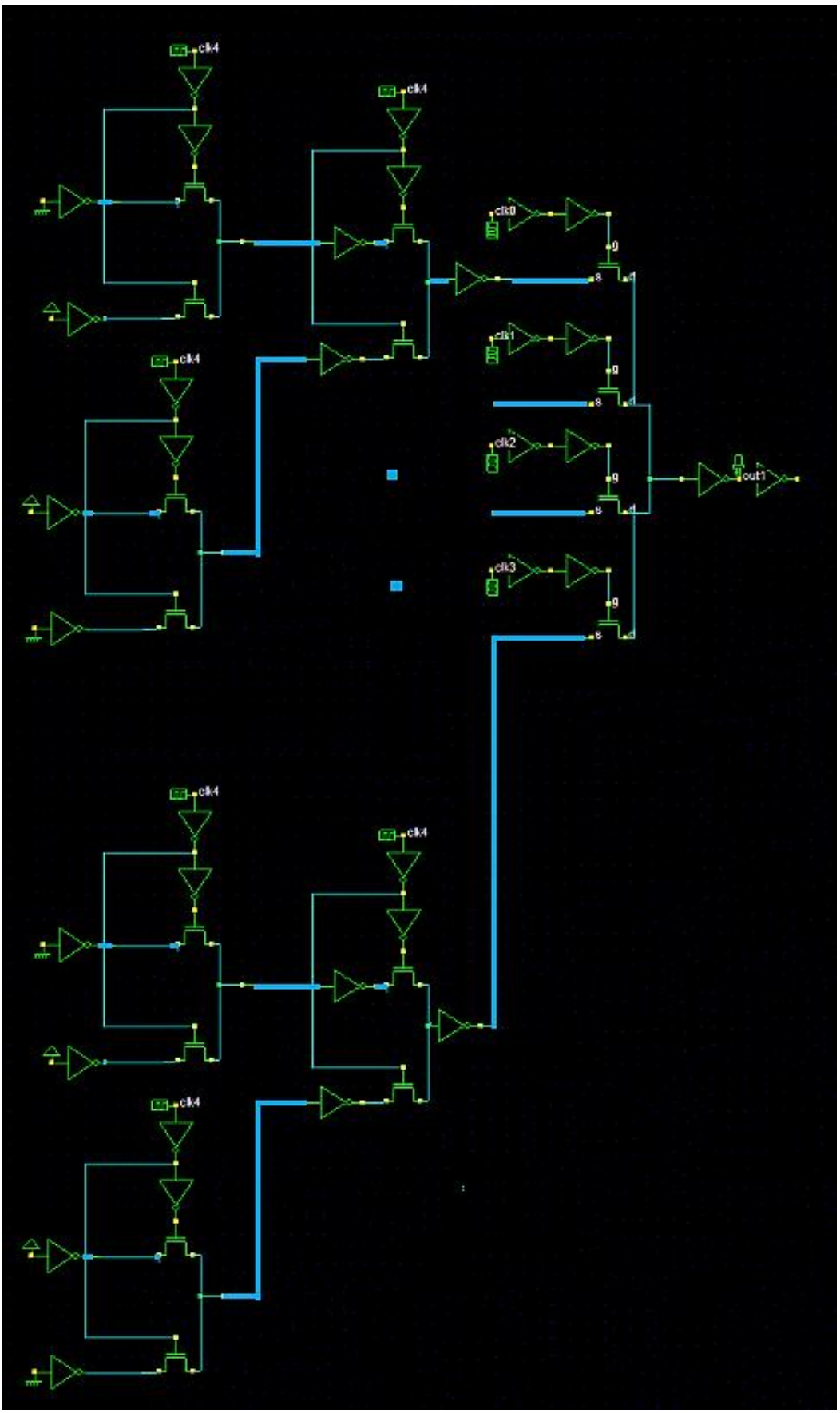


Рисунок 4.37 – Предлагаемый комбинированный 4-LUT№4 (унитарный блок на 4 унитарных разряда справа)

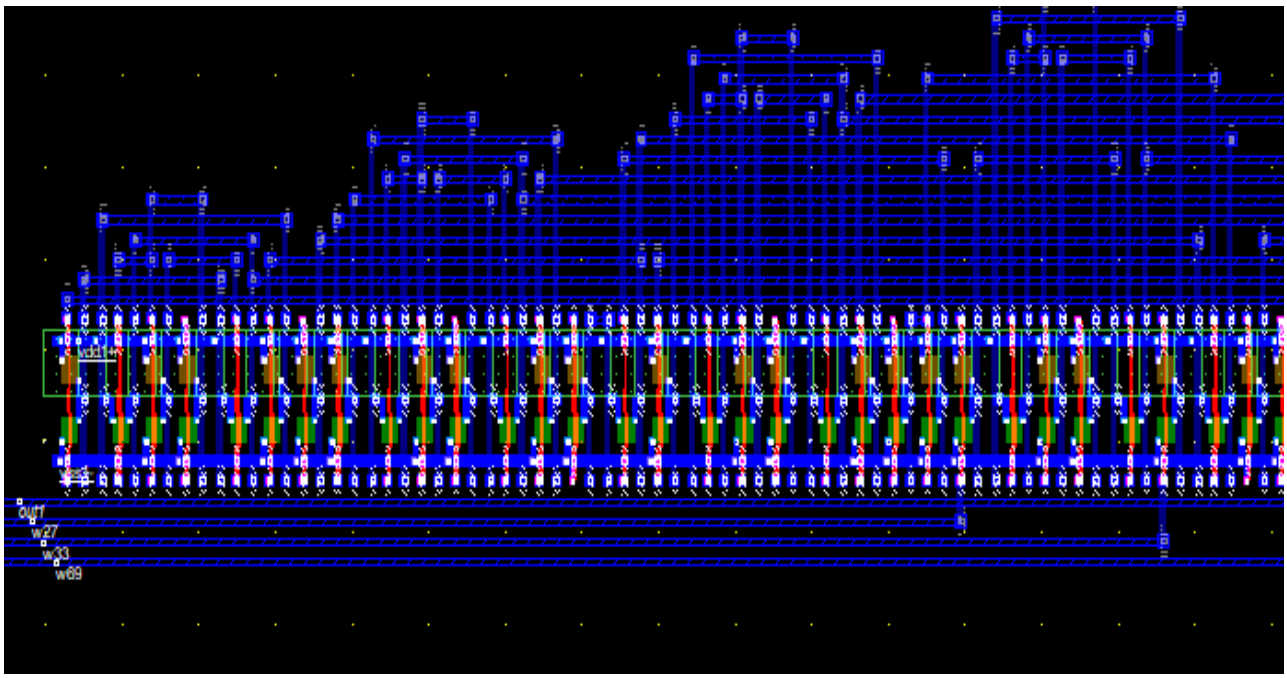


Рисунок 4.38 – Фрагмент топологии четвертого варианта предлагаемого комбинированного 4-LUT №4

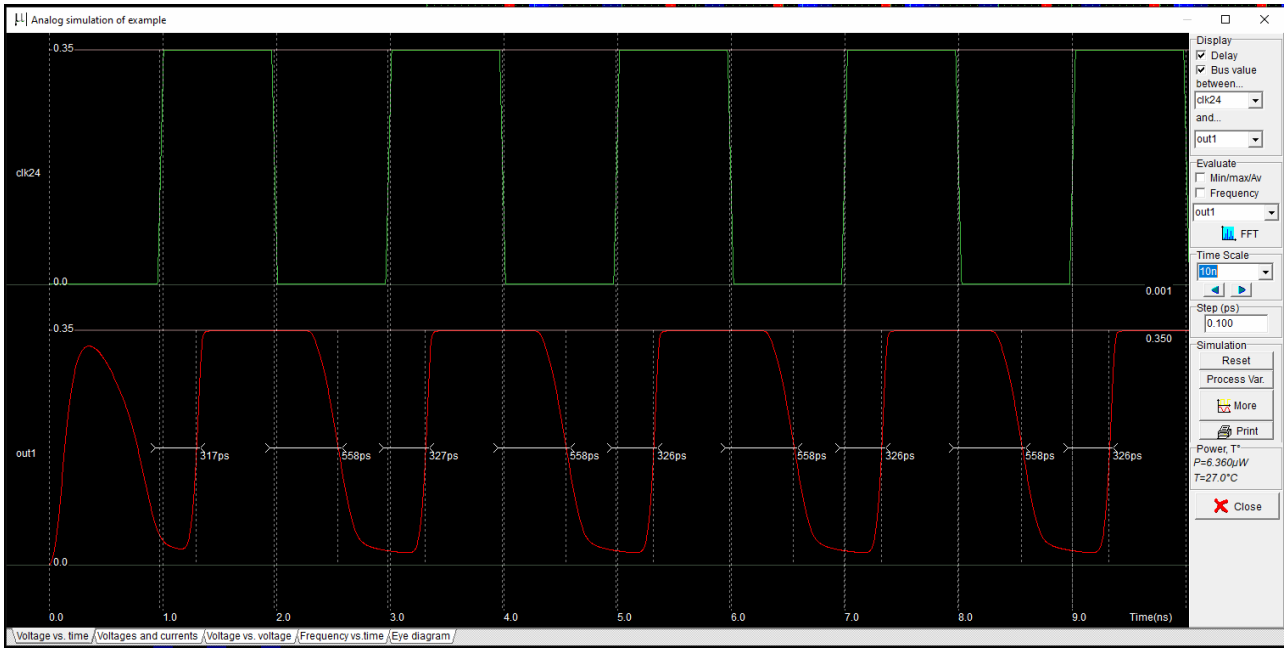


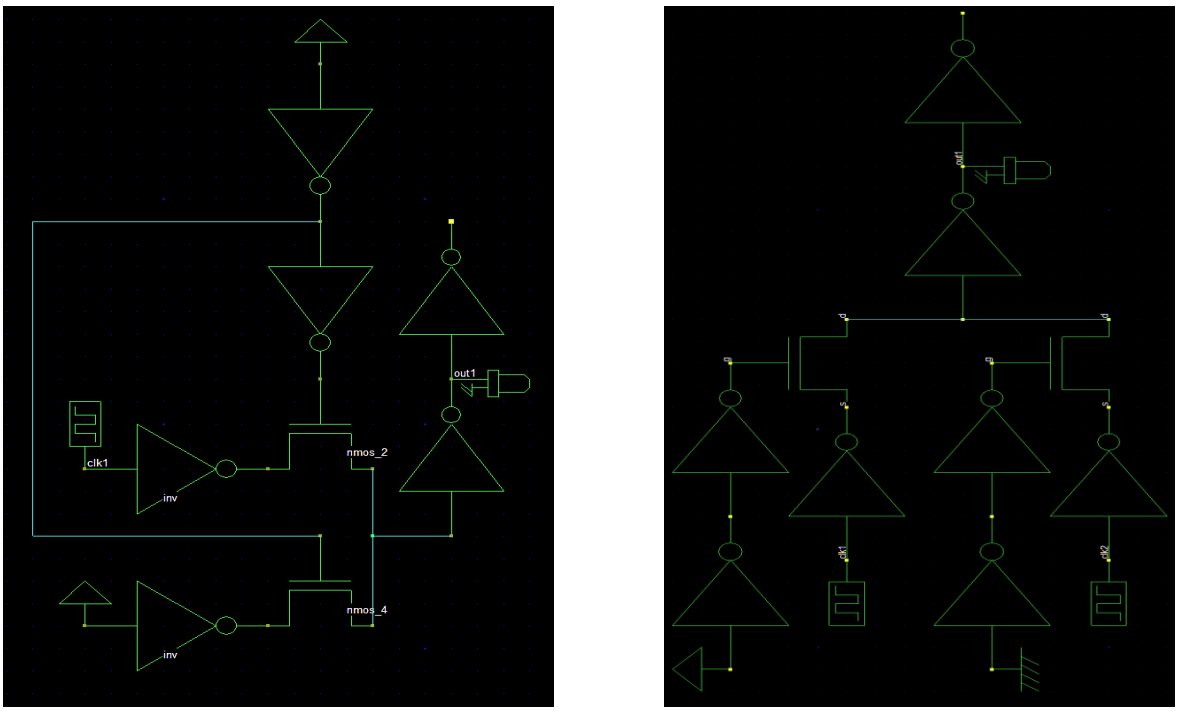
Рисунок 4.39 – Временная диаграмма четвертого варианта предлагаемого комбинированного 4-LUT №4

В этих двух вариантах №3, №4 по пути сигнала по три транзистора. Увеличение бинарных LUT до трех разрядов ни имеет смыслы, ибо тогда в пути будет 4 транзистора (ограничение Мида-Конвея).

4.4 Моделирование в системе топологического моделирования Microwind элементов для коммутации сигналов

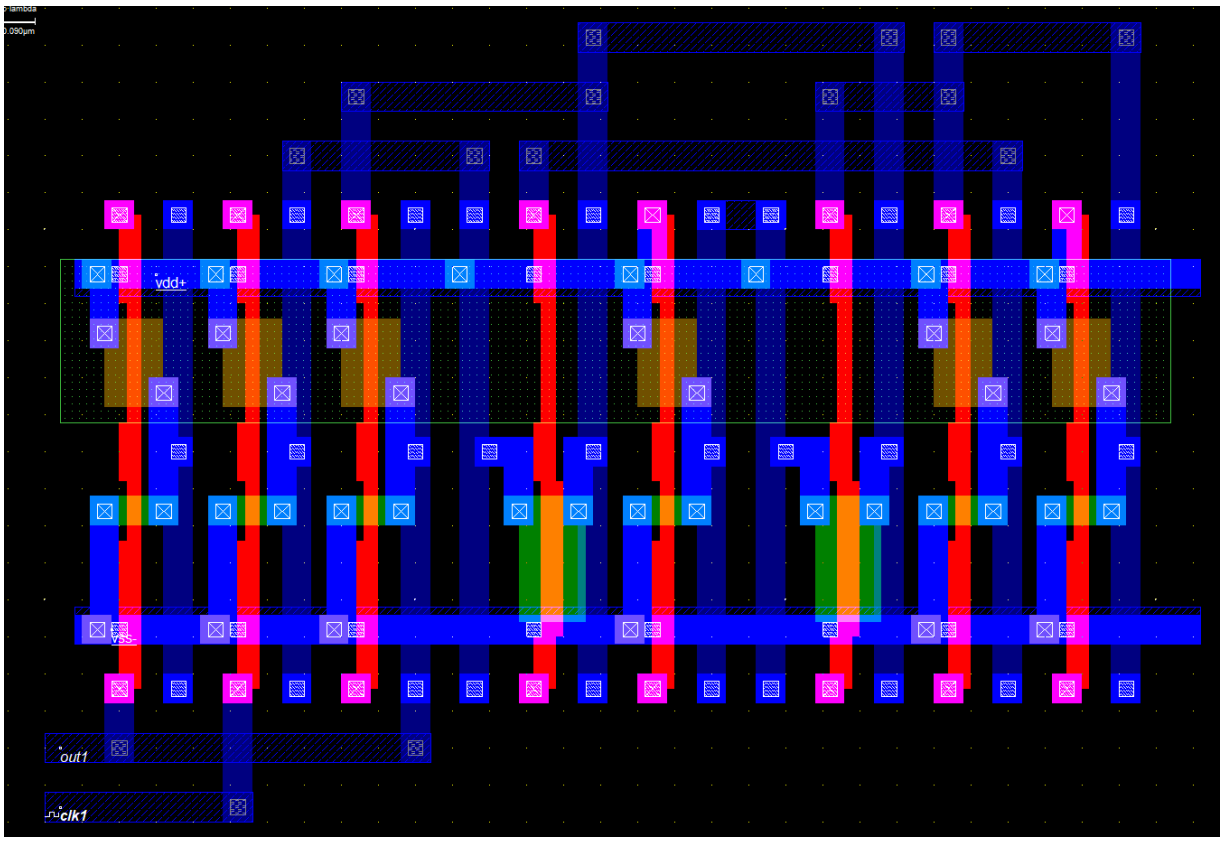
4.4.1 Моделирование в Microwind элементов для коммутации сигналов на одну переменную

При коммутации сигналов на входы переменных элементов, используемых для вычисления логической функции подаются значения (константы), записанные в конфигурационной памяти коммутаторов, а сами сигналы поступают на входы настройки элементов, используемых для вычисления логической функции, то есть при данной настройке всегда активируется один путь в дереве предающих транзисторов. Элемент на 1 переменную показан на рисунках 4.40 и 4.41. На рисунке 4.42. показана временная диаграмма работы. Построенная по технологии CMOS 32 nm, далее все элементы будут построены по такой же технологии при напряжении источника питания 0,35 В, частоте 0,5 ГГц, модель BSim4).

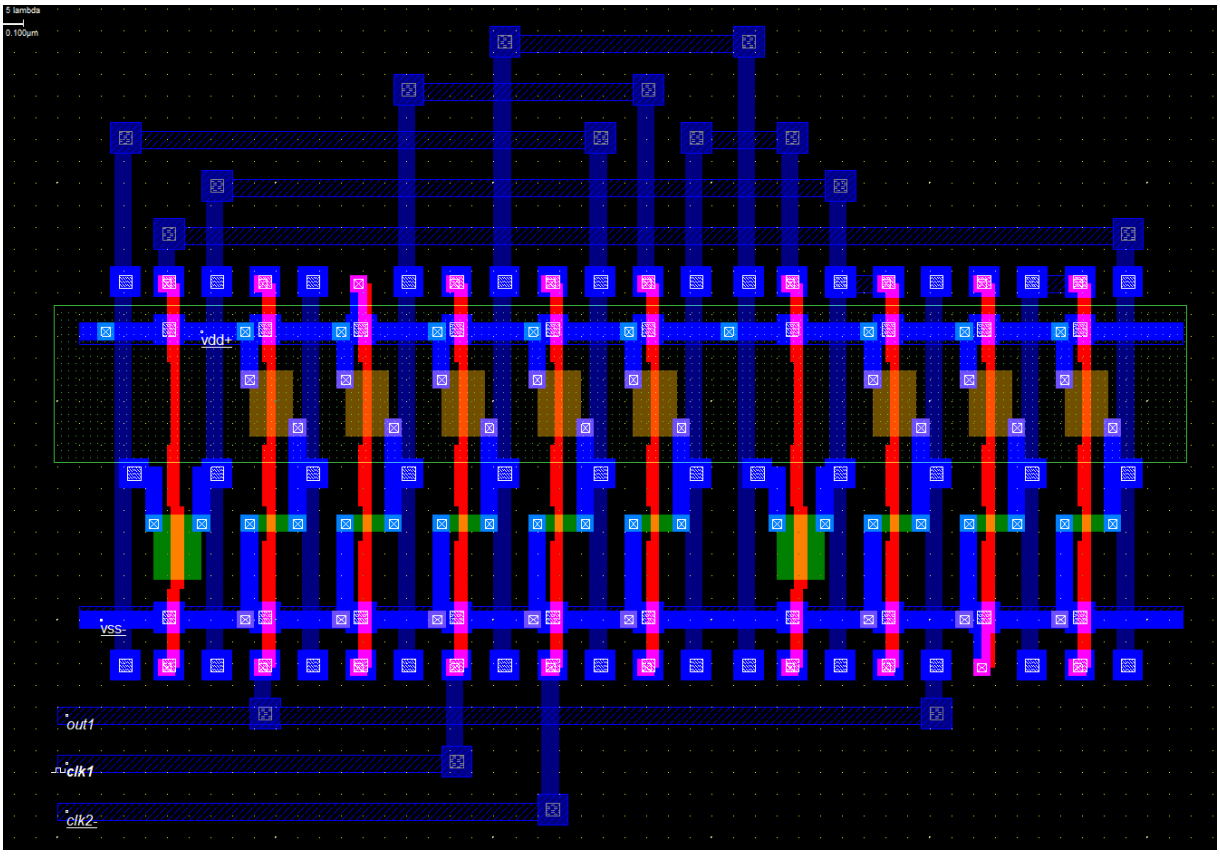


а) б)

Рисунок 4.40 – Схема электрическая функциональная коммутатора 1-LUT, а) позиционного; б) унитарного

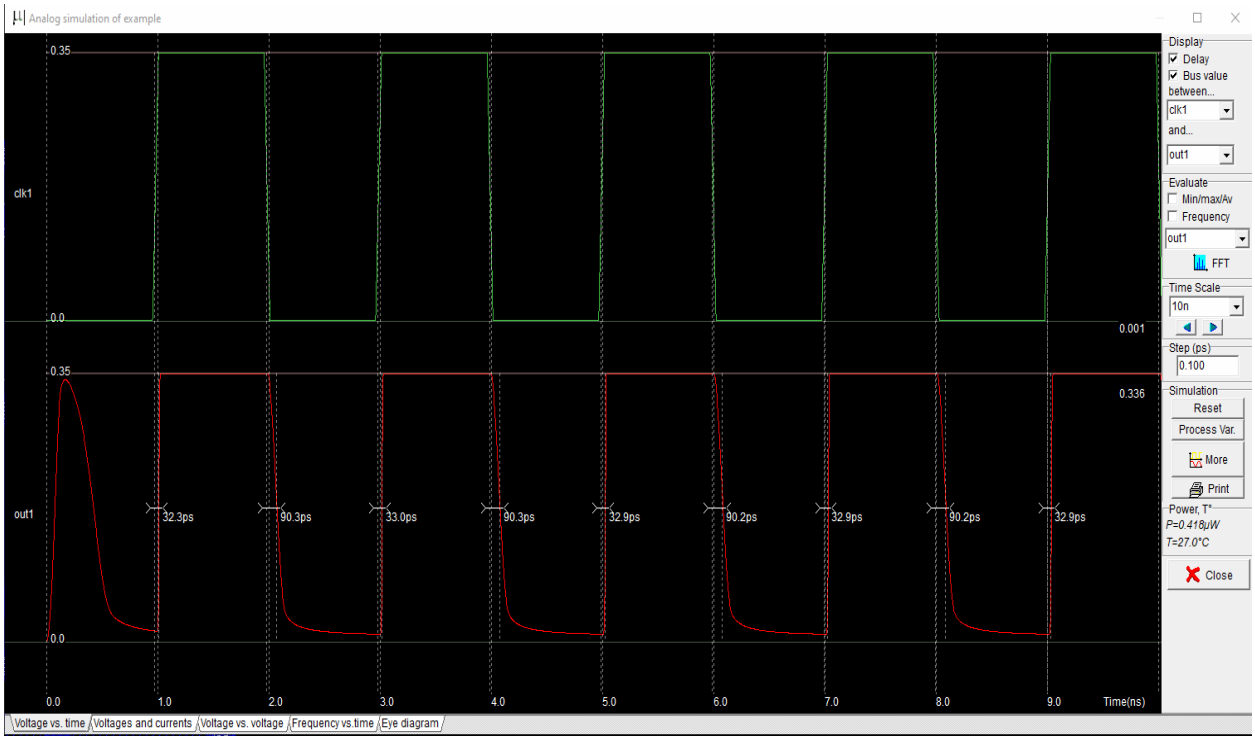


а)

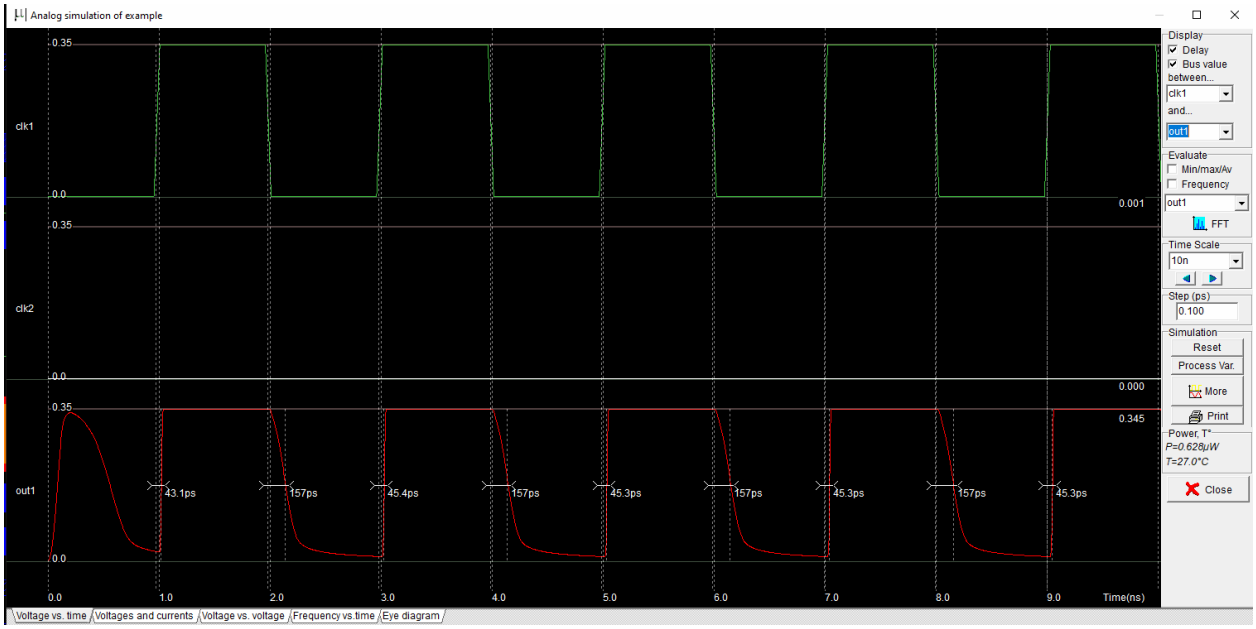


б)

Рисунок 4.41 – Топология коммутатора 1-LUT: а) позиционного; б) унитарного



а)

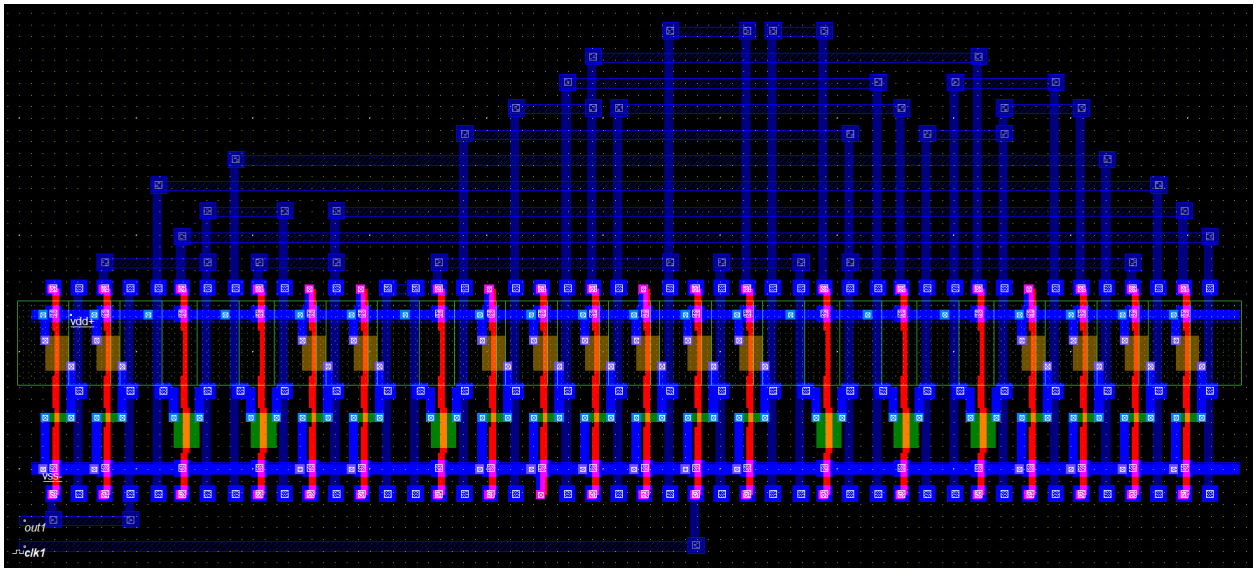


б)

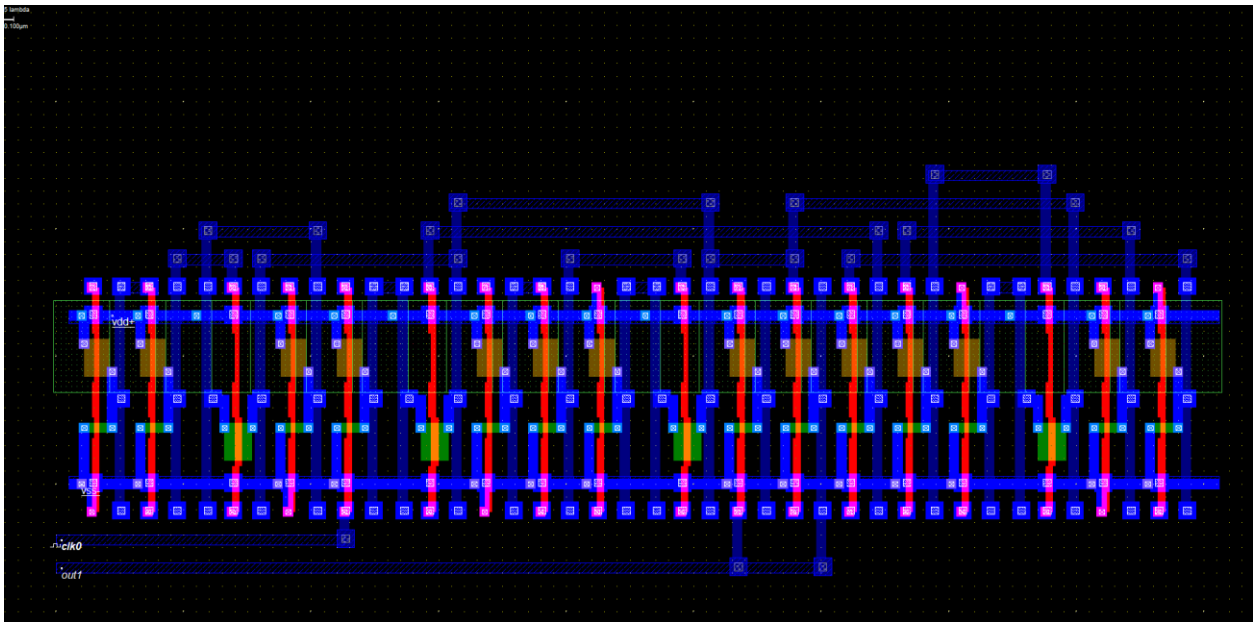
Рисунок 4.42 – Временная диаграмма работы: а) позиционного коммутатора на основе 1-LUT CMOS 32 nm; б) унитарного коммутатора на одну переменную CMOS 32 nm

4.4.2 Моделирование в Microwind элементов для коммутации сигналов на две переменные

Предлагаемый элемент позиционный и унитарный для коммутации сигналов на 2 переменные – рисунки 4.43, 4.44.



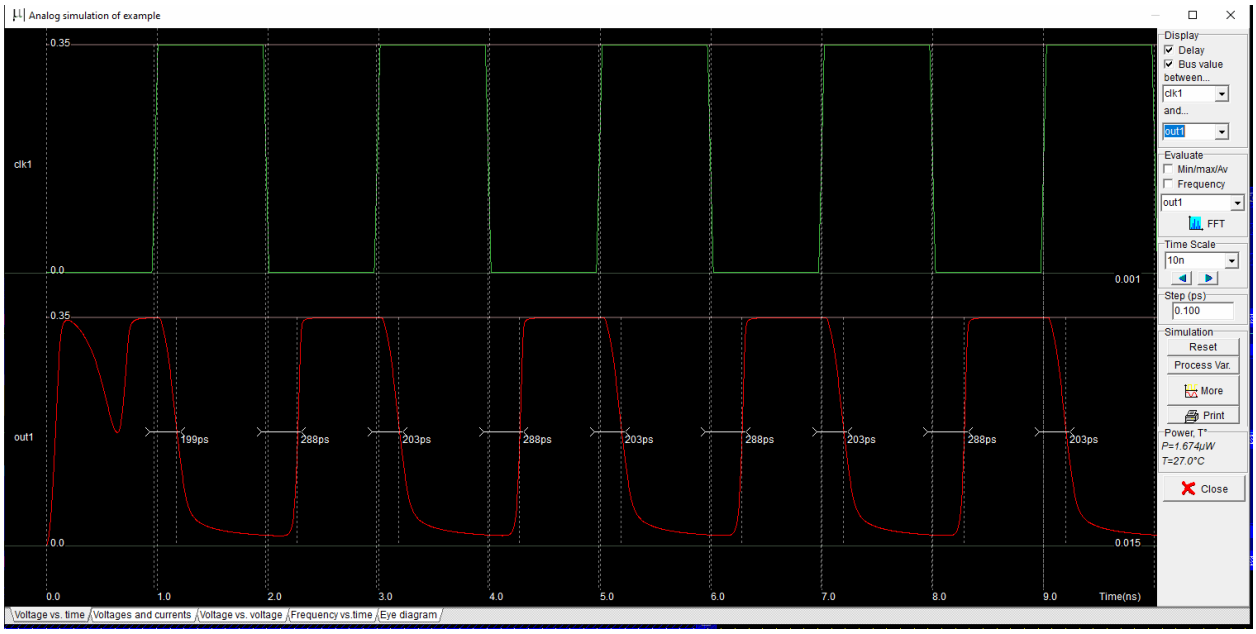
а)



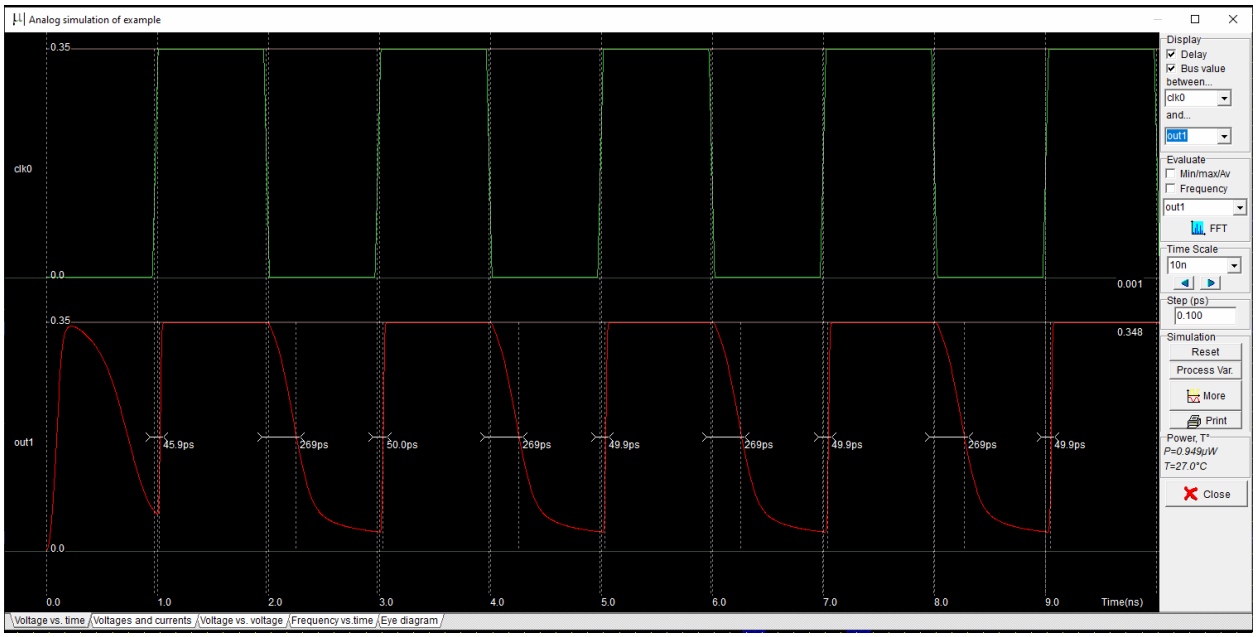
б)

Рисунок 4.43 – Топология коммутатора 2-LUT а) позиционного; б) унитарного

На рисунке 4.44. показана временная диаграмма работы коммутатора на основе 2-LUT для позиционного и унитарного видов кодирования.



а)

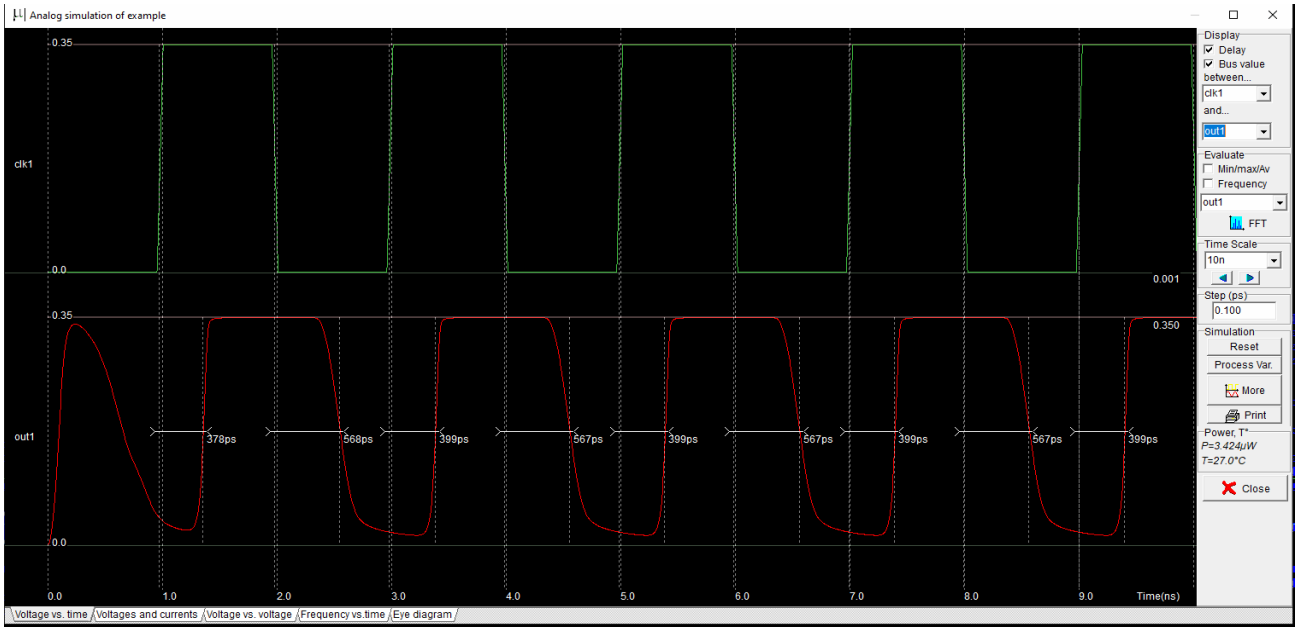


б)

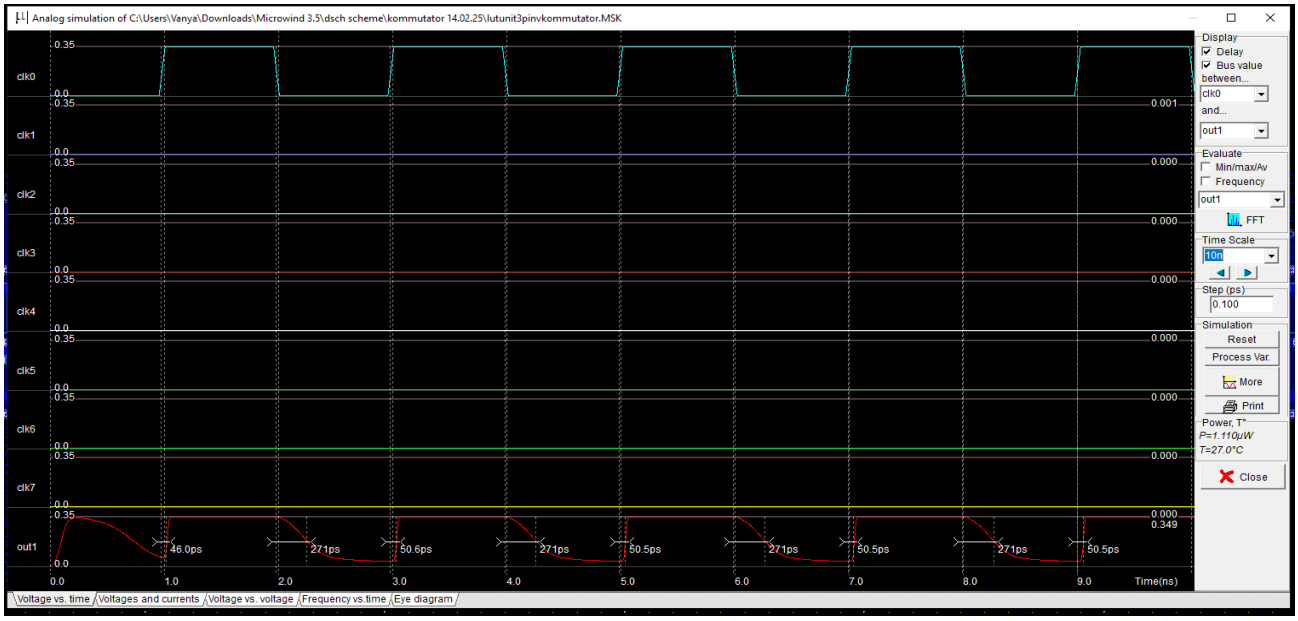
Рисунок 4.44 – Временная диаграмма работы а) позиционного; б) унитарного

4.4.3 Моделирование в Microwind элементов для коммутации сигналов на три переменные

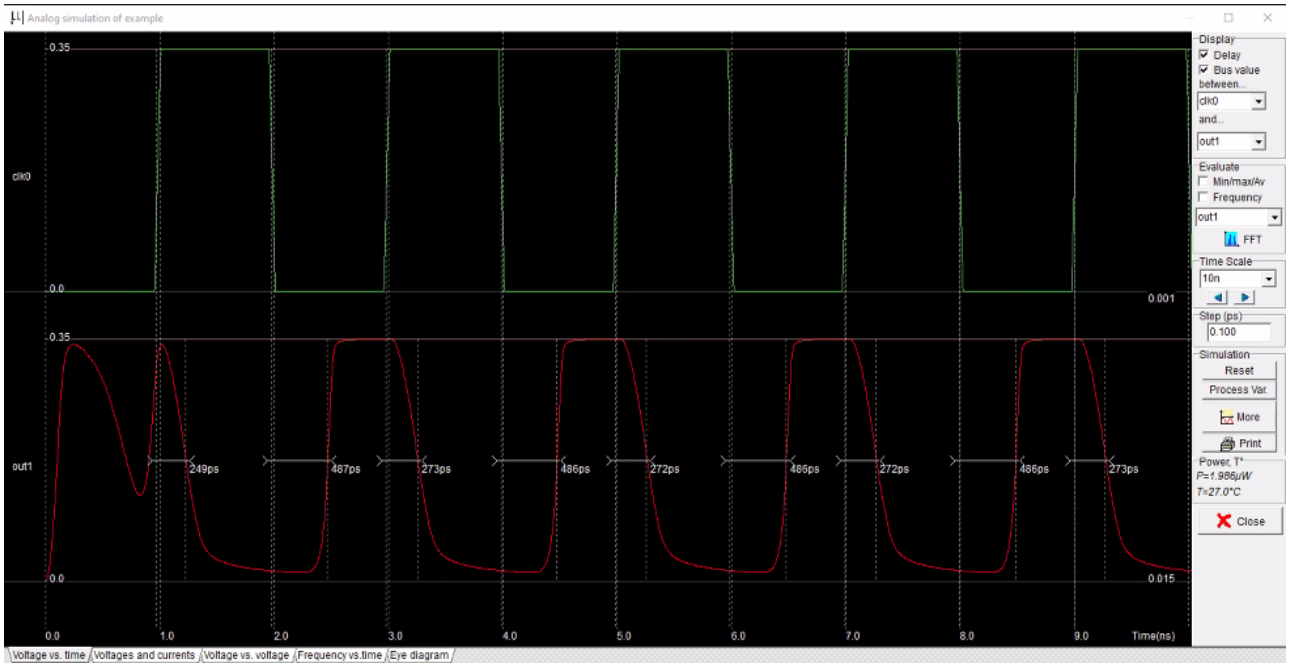
На рисунке 4.45. показана временная диаграмма работы а) коммутатора на основе 3-LUT; б) унитарного коммутатора на три переменных; в) комбинированного 3-LUT №1 (два унитарных блока 4 слева, один 1-LUT справа); г) комбинированного 3-LUT №2 (один унитарный блок 4 справа, четыре 1-LUT слева).



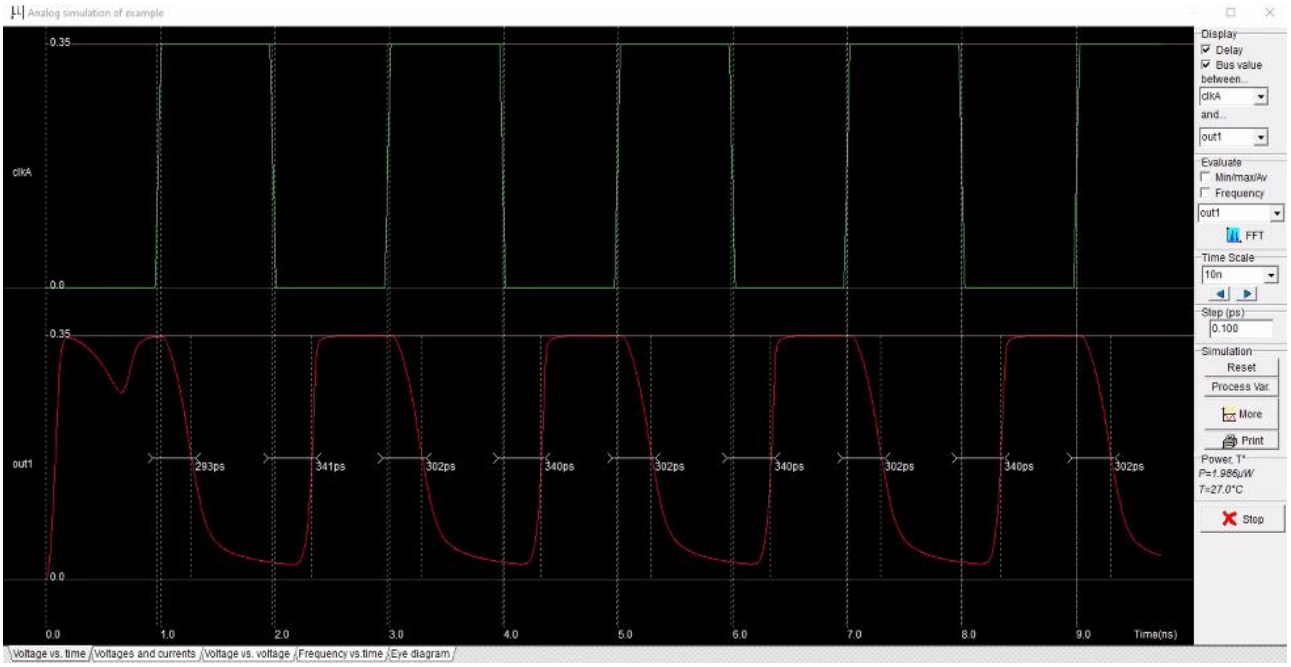
а)



б)



в)

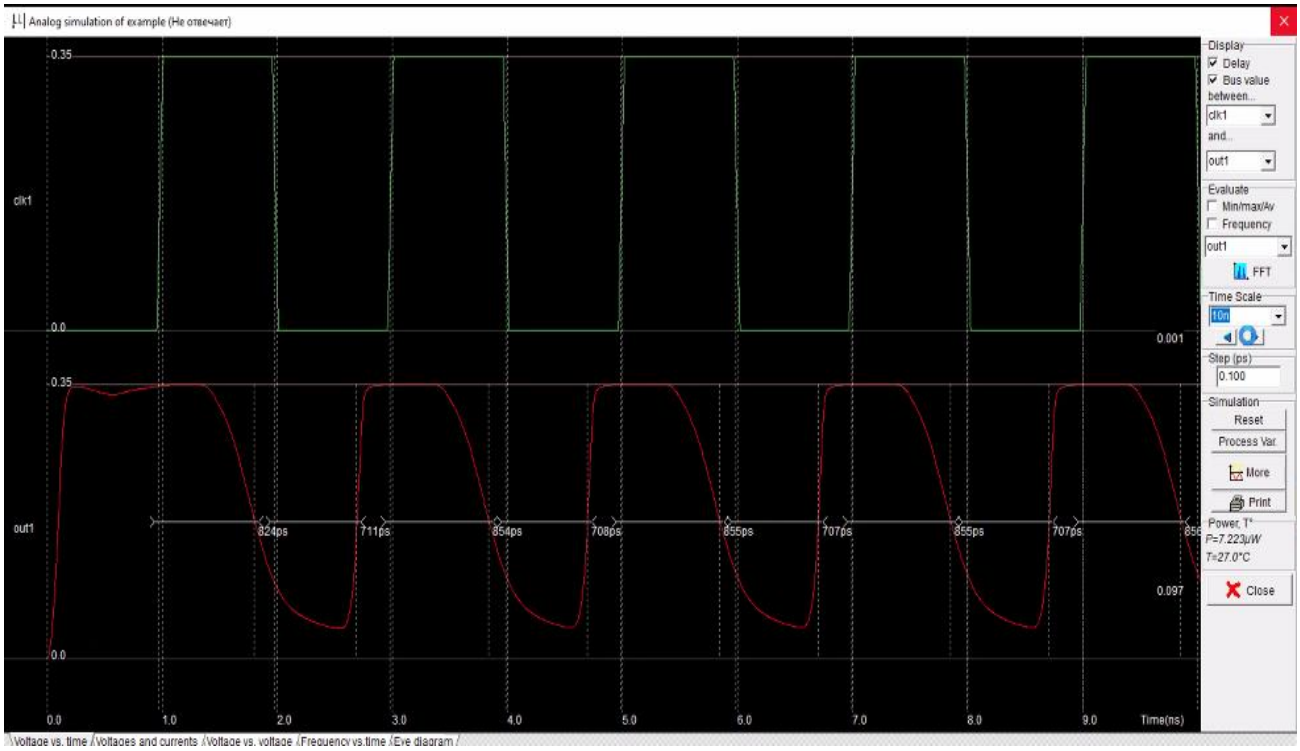


г)

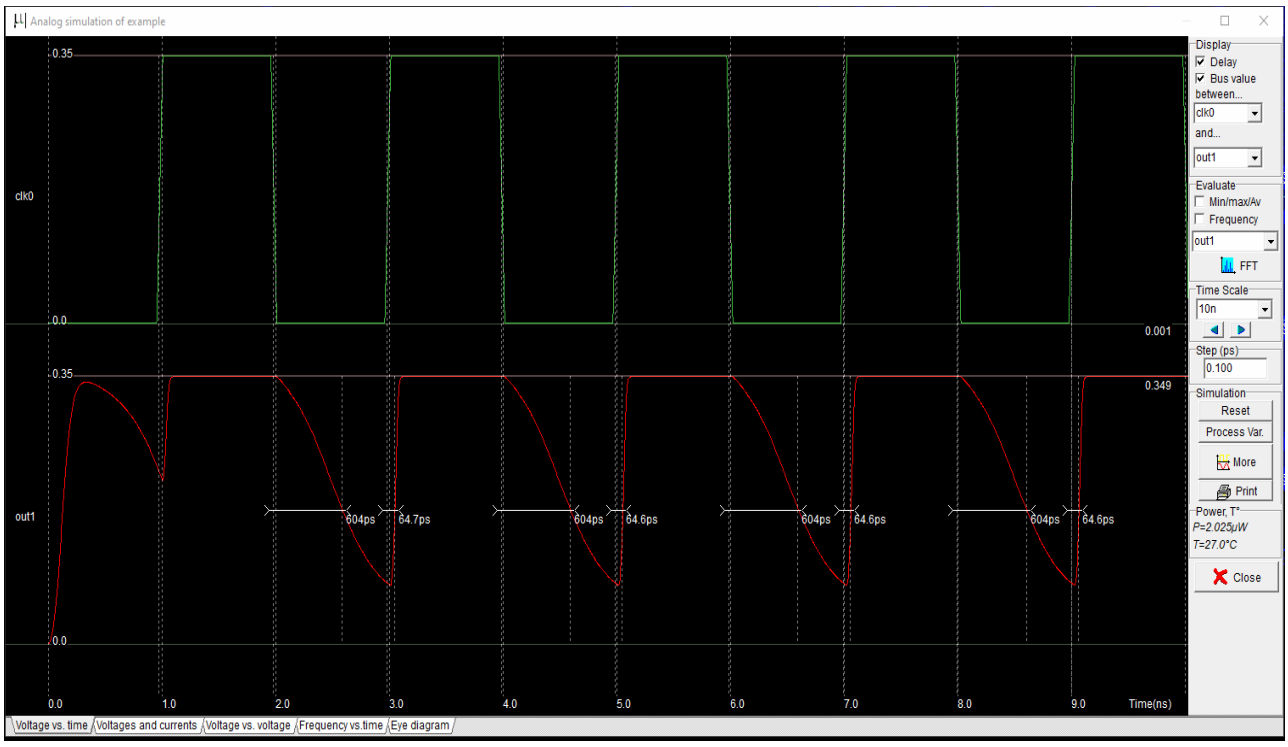
Рисунок 4.45 – Временная диаграмма работы: а) коммутатора на основе 3-LUT; б) унитарного коммутатора на три переменных; в) комбинированного 3-LUT №1 (два унитарных блока 4 слева, один 1-LUT справа); г) комбинированного 3-LUT №2 (один унитарный блок 4 справа, четыре 1-LUT слева)

4.4.4 Моделирование в Microwind элементов для коммутации сигналов на четыре переменные

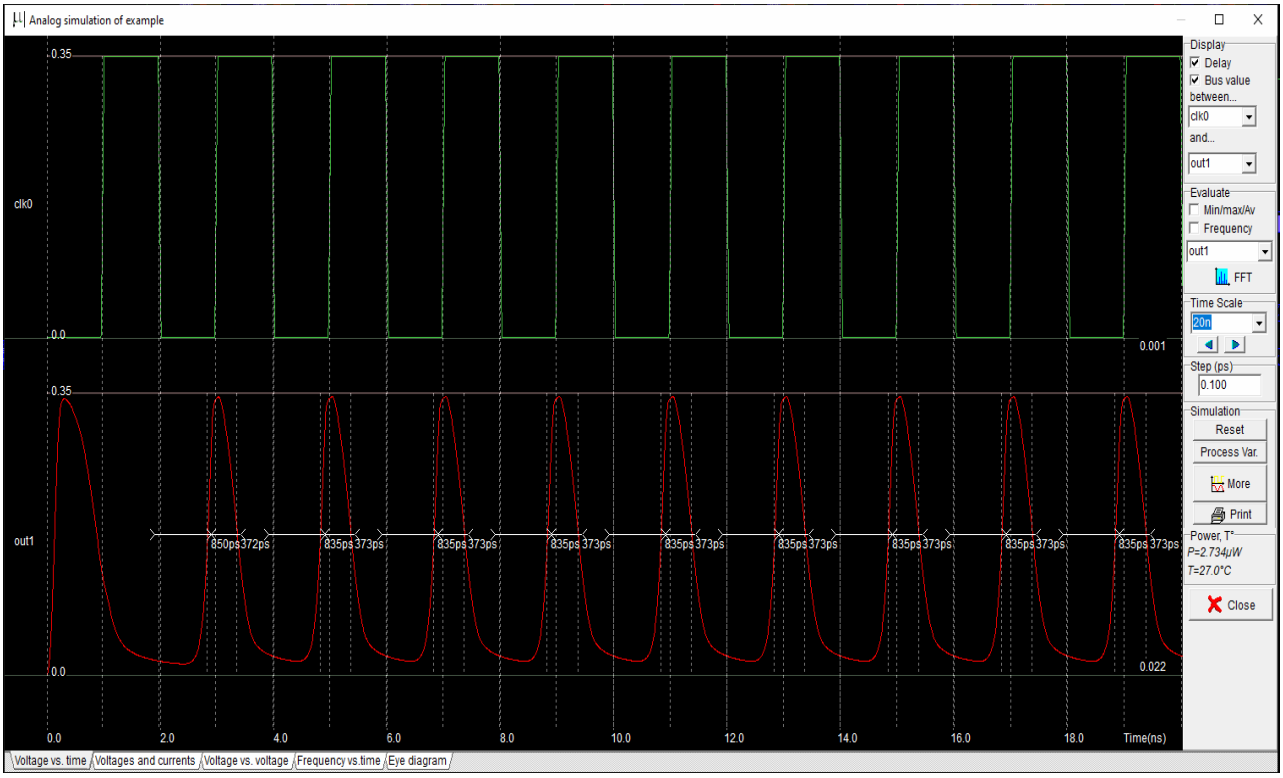
На рисунке 4.46. показана временная диаграмма работы а) коммутатора на основе 4-LUT; б) унитарного коммутатора на четыре переменных; в) комбинированного 4-LUT №1 (два унитарных блока 8 слева, один 1-LUT справа); г) комбинированного 4-LUT №2 (один унитарный блок 8 справа, восемь 1-LUT слева); д) комбинированного 4-LUT №3 (четыре унитарных блока 4 слева, один 2-LUT справа); е) комбинированного 4-LUT №4 (один унитарный блок 4 справа, четыре 2-LUT справа). В приложении Б продемонстрированы схемы электрические функциональные и топологии.



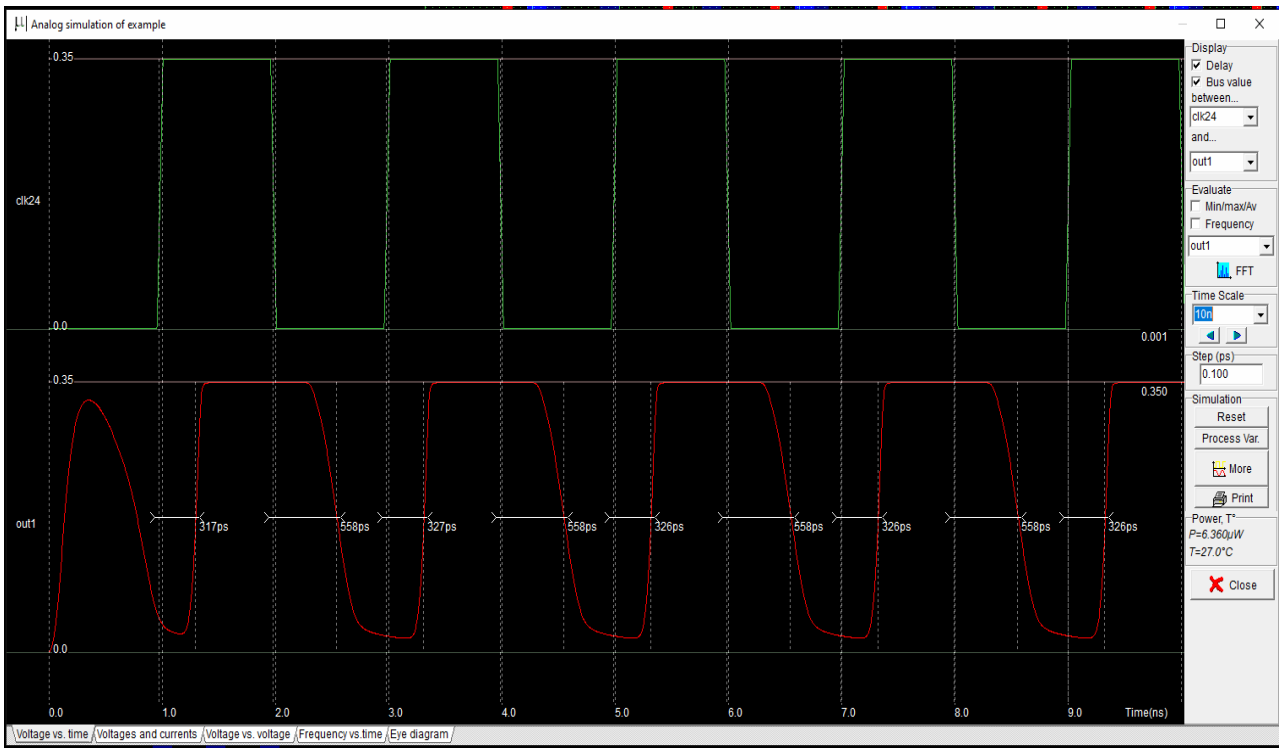
а)


Analog simulation of example
clk0
out1
0.35
0.0
0.001
0.349
604ps
64.7ps
64.6ps
Time(ns)
Display
Delay
Bus value between...
clk0
and...
out1
Evaluate
Min/max/Av
Frequency
FFT
Time Scale
10n
Step (ps)
0.100
Simulation
Reset
Process Var.
More
Print
Power, T°
P=2.025µW
T=27.0°C
Close
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram

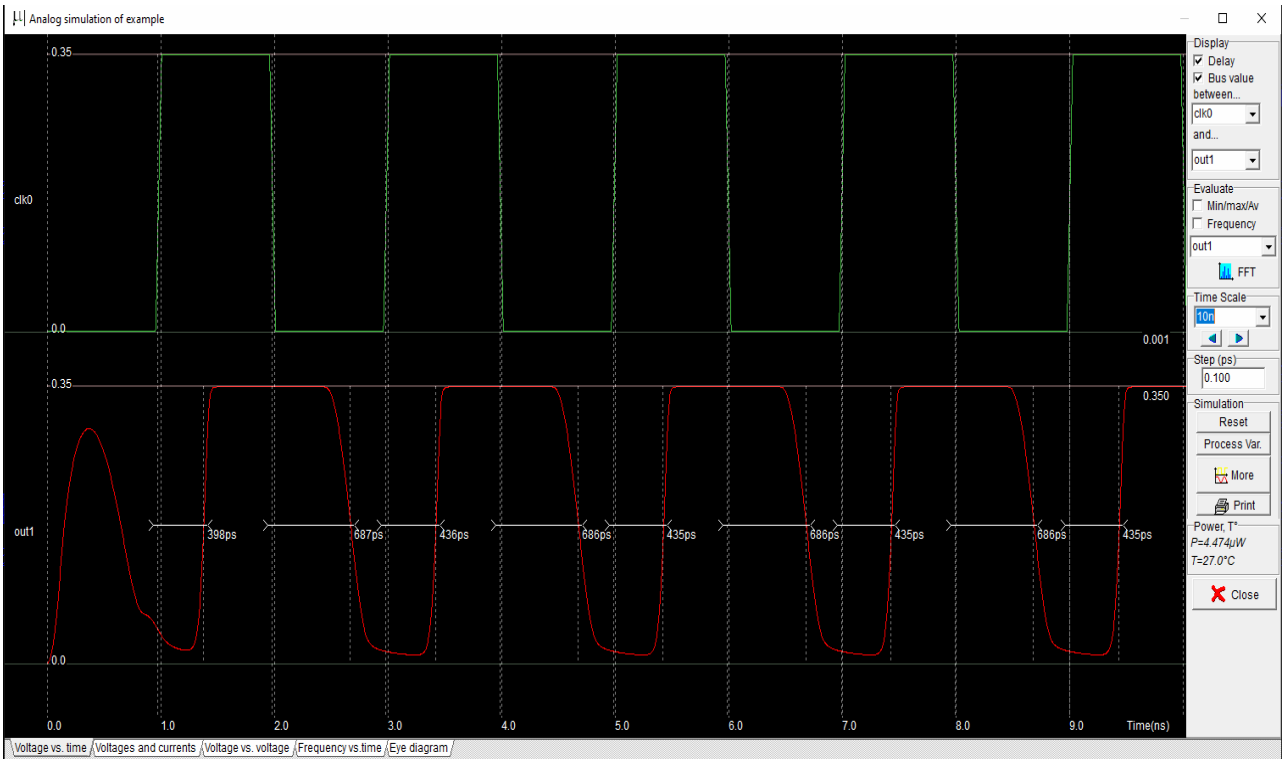
б)


Analog simulation of example
clk0
out1
0.35
0.0
0.001
0.022
850ps 372ps
835ps 373ps
Time(ns)
Display
Delay
Bus value between...
clk0
and...
out1
Evaluate
Min/max/Av
Frequency
FFT
Time Scale
20n
Step (ps)
0.100
Simulation
Reset
Process Var.
More
Print
Power, T°
P=2.734µW
T=27.0°C
Close
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram

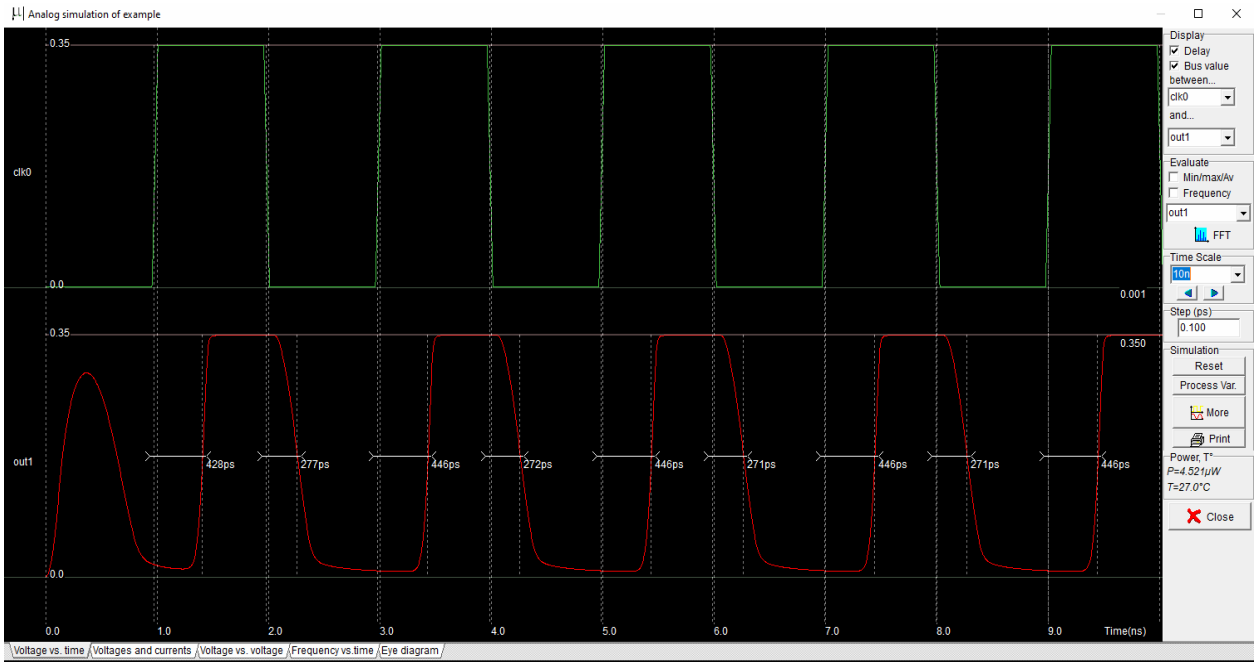
в)


Analog simulation of example
clk24
out1
0.35
0.0
0.001
0.350
317ps
558ps
327ps
558ps
326ps
558ps
326ps
558ps
326ps
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
Time(ns)
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram
Display
Delay
Bus value between...
clk24
and...
out1
Evaluate
Min/max/Av
Frequency
out1
FFT
Time Scale
10n
Step (ps)
0.100
Simulation
Reset
Process Var.
More
Print
Power, T°
P=6.360µW
T=27.0°C
Close

г)


Analog simulation of example
clk0
out1
0.35
0.0
0.001
0.350
398ps
687ps
436ps
686ps
435ps
686ps
435ps
686ps
435ps
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
Time(ns)
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram
Display
Delay
Bus value between...
clk0
and...
out1
Evaluate
Min/max/Av
Frequency
out1
FFT
Time Scale
10n
Step (ps)
0.100
Simulation
Reset
Process Var.
More
Print
Power, T°
P=4.474µW
T=27.0°C
Close

д)



е)

Рисунок 4.46 – Временная диаграмма работы: а) коммутатора на основе 4-LUT; б) унитарного коммутатора на четыре переменных; в) комбинированного 4-LUT №1 (два унитарных блока 4 слева, один 1-LUT справа); г) комбинированного 4-LUT №2 (один унитарный блок 4 справа, четыре 1-LUT слева); д) комбинированного 4-LUT №3(четыре унитарных блока 4 слева, один 2-LUT справа); е) комбинированного 4-LUT №4 (один унитарный блок 4 справа, четыре 2-LUT справа)

4.5 Выводы по главе 4

1) Схемотехническое моделирование подтвердило работоспособность унитарных и комбинированных элементов при вычислении заданной логической функции и правильность теоретического анализа функционирования, предложенных устройств, в статическом и динамическом режимах.

2) Унитарные элементы, как правило, обладают меньшей временной задержкой, чем позиционные, однако по количеству линий связи, площади кристалла и объему конфигурационной памяти коммутаций, они значительно проигрывают.

3) Промежуточный вариант с комбинированным кодированием показал теоретически предсказанные результаты: по числу линий связи (или числу конфигурационных бит) он обладает средними показателями, выигрывая у позиционного элемента по временной задержке.

4) Топологическое моделирование подтвердило правильность теоретических решений и позволило определить показатели качества различных предложенных вариантов реализации: площадь кристалла, потребляемую мощность, временную задержку с учетом доступных моделей транзисторов.

5) Целесообразно использовать разработанные топологии при проектировании архитектур новых, перспективных ПЛИС. В дальнейшем необходимо уточнить полученные оценки для более современных технологий и с учетом восстановителей логического уровня сигналов для большей размерности с учетом ограничений Мида-Конвей.

6) Предлагаемый универсальный элемент с выбором варианта кодирования сложнее и унитарного, и позиционного, но у него имеется новое свойство: он универсальный, пользователь сам выбирает, какой нужен вариант путем настройки.

7) Необходимо оценить сложность реализации полученных вариантов в зависимости от числа переменных или конфигурационной памяти межсоединений по показателям количества транзисторов (получить соответствующие формулы), площади кристалла, потребляемой мощности, временной задержке при топологическом моделировании.

ГЛАВА 5. ОЦЕНКА ЭФФЕКТИВНОСТИ РЕАЛИЗАЦИИ ЭЛЕМЕНТА, ИСПОЛЬЗУЮЩЕГО КОМБИНИРОВАННОЕ КОДИРОВАНИЕ

5.1 Сравнительные оценки сложности реализации предлагаемого элемента в количестве транзисторов

Оценим количество передающих транзисторов необходимых только для реализации деревьев вычислений обычного LUT с помощью выражения (5.1):

$$L_*(n) = 2^{n+1} - 2. \quad (5.1)$$

Для унитарного LUT – почти в два раза меньше и задержка – всего один транзистор против n, выражение (5.2):

$$L_{oh.*}(n) = 2^n. \quad (5.2)$$

Однако, с учетом конфигурационной памяти CRAM (SRAM, шести-транзисторные ячейки) и инверторов-восстановителей, получается иначе. обычный LUT имеет сложность, выражение (5.3):

$$L\ (n) = 2^{n+1} - 2 + 8 \cdot 2^n + 4n + 2 = 10 \cdot 2^n + 4n, \quad (5.3)$$

где $2^{n+1} - 2$ – сложность самого n уровневого дерева передающих транзисторов (30), $8 \cdot 2^n$ – сложность настройки (конфигурационной памяти) с учетом инверторов-восстановителей, $4n$ – сложность инверторов-восстановителей по входным переменным, 2 – сложность выходного инвертора. Все это без учета декомпозиции деревьев при n >3. Такая декомпозиция не требуется при унитарном кодировании.

Для реализации одновыходного унитарного LUT_{oh} на n позиционных переменных имеем выражение (5.4):

$$L_{oh}(n) = 8 \cdot 2^n + 4 \cdot 2^n + 2^n + 2 = 13 \cdot 2^n + 2, \quad (5.4)$$

где $8 \cdot 2^n$ – сложность настройки (конфигурационной памяти) с учетом инверторов-восстановителей, она такая же, как и у (5.2), $4 \cdot 2^n$ – сложность инверторов-восстановителей по входным переменным, их теперь больше, 2^n –

сложность одноуровневого дерева передающих транзисторов, 2 – сложность выходного инвертора.

Сравнение вариантов выполняется с использованием системы компьютерной алгебры Mathcad. LUT с унитарным кодированием проигрывает по сложности обычному – рисунок 5.1. На рисунке зеленым цветом обозначен обычный (позиционный) вид кодирования, а черным цветом – унитарный вид.

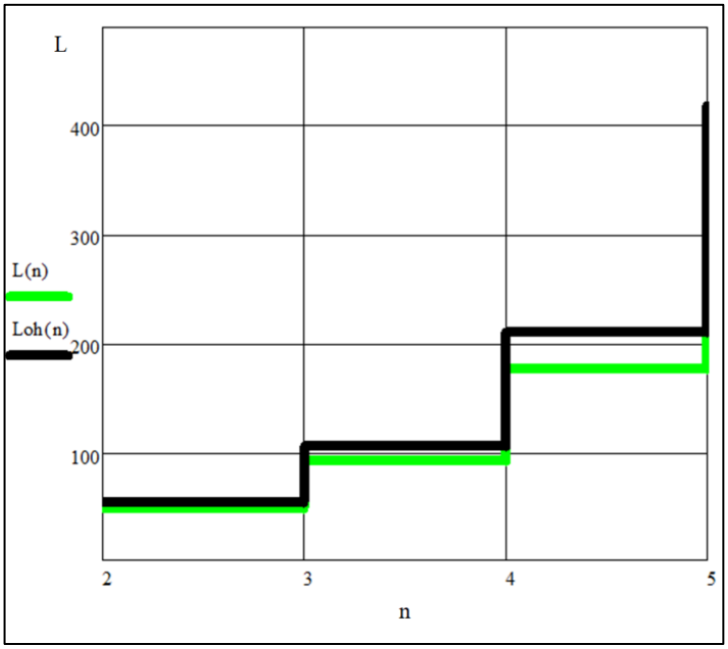


Рисунок 5.1 – Обычный LUT проще унитарного LUT_{oh} при реализации одной функции

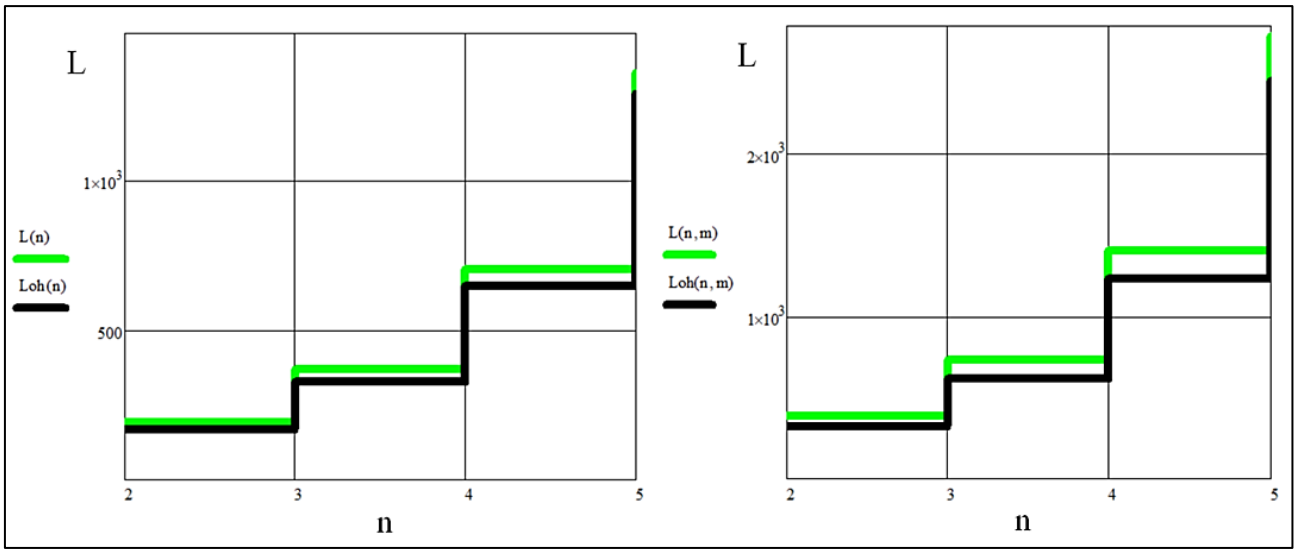
Но зато выигрывает в задержке в дереве передающих транзисторов, как уже указывалось, в n раз. При реализации системы из m функций (рисунок 5.2,б) в унитарном коде получаем выражение (5.5), если использовать m копий одноуровневого дерева унитарного LUT управляемых унитарными входными переменными (с соответствующей настройкой), то:

$$L_{oh}(n,m) = (8 \cdot 2^n + 2^n + 2) \cdot m + 4 \cdot 2^n. \quad (5.5)$$

В обычном LUT такая возможность не используется, поэтому необходимо m для реализации m функций, выражение (5.6):

$$L\ (n,m) = m[10 \cdot 2^n + 4 \cdot n]. \quad (5.6)$$

На рисунке 5.2 зеленым цветом обозначен обычныйы (позиционный) элемент кодирования, а черным цветом – унитарный элемент.



а) б)

Рисунок 5.2 – Реализация системы из m функций в обычном LUT сложнее, чем в унитарном LUT_{oh}: а) при m =4; б) при m =8

При реализации системы из m функций в унитарном коде в унитарном LUT получаем выражение (5.7):

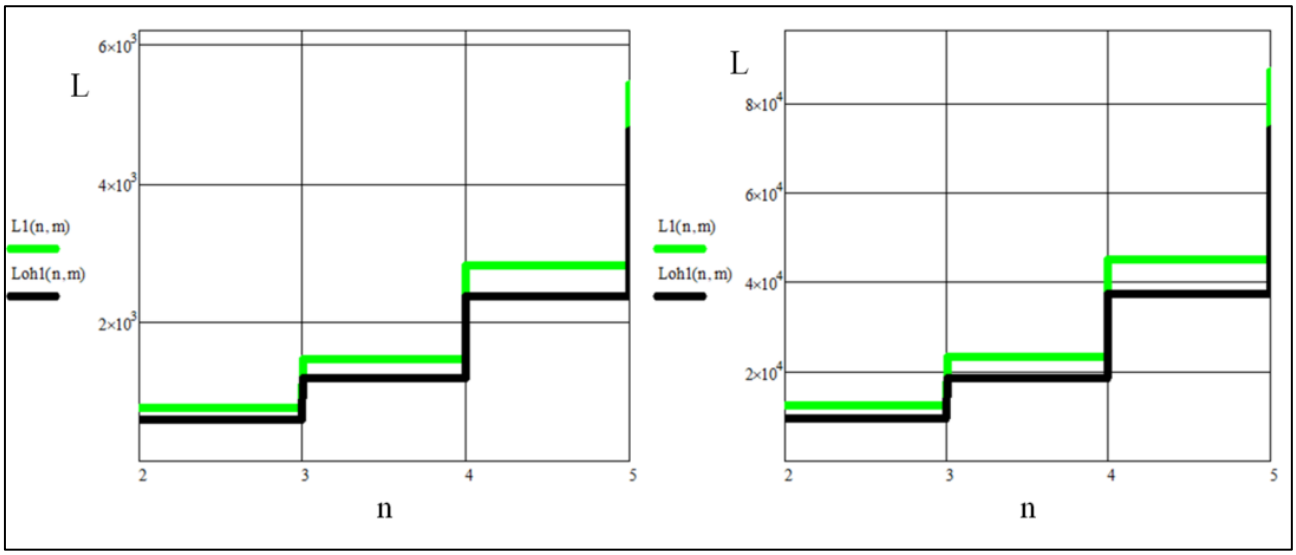
$$L_{oh}(n,m) = 8 \cdot 2^{n+m} + 4 \cdot 2^{n} + 2^{m+n} + 2^{m} \cdot 2 = \\ = 9 \cdot 2^{n+m} + 13 \cdot 2^{n} + 4 \cdot 2^{n} + 2^{m+n} + 2^{m} \cdot 2. \quad (5.7)$$

При реализации системы из m функций в позиционном коде требуется 2^m обычных LUT получаем выражение (5.8):

$$L\ (n,m) = 2^{m} \cdot [10 \cdot 2^{n} + 4n - 2]. \quad (5.8)$$

На рисунке 5.3 зеленым цветом обозначены обычный (позиционный) элемент кодирования, а черным цветом – унитарный элемент при разном числе m.

Выигрыш при реализации систем функций будет еще больше, если учитывать затраты на декомпозицию дерева транзисторов при построении LUT для большого числа переменных.



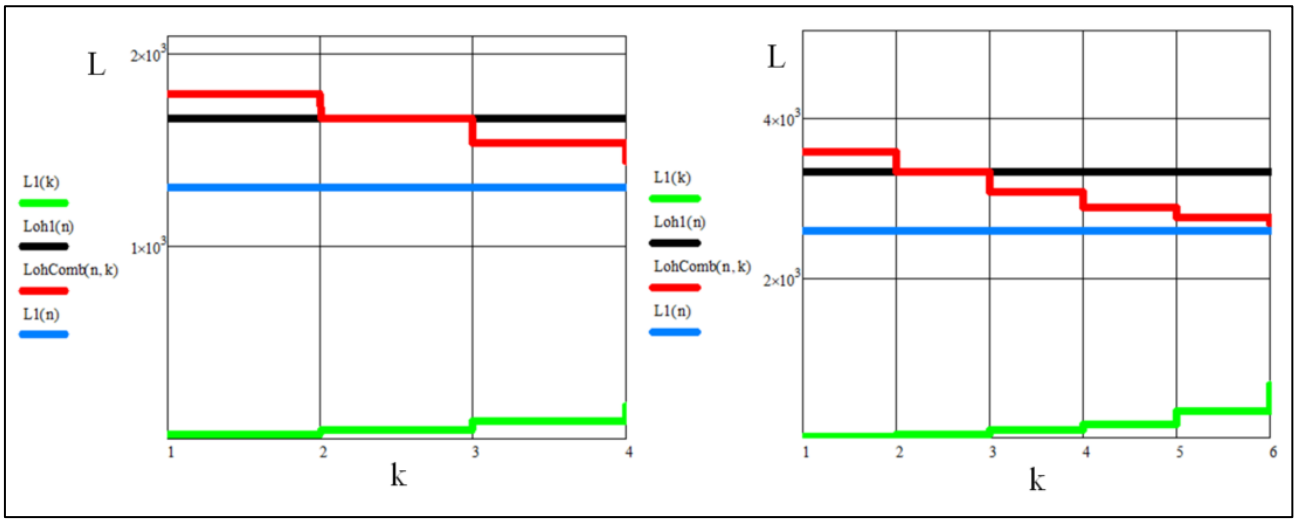
а) б)

Рисунок 5.3 – Реализация системы из *m* функций при их унитарном кодировании в обычном LUT также сложнее, чем в унитарном LUT_{oh}: а) при *m*=4; б) при *m*=8

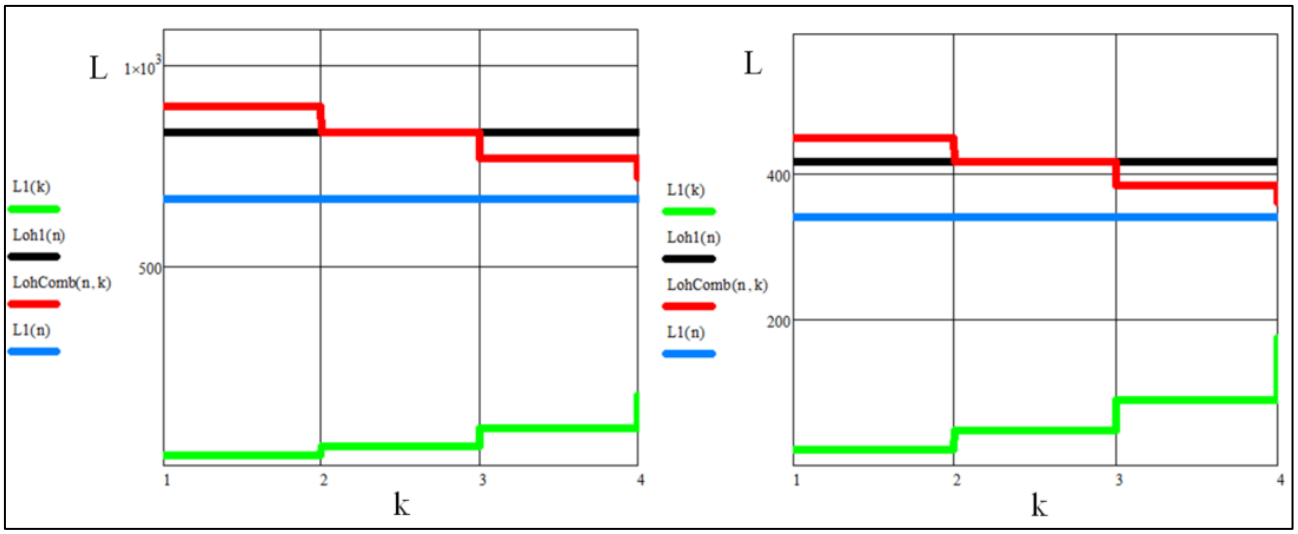
При реализации одной функции в комбинированном коде, с использованием *k* разрядов, вычисляемых с помощью позиционного LUT, адресуемого унитарным кодом, получаем с учетом того, что сложность настройки учитывают позиционные LUT, подключаемые унитарным, выражение (5.9):

$$L_{ohComb}(n,k) = L(k) \cdot 2^{n-k} + 4 \cdot 2^{n-k}. \quad (5.9)$$

На рисунке 5.4 зеленым цветом обозначен обычный (позиционный) элемент кодирования для *k* разрядов, черным цветом – унитарный элемент, красным цветом – комбинированный элемент, синим цветом – обычный (позиционный) элемент от n переменных.



а) б)



в) г)

Рисунок 5.4 – Реализация одной функции при комбинированном кодировании: а) n=7; б) n=8; в) n=6; г) n=5

Видим, что выигрыш получается при k >3. В этом случае условная задержка увеличивается минимум на три единицы за счет обычного LUT с n=3. При n=1 получается даже хуже чем унитарный код.

При реализации системы из m функций в комбинированном коде, используя k разрядов, вычисляемых с помощью позиционного LUT, адресуемого унитарным кодом получаем выражение (5.10):

$$L_{ohComb}(n,k,m) = (L(k) \cdot 2^{n-k}) \cdot m + 4 \cdot 2^{n-k}. \quad (5.10)$$

Однако уже при двух функциях получаем проигрыш, небольшой выигрыш имеет место при вычислении функций большой размерности при k, равных примерно половине переменных, однако при увеличении числа функций, выигрыша не будет. На рисунке 5.5 зеленым цветом обозначен обычный (позиционный) элемент кодирования для k разрядов, черным цветом – унитарный элемент, красным цветом – комбинированный элемент, синим цветом – обычный (позиционный) элемент от n переменных.

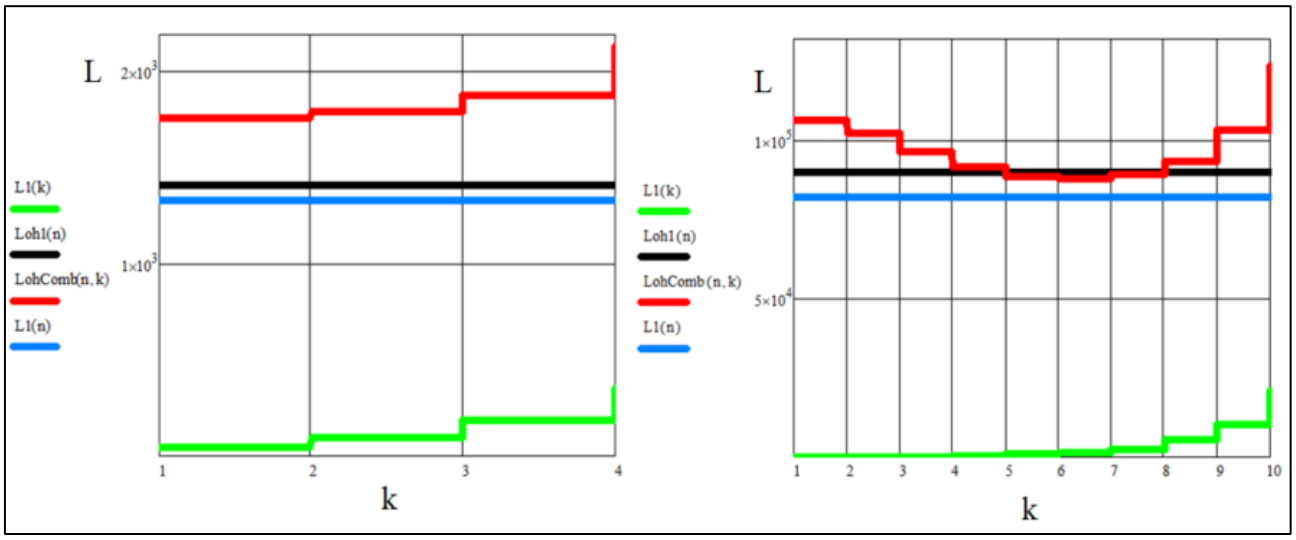


Рисунок 5.5 – Реализация системы из m функций при комбинированном кодировании: а) m=2; n=6; б) m=2; n=12; в) m=4; n=12 ($L_1(n)$, $L_{oh1}(n)$ совпадают на графике); г) m=2; n=8, унитарное кодирование системы

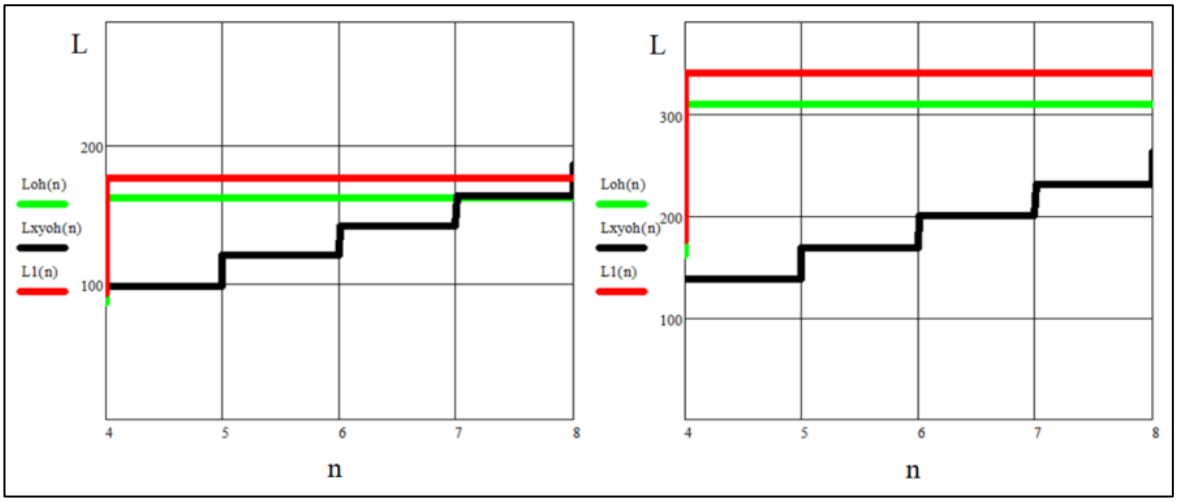
Очевидно, что такой же результат будет и при унитарной реализации системы функций. При реализации двухкоординатного LUT для реализации одной функции с длиной унитарных кодов соответственно n и k могут быть оценены выражениями (5.11)-(5.13):

$$L_{xyoh}(n,k) = 8 \cdot n \cdot k + n \cdot k + 4(n+k) + 2; \quad (5.11)$$

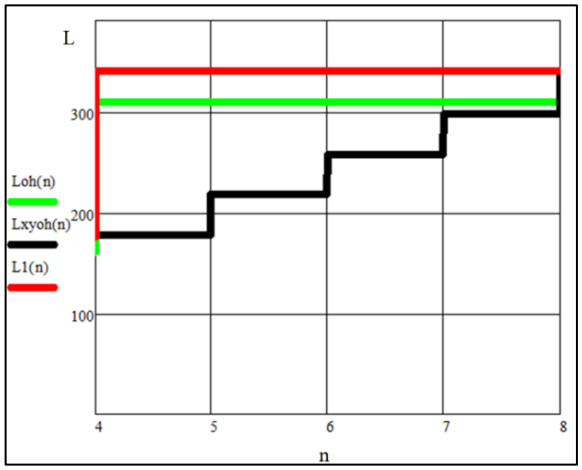
$$L_{oh}(n,k) = 8 \cdot 2^{\lceil \log_2 n \rceil + \lceil \log_2 k \rceil} + 2^{\lceil \log_2 n \rceil + \lceil \log_2 k \rceil} + 4(n+k) + 2; \quad (5.12)$$

$$L_1(n,k) = 2^{\lceil \log_2 n \rceil + \lceil \log_2 k \rceil + 1} + 8 \cdot 2^{\lceil \log_2 n \rceil + \lceil \log_2 k \rceil} + 4(\lceil \log_2 n \rceil + \lceil \log_2 k \rceil). \quad (5.13)$$

Получаем существенный выигрыш перед обычным и унитарным LUT за счет некоторого снижения быстродействия – рисунок 5.6:



а) б)



в)

Рисунок 5.6 – Реализация двух координатного унитарного LUT L_{xyoh} а) при k=2; б) при k=3; в) при k=4

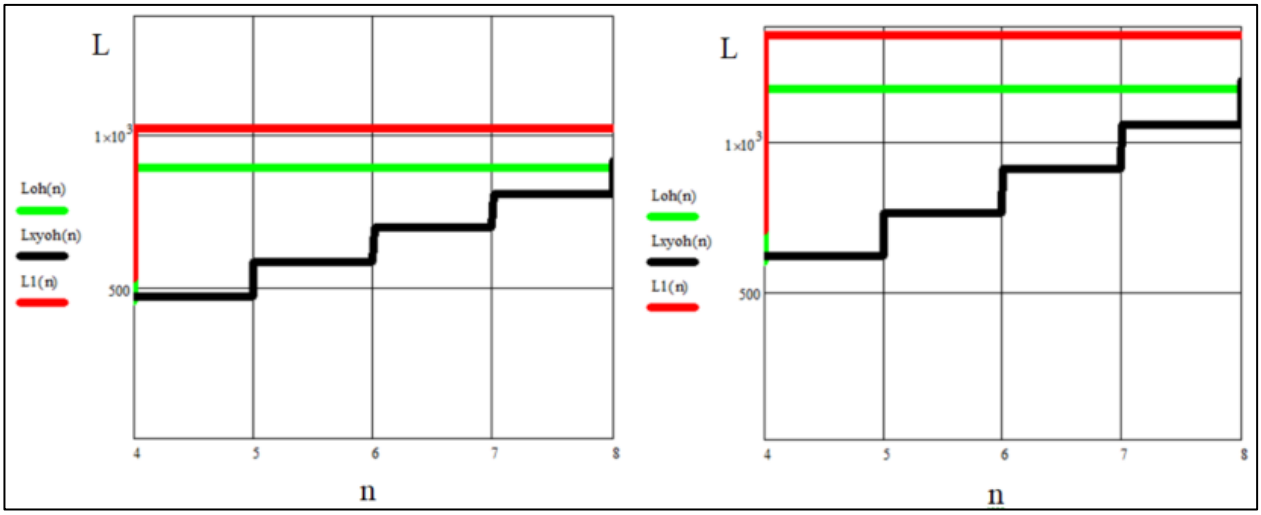
На рисунке 5.6 и 5.7 зеленым цветом обозначен унитарный элемент, черным цветом – унитарный двухкоординатный элемент, красным цветом – обычный (позиционный) элемент.

При реализации системы функций также имеется выигрыш в сложности перед обычным (позиционным) и унитарным LUT, выражения (5.14)-(5.16):

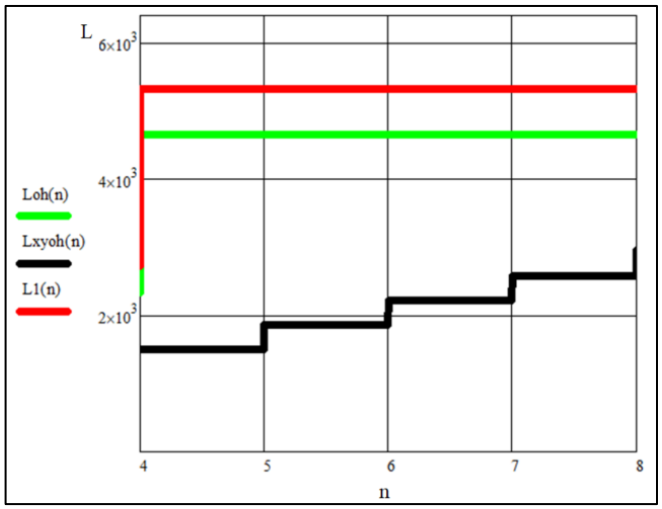
$$L_{xyoh}(n,k,m)=\left(8\cdot n\cdot k+n\cdot k+2\right)\cdot m+4(n+k); \quad (5.14)$$

$$L_{oh}(n,k,m)=\left(8\cdot 2^{\lceil\log_2 n\rceil+\lceil\log_2 k\rceil}+2^{\lceil\log_2 n\rceil+\lceil\log_2 k\rceil}+2\right)\cdot m+4(n+k); \quad (5.15)$$

$$L_1(n,k,m)=\left(2^{\lceil\log_2 n\rceil+\lceil\log_2 k\rceil+1}+8\cdot 2^{\lceil\log_2 n\rceil+\lceil\log_2 k\rceil}+4(\lceil\log_2 n\rceil+\lceil\log_2 k\rceil)\right)\cdot m. \quad (5.16)$$



а) б)



в)

Рисунок 5.7 – Реализация двух координатного унитарного LUT L_{xyoh} а) при k=4, m=3; б) при k=3; m=4; в) при k=5; m=8

Очевидно, что выигрыш сохранится и при реализации системы функций в унитарном коде (заменяем $m \to 2^m$), выражения (5.17)-(5.19):

$$L_{xyoh2^m}(n,k,m) = \left(8 \cdot n \cdot k + n \cdot k + 2\right) \cdot 2^m + 4(n+k); \quad (5.17)$$

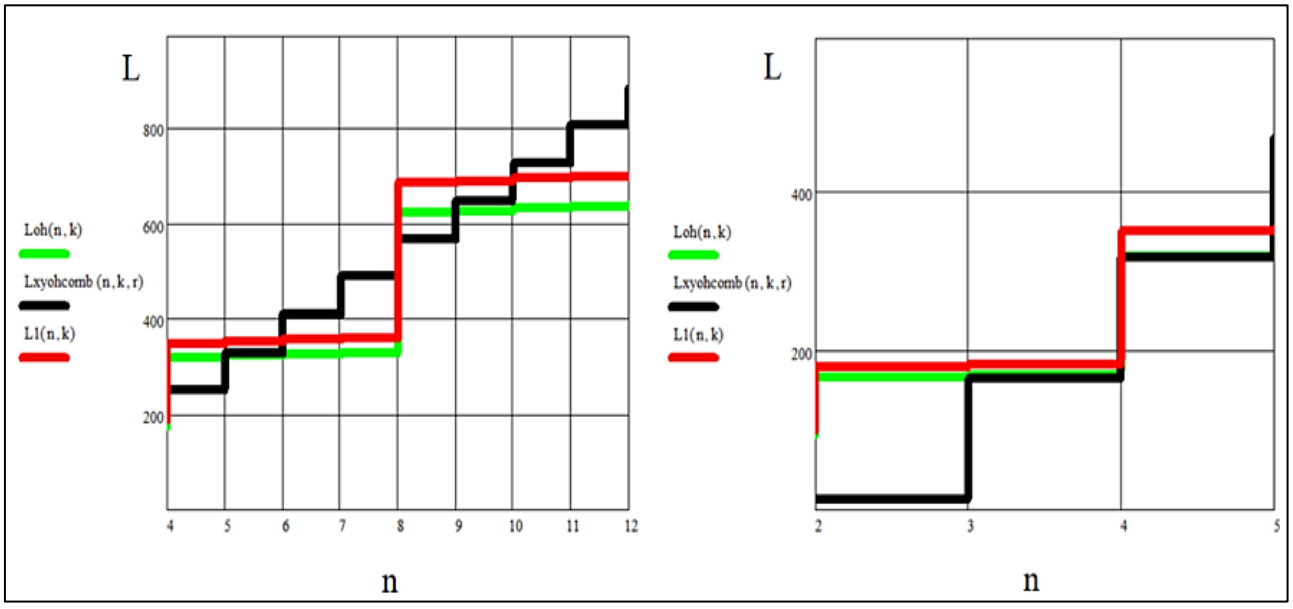
$$L_{oh2^m}(n,k,m) = \left(8 \cdot 2^{\lceil \log_2 n \rceil + \lceil \log_2 k \rceil} + 2^{\lceil \log_2 n \rceil + \lceil \log_2 k \rceil} + 2\right) \cdot 2^m + 4(n+k); \quad (5.18)$$

$$L_{1.2^m}(n,k,m) = \left(2^{\lceil \log_2 n \rceil + \lceil \log_2 k \rceil + 1} + 8 \cdot 2^{\lceil \log_2 n \rceil + \lceil \log_2 k \rceil} + 4(\lceil \log_2 n \rceil + \lceil \log_2 k \rceil)\right) \cdot 2^m. \quad (5.19)$$

При комбинированной реализации двух координатного LUT с обычными LUT на r переменных для реализации одной функции с длиной унитарных кодов соответственно n и k может быть оценена выражением (5.20):

$$L_{xyohcomb}(n,k,m) = L(r) \cdot (n-r) \cdot k + (n-r) \cdot k + 4(n-r+k) + 2. \quad (5.20)$$

Сравнение с обычным (позиционным) и унитарным LUT представлено на рисунке 5.8. На рисунке зеленым цветом обозначен унитарный элемент, черным цветом – унитарный комбинированный двухкоординатный элемент, красным цветом – обычный (позиционный) элемент.



а) б)

Рисунок 5.8 – Реализация двух координатного комбинированного LUT $L_{xyohcomb}$ а) при k=3, r=1; б) при k=3, r=2

Таким образом, получение выигрыша зависит от параметров логического элемента, но быстродействие снижается еще больше, чем у комбинированного однокоординатного варианта.

Оценим эффективность комбинированного 2 LUT. Обычный LUT имеет сложность – выражение (5.21):

$$L\ (n) = 2^{n+1} - 2 + 8 \cdot 2^n + 4n + 2 = 10 \cdot 2^n + 4n. \qquad (5.21)$$

Унитарный LUT – выражение (5.22):

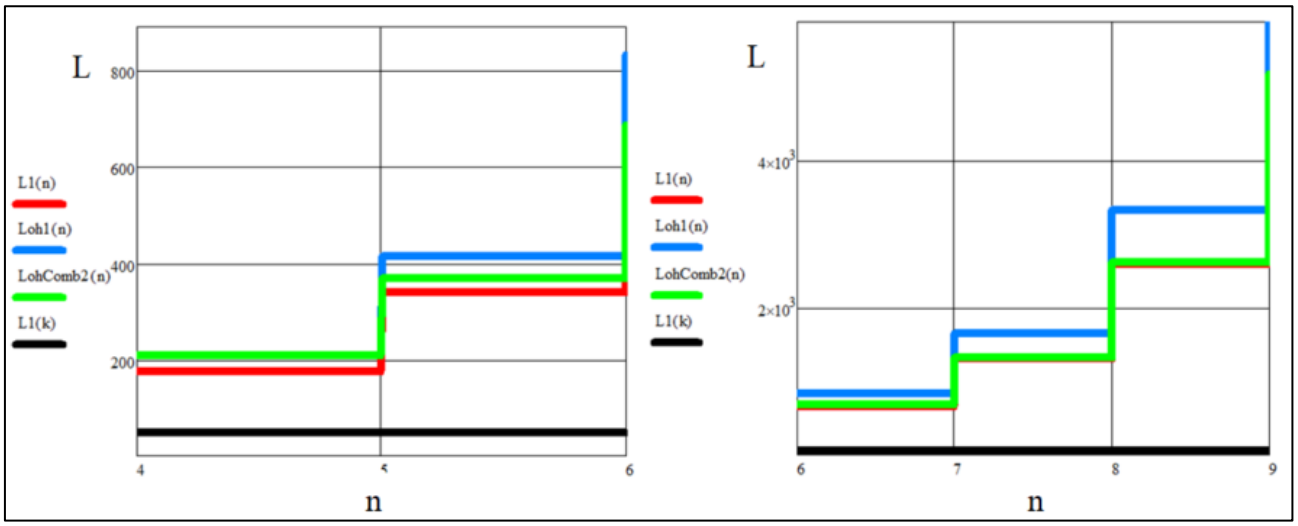
$$L_{oh}(n) = 8 \cdot 2^n + 4 \cdot 2^n + 2^n + 2 = 13 \cdot 2^n + 2. \qquad (5.22)$$

Комбинированный 2 LUT – выражение (5.23):

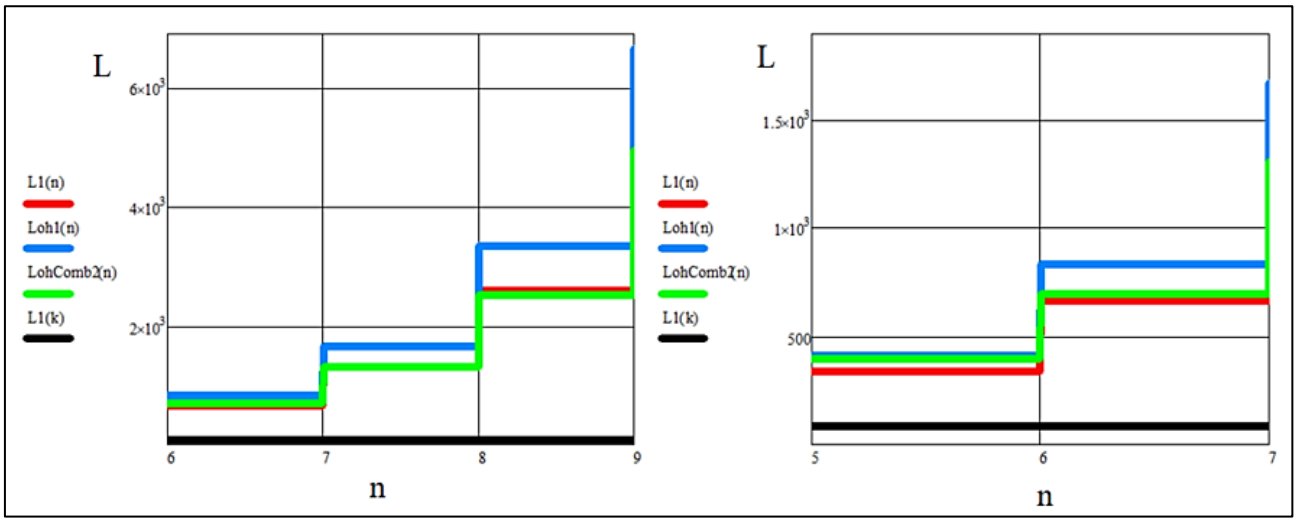
$$L_{ohComb2}(n) = 8 \cdot 2^n + 4 \cdot 2^{n-k} + (2^{n-k}) \cdot 2^k + (L_k). \qquad (5.23)$$

Соответствующие графики имеет вид – рисунок 5.9.

На рисунке 5.9 и рисунке 5.10 черным цветом обозначен обычный (позиционный) элемент на k разрядов, синим цветом – унитарный элемент, красным цветом – обычный (позиционный) элемент, зеленым цветом – унитарный комбинированный элемент.



а) б)



в) г)

Рисунок 5.9 – Реализация комбинированного 2 LUT: а), б) k=2; в), г) k=3

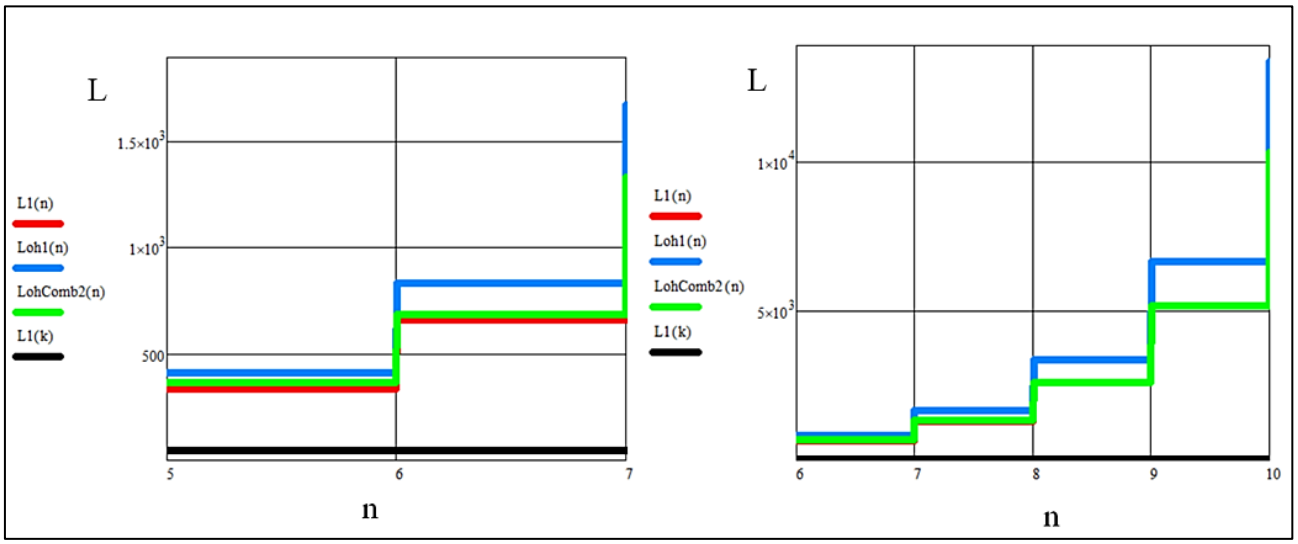
Комбинированный элемент 2 LUT L_k – без сложности настройки, она уже учтена. Комбинированный 2 LUT с использованием двухуровневого унитарного дерева. В этом случае входные переменные n делятся на две подгруппы n_1 и n_2; n_1+n_2=n, выражения (5.24)-(5.26):

$$L_{oh}(k) = 4 \cdot 2^k + 2^k + 2; \qquad (5.24)$$

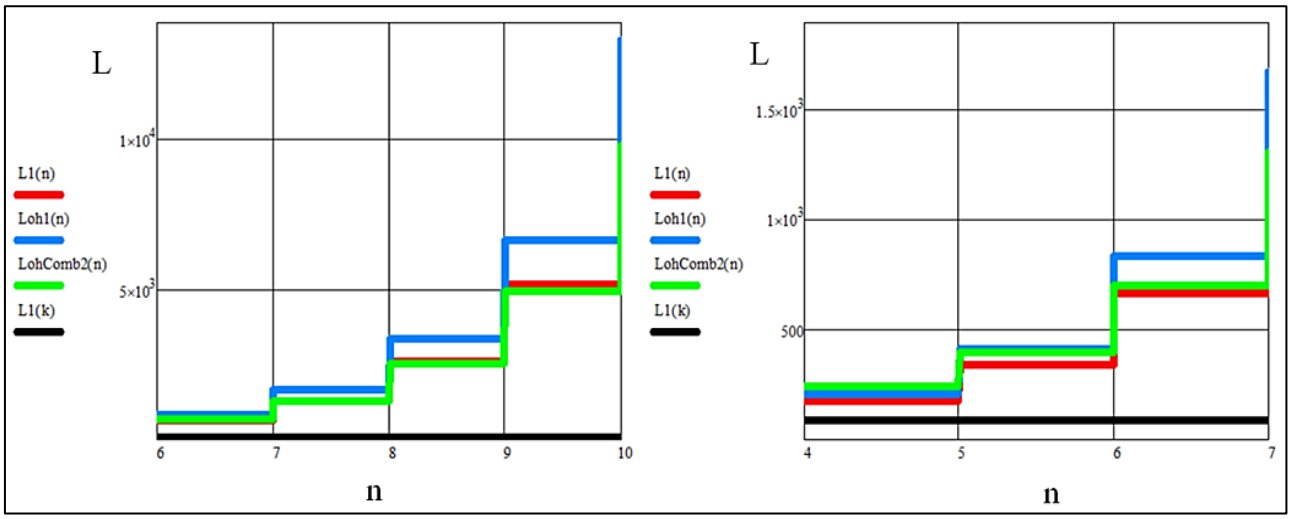
$$L_{oh}(n_1) = 4 \cdot 2^{n_1} + 2^{n_1} + 2; \qquad (5.25)$$

$$L_{ohComb2}(n - n_1) = 8 \cdot 2^n + 4 \cdot 2^{n-n_1} + (2^{n-n_1}) \cdot 2^{n_1} + L_{oh}(n_1). \qquad (5.26)$$

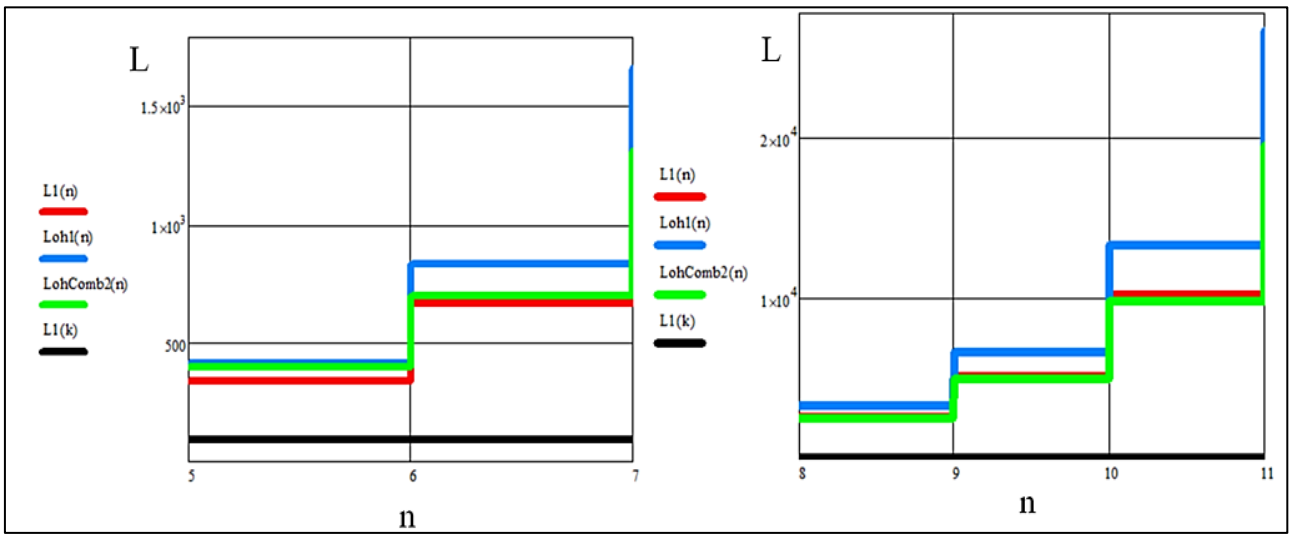
Соответствующие графики имеют вид:



а) б)



в) г)



д) е)

Рисунок 5.10 – Реализация комбинированного 2-1 LUT: а), б) n_1=k=2; в), г), д), е) n_1=k=3

Получается даже лучше обычного LUT, при значительном снижении задержки.

5.2 Сравнение двух вариантов реализации комбинированного элемента в зависимости от расположения n_1, n_2

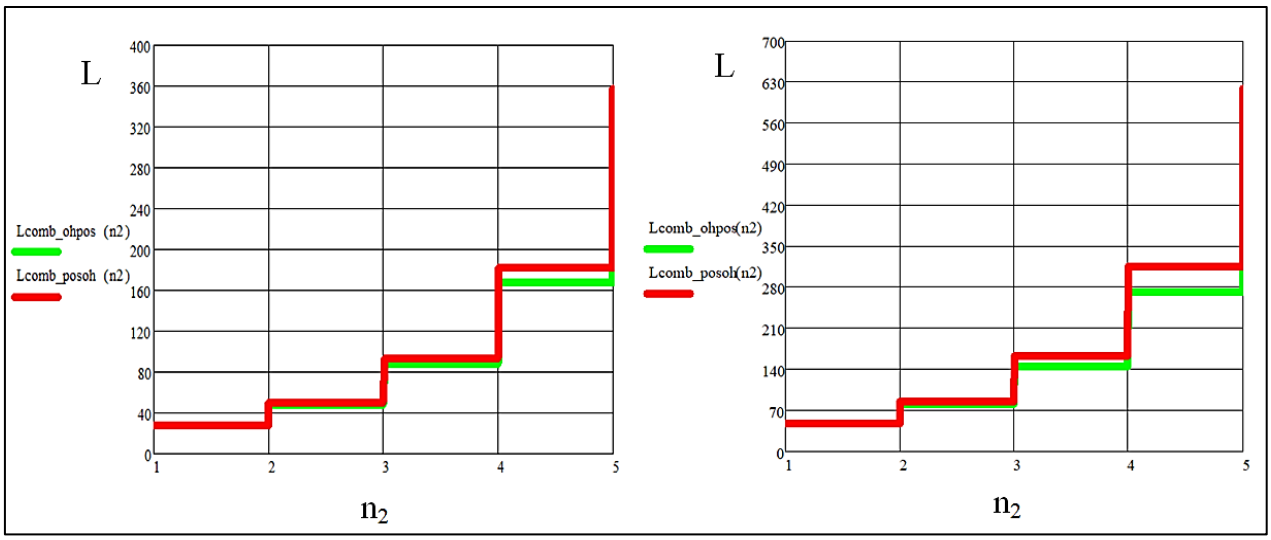
Входные переменные n делятся на две подгруппы n_1 (обычный, позиционный блок, n_1LUT, у которого n_1 входов) и n_2 (унитарный блок, унитарный n_2LUT, у которого 2 n_2 входов); n_1+n_2=n.

Возможны два варианта реализации комбинирования, в зависимости от того, что из n_1, n_2 находится слева и что справа ЛЭ, выражение (5.27):

$$L_{comb:oh/pos} = [(2^{n_1+1} - 2) + 2] + 4(n_1 + 2^{n_2}) + 2^{n_1+n_2} + 2^{n_1+n_2+1};$$
$$L_{comb:pos/oh} = 2^{n_2}(2^{n_1+1} - 2) + 4(n_1 + 2^{n_2}) + 2^{n_2} + 2^{n_1+n_2+1} + 2. \quad (5.27)$$

где $[(2^{n_1+1}) + 2]$– сложность обычного n_1 LUT: многоуровневого дерева передающих транзисторов сложностью $(2^{n_1+1}) - 2$ с одним инвертором на выходе (сложность =2); $4(n_1 + 2^{n_2})$ – сложность пар инверторов по всем входам; $(2^{n_1+n_2})$ – сложность 2^{n_1} деревьев унитарных n_2 LUT; $2^{n_2}(2^{n_1+1} - 2)$ – сложность 2^{n_2} обычных n_1 LUT; $(2^{n_1+n_2+1})$ – сложность настройки логической функции (она одинакова), у коммутаторов связей настройка осуществляется по входам переменных: $2(2^{n_2} + n_1)$.

Сравнение двух вариантов комбинированной реализации в зависимости от расположения n_1, n_2 при n_1 =1 и при n_1 =2 показано на рисунке 5.11,а и рисунке 5.11,б соответственно. Лучше первый вариант – зеленый цвет (унитарный блок слева) для обоих n_1.[103].



а) б)

Рисунок 5.11 – График сравнения сложности L (в количестве транзисторов) вариантов реализации (5.27) в зависимости от количества переменных унитарного блока n_2 при количестве переменных позиционного блока при а) n_1 =1, б) n_1 =2

Далее показано сравнение двух вариантов комбинированной реализации в зависимости от расположения n_1, n_2 при n_1 =3 и n_1 = 4 на рисунке 5.12,а и рисунке 5.12,б соответственно. Лучше первый вариант – зеленый цвет (унитарный блок слева) для обоих n_1.

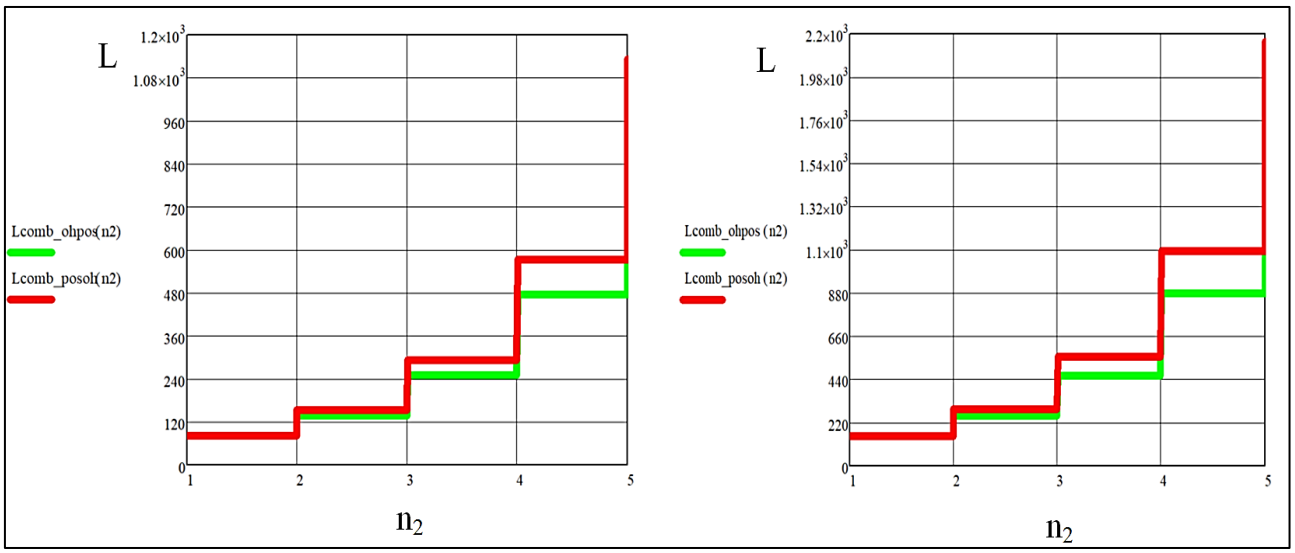


Рисунок 5.12 – График сравнения сложности L (в количестве транзисторов) вариантов реализации (5.27) в зависимости от количества переменных унитарного блока n_2 при количестве переменных позиционного блока при а) n_1 =3, б) n_1 = 4

Красный график (*comb:pos/oh*) – это вид комбинирования, где позиционная часть слева, а унитарная справа, зеленый график (*comb:oh/pos*) – унитарная слева, позиционная справа.

Сравнение двух вариантов комбинированной реализации в зависимости от расположения n_1, n_2 при n_1 = 5 показано на рисунке 5.13. Лучше первый вариант – зеленый цвет (унитарный блок слева).

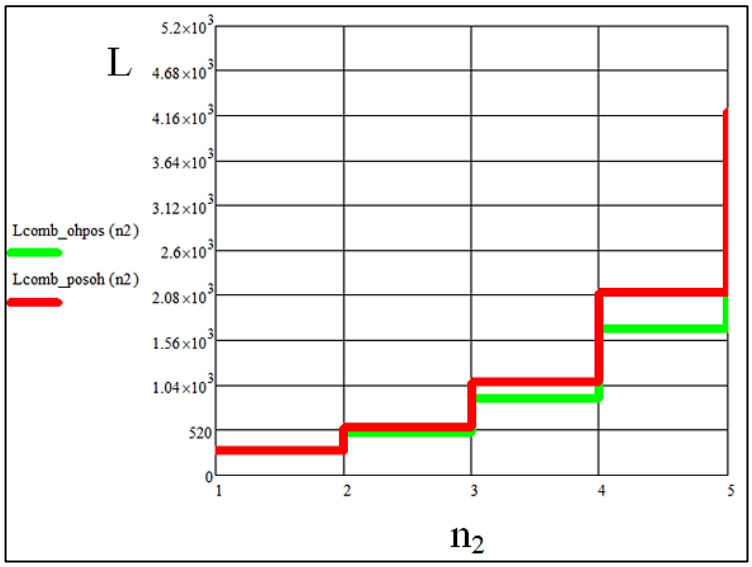


Рисунок 5.13 – Сравнение двух вариантов комбинированной реализации в зависимости от расположения n_1, n_2 при n_1 =5

Выигрыш при n_1=4 получается 20% и он растет при увеличении n_1. Задержка во всех случаях равна суммарной задержке на (n_1+1) транзисторах.

Сравним первый вариант (унитарный блок слева) с обычным (n_1+n_2) LUT_{pos} и унитарным (n_1+n_2) LUT_{oh}, выражение (5.28):

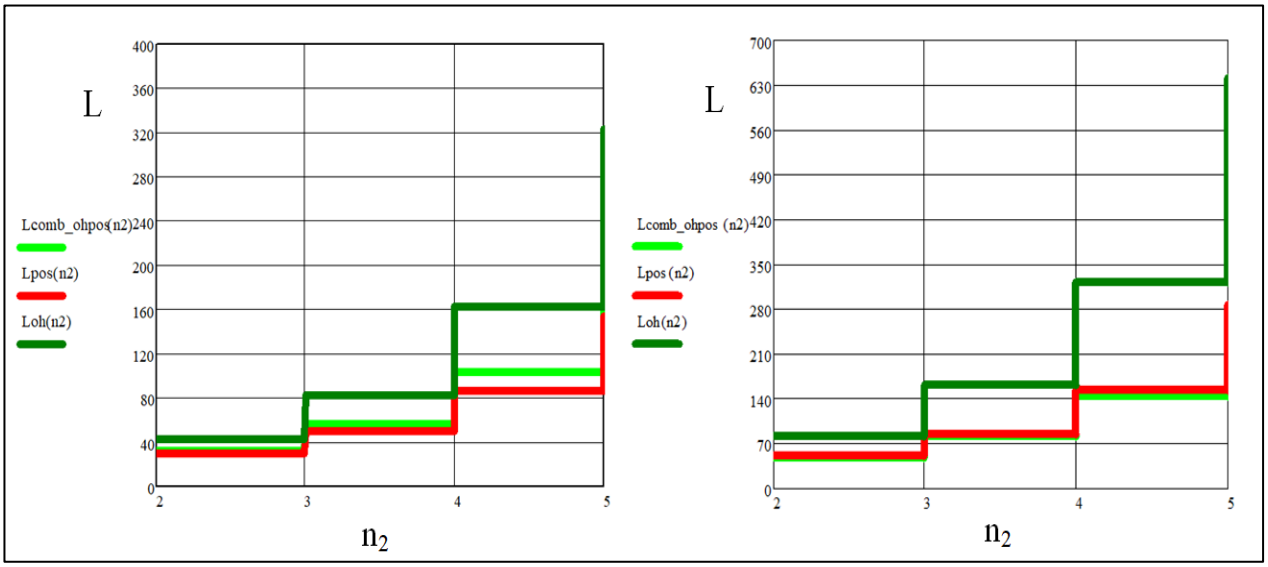
$$L_{pos} = [(2^{n_1+n_2+1} - 2) + 2] + 4(n_1 + n_2) + 2^{n_1+n_2+1};$$
$$L_{oh} = 2^{n_2+n_1} + 4(2^{n_2+n_1}) + 2^{n_1+n_2+1} + 2. \quad (5.28)$$

Задержка для (n_1+n_2) LUT_{pos} равна суммарной задержке на (n_1+n_2) транзисторах. Задержка для (n_1+n_2) LUT_{oh} равна задержке одного транзистора.

Комбинированный элемент $LUT_{comb:oh/pos}$ лучше унитарного, но проигрывает по сложности позиционному, хотя и лучше по задержке в количестве задержек транзисторов в цепочке одной ветви дерева.

Сравнение первого варианта $LUT_{comb:oh/pos}$ (унитарный блок слева) с позиционным LUT_{pos} и унитарным LUT_{oh} при n_1 =1 и при n_1 =2 показано на рисунке 5.14,а и рисунке 5.14,б соответственно. Комбинированный элемент лучше и унитарного и обычного, и лучше по задержке в количестве задержек транзисторов в цепочке одной ветви дерева для обоих n_1.

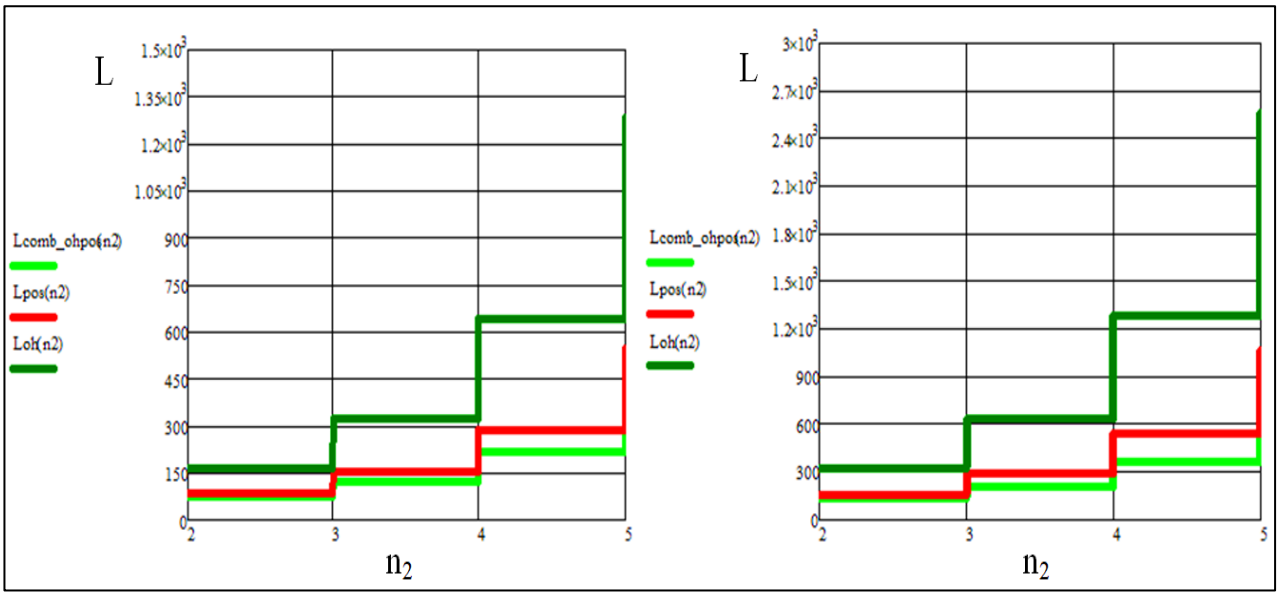
Зеленый график (*comb:oh/pos*) – комбинированный элемент, где унитарная часть слева, позиционная справа, красный график (*pos*) – позиционный вид, темно-зеленый (*oh*) – унитарный.



а) б)

Рисунок 5.14 – Графики сравнения сложности *L* (в количестве транзисторов) варианта $LUT_{comb:oh/pos}$ (унитарный блок слева) с позиционным LUT_{pos} и унитарным LUT_{oh} в зависимости от количества переменных унитарного блока n_2 при количестве переменных позиционного блока а) $n_1 = 1$, б) $n_1 = 2$

Сравнение первого варианта $LUT_{comb:oh/pos}$ (унитарный блок слева) с позиционным LUT_{pos} и унитарным LUT_{oh} при n_1 =3 и при n_1 =4 показано на рисунке 5.15,а и рисунке 5.15,б соответственно. Комбинированный элемент лучше и унитарного и обычного, и лучше по задержке в количестве задержек транзисторов в цепочке одной ветви дерева для обоих n_1..



а) б)

Рисунок 5.15 – Графики сравнения сложности L (в количестве транзисторов) позиционного, унитарного и комбинированного элемента (унитарные блоки слева) в зависимости от количества переменных унитарного блока n_2 при количестве переменных позиционного блока а) $n_1 = 3$, б) $n_1 = 4$

5.3 Оценка увеличения числа связей и конфигурационной памяти

Увеличение числа связей (или настройки для коммутатора) будет в выражении (5.29) в

$$\frac{2^n}{n} \tag{5.29}$$

раз для унитарного логического элемента LUT по сравнению с позиционным и в выражении (5.30)

$$\frac{2^{n_2} + n_1}{n_1 + n_2} \tag{5.30}$$

Раз для комбинированного элемента по сравнению с существующим значением в выражении (5.31):

$$n_1 + n_2 = n, \tag{5.31}$$

где n_1 – часть кода для позиционного кодирования (позиционный блок), n_2 – часть унитарного кодирования (унитарный блок).

При этом в случае получаем временную задержку вычисления логической функции на одном *p*-МОП транзисторе, в случае комбинирования получаем временную задержку вычисления логической функции на n_1+1 *p*-МОП транзисторах.

При реализации коммутатора целесообразно n_1+1<=3. Оценим уменьшение конфигурационной памяти при реализации коммутатора в случае использования комбинированного элемента. Полностью унитарный элемент требует $2^{n_1+n_2}$ ячеек конфигурационной памяти. Комбинированный элемент требует $2^{n_2}+n_1$ ячеек конфигурационной памяти. То есть выигрыш – выражение (5.32):

$$\frac{2^{n_1+n_2}}{2^{n_2}+n_1} \tag{5.32}$$

Введем параметр части основного кода *k* – выражение (5.33). Пусть

$$n_1=\left\lfloor \frac{n}{k} \right\rfloor; n_2=n-\left\lfloor \frac{n}{k} \right\rfloor, \tag{5.33}$$

где$\lfloor\ \rfloor$ ближайшее меньшее натуральное число.

Тогда длина унитарного кода равна выражению (5.34):

$$2^{\left\lceil \log(n-\left\lfloor \frac{n}{k} \right\rfloor,2) \right\rceil}, \tag{5.34}$$

где$\lceil\ \rceil$ ближайшее большее натуральное число, $\log(n-\left\lfloor \frac{n}{k} \right\rfloor, 2)$ – двоичный логарифм.

Поэтому показатель уменьшения конфигурационной памяти примет вид, выражение (5.35):

$$\frac{2^n}{2^{\left\lceil \log(n-\left\lfloor \frac{n}{k} \right\rfloor,2) \right\rceil}+\left\lfloor \frac{n}{k} \right\rfloor}. \tag{5.35}$$

Причем, временная задержка в количестве *p*-МОП транзисторов имеет вид – выражение (5.36):

$$\left\lfloor \frac{n}{k} \right\rfloor + 1. \qquad (5.36)$$

Таким образом, можно для заданных n и k и τ определить выигрыш в памяти – выражение (5.37):

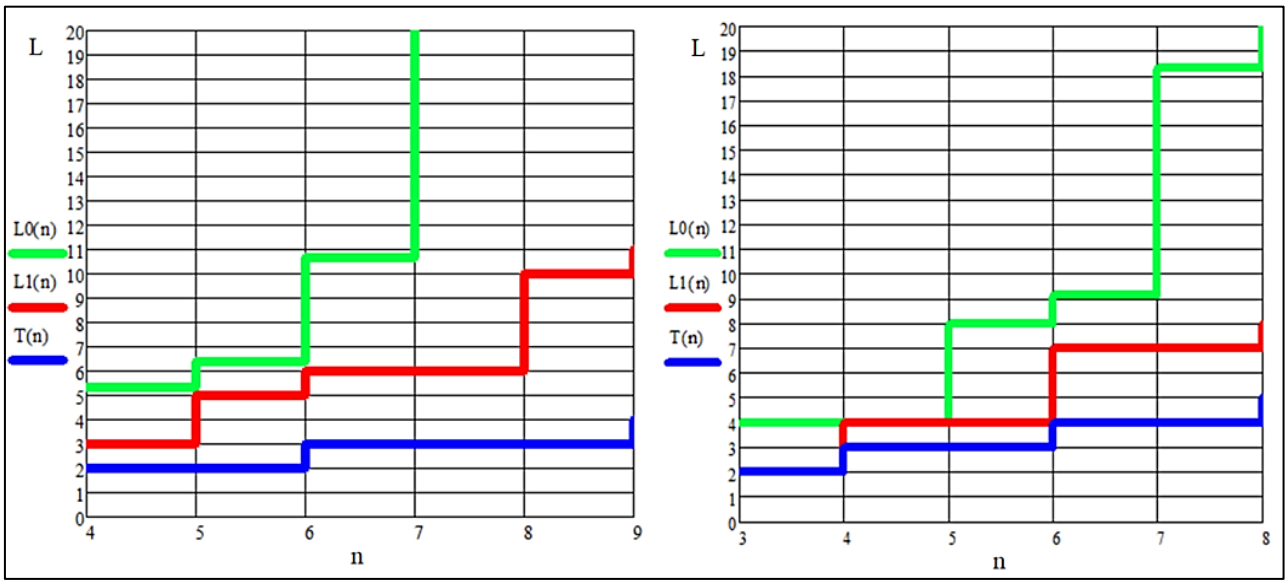
$$L0(n) = \frac{2^n}{2^{\left\lceil \log\left(n - \left\lfloor \frac{n}{k} \right\rfloor, 2\right) \right\rceil} + \left\lfloor \frac{n}{k} \right\rfloor} \to \max; при\, T(n) = \left\lfloor \frac{n}{k} \right\rfloor + 1 \le \tau., \qquad (5.37)$$

Здесь выражение (5.38) обозначает длину кода

$$L1(n) = 2^{\left\lceil \log\left(n - \left\lfloor \frac{n}{k} \right\rfloor, 2\right) \right\rceil} + \left\lfloor \frac{n}{k} \right\rfloor. \qquad (5.38)$$

Зеленый график – обозначает выигрыш в памяти, красный график – обозначает длину кода, синий – временную задержку.

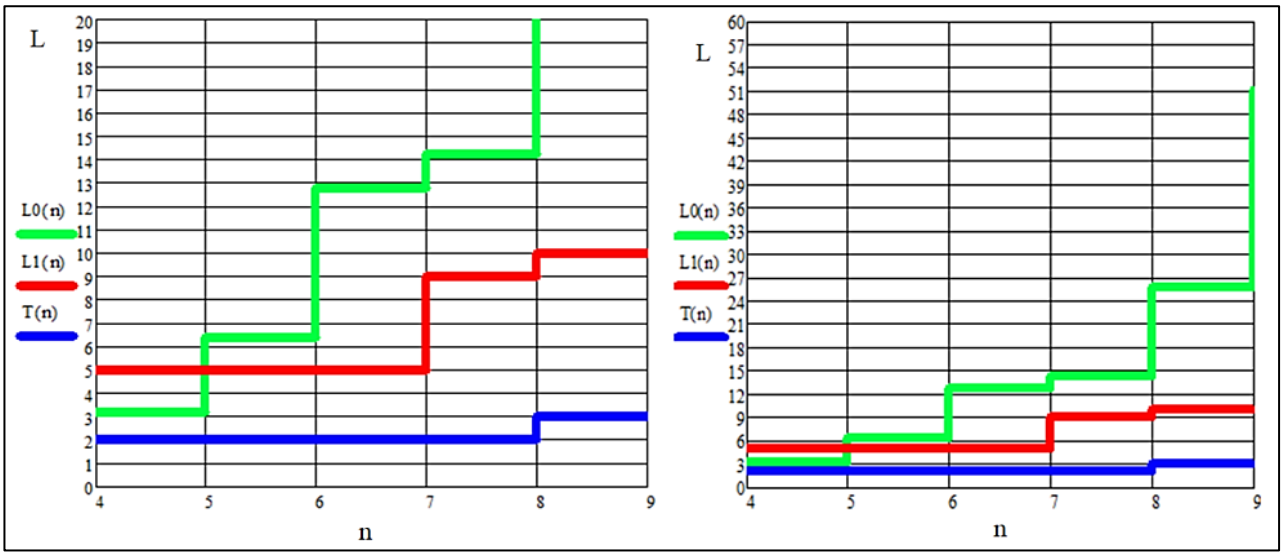
Так, при k=3 и k=2 получаем (рисунок 5.16):



а) б)

Рисунок 5.16 – Изменение выигрыша в памяти L0(n) при длине кода L1(n) и временной задержке T(n) в зависимости от n при а) k=3, б) k=2

При k=4 получаем (рисунок 5.17):



а) б)

Рисунок 5.17 – Изменение выигрыша в памяти L0(n) при длине кода L1(n) и временной задержке T(n) в зависимости от n при а) k=4, б) k=4 в масштабе по оси y от 0 до 60

Если же разрешается путь в три транзистора, то выигрыш в памяти еще больше (более чем в 50 раз) – рисунок 5.18, б. При k=5 получаем рисунок 5.18:

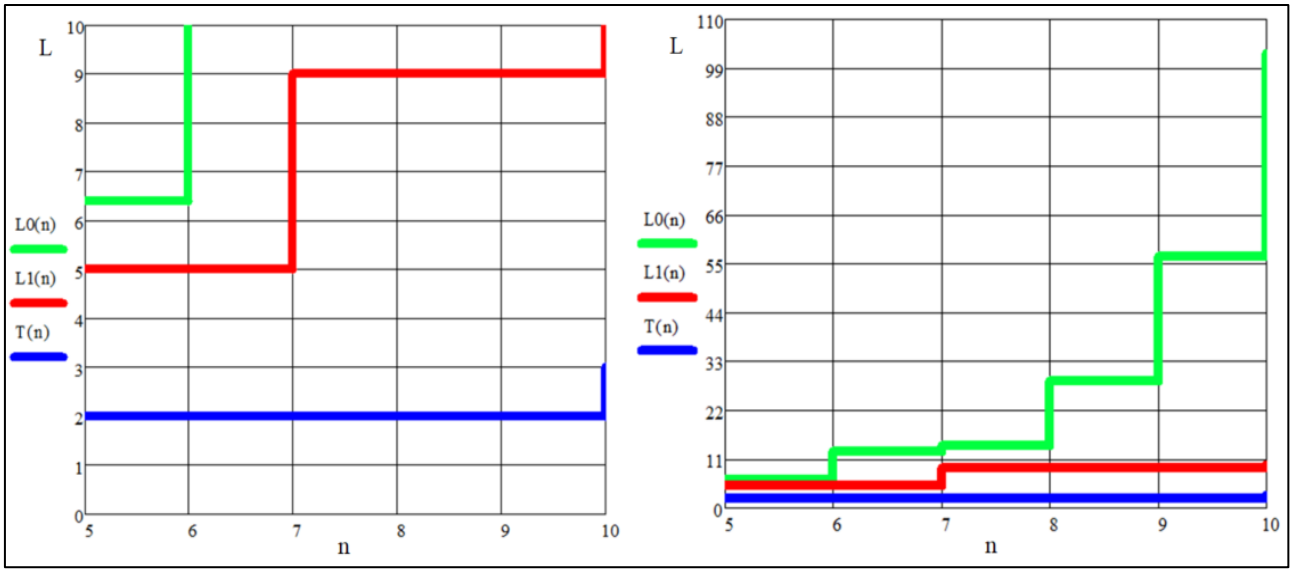


Рисунок 5.18 – Изменение выигрыша в памяти L0(n) при длине кода L1(n) и временной задержке T(n) в зависимости от n при k=5: а) в масштабе по оси y от 0 до 10; б) в масштабе по оси e от 0 до 100

Однако необходимо принять во внимание топологические особенности и характеристики: площадь кристалла, реальную временную задержку (она может изменяться в зависимости не только от количества транзисторов в цепочке, например, может влиять так называемая паразитная емкость), потребляемую мощность на разных частотах. Поэтому требуется учет результатов топологического моделирования и показателей, помимо теоретических в виде количества транзисторов и числа транзисторов в цепочке передачи сигнала.

5.4 Сравнение показателей с учетом результатов топологического моделирования в системе Microwind

В процессе моделирования кроме временной задержки оценивались следующие показатели: занимаемая площадь на кристалле, потребляемая мощность и количество транзисторов. Сравнение полученных показателей разработанных элементов для вычисления логических функций и для коммутации сигналов при напряжении 0,35 В и частоте 0,5 ГГц и модели BSim4, технологии CMOS 32nm приведено в таблице 5.1 и таблице 5.2 соответственно.

Описание полученных показателей:

— Вариант элемента для вычисления логической функции.

— Временная задержка по фронту (Propagation Delay – plh), ps.

— Временная задержка по срезу (Propagation Delay – phl), ps.

— Площадь, nm^2.

— Потребляемая мощность microWatt.

— Число транзисторов в модели (без конфигурационной памяти) с нагрузочным инвертором.

— Число линий связи для переменных при вычислении логической функции или число конфигурационных бит для коммутации.

Таблица 5.1 – Сравнение полученных показателей разработанных топологий элементов для вычисления логических функций

	№ варианта элемента ЛФ	**T фронт, ps**	**T срез, ps**	**S, nm^2**	**W, µW**	**L, шт**	**U, шт**
1	1-LUT	34	187	13,0	0,714	14	1
2	Унитарный 1-LUT	45,7	184	19,0	0,832	18	2
3	2-LUT	270	103	43	1,4	34	2
4	Унитарный 2-LUT	56.6	295	35,5	1,161	32	4
5	3-LUT	349	556	138,6	3,689	74	3
6	Унитарный 3-LUT	50,6	354	88,8	1,832	60	8
7	Комбинированный 3-LUT №1 (два унитарных блока 4 слева, один 1-LUT справа)	312	177	66,8	1,604	70	5
8	Комбинированный 3-LUT №2 (один унитарный блок 4 справа, четыре 1-LUT слева)	55.2	223	111,7	2,202	72	5
9	4-LUT	532	1497	730	9,519	154	4
10	Унитарный 4-LUT	107	462	266,7	1,857	116	16
11	Комбинированный 4-LUT №1 (два унитарных блока 8 слева, один 1-LUT справа)	749	240	243,7	3,847	126	9
12	Комбинированный 4-LUT №2 (один унитарный блок 8 справа, восемь 1-LUT слева)	374	726	369,9	4,739	140	9
13	Комбинированный 4-LUT №3 (четыре унитарных блока 4 слева, один 2-LUT справа)	446	272	449,2	4,521	146	6
14	Комбинированный 4-LUT №4 (один унитарный блок 4 справа, четыре 2-LUT слева)	431	771	412,2	5,983	152	6

Таблица 5.2 – Сравнение полученных показателей разработанных топологий элементов для коммутации сигналов

	№ варианта коммутатора	**T фронт, ps**	**T срез, ps**	**S, nm^2**	**W, µW**	**L, шт**	**U, шт**
1	1-LUT	32,3	90,3	6,7	0,418	14	1
2	Унитарный 1-LUT	45,4	157	19,0	0,628	18	2
3	2-LUT	288	203	47.6	1,674	34	2
4	Унитарный 2-LUT	50	269	29,5	0,949	32	4
5	3-LUT	399	567	128,6	3,424	74	3
6	Унитарный 3-LUT	50,6	271	46,6	1,110	60	8
7	Комбинированный 3-LUT №1 (два унитарных блока 4 слева, один 1-LUT справа)	486	272	82,7	1,986	70	5
8	Комбинированный 3-LUT №2 (один унитарный блок 4 справа, четыре 1-LUT слева)	340	302	92,8	2,261	72	5
9	4-LUT	711	854	680	7,223	154	4
10	Унитарный 4-LUT	64,7	604	138,7	2,025	116	16
11	Комбинированный 4-LUT №1 (два унитарных блока 8 слева, один 1-LUT справа)	835	373	200,1	2,734	126	9
12	Комбинированный 4-LUT №2 (один унитарный блок 8 справа, восемь 1-LUT слева)	344	719	369,9	3,762	140	9
13	Комбинированный 4-LUT №3 (четыре унитарных блока 4 слева, один 2-LUT справа)	436	686	449,2	4,474	146	6
14	Комбинированный 4-LUT №4 (один унитарный блок 4 справа, четыре 2-LUT слева)	326	558	412,2	6,360	152	6

Предлагаемый комбинированный элемент характеризуется уменьшением числа линий связи для переменных соответствующего логического элемента либо уменьшением объема памяти конфигураций соответствующего коммутатора.

Таким образом, по результатам топологического моделирования при реализации логической функции трех переменных (или коммутации 8 связей) минимальную временную задержку 223 ps обеспечивает комбинированный вариант №2 (таблица 1 строка №8), временная задержка оказалось меньше даже унитарного варианта (таблица 1 строка №6, 354 ps). По числу линий связи (5 шт.) комбинированные варианты №1 и №2 демонстрируют средние показатели относительно унитарного и позиционного элементов. Обычный (позиционный) элемент (таблица 1 строка №5) дает минимальное число связей переменных – 3 шт. (или конфигурационных бит), но характеризуется большими: временной задержкой (556 ps), занимаемой площадью (138,6 nm^2), количеством транзисторов (74 шт.) и потребляемой мощностью (3,689 μW).

Для элементов с реализацией четырех переменных были получены следующие результаты: минимальную временную задержку 446 ps обеспечивает комбинированный вариант №3 (таблица 1 строка №13), но проигрывает всем вариантам по занимаемой площади. По числу линий связи (6 шт.) комбинированные варианты №3 и №4 демонстрируют средние показатели относительно унитарного и позиционного элементов. Обычный (позиционный) элемент (таблица 1 строка №9) дает минимальное число связей переменных – 4 шт. (или конфигурационных бит), но характеризуется большими значениями по остальным показателям.

По таблице 5.3 может быть осуществлен ориентировочный выбор варианта по постановке задачи (3), (4), если задана требуемая разрядность $n>3$ и некоторые ограничения. Например, если задано $n=5$, то минимизация временной задержки с ограничением на число линий связи переменных (<=10) дает вариант №4 (3 транзистора в пути сигнала) при одинаковой сложности в

числе транзисторов с вариантом №3 «2 транзистора в пути сигнала», имеющего 17 линий связи. Если минимизируется число линий связи при ограничении на задержку, например, при n=6 (<=3 транзистора в пути сигнала) то получаем вариант №7, то есть 18 линий связи при одинаковой сложности в числе транзисторов с вариантом №6 «2 транзистора в пути сигнала», имеющего 33 линии связи. Увеличение числа линий связи определяет также увеличение сложности соответствующих коммутаторов, не указанное в таблице 5.2.

Таблица 5.3 – Ориентировочные значения показателей сложности типовых вариантов реализации элементов для вычисления логических функций n>3 или коммутации связей.

№ п/п	Разрядность			Число линий связи / объем конфигурационной памяти			Ориентировочная временная задержка в числе транзисторов на пути сигнала (без декомпозиции)			Ориентировочное число транзисторов (без декомпозиции)		
	Позиц. часть	Унит. часть	Общая									
	n_1	n_2	n	Унит.	Позиц.	Комб.	Унит.	Позиц.	Комб.	Унит.	Позиц.	Комб.
1	1	2	3	8	3	5	1	3	2	58	46	48
2	2	1	3	8	3	4	1	3	3	58	46	48
3	1	3	4	16	4	9	1	>4	2	114	82	88
4	3	1	4	16	4	5	1	>4	>4	114	82	88
5	2	2	4	16	4	3	1	2	3	114	82	80
6	1	4	5	32	5	17	1	>5	2	226	150	168
7	2	3	5	32	5	10	1	>5	3	226	150	168
8	3	2	5	32	5	4	1	>5	>4	226	150	140
9	1	5	6	64	6	33	1	>6	2	450	282	328
10	2	4	6	64	6	18	1	>6	3	450	282	328
11	3	3	6	64	6	11	1	>6	>4	450	282	252
12	4	2	6	64	6	12	1	>6	>5	450	282	256
13	5	1	6	64	6	7	1	>6	>6	450	282	284
14	1	6	7	128	7	65	1	>7	2	898	542	648
15	2	5	7	128	7	34	3	>7	3	898	542	528
16	3	4	7	128	7	19	1	>7	>4	898	542	476
17	4	3	7	128	7	12	1	>7	>5	898	542	464
18	5	2	7	128	7	9	1	>7	>6	898	542	484

Продолжение таблицы 5.3

№ п/п	Разрядность			Число линий связи / объем конфигурационной памяти			Ориентировочная временная задержка в числе транзисторов на пути сигнала (без декомпозиции)			Ориентировочное число транзисторов (без декомпозиции)		
	Позиц. часть	Унит. часть	Общая									
	n_1	n_2	n	Унит.	Позиц.	Комб.	Унит.	Позиц.	Комб.	Унит.	Позиц.	Комб.
19	6	1	7	128	7	8	1	>7	>7	898	542	544
20	1	7	8	256	8	257	1	>8	2	1794	1058	1288
21	2	6	8	256	8	66	1	>8	3	1794	1058	1040
22	3	5	8	256	8	33	1	>8	>4	1794	1058	924
23	4	4	8	256	8	20	1	>8	>5	1794	1058	880
24	5	3	8	256	8	13	1	>8	>6	1794	1058	884
25	6	2	8	256	8	10	1	>8	>7	1794	1058	936
26	7	1	8	256	8	9	1	>8	>8	1794	1058	1060

5.5 Выводы по главе 5

1) Ориентировочные расчеты по выведенным формулам оценок сложности в системе компьютерной алгебры Mathcad показывают, что унитарный LUT выигрывает в быстродействии (при учете только пути в количестве транзисторов) у обычного (позиционного), хотя и проигрывает в сложности (по количеству транзисторов). Однако при реализации систем функций унитарный LUT выигрывает и в быстродействии и в сложности у обычного LUT. Но при этом возрастает количество и сложность связей. Значение k с которого начинается выигрыш комбинированного LUT, реализующего всего одну функцию, не зависит от числа переменных и начинается всегда с k=3. При k=2 выигрыша нет, а при k=1 получается даже сложнее исходного унитарного варианта.

2) Комбинированный LUT по показателю аппаратных затрат в количестве транзисторов целесообразен лишь при реализации одной функции. При этом в случае n =1 получается даже хуже и по сложности, и по быстродействию, чем унитарный LUT. В случае n=2 получается эквивалентно по сложности унитарному LUT, но хуже по быстродействию (при учете только пути в количестве транзисторов). Такой вариант имеет смысл только n >=3. В случае двух и более функций он проигрывает и по сложности, и по быстродействию перед полностью унитарным LUT в диапазоне переменных, реализованных в ПЛИС в настоящее время.

3) Комбинированный LUT позволяет получить выигрыш (по количеству транзисторов) реализации системы функций перед полностью унитарным LUT только при очень большом значении переменных (больше 11, что в настоящее время не реализовано) и небольшом числе функций, но при резко снижается быстродействие (при учете только пути в количестве транзисторов).

4) Двух координатный LUT даёт существенный выигрыш (по количеству транзисторов)перед обычным и унитарным LUT за счет некоторого снижения

быстродействия (при учете только пути в количестве транзисторов) как при реализации одной функции, так и для систем функций.

5) Комбинированный LUT в ряде случаев по сложности (в количестве транзисторов) выигрывает у обычного за счет некоторого снижения задержки (при учете только пути в количестве транзисторов).

6) Более точные оценки требуют относительно сложного топологического моделирования рассматриваемых при проектировании вариантов с учетом проектных норм < 10 нм, осложняемого существующими в настоящее время технологическими ограничениями в РФ.

7) Моделирование с использованием модели BSim4 и технологии CMOS 32nm показало, что разработанные соответствующие комбинированные элементы в некоторых вариантах по временной задержке превосходят унитарные элементы, например, комбинированный 4-LUT №3 (четыре унитарных блока 4 слева, один 2-LUT справа) лучше унитарного, но проигрывает в площади кристалла.

ЗАКЛЮЧЕНИЕ

Представленная диссертационная работа посвящена решению актуальной научной задачи разработки научно-методического аппарата синтеза элементов ПЛИС с использованием комбинированного кодирования, отличающихся лучшими характеристиками по быстродействию при допустимом увеличении сложности. Комбинированный элемент 3-LUT позволяет получить задержку более чем в 2,4 раза меньше, чем у бинарного за счет увеличения числа линий связи, а число связей уменьшить в 1,6 по сравнению с вариантом элемента с унитарным кодированием, проигрывая ему в площади, потребляемой мощности и количестве транзисторов, но выигрывая в задержке.

Получен патент на разработанный по предложенному методу элемент, в котором снижена временная задержка при отсутствии увеличения сложности в количестве транзисторов. Предложенные элементы могут быть использованы как для вычисления логических функций, так и для маршрутизации связей.

Разработаны схемы электрические функциональные и принципиальные, а также топологии новых элементов и программа их синтеза. В результате топологического моделирования получены показатели временной задержки, потребляемой мощность и занимаемой площади на кристалле, подтверждающие эффективность предложенного метода для реализации элементов FPGA. Исходя из результатов внедрения в проекте отдела 52 «Архитектура и схемотехника инновационных вычислительных систем» ФИЦ ИУ РАН по теме государственного задания «Информационные, управляющие и телекоммуникационные системы 2024-2028» временная задержка снижается более, чем на 15%, а также снижаются аппаратурные затраты более чем на 20% в зависимости от разрядности унитарной и позиционной части.

Разработана новая модель элемента ПЛИС, которая позволяет комбинировать позиционное и унитарное кодирование.

Метод и алгоритм синтеза позволяют создавать новые элементы, отличающиеся лучшими характеристиками по быстродействию при допустимом увеличении сложности.

Получены математические выражения оценок сложности логических элементов LUT с комбинированным кодированием в количестве требуемых транзисторов, позволяющие выбирать заданный вариант кодирования.

Научная новизна в представленных результатах состоит том, что предложенные модели, метод, алгоритм и оценки сложности дополняют и развивают существующий научно-методический аппарат синтеза элементов ПЛИС средствами, обеспечивающими повышение быстродействия при вычислении логических функций или обеспечивающими снижение объёма памяти и аппаратных затраты на реализацию межсоединений в коммутаторе при ограничениях на временную задержку. Дальнейшие исследования целесообразно продолжать в направлении улучшения и уточнения оценок характеристик ПЛИС, построенных по технологическим нормам < 10 нм и с использованием новых «объёмных» транзисторов типа Tri-Gate, Gate-All-Around.

Список сокращений

АЛМ – адаптивный логический модуль;

БИС – большая интегральная схема;

БМК – базовые матричные кристаллы;

ДНФ – дизъюнктивная нормальная форма;

ИМС – интегральная микросхема;

ИПУ РАН – Институт проблем управления Российской академии наук;

КЛБ – конфигурируемый логический блок;

КМОП – комплементарная структура металл-оксид-полупроводник;

ЛЭ – логический элемент;

ЛС – логическая схема;

ЛФ – логическая функция;

МОП – металл-оксид-полупроводник;

ОЗУ – оперативное запоминающее устройство;

ПЗУ – постоянное запоминающее устройство;

ППЗУ – перепрограммируемое постоянное запоминающее устройство;

ПЛМ – программируемая логическая матрица;

ПМЛ – программируемая матрица логики;

ПЛИС – программируемые логические интегральные схемы;

ПАИС – программируемые аналоговые интегральные схемы;

ППВМ – программируемые пользователем вентильные матрицы;

САПР – Система автоматизированного проектирования;

СДНФ – совершенная дизъюнктивная нормальная форма;

ССМ – система схемотехнического моделирования;

ФГБОУ ВО ПНИПУ – Федеральное государственное бюджетное;

образовательное учреждение высшего образования Пермский национальный исследовательский политехнический университет;

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

ALM – Adaptive logic modules;

ASIC – application-specific integrated circuit;

BM – Benchmark Circuit – Benchmark;

CB – Connection Box;

CNTFET – Carbon nanotube field-effect transistor;

CPLD – Complex Programmable Logic Device;

FPGA – Field-Programmable Gate Array;

LAB – logic array block;

LE – Logic Element (Basic Logic Element);

LUT – Look Up Table;

MLAB – Memory LAB;

PAL – Programmable Array Logic;

PLA – Programmable logic array;

SB – Switch Box;

SRAM – Static Random Access Memory;

ULA – Uncommited Logic Array.

Библиографический список

1. Угрюмов Е.П. Цифровая схемотехника: учеб. пособие. – 3-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2010. – 782 с.

2. Строгонов А.В., Цыбин С.А. Программируемая коммутация ПЛИС: взгляд изнутри [Электронный ресурс]. – URL: https://elibrary.ru/download/elibrary_15643673_59173104.pdf (дата обращения: 17.06.2024).

3. Строгонов А.В., Городков П. Современные тенденции развития ПЛИС: от системной интеграции к искусственному интеллекту. Электроника: Наука, технология, бизнес. 2020. № 4 (195). С. 46-56.

4. Intel® FPGAs and SoC FPGAs [Электронный ресурс]. – URL: https://www.intel.in/content/www/in/en/products/details/fpga.html (дата обращения: 17.11.2021).

5. Intel® Stratix® 10 Logic Array Blocks and Adaptive Logic Modules [Электронный ресурс]. – URL: User Guidefile:///C:/Users/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C/Downloads/ug-s10-lab-683699-666917.pdf(дата обращения: 18.11.2021)

6. FPGA Architecture White Paper. [Электронный ресурс]. – Режим доступа: https://www.altera.com/en_US/pdfs/literature/wp/wp-01003.pdf (дата обращения: 21.10.2020).

7. Микросхемы Xilinx [Электронный ресурс]. – URL: https://acomsupply.com/proizvoditely/xilinx/?yclid=14716191392403030015 (дата обращения: 30.06.2024).

8. 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.

9. Daniel Song, Eugene Wu, William Song, Bow-Nan Cheng. Resource-Efficient and Power-Efficient FPGA Frequency Channelizer Using Novel Systolic Array Architectures. DOI: 10.1109/MILCOM55135.2022.10017771. [Электронный ресурс]. – URL: https://ieeexplore.ieee.org/document/10017771 (дата обращения: 01.07.2024).

10. Narasimhulu Pillutla; Shishir Kumar. FPGA Implementation of High-speed Communication End System (ES) Interface for Avionics Application. DOI: 10.1109/ICARES60489.2023.10329895 [Электронный ресурс]. – URL: https://ieeexplore.ieee.org/document/10329895 (дата обращения: 01.07.2024).

11. Carl Carmichael. Triple Module Redundancy Design Techniques for Virtex FPGAs [Электронный ресурс]. – URL: https://www.xilinx.com/support/documentation/application_notes/xapp197.pdf (дата обращения: 02.07.2024).

12. Intel Reliability Report. [Электронный ресурс]. – URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/rr/rr.pdf (дата обращения: 04.02.2021).

13. A.V. Grekov, S. F. Tyurin. Fault tolerant electronic engine controller. DOI: 10.1109/DESSERT.2018.8409132. [Электронный ресурс]. – URL: https://ieeexplore.ieee.org/document/8409132 (дата обращения: 01.07.2024).

14. Арбузов И., Строгонов А., Городков П. Пример разработки проекта в базисе ПЛИС 5578ТС024. Компоненты и технологии. 2019. № 7 (216). С. 66-69.

15. Строгонов А., Городков П. Обзор ПЛИС китайских производителей. [Электронный ресурс]. – URL: https://www.elibrary.ru/download/elibrary_48565021_33092934.pdf (дата обращения: 17.11.2022).

16. Строгонов А., Городков П. ПЛИС компании GUANGDONG GOWIN SEMICONDUCTOR CORPORATION. Компоненты и технологии. 2020. № 1 (222). С. 84-86.

17. Industrial FPGA market [Электронный ресурс]. – URL: https://www.industryarc.com/Research/industrial-fpga-market-research-800383 (дата обращения: 08.01.2025).

18. Денисов А.Н., Коняхин В.В. Полузаказные БИС на БМК серий 5503 и 5507. Серия практических пособий в 4 книгах / Сер. Мир электроники Том 1 Методология проектирования и освоение производства. Москва, 2019.

19. Аксенова, Г.П. Метод параллельно-последовательного самотестирования в интегральных схемах типа FPGA/ Г.П. Аксенова, В.Ф. Халчев //Автоматика и телемеханика. 2007. – №1. – С. 163-174.

20. Аксенова, Г.П. Контролепригодная архитектура для самотестирования в программируемых логических матричных структурах/ Г.П. Аксенова //Автоматика и телемеханика. 2010. – №12. – С. 154-165.

21. Скобцов, Ю.А. Генетический алгоритм построения функциональных тестов арифметико-логических устройств / Ю.А. Скобцов, Д.Е. Иванов, В.Ю. Скобцов // Восточно-Европейский журнал передовых технологий. – 2014. – Т. 2. № 9 (68). – С. 9-13.

22. Хаханов, В.И. Инфраструктура диагностического обслуживания SoC. / В.И. Хаханов //Вестник Томского университета 2008, №4(5) [Электронный ресурс]. – Режим доступа: http://sun.tsu.ru/mminfo/000063105/inf/05/image/05-074.pdf.

23. A. Drozd, M. Drozd, M. Kuznietsov, “Use of Natural LUT Redundancy to Improve Trustworthiness of FPGA Design,” CEUR Workshop Proceedings, vol. 1614, 2016, pp. 322–331.

24. Kharchenko, V. Design and testing technique of FPGA-based critical systems. 2009 10th International Conference - The Experience of Designing and Application of CAD Systems in Microelectronics [Electronic resource] / V. Kharchenko, O. Siora, V. Sklyar. – Access mode: https://ieeexplore.ieee.org/document/4839839. –28.12.2020.

25. Yervant Zorian. Gest editors’ introduction: Design for Yield and reliability / Z. Yervant, G. Dmytris // IEEE Design & Test of Computers. – May–June 2004. – Pp. 177-182.

26. Дж. Д.Ульмана (Jeffrey David Ullman) [Compilers: Principles, Techniques, and Tools (with A. V. Aho and R. Sethi), Addison-Wesley, Reading MA, 1977, 1986. Computational Aspects of VLSI, Computer Science Press, 1984 ISBN 978-0-914894-95-7

27. Mehta, Nikil. An ultra-low-energy, variation-tolerant FPGA architecture using component-specific mapping. Dissertation (Ph.D.), California Institute of Technology [Электронный ресурс]. – URL: http://thesis.library.caltech.edu/7226/1/Nikil-Mehta-2013.pdf (дата обращения: 01.07.2024).

28. Mehta, Nikil. Exploiting partially defective LUTs: Why you don't need perfect fabrication. DOI: 10.1109/FPT.2013.6718323 [Электронный ресурс]. – URL: https://ieeexplore.ieee.org/document/6718323 (дата обращения: 01.07.2024).

29. Monther Abusultan, Sunil P. Khatri. A comparison of FinFET based FPGA LUT Texas A&M University, College Station, TX, USA. Published in ACM Great Lakes Symposium on VLSI 2014. DOI:10.1145/2591513.2591596 [Электронный ресурс]. – URL: Designshttps://dl.acm.org/citation.cfm?doid=2591513.2591596 (дата обращения: 18.11.2021)

30. Mohammad Ebrahimi, Rezgar Sadeghi, Zainalabedin Navabi. LUT Input Reordering to Reduce Aging Impact on FPGA LUTs. DOI: 10.1109/TC.2020.2974955 [Электронный ресурс]. – URL: https://ieeexplore.ieee.org/document/9001149/citations?tabFilter=papers#citations (дата обращения: 01.07.2024).

31. Dongsu Kim, Taehwan Kim, Yunho Jang, Jongsun Park. SOT-MRAM Based LUT Cell Design for Area and Energy Efficient FPGA. DOI: 10.1109/TCSII.2024.3386926 [Электронный ресурс]. – URL: https://ieeexplore.ieee.org/document/10496443/authors#authors (дата обращения: 01.07.2024).

32. Nirmal Vinod, KV Abhishek Neelakandan, R Udith, K Sayooj Devadas, Kovvuri Dinesh, Anu Chalil, KN Sreehari. Performance Evaluation of LUTs in FPGA in Different Circuit Topologies. DOI: 10.1109/ICCSP48568.2020.9182074 [Электронный ресурс]. –

URL: https://ieeexplore.ieee.org/document/9182074/authors#authors (дата обращения: 01.07.2024).

33. Tyurin S.F. LUT's Sliding Backup. IEEE transactions on device and materials reliability, 2019, vol. 19, pp. 221-225. DOI: 10.1109/TDMR.2019.2898724

34. Tyurin S.F. Green Logic: Green LUT FPGA Concepts, Models and Evaluations. Green IT Engineering: Components, Networks and Systems Implementation, 2017, vol. 105, pp. 241-261. DOI: 10.1007/978-3-319-55595-9

35. Греков, А.В. Повышение отказоустойчивости конфигурируемых блоков программируемых логических интегральных схем на основе функционально полных толерантных элементов: диссертация на соискание учёной степени кандидата технических наук / А.В. Греков. – Пермь, 2011. – 167 с.

36. Громов О.А. Повышение отказоустойчивости программируемых логических интегральных схем на основе КМОП элементов с избыточным базисом: диссертация на соискание ученой степени канд. техн. наук / Громов Олег Александрович, Пермь, 2013 – 157 с.

37. Городилов А.Ю. Методы и алгоритмы диагностирования и реконфигурации логики высоконадёжных ПЛИС: диссертация на соискание ученой степени канд. техн. наук / Городилов Алексей Юрьевич, Пермь, 2016 – 145 с.

38. Каменских А.Н. Комбинированное резервирование самосинхронных схем: диссертация на соискание ученой степени канд. техн. наук / Каменских Антон Николаевич Пермь, 2017 – 138 с.

39. Вихорев Р.В. Логические элементы ПЛИС FPGA для реализации систем функций: диссертация на соискание ученой степени канд. техн. наук / Вихорев Руслан Владимирович. Пермь, 2019. – 192 с.

40. Vikhorev R. Universal logic cells to implement systems functions. Conference of Russian Young Researchers in Electrical and Electronic Engineering. IEEE, 2016. pp. 404–406. DOI: 10.1109/EIConRusNW.2016.7448197

41. Vikhorev R. Improved FPGA logic elements and their simulation. Conference of Russian Young Researchers in Electrical and Electronic Engineering, IEEE, 2018. pp. 275–280. DOI: 10.1109/EIConRus.2018.8317080

42. Tyurin S.F., Vikhorev R.V. A Decoder – Look up Tables for FPGAs // IJC, vol. 20, no. 3, P. 365-373, 2021. doi: 10.47839/ijc.20.3.2282.

43. Скорнякова А.С. Логические элементы ПЛИС FPGA для самосинхронных схем: диссертация на соискание ученой степени канд. техн. наук / Скорнякова Александра Юрьевна. Пермь, 2020. – 185 с.

44. Skornyakova A.Yu., Vikhorev R.V. Self-Timed LUT Layout Simulation. Conference of Russian Young Researchers in Electrical and Electronic Engineering, IEEE, 2020. pp. 176–179. DOI: 10.1109/EIConRus49466.2020.9039374

45. Советов С.И. Логические элементы ПЛИС FPGA, реализующие несколько функций одновременно: диссертация на соискание ученой степени канд. техн. наук / Советов Станислав Игоревич. Пермь, 2024. – 190 с.

46. Советов, С.И. Разработка топологии многофункционального логического элемента плис / С.И. Советов // Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления. – 2023. – № 48. – С. 30–49. DOI: 10.15593/2224-9397/2023.4.02

47. Barkalov A., Titarenko L., Krzywicki K., Mielcarek K. Using codes of output collections for hardware reduction in circuits of LUT-based finite state machines. Electronics. 2022. Т. 11. № 13. С. 2050.

48. FPGA's Programmable Technology. [Электронный ресурс]. – URL: https://www.fpgakey.com/technology/details/fpga-programmable-technology(дата обращения: 08.01.2025).

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

50. Tyurin, S.F. Study of the Multi-Input LUT Complexity / S.F. Tyurin, A.V. Grekov // Radio Electronics, Computer Science, Control. 2018. № 1 (44). – P. 14-21. DOI: 10.15588/1607-3274-2018-1-2.

51. Золотухо Р. Stratix III — новое семейство FPGA фирмы Altera [Электронный ресурс] / Р. Золотуха, Д. Комолов – Режим доступа: http://kit-e.ru/assets/files/pdf/2006_12_30.pdf (дата обращения: 21.10.2020).

52. Современные реализации ПЛИС. [Электронный ресурс]. – URL: http://fpga.parallel.ru/devices.html (дата обращения: 02.07.2024).

53. Programmable Logic Devices [Электронный ресурс]. – URL: http://ee.sharif.edu/~logic_circuits_t/readings/PLD.pdf (дата обращения: 04.11.2018).

54. Mead C. A., Conway L. Introduction to VLSI Systems. [Электронный ресурс] – URL:https://www.researchgate.net/publication/234388249_Introduction_to_VLSI_systems (дата обращения: 18.11.2022)

55. Тюрин, С.Ф. Особенности синтеза по STATE MACHINE FILE в системе QUARTUS II фирмы ALTERA / С.Ф. Тюрин, И.И. Безукладников, О.В. Гончаровский // Информационно-измерительные и управляющие системы. 2016. Т. 14. № 9. С. 39-47.

56. Shannon, Claude E. Von Neumann's Contributions to automata theory [Electronic resource] / Claude E. Shannon. – Access mode: https://pdfs.semanticscholar.org/3903/d10dfccfe2c3e5bee9603644c9ef2a45b9e7.pdf. (дата обращения: 02.07.2024)

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

58. Dongliang Zhang, Kunpeng Wu, Jie Wu; Hanming Tao. Research on Interleaved Hamming Code Verification of FPGA and Reliability Analysis. [Электронный ресурс]. – URL: DOI: 10.1109/ICFTIC59930.2023.10456089 (дата обращения: 02.07.2024)

59. Giovanni Quintarelli, Matteo Bertolucci, Pietro Nannipieri. Design and Implementation of a DVB-S2 Reconfigurable Datapath BCH Encoder for High Data-Rate Payload Data Telemetry. DOI: 10.1109/ACCESS.2023.3327786 [Электронный ресурс]. – URL: https://ieeexplore.ieee.org/document/10296904 (дата обращения: 02.07.2024)

60. A. Hocquenghem. Codes correcteurs d'erreurs // Chiffres. — Paris, 1959. — Septembre (vol. 2). — P. 147–156.

61. R. C. Bose, D. K. Ray-Chaudhuri. On A Class of Error Correcting Binary Group Codes // Information and Control. — 1960. [Электронный ресурс]. – URL: https://repository.lib.ncsu.edu/server/api/core/bitstreams/2fac7e12-6b78-403c-af0f-b19a91282a2b/content (дата обращения: 03.07.2024)

62. Цифровая обработка сигналов : учебное пособие / В. И. Фрейман ; М-во науки и высш. образования Рос. Федерации, Перм. нац. исслед. политехн. ун-т. - Пермь : Изд-во ПНИПУ, 2021. - 114 с. - Утверждено рис. ун-та в качестве учеб. пособия. - ISBN 978-5-398-02542-2.

63. I. Reed, G. Solomon. «Polynomial Codes over Certain Finite Fields». [Электронный ресурс]. – URL: https://www.semanticscholar.org/paper/Polynomial-Codes-Over-Certain-Finite-Fields-Reed-Solomon/5afa7c41ecd3c97a55bc5088e0070fe927133e43 (дата обращения: 03.07.2024)

64. Siva Satya Sri Ganesh Seeram; Shanmukha Naga Naidu Polireddi; Geethu Remadevi Somanathan; Ramesh Bhakthavatchalu. Synthesis of Synchronous Gray Code Counters by Combining Mentor Graphics HDL Designer and Xilinx VIVADO FPGA Flow. DOI: 10.1109/ICCSP48568.2020.9182333 [Электронный ресурс]. – URL: https://ieeexplore.ieee.org/document/9182333 (дата обращения: 02.07.2024)

65. Gray code. [Электронный ресурс]. – URL: https://xlinux.nist.gov/dads/HTML/graycode.html (дата обращения: 03.07.2024)

66. Robert Royce Johnson, «Electronic counter», US Patent No. 3030581, filed in 1953. [Электронный ресурс]. – URL: https://www.legacy.com/us/obituaries/saltlaketribune/name/robert-johnson-obituary?id=21254666 (дата обращения: 03.07.2024)

67. Код Айкена. Уайт, Гарланд С. "Кодированные десятичные системы счисления для цифровых компьютеров". Труды Института радиоинженеров. 41 (10). Институт радиоинженеров (IRE): 1450-1452. doi: 10.1109/JRPROC.1953.274330. eISSN 2162-6634. ISSN 0096-8390. S2CID 51674710.

68. Oleksii Borysenko, Svitlana Matsenko, Sandis Spolitis, Vjaceslavs Bobrovs. Development of the Fibonacci-Octal Error Detection Code for Telecommunication Systems. [Электронный ресурс]. – URL: https://ieeexplore.ieee.org/document/9141620 (дата обращения: 03.07.2024)

69. Xifan Tang, Giovanni De Micheli, Pierre-Emmanuel Gaillardon. A High-Performance FPGA Architecture Using One-Level RRAM-Based Multiplexers. [Электронный ресурс]. – URL: https://ieeexplore.ieee.org/document/7747469/authors#authors DOI: 10.1109/TETC.2016.2630121 (дата обращения: 04.07.2024)

70. Ryan Kenny, Jeff Watt. The Breakthrough Advantage for FPGAs with Tri-Gate Technology [Электронный ресурс]. –https://www.semanticscholar.org/paper/The-Breakthrough-Advantage-for-FPGAs-with-Tri-Gate-Kenny/99f6d408f0f3a33c9b44e789f626ee2bc80c30b3 (дата обращения: 04.07.2024).

71. Трёхмерные транзисторы [Электронный ресурс]. – URL: https://habrahabr.ru/company/intel/blog/118816/ (дата обращения: 04.07.2024).

72. Интегрированные транзисторы CMOS tri-gate [Электронный ресурс]. – URL: http://compress.ru/article.aspx?id=16789 (дата обращения: 04.07.2024).

73. Yuan-Yu Huang, Po-Tsang Huang, Po-Yi Lee, Pin Su. Novel Complementary FeFET- based Lookup Table and Routing Switch Design and their Applications in Energy/Area-Efficient FPGA. [Электронный ресурс]. – URL: https://ieeexplore.ieee.org/document/10103081/authors#authors DOI: 10.1109/EDTM55494.2023.10103081 (дата обращения: 04.07.2024)

74. Jide Zhang, Kaixiang Zhu, Kaichuang Shi, Lingli Wang, Hao Zhou. Efficient FPGA Routing Architecture Exploration Based on Two-Stage MUX. [Электронный

ресурс]. – URL: eshttps://ieeexplore.ieee.org/document/10395964/authors#authors DOI: 10.1109/ASICON58565.2023.10395964 (дата обращения: 04.07.2024)

75. Palle Mahendra, S R Ramesh. FPGA Implementation of High Performance Precise Signed and Unsigned Multiplier using Ternary 6-LUT Architecture. [Электронный ресурс]. – URL: https://ieeexplore.ieee.org/document/9850686 DOI: 10.1109/ICICT54344.2022.9850686 (дата обращения: 04.07.2024)

76. Zeinab Seifoori, Behzad Omidi, Hossein Asadi. PERA: Power-Efficient Routing Architecture for SRAM-Based FPGAs in Dark Silicon Era. [Электронный ресурс]. – URL: https://ieeexplore.ieee.org/document/10273211DOI: 10.1109/TVLSI.2023.3303352 (дата обращения: 04.07.2024)

77. 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.

78. 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.

79. Программируемое логическое устройство / С.Ф. Тюрин, И.А. Васенин, Ю.А. Степченков, Ю.Г. Дьяченко, С.И. Советов // Патент на изобретение RU 2811404 C1. – 11.01.2024. – Заявка от 02.08.2023.

80. National Instruments. Multisim. [Electronic resource]. – Access mode: // http://www.ni.com/multisim/

81. Microwind & Dsch Version 3.5. Available at: https://www.yumpu.com/en/document/view/40386405/microwind-manual-lite-v35pdf-moodle (accessed 5 November 2022).

82. N. Paydavosi. BSIM4v4.8.0 MOSFET Model -User's Manual 2013 Available at: http://bsim.berkeley.edu/BSIM4/BSIM480.zip (accessed: 16.04.2020).

83. Васенин, И.А. Унитарное программирование LUT таблиц для ПЛИС / И.А. Васенин, С.Ф. Тюрин // Инновационные технологии: теория, инструменты, практика : материалы XIV Междунар. интернет-конф. молодых ученых, аспирантов и студентов (16 нояб.-31 дек. 2022 г.) / М-во науки и высш. образования Рос. Федерации, Перм. нац. исслед. политехн. ун-т. - Пермь : ПНИПУ, 2023. - С. 100-107.

84. Васенин, И.А. Топологическое моделирование элемента ПЛИС с комбинированным кодированием входных переменных / И.А. Васенин, С.Ф. Тюрин // Автоматизированные системы управления и информационные технологии : материалы всерос. науч.-практ. конф. (г. Пермь,7-9 июня 2023 г.). Т. 1 / М-во науки и высш. образования Рос. Федерации, Перм. нац. исслед. политехн. ун-т. - Пермь : Изд-во ПНИПУ, 2023. - С. 44-49.

85. Andrew Boutros, Vaughn Betz. FPGA Architecture: Principles and Progression. October 2022IEEE, Circuits and Systems M19agazine 21(2):4-29. DOI: 10.1109/MCAS.2021.3071607 Lab: Vaughn Betz's Lab [Электронный ресурс]. – URL: https://www.researchgate.net/publication/364082937_FPGA_Architecture_Principles_and_Progression (дата обращения: 10.07.2024)

86. Рекордный транзистор со свойствами мемристора не требует постоянного питания в устройствах 6G. [Электронный ресурс]. – URL: https://www.pravda.ru/news/science/2061003 tranzistor/https://www.pravda.ru/news/science/2061003-tranzistor/ (дата обращения: 12.07.2024)

87. Реализация цифровых автоматов в системе Quartus фирмы Altera: учеб. Пособие / С.Ф. Тюрин, А.В. Греков, О.А. Громов – Пермь: Изд-во Перм. гос. техн. ун-та, 2011. – 134с.

88. Тюрин, С. Ф. Схемотехника: учеб. пособие / С. Ф. Тюрин. – Пермь: Изд.-во Перм. нац. исслед. политехн. ун.-та, 2017. – 170 с. – ISBN 978-5-398-01702-1.

89. Тюрин С.Ф., Городилов А.Ю., Вихорев Р.В. Программируемое логическое устройство: патент РФ №2547229; опубл. 10.04.2015, Бюл. №10.

90. S.F. Tyurin, “Hyper redundancy for super reliable FPGAs,” Radioelectron. comput. syst., vol. 97, no. 1, pp. 119–132, Jan. 15. doi: 10.32620/reks.2021.1.11.

91. Sklyar V. Safety-critical Certification of FPGA-based Platform against Requirements of U.S. Nuclear Regulatory Commission (NRC): Industrial Case Study [Electronic resource] / V. Sklyar. – Access mode: http://ceur-ws.org/Vol-1614/paper_32.pdf

92. Проектирование на ПЛИС в МГУ. [Электронный ресурс]. – URL: https://engineering.phys.msu.ru/ru/programmirovanie-dlya-sstudentov-2-kursa/hard-n-soft/fpga (дата обращения: 02.07.2024).

93. Jeffrey D. Ullman. Computational Aspects of VLSI, Computer Science Press, 1984 ISBN 978-0-914894-95-7. [Электронный ресурс]. – URL: https://archive.org/details/computationalasp0000ullm (дата обращения: 03.07.2024)

94. Mead C.A., Conway L. Introduction to VLSI Systems, available at: https://www.researchgate.net/publication/234388249_Introduction_to_VLSI_systems (дата обращения 12.06.2023).

95. Etienne Sicard, Microwind & Dsch User's Manual Version 2, available at: https://www.eletrica.ufpr.br/marlio/cidigital/projeto/manualAll.pdf (дата обращения 12.06.2023).

96. O. Drozd, O. Ivanova, K. Zashcholkin et al. ―Checkability Important for Fail-Safety of FPGA-based Components in Critical Systems [Возможности проверки важна для отказоустойчивости компонентов на базе ПЛИС в критически важных системах]‖, (in Russian), Intelligent Information Technologies & Systems of Information Security (IntelITSIS) 2021: International, 24-26 March 2021: proceedings.– Khmelnytskyi, Ukraine: IEEE, 2021. – P. 471-480

97. Tyurin, S.F. FPGA gates using combined variables coding / S.F. Tyurin, I.A. Vasenin, S.I. Sovetov // Perm National Research Polytechnic University Bulletin. Electrotechnics, information technologies, control systems, 2023, no. 46, pp. 83-107. DOI: 10.15593/2224-9397/2023.2.04

98. Васенин, И.А. Топологическое моделирование логических элементов LUT, использующих позиционный код набора переменных и LUT с унитарным кодированием. / И.А. Васенин, С.Ф. Тюрин // Автоматизированные системы управления и информационные технологии: материалы всерос. науч.-практ. конф. (г. Пермь, 2024 г.). С. 614-620

99. One-hot Programming LUT for FPGAs / I. Vasenin, S. F. Tyurin // Proceedings of the Seminar on Microelectronics, Dielectrics and Plasmas (theory and practical applications) [Electronic resource] : November 20, 2023 St. Petersburg Russia 2023 / IEEE Russia North-West section, St. Petersburg Electrotechn. Univ. "LETI". - [s. l.]: IEEE, 2023 - P. 132-135. - URI: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10424393 (дата обращения: 14.02.2024). - DOI 10.1109/MDP60436.2023.

100. Программируемое логическое устройство / С.Ф. Тюрин, Ю.А. Степченков, Ю.Г. Дьяченко, С.И. Советов, И.А. Васенин // Патент на изобретение RU 2826302 C1. – 09.09.2024. – Заявка от 27.10.2023.

101. Васенин, И.А. Моделирование элементов ПЛИС, использующих комбинированное кодирование / И.А. Васенин // Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления. – 2025. – № 53. – С. 157–176. DOI: 10.15593/2224-9397/2025.1.08

102. Васенин, И.А. Синтез элементов ПЛИС, использующих комбинированное кодирование / И.А. Васенин, О.В. Гончаровский, С.Ф. Тюрин // Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления. – 2025. – № 53. – С. 111–129. DOI: 10.15593/2224-9397/2025.1.06

103. Тюрин, С.Ф. Комбинированное кодирование в элементах ПЛИС / С.Ф. Тюрин, И.А. Васенин, Ю.А. Степченков, Ю.Г. Дьяченко.// Системы и средства информатики. 2025. Т. 35. № 1. С. 3-19.

ПРИЛОЖЕНИЕ А

Программа синтеза схем с использованием двух видов кодирования для получения комбинированных схем с помощью универсальных блоков

1. Описание программы

1.1. Общие сведения

Программа синтеза схем, где используется два вида кодирования для получения комбинированных схем с помощью универсальных блоков. Программа предназначена для создания схем на несколько переменных для вычисления логических функций или коммутации сигналов за счет реализации двух видов кодирования: унитарного и позиционного (бинарного).

1.2. Функциональное назначение

В зависимости от введенных данных, которые обозначают количество переменных (n1 и n2) отображаются универсальные элементы в одном из вариантов работы: унитарный или позиционный (бинарный), а также соединения этих блоков между собой.

1.3. Описание логической структуры

В программе реализованы следующие функции: ввод количества переменных (n1 и n2), обозначающих один из вариантов работы универсального блока, построчный вывод в консоль, вывод синтезируемых схем одного из трех видов кодирования: унитарный, позиционный (бинарный), комбинированный.

1.4. Используемые технические средства

Для того, чтобы запустить и исполнить программу необходим ЭВМ с интерпретатором языка C++. Программа может работать на любой ЭВМ под управлением операционных систем семейств Windows, macOS, GNU/Linux, BSD и др.

1.5. Вызов и загрузка

Вызов программы может осуществляться из консоли с помощью интерпретатора языка C++.

1.6. Входные данные

Количество переменных – n1 и n2, где n1 – число позиционных (бинарных) переменных и n2 – количество унитарных переменных,

1.7. Выходные данные

Выводится общее число переменных n (n=n1+n2), полученные схемы из универсальных элементов с реализацией одного из вида кодирования: унитарный, позиционный (бинарный) или комбинированный.

2. Листинг программы

```
#include <iostream>
#include <cmath>
using namespace std;
char** matrixU(int n2) {
    cout << "СХЕМА ТИПА 'U'" << endl;
    if (n2 <= 0) {
        throw invalid_argument("n2 не может быть равно 0.");
    }

    int rows = pow(2, n2) - 1; // rows = 2^n2 - 1
    int cols = n2; // cols = n2
    char** matrix = new char* [rows];
    for (int i = 0; i < rows; i++) {
        matrix[i] = new char[cols];
    }
    int pos1 = 1, pos2 = 3, pos3 = 7;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            matrix[i][j] = ' ';
        }
    }
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (matrix[i][j] == ' ') {
                if (i % 2 == 0 && j == 0) {
                    matrix[i][j] = 'u';
                }
                else if (j == 0 && i == pos1) {
                    matrix[i][j] = '|';
                    matrix[i][j + 1] = '-';
                    pos1 += 4;
                }
```

```
                else if (j == 1 && i == pos2) {
                    matrix[i][j] = '|';
                    matrix[i][j + 1] = '-';
                    pos2 += 8;
                }
                else if (j == 2 && i == pos3) {
                    matrix[i][j] = '|';
                    matrix[i][j + 1] = '-';
                    pos3 += 16;
                }
                else if (j == 3 && i == 15) {
                    matrix[i][j] = '|';
                    matrix[i][j + 1] = '-';
                }
            }
        }
    }
    return matrix;
}
char** matrixB(int n1) {
    cout << "СХЕМА ТИПА 'B'" << endl;
    if (n1 <= 0) {
        throw invalid_argument("n1 не может быть равно 0.");
    }
    int rows = static_cast<int>(pow(2, n1)) - 1; // rows = 2^n1 - 1
    int cols = 1;
    for (int i = 0; i < n1 - 1; i++) {
        cols += 2;
    }
    char** matrix = new char* [rows];
    for (int i = 0; i < rows; i++) {
        matrix[i] = new char[cols];
    }
    int pos1 = 1, pos2 = 3, pos3 = 7;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            matrix[i][j] = ' ';
        }
    }
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (matrix[i][j] == ' ') {
                if (i % 2 == 0 && j == 0) {
                    matrix[i][j] = 'b';
```

```
                }
                else if (j == 0 && i == pos1) {
                    matrix[i][j] = '|';
                    matrix[i][j + 1] = '-';
                    matrix[i][j + 2] = 'b';
                    pos1 += 4;
                }
                else if (j == 2 && i == pos2) {
                    matrix[i][j] = '|';
                    matrix[i][j + 1] = '-';
                    matrix[i][j + 2] = 'b';
                    pos2 += 8;
                }
                else if (j == 4 && i == pos3) {
                    matrix[i][j] = '|';
                    matrix[i][j + 1] = '-';
                    matrix[i][j + 2] = 'b';
                    pos3 += 16;
                }
                else if (j == 6 && i == 15) {
                    matrix[i][j] = '|';
                    matrix[i][j + 1] = '-';
                    matrix[i][j + 2] = 'b';
                }
            }
        }
    }
    return matrix;
}
char** mixed_matrix(int n1, int n2) {
    cout << "КОМБИНИРОВАННАЯ СХЕМА" << endl;
    // n1 - количество 'b', n2 - количество 'u'
    int rows, cols;
    if (n1 == 1 && n2 == 3) {
        rows = 15;
        cols = 5;
    }
    else if (n1 == 2 && n2 == 2) {
        rows = 15;
        cols = 6;
    }
    else if (n1 == 3 && n2 == 1) {
        rows = 15;
        cols = 7;
```

```
}
else if (n1 == 1 && n2 == 2) {
    rows = 7;
    cols = 4;
}
else if (n1 == 2 && n2 == 1) {
    rows = 7;
    cols = 5;
}
else {
    cout << "Такой схемы не существует" << endl;
}
cout << n1 << "b + " << n2 << "u" << endl;
char** matrix = new char* [rows];
for (int i = 0; i < rows; i++) {
    matrix[i] = new char[cols];
}
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        matrix[i][j] = ' ';
    }
}
for (int i = 0; i < rows; i++) {
    if (i % 2 == 0) {
        matrix[i][0] = 'u';
    }
    else if (i % 4 == 1) {
        matrix[i][0] = '|';
        matrix[i][1] = '-';
    }
    else {
        matrix[i][0] = ' ';
    }
}
if (n1 == 1 && n2 == 3) {
    matrix[3][1] = '|';
    matrix[3][2] = '-';
    matrix[11][1] = '|';
    matrix[11][2] = '-';
    matrix[7][2] = '|';
    matrix[7][3] = '-';
    matrix[7][4] = 'b';
}
if (n1 == 2 && n2 == 2) {
```

```
            matrix[3][1] = '|';
            matrix[3][2] = '-';
            matrix[3][3] = 'b';
            matrix[11][1] = '|';
            matrix[11][2] = '-';
            matrix[11][3] = 'b';
            matrix[7][3] = '|';
            matrix[7][4] = '-';
            matrix[7][5] = 'b';
        }
        if (n1 == 3 && n2 == 1) {
            matrix[1][2] = 'b';
            matrix[5][2] = 'b';
            matrix[9][2] = 'b';
            matrix[13][2] = 'b';
            matrix[3][2] = '|';
            matrix[3][3] = '-';
            matrix[3][4] = 'b';
            matrix[11][2] = '|';
            matrix[11][3] = '-';
            matrix[11][4] = 'b';
            matrix[7][4] = '|';
            matrix[7][5] = '-';
            matrix[7][6] = 'b';
        }
        if (n1 == 1 && n2 == 2) {
            matrix[3][1] = '|';
            matrix[3][2] = '-';
            matrix[3][3] = 'b';
        }
        if (n1 == 2 && n2 == 1) {
            matrix[1][2] = 'b';
            matrix[5][2] = 'b';
            matrix[3][2] = '|';
            matrix[3][3] = '-';
            matrix[3][4] = 'b';
        }

        return matrix;
    }
    int main() {
        setlocale(LC_ALL, "Russian");
        char** matrix = NULL;
        int rows, cols;
```

```
        int n1 = 0, n2 = 0;
        while (n1 == 0 && n2 == 0) {
            cout << "Input n1 and n2:" << endl << "n1 = ";
            cin >> n1; // Ввод n1 от пользователя
            cout << "n2 = "; cin >> n2;
            if (n2 != 0 && n1 == 0) {
                if (n2 > 5) {
                    cout << "n2 превышает допустимое значение! Повторите ввод."
<< endl;
                    n2 = 0;
                }
                else {
                    rows = pow(2, n2) - 1;
                    cols = n2;
                    matrix = matrixU(n2);
                }
            }
            else if (n1 != 0 && n2 == 0) {
                if (n1 > 5) {
                    cout << "n1 превышает допустимое значение! Повторите ввод."
<< endl;
                    n1 = 0;
                }
                else {
                    rows = static_cast<int>(pow(2, n1)) - 1;
                    cols = 1;
                    for (int i = 0; i < n1 - 1; i++) {
                        cols += 2;
                    }
                    matrix = matrixB(n1);
                }
            }
            else if (n1 == 0 && n2 == 0) {
                cout << "Оба числа равны нулю! Повторите ввод." << endl;
            }
            else if (n1 > 5 || n2 > 5) {
                cout << "Одно из чисел превышает допустимое значние!
Повторите ввод." << endl;
            }
            else {
                cout << "n = n1 + n2 = " << n1 + n2 << endl;
                if (n1 == 1 && n2 == 3) {
                    rows = 15;
                    cols = 5;
```

```
            }
            else if (n1 == 2 && n2 == 2) {
                rows = 15;
                cols = 6;
            }
            else if (n1 == 3 && n2 == 1) {
                rows = 15;
                cols = 7;
            }
            else if (n1 == 1 && n2 == 2) {
                rows = 7;
                cols = 4;
            }
            else if (n1 == 2 && n2 == 1) {
                rows = 7;
                cols = 5;
            }
            else {
                cout << "Такой схемы не существует! Повторите ввод." << endl;
                n1 = 0; n2 = 0;
            }
            if (n1 != 0 && n2 != 0) {
                matrix = mixed_matrix(n1, n2);
            }
        }
    }
    for (int i = 0; i < (rows); i++) {
        for (int j = 0; j < cols; j++) {
            cout << matrix[i][j];
        }
        cout << endl;
    }
    cout <<
    cin.ignore();
    cin.get();
    for (int i = 0; i < (static_cast<int>(pow(2, n1)) - 1); i++) {
        delete[] matrix[i];
    }
    delete[] matrix;
    return 0;
}
```

ПРИЛОЖЕНИЕ Б

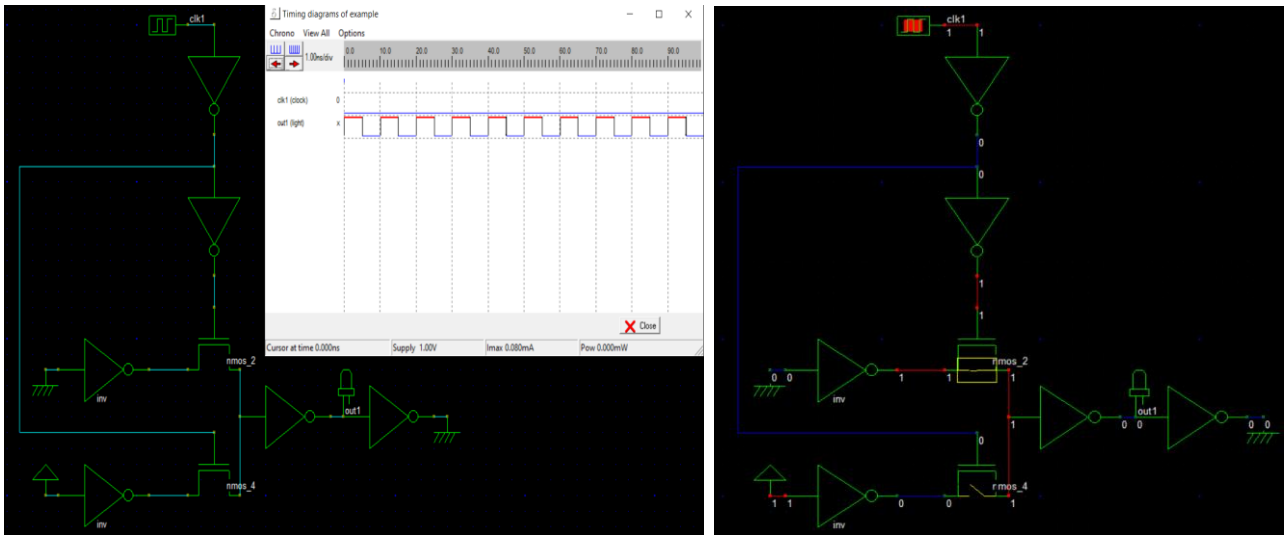
Моделирование в системе MicroWind

1. Моделирование схем электрических функциональных в системе DSCH

Моделирование схем электрических принципиальных осуществлялось в программе DSCH, которая является встроенным пакетом в программу для моделирования MicroWind.

В DSCH имеется функционал, с помощью которого можно создавать и моделировать схемы из базовых логических и электрических элементов и компонентов.

Программа DSCH представляет собой логический редактор и симулятор. DSCH используется для проверки архитектуры логической схемы перед началом проектирования микроэлектроники. DSCH предоставляет удобную для пользователя среду для проектирования иерархической логики и быстрого моделирования с анализом задержек, что позволяет проектировать и проверять сложные логические структуры. С помощью библиотеки символов создается схема.



а) б)

Рисунок Б.1 – а) Режим отладки, получение временной диаграммы, б) Процесс отладки и переключение сигналов на элементах схемы

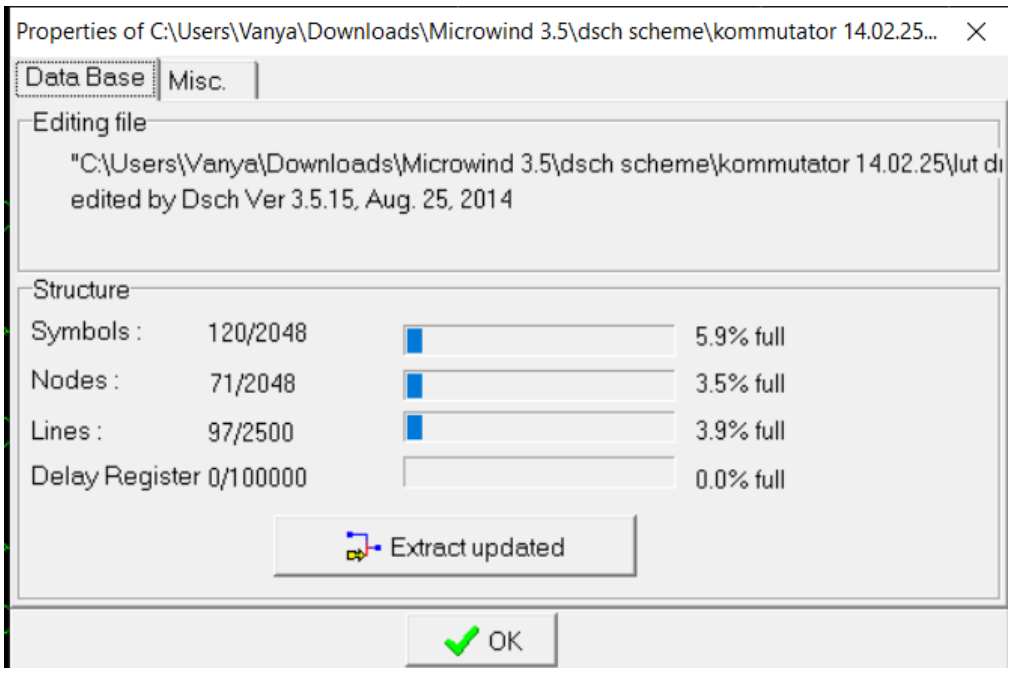
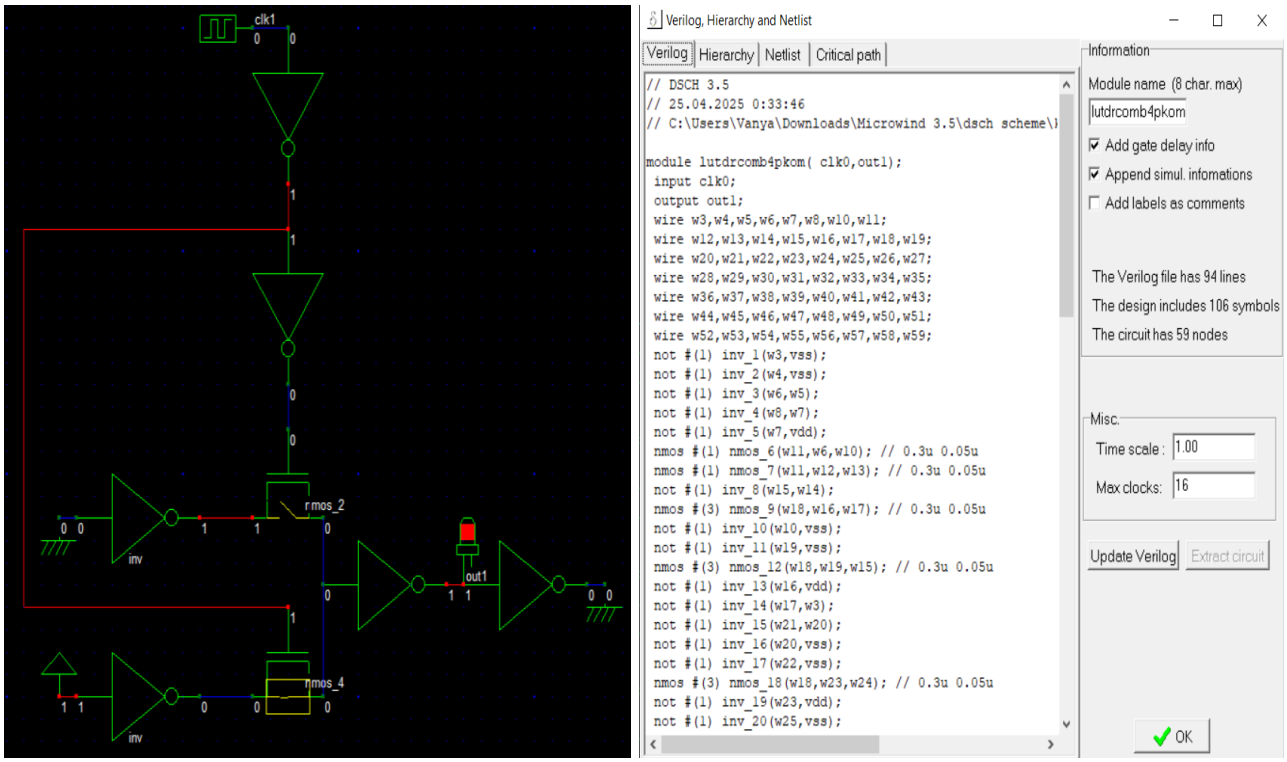


Рисунок Б.2. Окно с параметрами созданной схемы



а) б)

Рисунок Б.3 – а) Смоделированная схема, б) Процесс генерации Verilog файла по составленной схеме

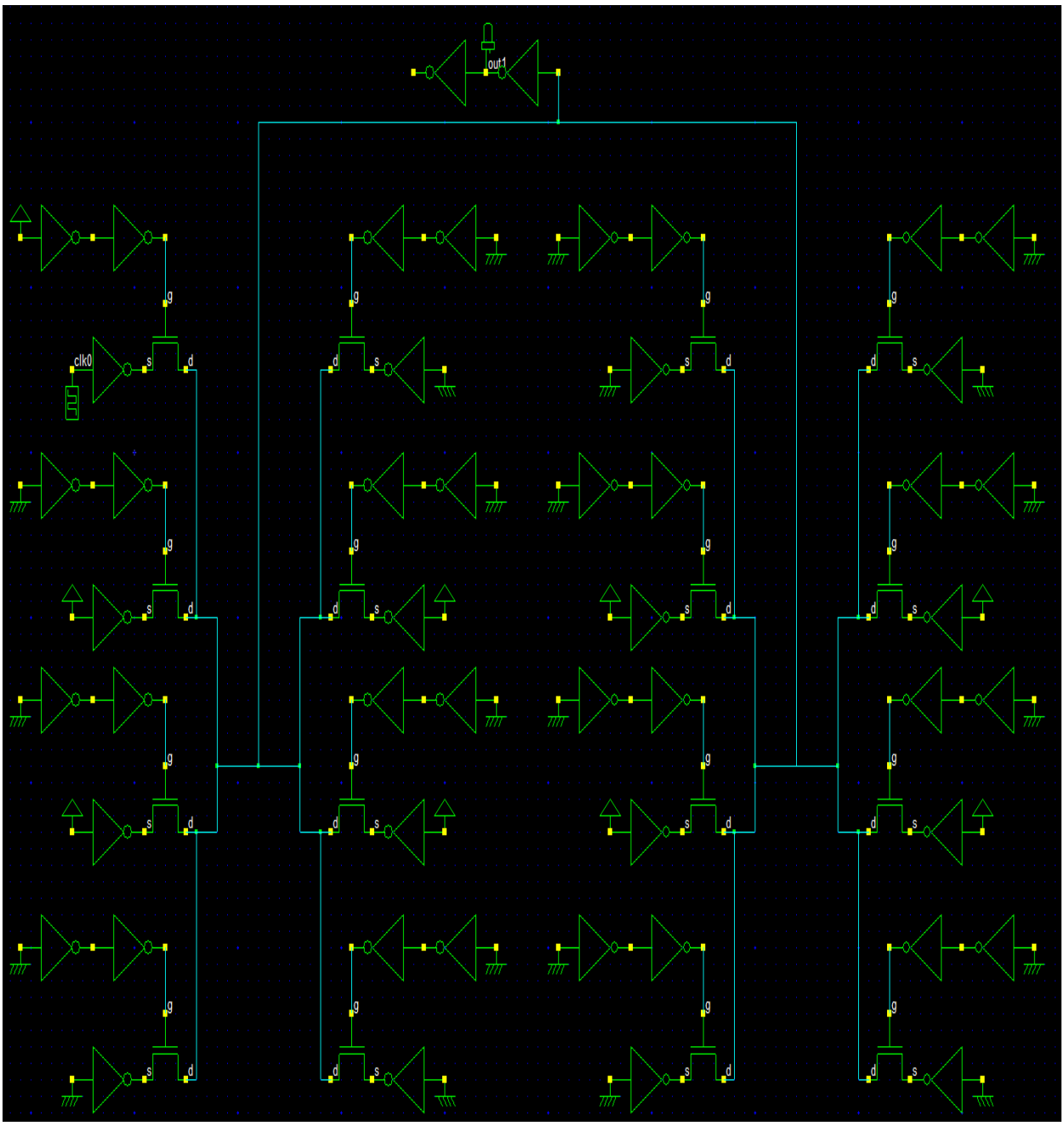


Рисунок Б.4 – Схема электрическая функциональная унитарного коммутатора на основе на 4 переменные

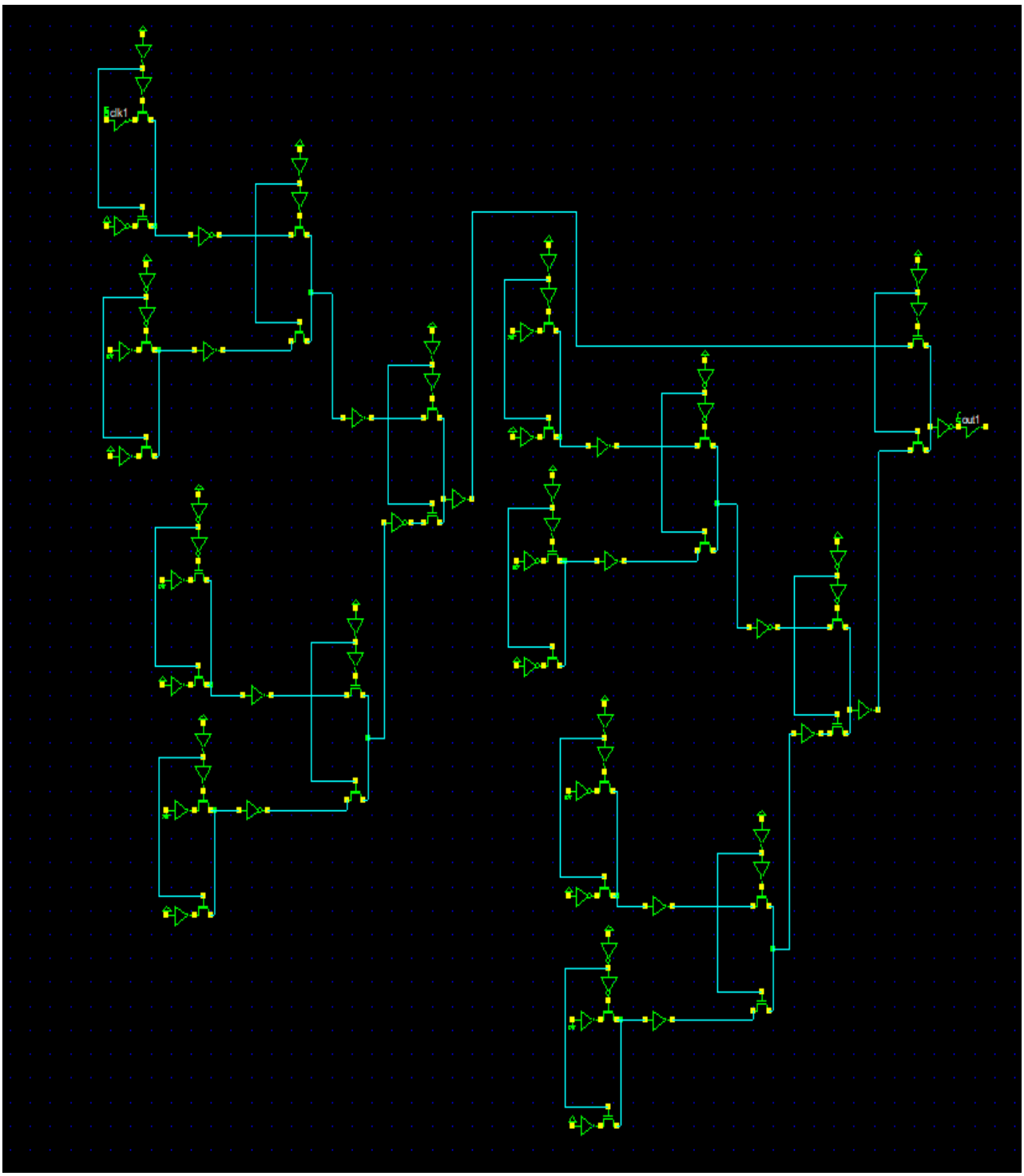


Рисунок Б.5 – Схема электрическая функциональная позиционного (бинарного) коммутатора на основе на 4 переменные

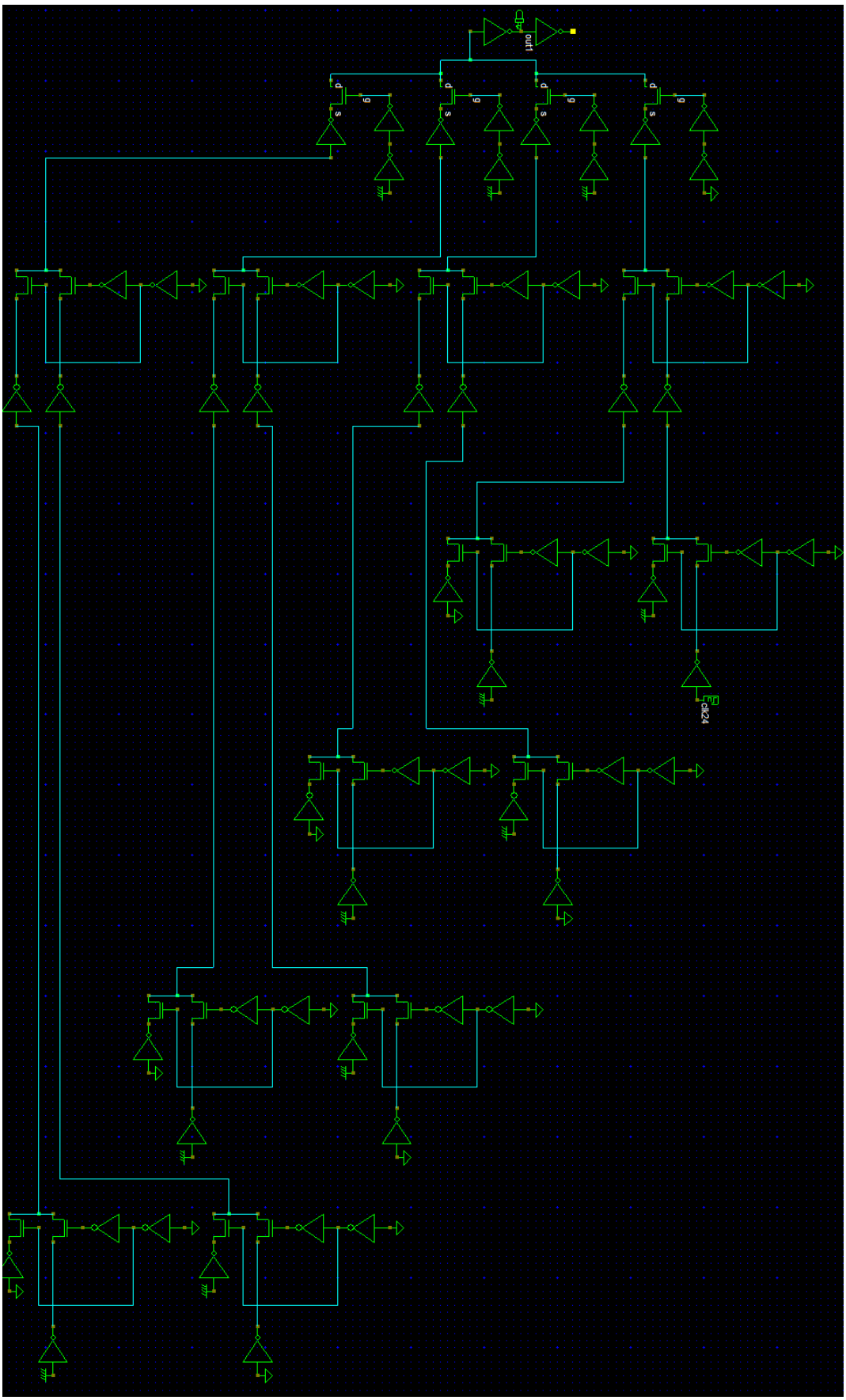


Рисунок Б.6 – Схема электрическая функциональная предлагаемого комбинированного 4-LUT №1

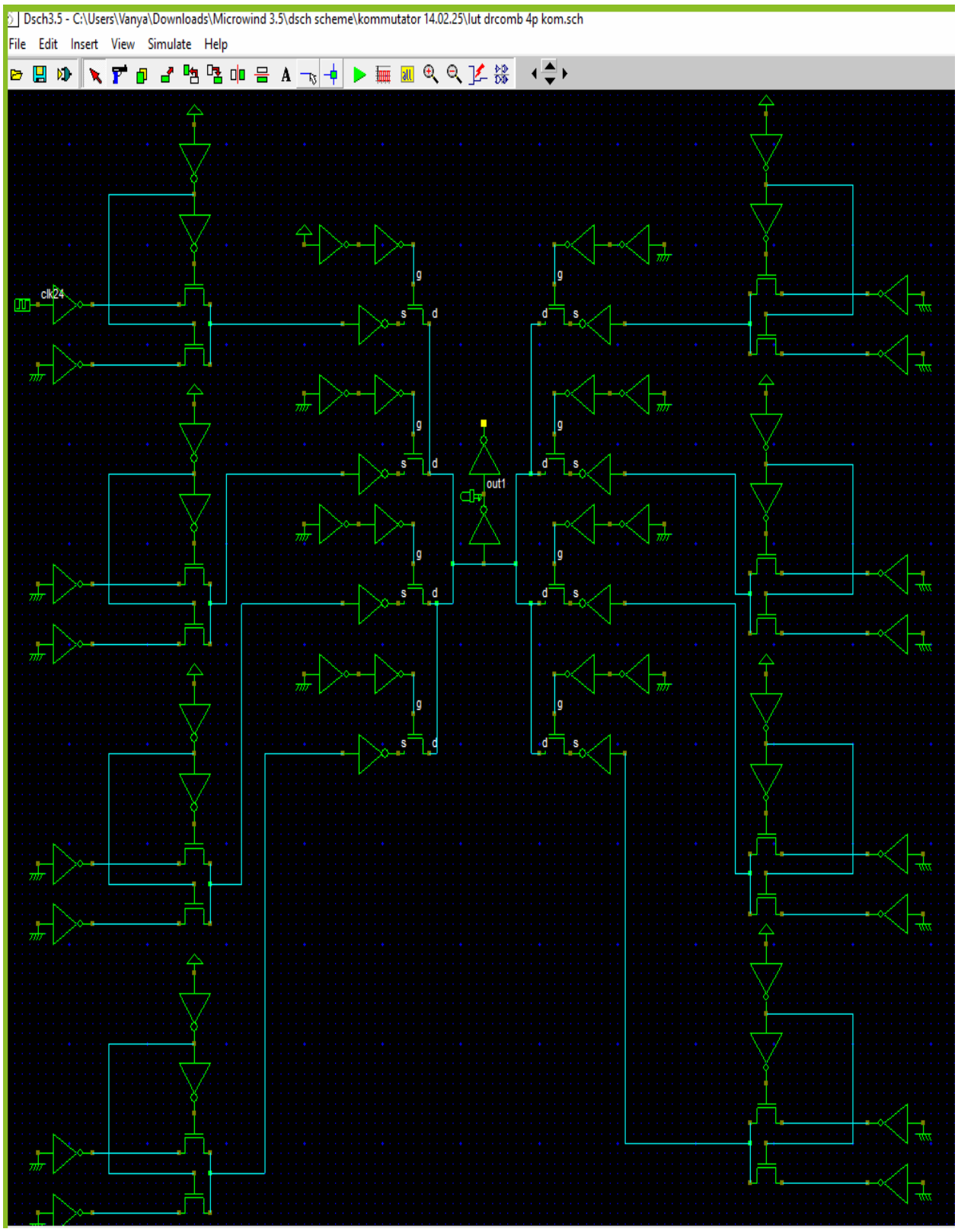


Рисунок Б.7 – Схема электрическая функциональная предлагаемого комбинированного 4-LUT №2

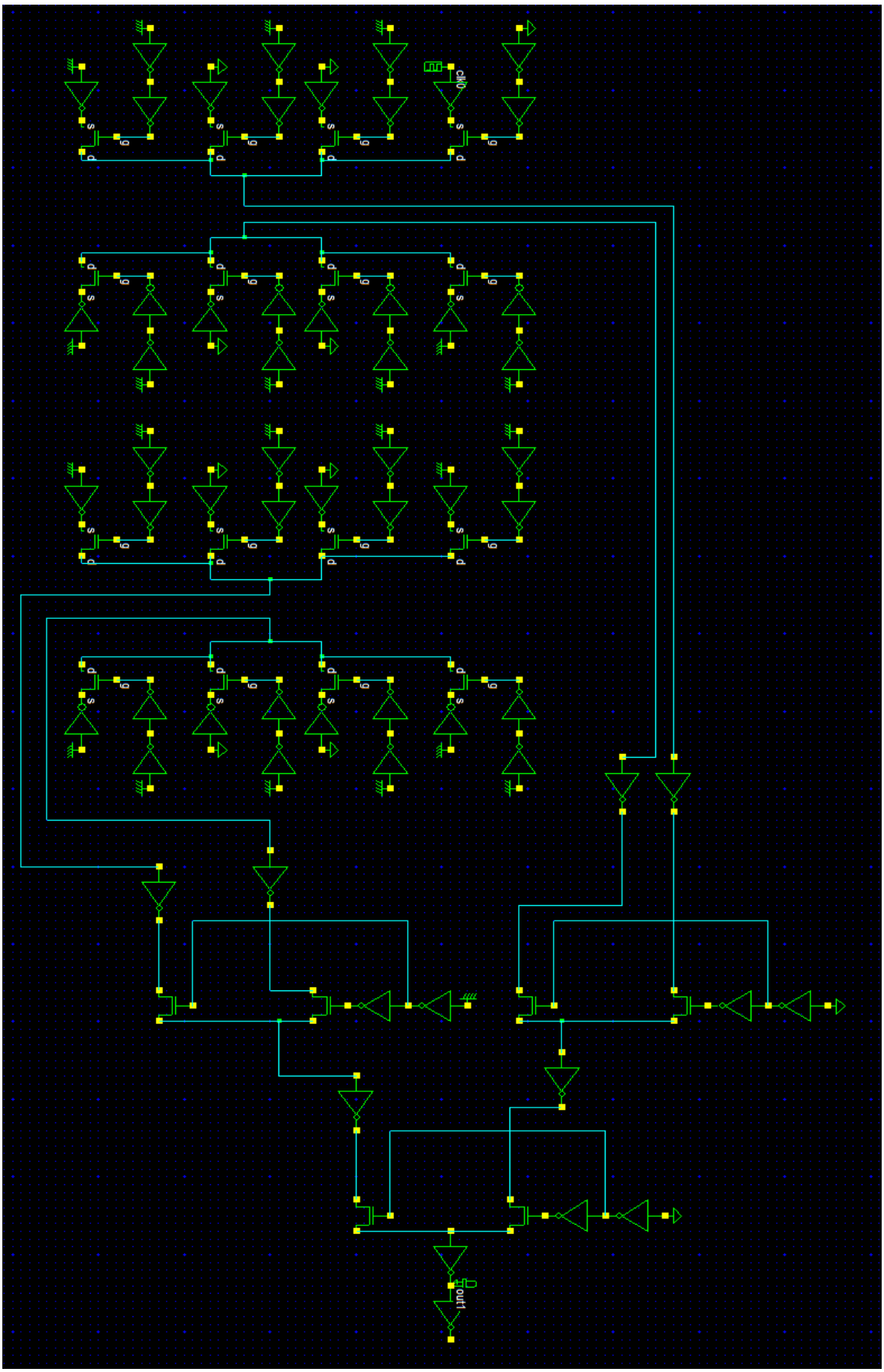


Рисунок Б.8 – Схема электрическая функциональная предлагаемого комбинированного 4-LUT №3

2. Настройки для моделирования схем в MicroWind

Microwind – интегрированное программное обеспечение, охватывающее проектирование микросхем от концепции до завершения и позволяющее разработчикам создавать микросхемы. Тесно объединяет реализацию смешанных сигналов с цифровой реализацией, моделированием схем, извлечением и проверкой на уровне транзисторов.

В данной программе присутствует большое число разделов с настройками, которые отвечают за работу программы. Перед началом работы необходимо выбрать технологические правила и библиотеку с теми правилами, при которых будет производиться топологическое моделирование схемы. В работе использовалась технология CMOS32nm.

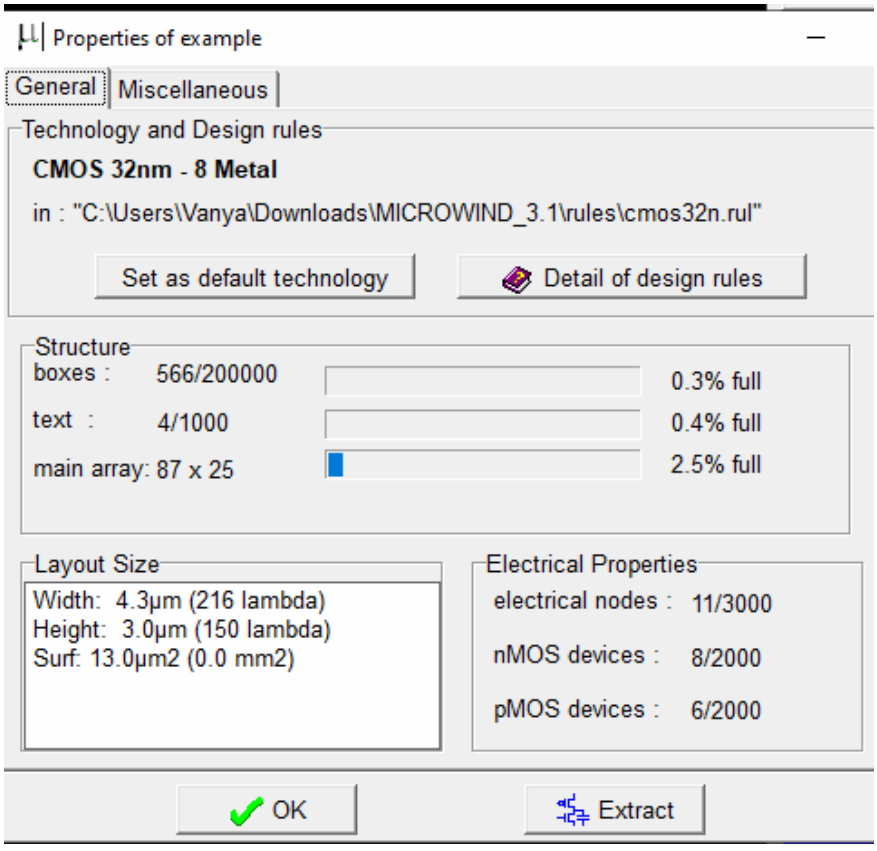


Рисунок Б.9 – Настройки технологии и правил моделирования

Размеры занимаемые схемой и электрические параметры (кол-во компонентов)

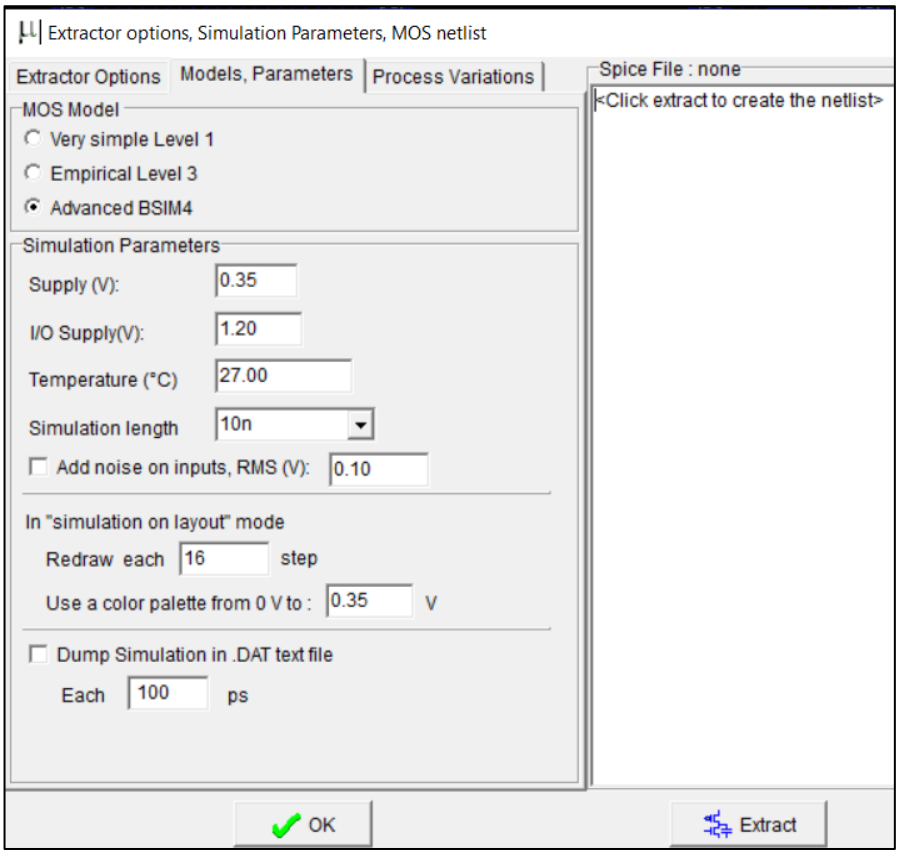


Рисунок Б.10– Окно с выбором параметров модели и параметров симуляции

Далее производиться настройка величины блока (источник), отвечающего за подачу такового сигнала/частоты для симуляции (Clock).

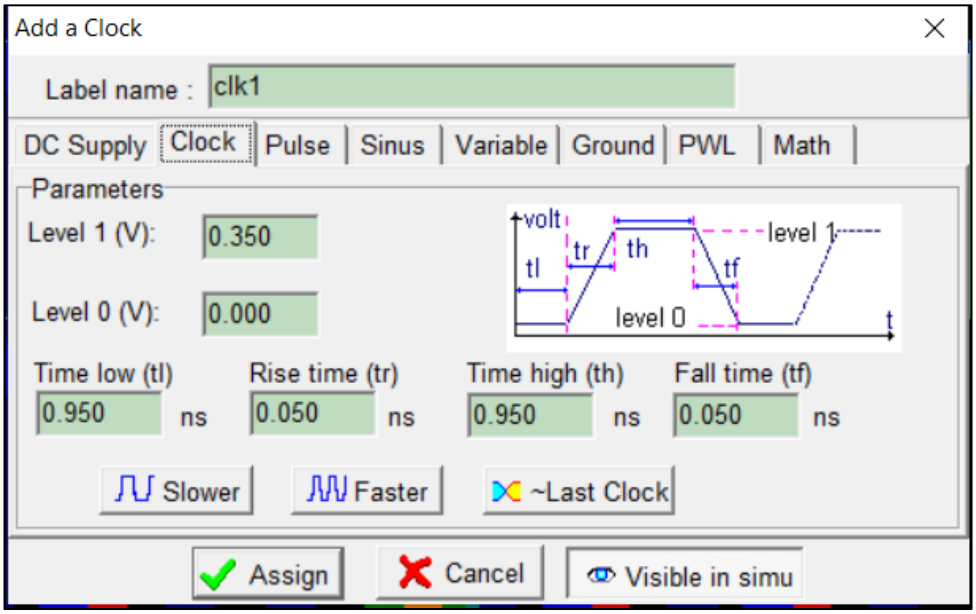


Рисунок Б.11 – Настройки моделирования

3. Получение топологий схем

Далее – этап создания топологий схем. С помощью встроенной библиотеки и палитры компонентов можно вручную собрать необходимые элементы. Или же воспользоваться функционалом автоматического моделирования с помощью компиляции файла Verilog и получить топологию элемента.

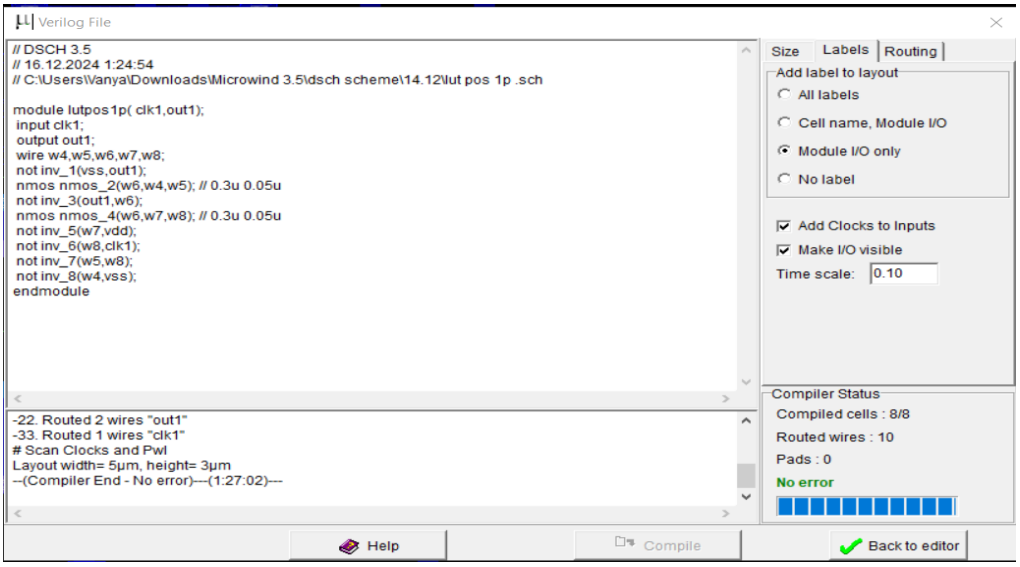


Рисунок Б.12 – Процесс компиляции файла Verilog

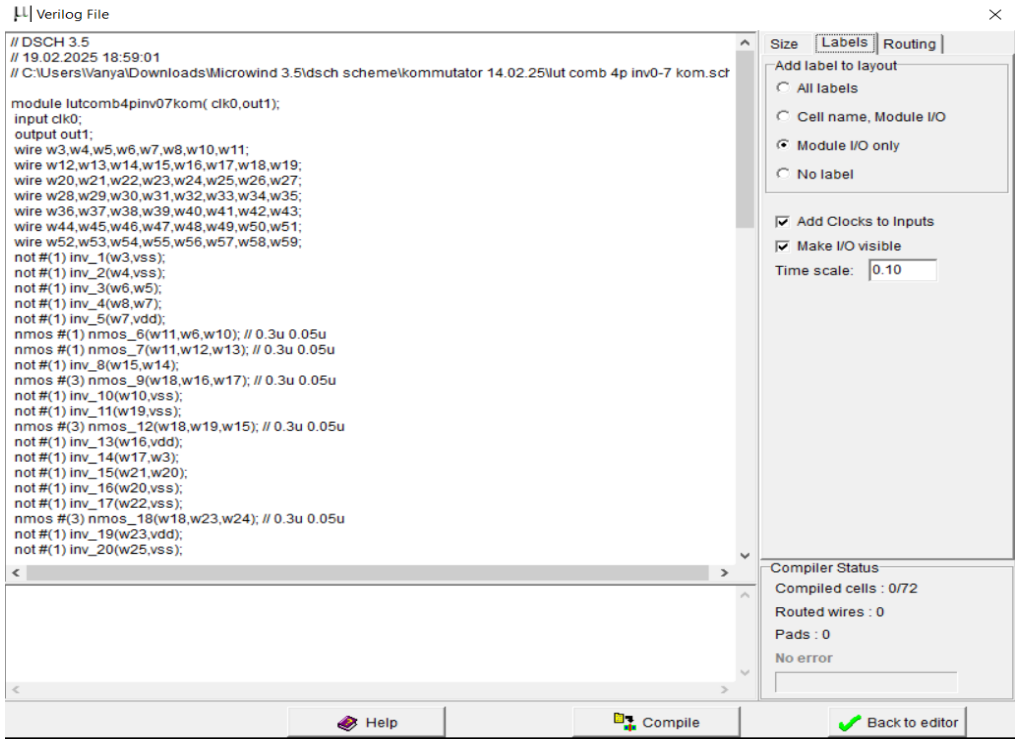


Рисунок Б.13 – Процесс компиляции файла Verilog

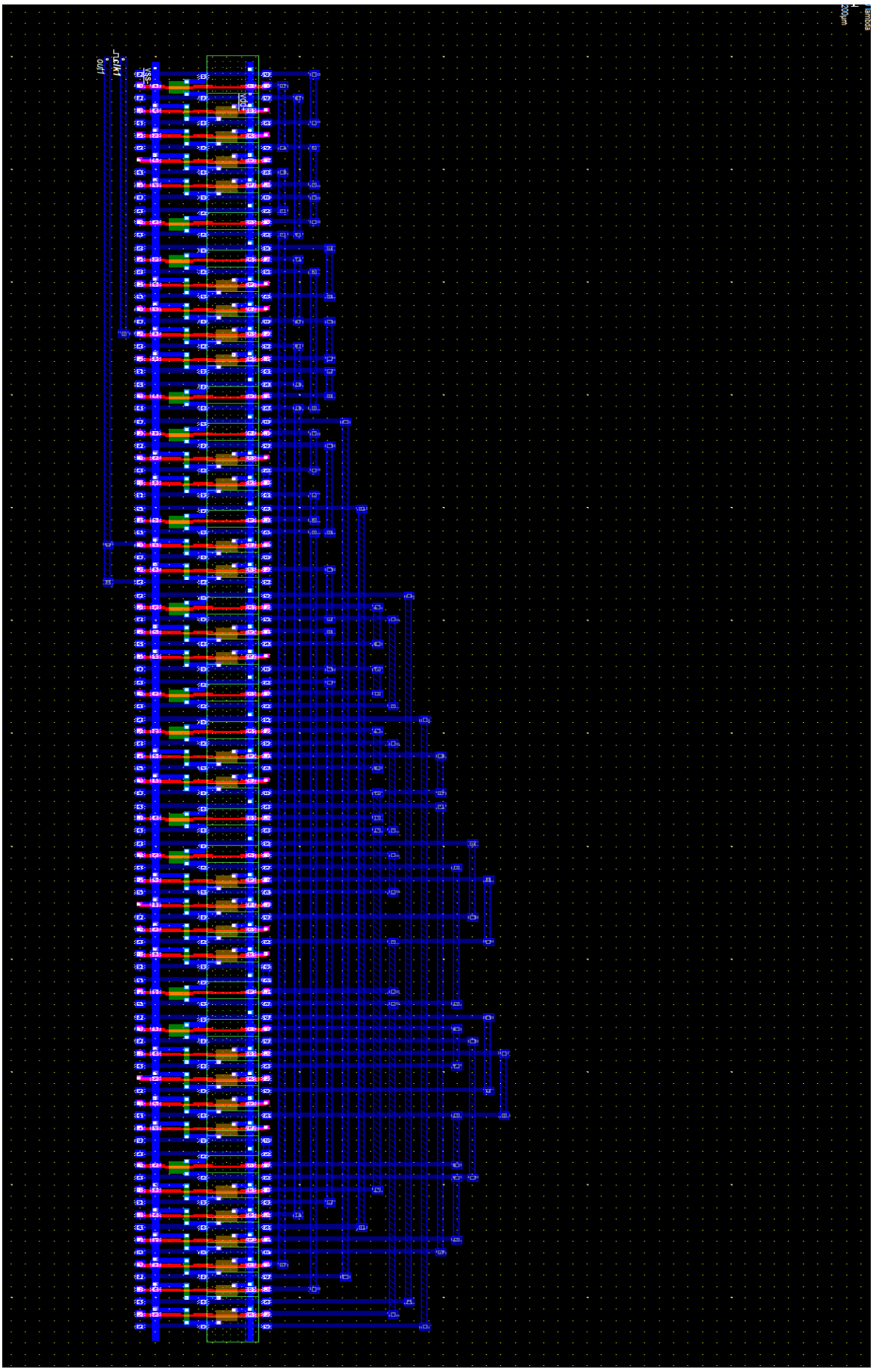


Рисунок Б.14 – Топология коммутатора 3-LUT комбинированный

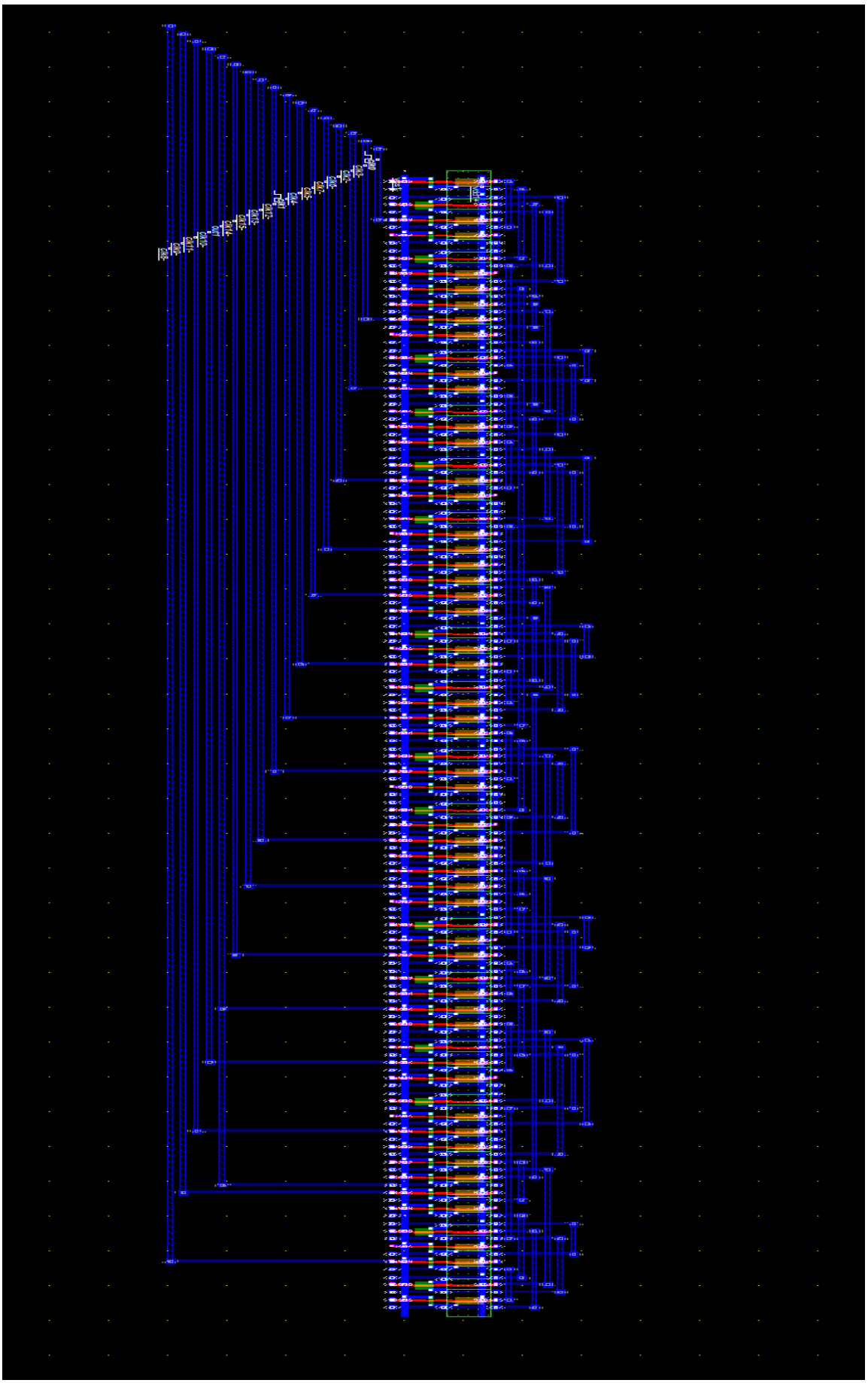


Рисунок Б.15 – Топология 4-LUT унитарного элемента

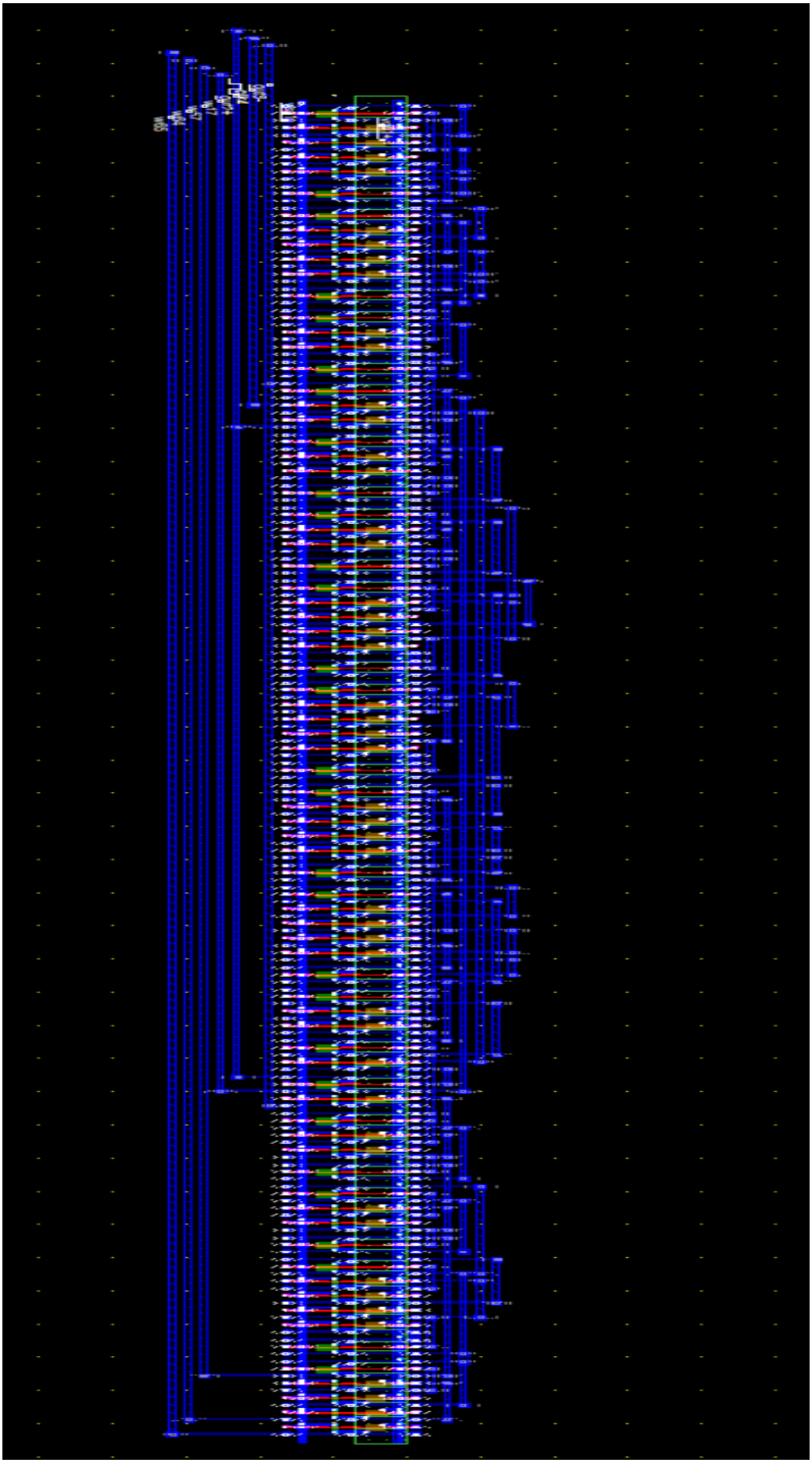


Рисунок Б.16 – Топология 4-LUT позиционного элемента

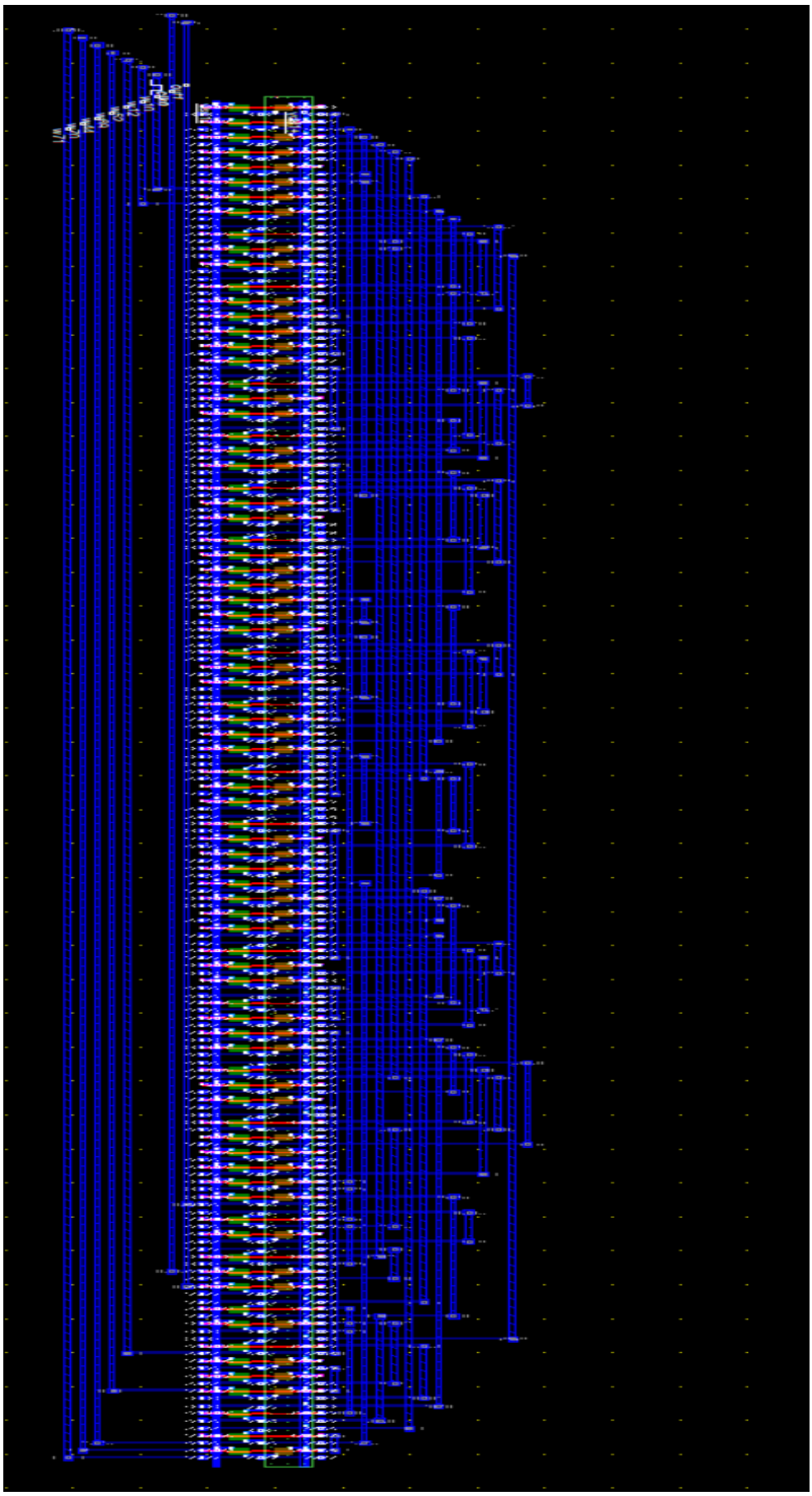


Рисунок Б.17 – Топология коммутатора на основе 4-LUT комбинированный №1

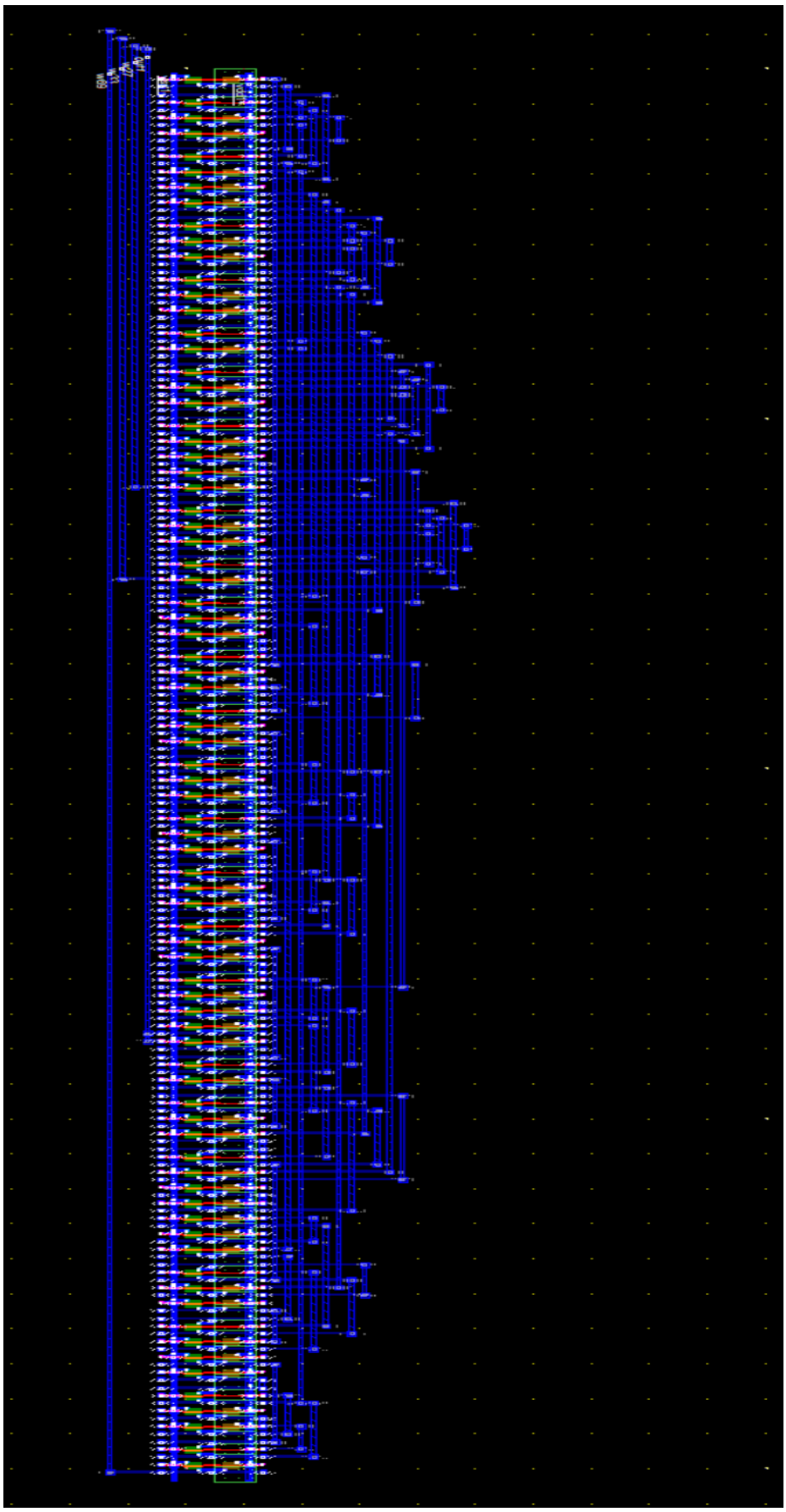


Рисунок Б.18 – Топология коммутатора на основе 4-LUT комбинированный №2

ПРИЛОЖЕНИЕ В

Акты о внедрении

1. Акт о внедрении результатов диссертационного исследования в учебный процесс кафедры «Автоматика и Телемеханика»

УТВЕРЖДАЮ

Проректор
по образовательной деятельности
Пермского национального
исследовательского
политехнического университета
доктор педагогических наук, доцент

/ И.Ю. Черникова /

«31» марта 2025 г.

АКТ
о внедрении результатов в учебный процесс кафедры «Автоматика и телемеханика» ФГАОУ ВО ПНИПУ, полученных Васениным Иваном Андреевичем при выполнении диссертационной работы на соискание ученой степени кандидата технических наук
«ЭЛЕМЕНТЫ ПЛИС С ИСПОЛЬЗОВАНИЕМ КОМБИНИРОВАННОГО КОДИРОВАНИЯ»

Комиссия в составе:
председатель комиссии: д.т.н., профессор Южаков А.А.,
члены комиссии:
д.т.н., доцент Фрейман В.И.,
к.т.н., доцент Гончаровский О.В.,
составила настоящий акт о том, что основные теоретические положения и практические результаты диссертационного исследования Васенина И.А. внедрены в учебный процесс кафедры «Автоматика и телемеханика» ФГАОУ ВО «Пермский национальный исследовательский политехнический университета» 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. Новая математическая модель элемента, отличающаяся тем, что описывает комбинированные варианты, использующие как позиционное, так

и унитарное кодирование в одном устройстве, а также универсальный элемент с настраиваемым типом кодирования;

2. Метод синтеза элементов с комбинированным и универсальным кодированием, отличающийся тем, что позволяет создавать новые устройства с лучшими характеристиками по быстродействию при допустимом увеличении сложности;

3. Алгоритм синтеза элемента с комбинированным кодированием, отличающийся тем, что обеспечивает по заданным параметрам требуемые соединения, используя предложенный элемент с конфигурируемым кодированием;

4. Оценки сложности новых элементов с комбинированным кодированием, позволяющие выбирать требуемый вариант комбинирования;

внедрены в практические и лабораторные занятия в виде методик расчетов новых предлагаемых элементов.

Эффект от внедрения результатов диссертационной работы заключается в повышении уровня знаний, умений и владений (освоения профессиональных компетенций и их компонентов в области методов синтеза элементов вычислительных систем в соответствии с требованиями Федеральных государственных образовательных стандартов высшего образования.

Результаты внедрения результатов диссертационной работы обсуждались на заседании кафедры «Автоматика и телемеханика» 24.03.2025, протокол № 11.

Председатель комиссии: _____ / Южаков А.А. /

Члены комиссии: _____ / Фрейман В.И. /

_____ / Гончаровский О.В. /

«31» марта 2025 г.

2. Акт о внедрении результатов диссертационного исследования в ООО «Динамика роста» (г. Пермь)

ООО «Динамика роста» ОГРН 1235900022478/ИНН 5906178786 Тел.: +7 (912) 987-11-14

динамика роста.

Акт о внедрении

результатов диссертационных исследований

Васенина Ивана Андреевича

Настоящим актом подтверждается, что ООО «Динамика роста» г. Пермь 25.07.24 г. приобрела лицензию у Пермского Национального Исследовательского Политехнического Университета (ПНИПУ) на право использования изобретения «Программируемое логическое устройство» (патент РФ № 2 811 404, приоритет от 02.08.2023 г., дата государственной регистрации 11.01. 2024 г. (авторы Тюрин С.Ф., Васенин И.А., Степченков Ю.А., Дьяченко Ю.Г., Советов С.И.), а в научно-исследовательской работе ООО «Динамика роста» использовались следующие научные результаты, полученные в кандидатской диссертации аспиранта кафедры «Автоматики и телемеханики» ПНИПУ Васенина Ивана Андреевича:

1. Математическая модель элемента с комбинированным и универсальным кодированием

2. Метод реализации элемента с комбинированным и универсальным кодированием

3. Оценки сложности и эффективности различных вариантов комбинированного кодирования

4. Алгоритм синтеза элемента с заданным вариантом кодирования.

Разработанная модель, метод, алгоритм и оценки сложности могут быть использованы при проектировании новых программируемых логических устройств. При этом временная задержка снижается от 15%, а также снижаются аппаратурные затраты от 20%.

УТВЕРЖДАЮ

Генеральный директор
Коногорова Л.В.
ООО «Динамика роста»
«17» февраля 2025 г.

Общество с ограниченной ответственностью Динамика роста Пермь ОГРН 1235900022478

Ознакомьтесь с успешными кейсами

3. Акт о внедрении результатов диссертационного исследования в ФИЦ ИУ РАН (г. Москва)

УТВЕРЖДАЮ
Директор ФИЦ ИУ РАН
_____ М.А. Посыпкин
«28» марта 2025 г.

Акт о внедрении

результатов диссертационных исследований

Васенина Ивана Андреевича

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 Васенина Ивана Андреевича: раздел 5.3 "Разработка логики самосинхронных ПЛИС отчета за 2024 год, № госрегистрации 124040200035-3:

1. Математическая модель элемента, отличающаяся тем, что описывает комбинированные варианты, использующие как позиционное, так и унитарное кодирование в одном устройстве, а также универсальный элемент с настраиваемым типом кодирования;

2. Метод синтеза элементов с комбинированным и универсальным кодированием, отличающийся тем, что позволяет создавать новые устройства с лучшими характеристиками по быстродействию при допустимом увеличении сложности;

3. Оценки сложности новых элементов с комбинированным кодированием, позволяющие выбирать требуемый вариант комбинирования;

4. Алгоритм синтеза элемента с комбинированным кодированием, отличающийся тем, что обеспечивает по заданным параметрам требуемые соединения, используя предложенный элемент с конфигурируемым кодированием.

Разработанные модель, метод, алгоритм и оценки сложности используются при проектировании программируемых логических устройств на основе отечественной элементной базы и позволяют компенсировать технологические ограничения по быстродействию, имеющиеся у отечественных разработчиков ПЛИС критического применения. При этом временная задержка снижается от 15%, а также в ряде случаев одновременно снижаются и аппаратурные затраты от 20%.

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

«28» марта 2025 г. _____ Ю.А. Степченков