Федеральное государственное бюджетное
образовательное учреждение высшего образования
ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

На правах рукописи

ВИХОРЕВ Руслан Владимирович

ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ ПЛИС FPGA ДЛЯ РЕАЛИЗАЦИИ СИСТЕМ ФУНКЦИЙ

05.13.05 – Элементы и устройства вычислительной техники и систем управления

ДИССЕРТАЦИЯ
на соискание ученой степени
кандидата технических наук

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

Пермь - 2019

Оглавление

Введение

Актуальность темы исследования. В настоящее время для реализации цифровых элементов и устройств вычислительной техники и систем управления широко используются программируемые логические интегральные схемы (ПЛИС). Основой ПЛИС типа Field-Programmable Gate Array (FPGA) является логический элемент (ЛЭ) – генератор функций Look Up Table (LUT), число входных переменных которого увеличилось за 30 лет с 4-х до 8, а количество ЛЭ в ПЛИС – до десятка миллионов. Разработаны адаптивные логические модули (АЛМ), в которых разрядность ЛЭ может гибко изменяться под конкретный проект. Даже при наличии таких возможностей ПЛИС в ряде областей применения, количества ЛЭ может оказаться недостаточно, например, в системах управления летательных аппаратов в связи с возрастанием возложенных на них задач и наличия очень жестких масса-габаритных ограничений. Таким образом задача улучшения технических характеристик ЛЭ ПЛИС является актуальной. Одним из сдерживающих факторов является то, что существующие ЛЭ ПЛИС типа FPGA не ориентированы на реализацию систем функций, зависящих от переменных одного ЛЭ. Для данного конфигурационного файла логический элемент FPGA реализует только одну логическую функцию в совершенной дизъюнктивной нормальной форме (СДНФ). Реализация систем из m логических функций выполняется на m ЛЭ, то есть каждая логическая функция реализуется на отдельном LUT, что приводит к значительной избыточности при вычислении систем логических функций, зависящих от одних и тех же переменных, особенно, когда количество переменных достаточно велико. В то же время реализация систем функций в дизъюнктивной нормальной форме (ДНФ) используется в ПЛИС типа Complex Programmable Logic Device (CPLD). Оба подхода имеют свои плюсы и минусы, однако особенности комплексирования подходов FPGA и CPLD до конца не выявлены.

Таким образом, актуальным является проведение исследований в области создания методов и средств реализации систем логических функций в ЛЭ ПЛИС FPGA, особенно в связи с известными проблемами импортозамещения в отечественной электронной компонентной базе.

Степень разработанности темы исследования. Особенности логики ПЛИС рассмотрены в работах З. Ерванта, Н. Мехта, А.В. Строгонова, С.А. Цыбина, А.Н. Денисова, В.И. Хаханова, В.С. Харченко, А.В. Дрозда, Г.П. Аксеновой, Д.Е. Иванова, Ю.А. Скобцова, M. Abusultan, S. P Khatri, M Turi, J. Delgado-Frias, N. Jha. Новые логические элементы ПЛИС разрабатывались и исследовались в работах С.Ф. Тюрина, А.В. Грекова, О.А. Громова, А.Ю. Городилова, А.Н. Каменских. Однако эти работы не затрагивают реализацию систем логических функций в LUT FPGA. Созданием гибридной FPGA (Hybrid FPGA), которая по сути является совокупностью ЛЭ LUT и встроенных программируемых логических матриц (PLA, PAL), занимались исследователи Alireza Kaviani, Stephen Brown, Chi Wai Yu.

Однако в полной мере вопросы реализации систем логических функций при комплексировании подходов FPGA и CPLD до сих пор не решены.

Объектом исследования являются логические элементы ПЛИС.

Предметом исследования являются методы реализации систем логических функций в ПЛИС.

Целью диссертационного исследования является решение научной задачи выбора оптимального набора логических элементов для адаптивно-гибридной реализации систем логических функций в ПЛИС FPGA.

Основные задачи исследования:

1. Усовершенствовать метод реализации логических функций в СДНФ на основе соответствующего адаптивного логического элемента.
2. Усовершенствовать метод реализации логических функций в ДНФ.
3. Произвести оценку сложности реализации систем логических функций на основе разработанных логических элементов.

4. Разработать и исследовать модели предложенных логических элементов FPGA для реализации систем логических функций в типовых проектах на ПЛИС.

5. Разработать алгоритм оптимизации набора логических элементов ПЛИС FPGA для реализации систем логических функций в типовых проектах на ПЛИС.

Научная новизна диссертационной работы заключается в следующем:

— Предложен усовершенствованный метод реализации логических функций в СДНФ на основе соответствующего адаптивного логического элемента, отличающийся тем, что в зависимости от настройки возможна реализация либо стандартного логического элемента LUT, либо дешифратора для реализации систем логических функций.

— Предложен усовершенствованный метод реализации систем логических функций в ДНФ, отличающийся тем, что использовано оригинальное кодирование конъюнкций ДНФ, обеспечивающее формирование значения конъюнкции без использования подтягивающего резистора.

— Получены оценки сложности реализации систем логических функций на основе разработанных логических элементов, отличающиеся декомпозицией, учитывающей ограничения Мида-Конвей на число последовательно соединенных передающих транзисторов.

— Выполнена оптимизация набора логических элементов ПЛИС FPGA на основе комбинирования ЛЭ, вычисляющих функции в СДНФ, и ЛЭ, вычисляющих функции в ДНФ, отличающаяся тем, что для этих целей модифицирован венгерский метод.

Теоретическая значимость диссертационной работы состоит в том, что развиты методы синтеза логических элементов, реализующих системы функций в ПЛИС. Разработаны и получены патентоспособные технические решения для реализации систем логических функций в ПЛИС типа FPGA,

полученные оценки сложности показывают предпочтительность их по аппаратным затратам по сравнению с существующими АЛМ.

Практическая значимость диссертационной работы состоит в том, что предложенные усовершенствованные методы позволили снизить аппаратные затраты на реализацию систем логических функций (акт Федерального исследовательского центра "Информатика и управление" Российской академии наук, акт Публичного акционерного общества «Пермская научно-производственная приборостроительная компания»).

Полученные научные и практические результаты используются в учебном процессе кафедры «Автоматика и телемеханика» Пермского национального исследовательского политехнического университета в рамках практических занятий профильных дисциплин «Электроника», «Проектирование дискретных устройств», «Схемотехника» у студентов направления 27.03.04 «Управление в технических системах» (акт Федерального государственного бюджетного образовательного учреждения высшего образования «Пермский национальный исследовательский политехнический университет»).

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

Область исследования, обозначенная в сформулированных задачах, соответствует п. 3 «Разработка принципиально новых методов анализа и синтеза элементов и устройств вычислительной техники и систем управления с целью улучшения их технических характеристик» паспорта научной специальности 05.13.05.

Основные положения, выносимые на защиту:

1. Усовершенствованный метод реализации логических функций в СДНФ на основе разработанного адаптивного логического элемента.

2. Усовершенствованный метод реализации логических функций в ДНФ.

3. Алгоритм оптимизации набора логических элементов ПЛИС FPGA для реализации типовых систем логических функций.

Степень достоверности результатов. Полученные в диссертационной работе результаты не противоречат теоретическим положениям, известным из научных публикаций отечественных и зарубежных исследователей, и подтверждаются результатами, полученными в двух системах моделирования, апробацией и внедрением предложенных в диссертации методов, моделей и алгоритма оптимизации.

Апробация результатов. Основные теоретические и практические результаты диссертационной работы докладывались на научно-технических конференциях: «Микроэлектроника и информатика», Зеленоград, 2015, 2017; ЛЭТИ - IEEE North West Russia Section Young Researchers in Electrical and Electronic Engineering Conference, EIConRusNW, Санкт-Петербург, 2016, 2018; 2-я Российско-белорусская научно-техническая конференция «Элементная база отечественной радиоэлектроники: импортозамещение и применение», Нижний Новгород, 2015; 14-я международная конференция «Авиация и космонавтика», Москва, 2015; на международном форуме «Микроэлектроника-2017» в рамках 3-й международной научной конференции «Электронная компонентная база и электронные модули», Республика Крым, г. Алушта, 2017; 14-я всероссийская школа-конференция молодых ученых «Управление большими системами УБС-2017», г. Пермь, 2017; на международной выставке SEMIEXPO RUSSIA 2017; на всероссийской научно-технической конференции «Автоматизированные системы управления и информационные технологии» г. Пермь, 2019; на семинарах в ИПИ РАН, ИПУ РАН, МИЭТ, ПНИПУ, других региональных и

всероссийских конференциях; на международном конкурсе научно-технических работ «Инновационная радиоэлектроника», 2017 г.

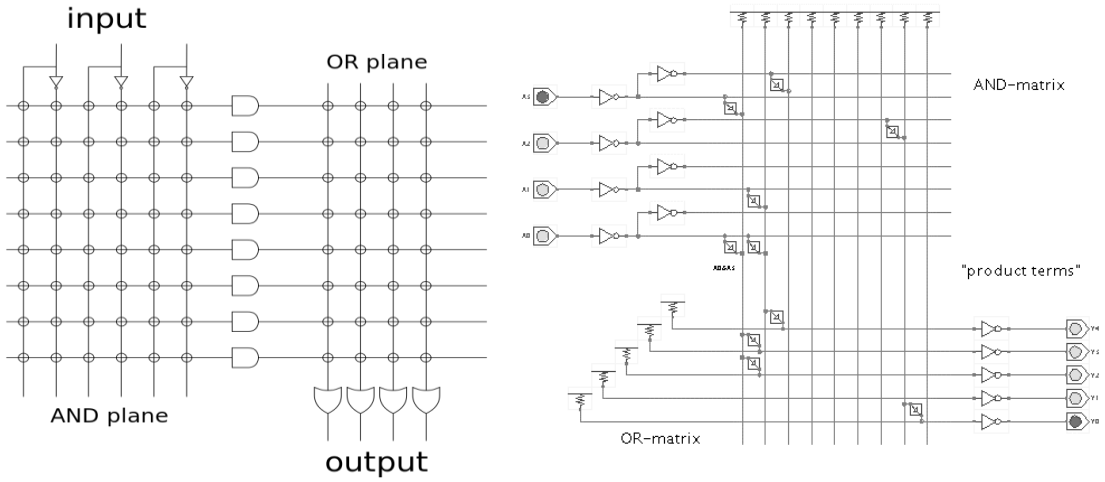
Публикации. Основные результаты диссертационной работы опубликованы в 21 печатной работе, из них 5 публикаций в ведущих рецензируемых научных изданиях, 2 публикации в изданиях, индексированных в международных базах цитирования Scopus и Web of Science, 3 патента на изобретение, 1 свидетельство о государственной регистрации программ для ЭВМ.

Структура и объем работы. Диссертационная работа состоит из введения, четырех глав, заключения, списка использованных источников и приложений. Содержит 180 страниц машинописного текста, из которых основной текст составляет 151 страницы, 152 рисунков и 6 таблиц, список литературы из 82 наименований, 3 приложений.

Глава 1. Исследование методов и средств реализации систем логических функций в существующих ПЛИС. Постановка задачи исследования

1.1. Анализ существующих БИС программируемой логики

В начале 70-х годов XX века компания Texas Instruments (TI) выпустила первую ПЛМ - программируемую (с помощью маски – mask - programmable) логическую матрицу, содержащую матрицу И – для программирования конъюнкций и матрицу ИЛИ – для программирования дизъюнкций. С ПЛМ (PLA, programmable logic array) и началась эпоха ПЛИС. Структура ПЛМ [1,2] изображена на рисунке 1.1.

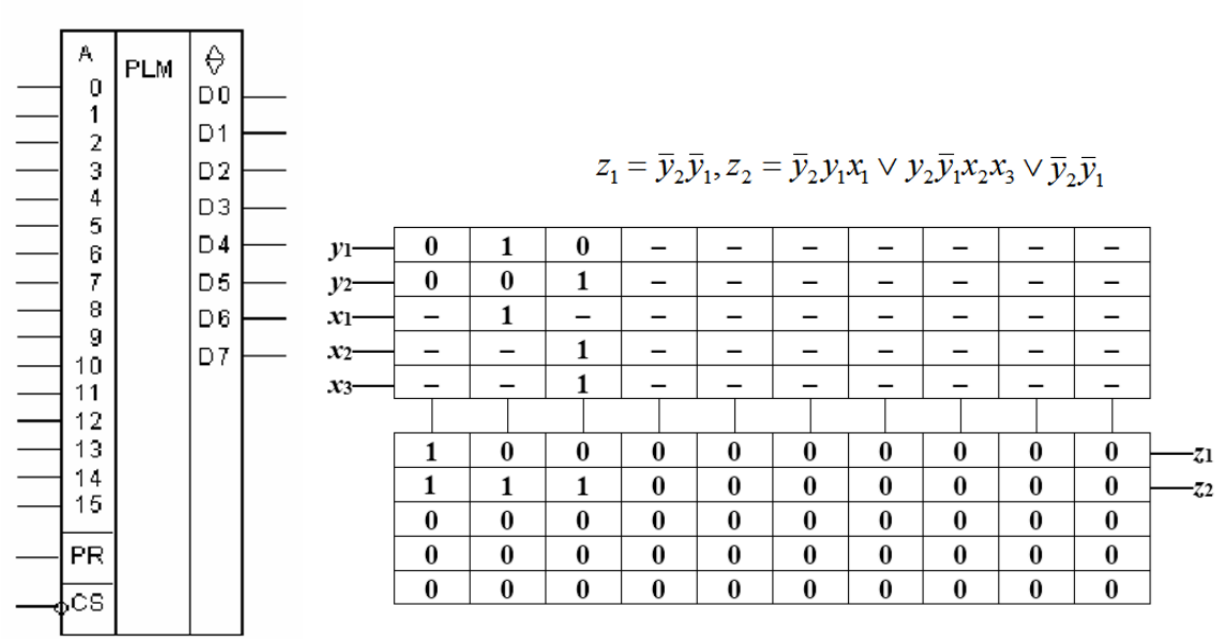


а) б)

Рисунок 1.1. Структура ПЛМ, а) упрощённая; б) с детализацией коммутаций

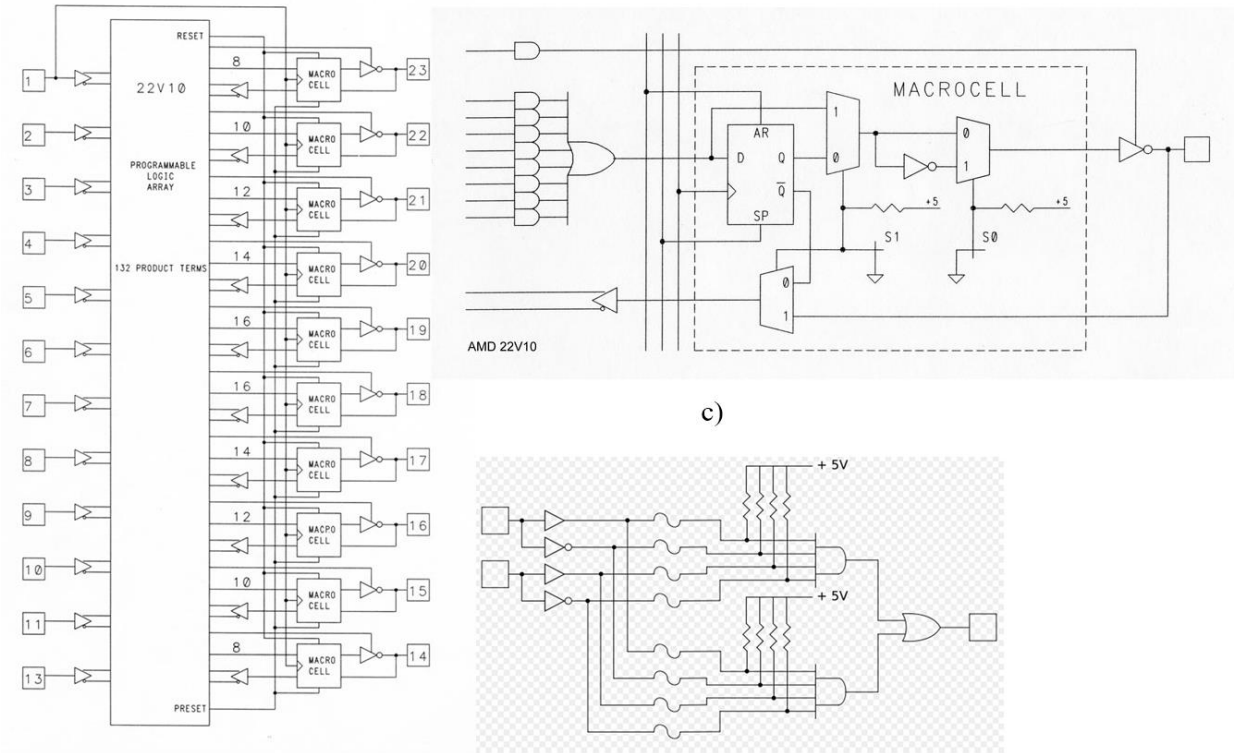
Как видно реализуется монтажная (проводная) логика – функции в дизъюнктивной нормальной форме – ДНФ. Программируются как матрица И, так и матрица ИЛИ. На рисунке 1.2 изображено условное графическое обозначение ПЛМ и пример программирования.

Структура другого типа программируемой логики – ПМЛ - программируемой матрицы логики (Programmable Array Logic - PAL) [3], где программируются только конъюнкции (Product terms) – матрица И, изображена на рисунке 1.3.



а) б)

Рисунок 1.2. ПЛМ: а) Условное графическое обозначение ПЛМ; б) пример программирования



с)

а) б)

Рисунок 1.3. Структура ПМЛ, а) структура; б) упрощённая схемы реализации ДНФ; с) макроячейка

ПЛМ и ПМЛ реализуют системы логических функций в ДНФ [4,5,6]. Другой вариант реализации логики – использование оперативной (ОЗУ) или постоянной памяти (ПЗУ) для записи таблицы истинности функции. В этом случае реализуется совершенная дизъюнктивная нормальная форма – СДНФ [6-8]. Так, например, для реализации мажоритарной функции (функции переноса) на ИМС средней степени интеграции-мультиплексоре[7,8], входы данных, соответствующие наборам 3,5,6,7 логической функции, зависящей от переменных a,b,c подключаются к константе 0 (так как выход мультиплексора инверсный), а входы 0,1,2,4 – к константе 1 – рисунок 1.4.

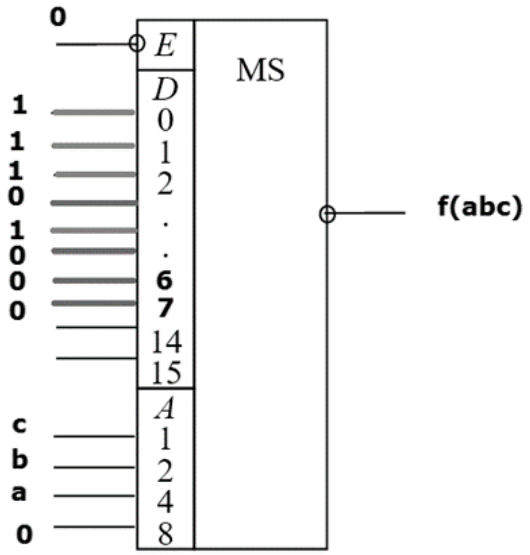


Рисунок 1.4. Структура однобитного мультиплексора логической функции не более, чем 4-х переменных

В принципе рисунок 1.4 - это однобитное ПЗУ. Для реализации большего числа функций можно использовать несколько схем показанных на рисунке 1.4 и получим многобитное ПЗУ. В этом случае на каждую функцию необходим один мультиплексор. Для реализации многобитной памяти используют дешифратор (DC) и элементы 2-И-НЕ [7,8]. – рисунок 1.5.

В случае программирования элементов 2-И-НЕ получаем ОЗУ. Это так называемая «гибкая логика», так как пользователь имеет возможность сам определять необходимые ему функции.

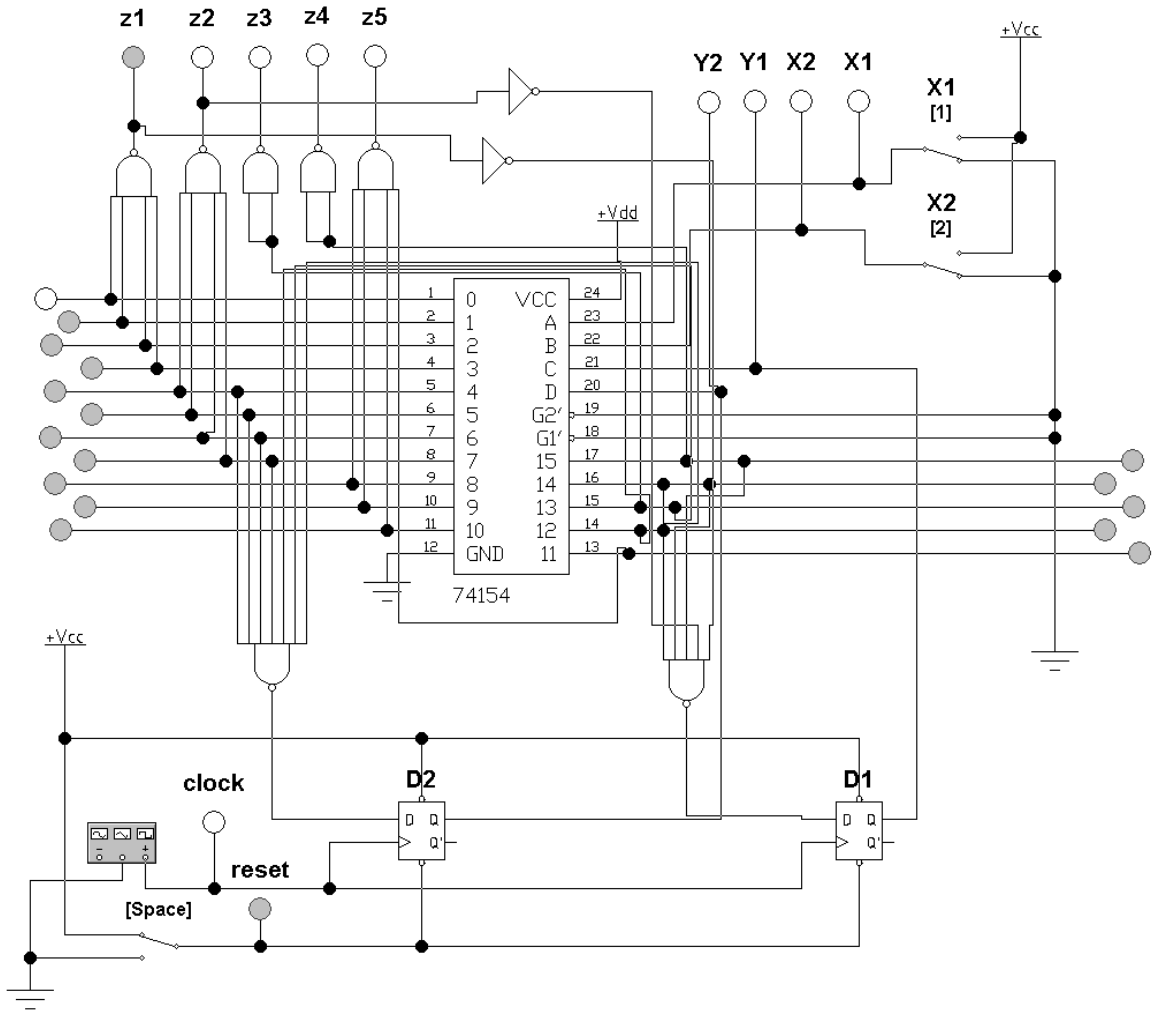


Рисунок 1.5 Реализация семиразрядного четырехадресного ПЗУ на дешифраторе 74154

С 70-х годов XX развивалось и другое направление - базовые матричные кристаллы (вентильные матрицы- gate array, uncommitted logic array - ULA), у которых в заводских условиях программировались связи логических элементов, имеющиеся на заготовке [6]. Это были так называемые полузаказные БИС средней стоимости, в отличие от самых дорогих заказных (application-specific integrated circuits -ASICs), в которых вся архитектура создаётся «с нуля». Такие БМК, выпускает ТЦ МИЭТ (г.Зеленоград), для которых используется САПР «Ковчег» [9], библиотеки элементов, например, приведены в [10]. В БМК используют даже не сами логические элементы, а четырехтранзисторные ячейки (два транзистора p-проводимости, два n-проводимости) – рисунок 1.6.

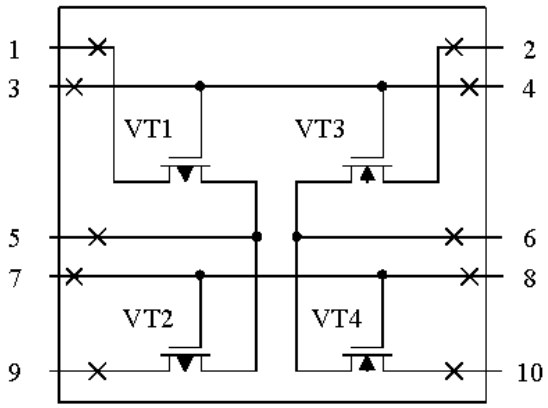


Рисунок 1.6. Четырех транзисторная ячейка БМК

Из таких ячеек создаются типовые (библиотечные) элементы, а уже их можно использовать для синтеза цифровой аппаратуры. Примеры разработки новых элементов БМК представлены в диссертации Каменских А.Н. [11].

В конце 80-х годов XX века появились технологические возможности пользователю самостоятельно программировать связи элементов. Все эти принципы легли в основу более сложных ИМС программируемой логики - ПЛИС FPGA (ППВМ–программируемые пользователем вентильные матрицы) [6,12-16] и CPLD – (Complex Programmable Logic Device) [6]. Фактически логика CPLD изображена на рисунке 1.3.

Программируемая логика развивалась и во многом обеспечивала прогресс микропроцессорной техники, разработку однокристальных микроЭВМ, которые сейчас называются микроконтроллерами, обусловила и прогресс сигнальных процессоров. В XXI веке интегральная технология позволила разрабатывать целые программируемые системы на кристалле[6]. Классификация основных программируемых цифровых БИС [17] представлена на рисунке 1.7.

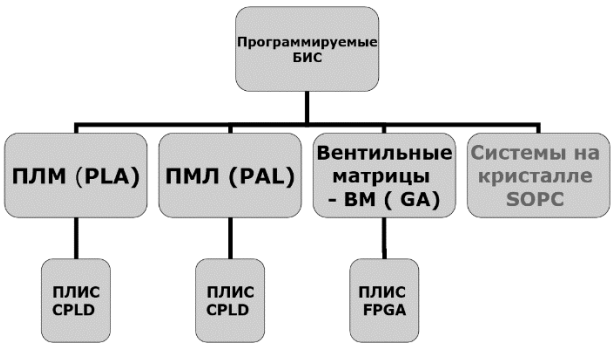


Рисунок 1.7. Классификация основных программируемых цифровых БИС

Архитектура конфигурируемого логического блока (КЛБ) ПЛИС FPGA [18-20] изображена на рисунке 1.8.

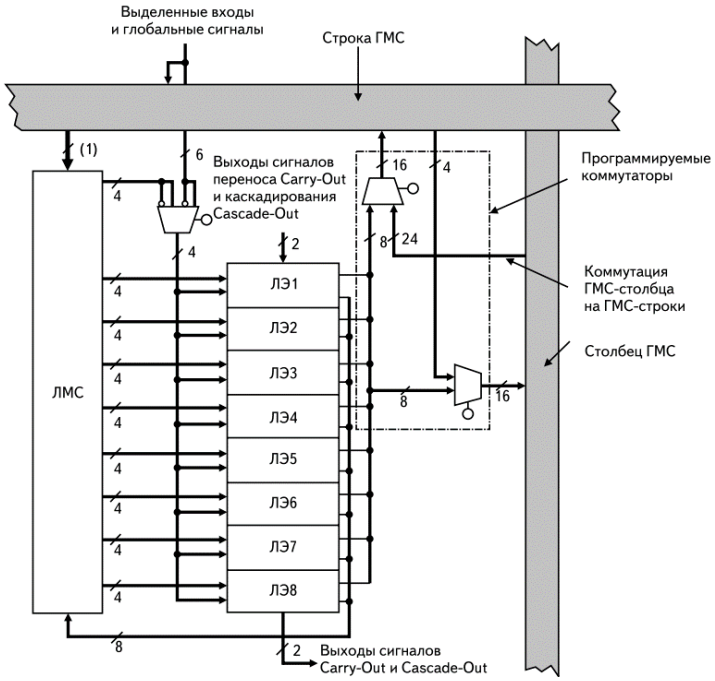


Рисунок 1.8. Архитектура типового конфигурируемого логического блока (КЛБ) ПЛИС FPGA

КЛБ содержит 8 логических элементов, которые, в свою очередь содержат следующие основные блоки: генератор функций четырех переменных – так называемый LUT (Look Up Table) и элемент памяти – D-триггер – рисунок 1.9.

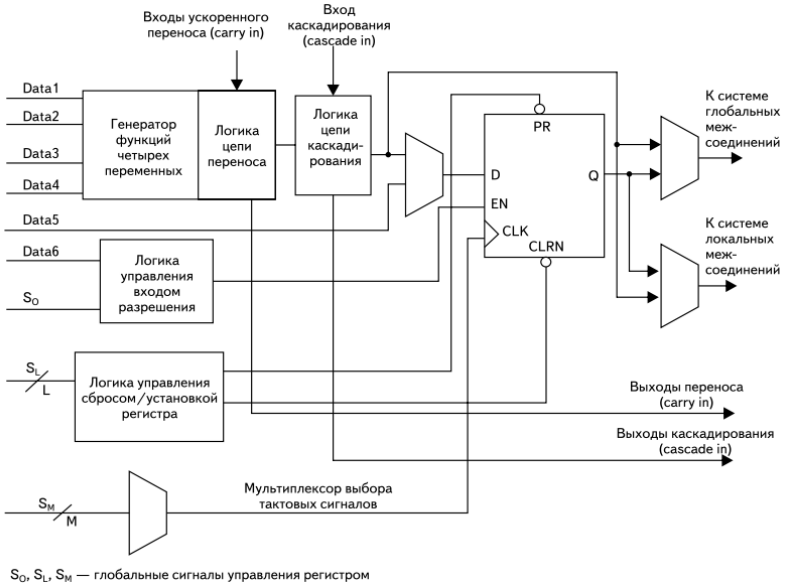


Рисунок 1.9. Архитектура типового ЛЭ КЛБ

Генератор функций четырех переменных строится на основе двух генераторов функций трех переменных, LUT - на три переменных (3-LUT) имеет вид [18-20], показанный на рисунке 1.10:

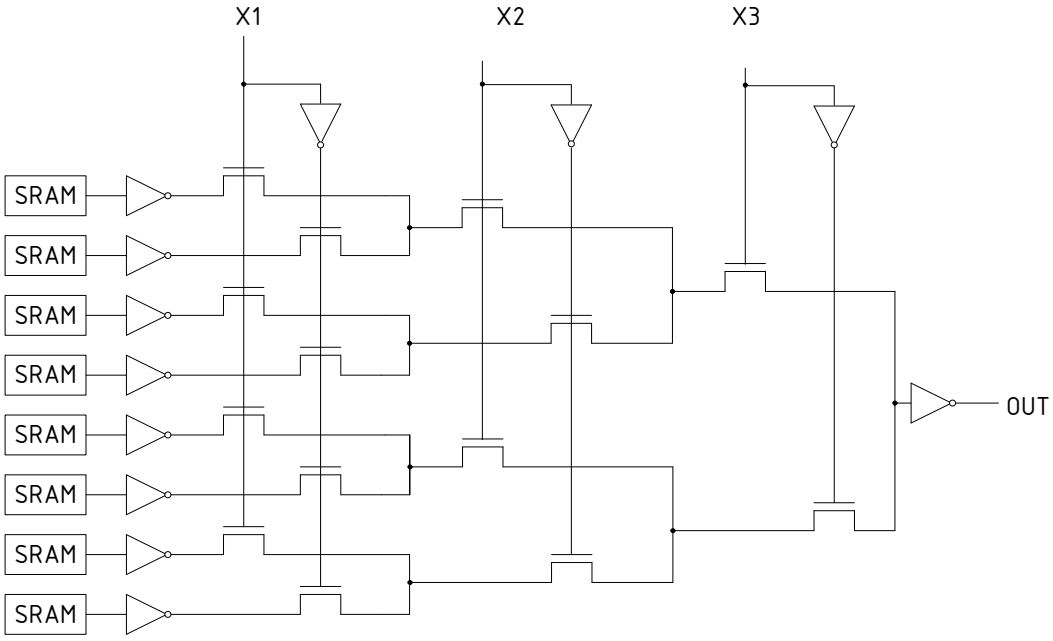


Рисунок 1.10. Генератор функций трех переменных - LUT на три переменные (3-LUT)

Настройка логической функции $z_{OUT}(x_3x_2x_1)$ определяется загрузкой конфигурационной оперативной памяти SRAM (a,b,c,d,e,f,g,h):

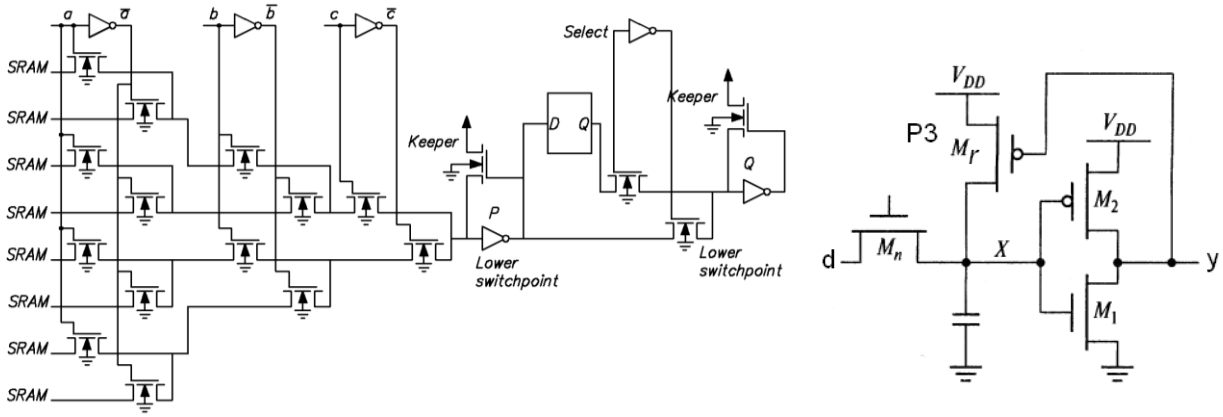
$$z_{OUT}(x_3x_2x_1) = a\bar{x}_3\bar{x}_2\bar{x}_1 \vee b\bar{x}_3\bar{x}_2x_1 \vee c\bar{x}_3x_2\bar{x}_1 \vee d\bar{x}_3x_2x_1 \vee ex_3\bar{x}_2\bar{x}_1 \vee fx_3\bar{x}_2x_1 \vee gx_3x_2\bar{x}_1 \vee hx_3x_2x_1. \quad (1.1)$$

Загружая необходимую настройку в ячейки SRAM, LUT реализует одну логическую функцию. Для реализации другой функции от этих же входных переменных необходимо либо загрузить новую настройку в ячейки памяти, либо задействовать дополнительный ЛЭ.

Более детальная структура генератора функций трех переменных - LUT на три переменные (3-LUT) с учётом D-триггера содержит двойные инверторы – буферы по входам переменных и восстановители по выходам LUT – рисунок 1.11.

Архитектура маршрутизации ПЛИС (FPGA Routing Architecture) включает программируемые коммутаторы связей (Programmable Routing

Interconnect, локальные и глобальные матрицы связей) ЛЭ[18]. Такой коммутатор имеет вид показанный на рисунке 1.12.



а) б)

Рисунок 1.11. 3-LUT: а) 3-LUT с двойными инверторами и триггером; б) восстановитель - буфер с запоминанием уровня

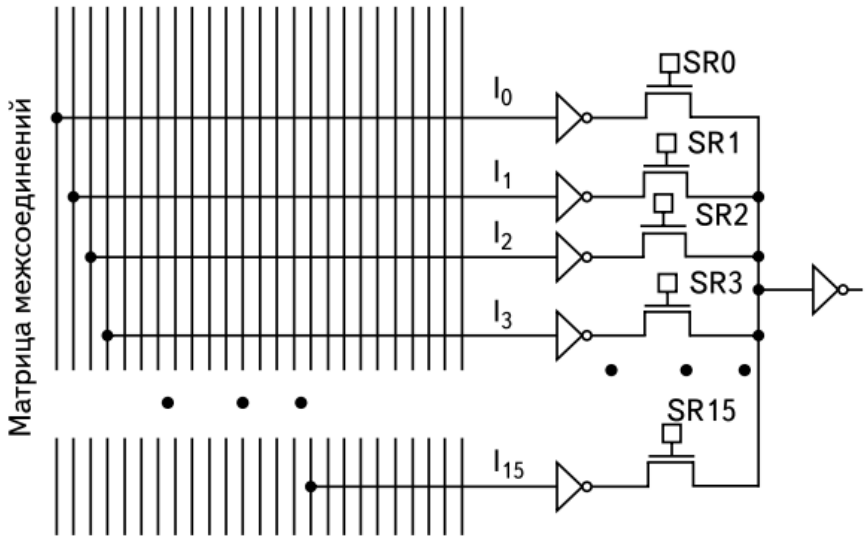


Рисунок 1.12. Программируемый коммутатор связей ПЛИС

По типу архитектуры связей ПЛИС делятся на островные (island-style, mesh-based FPGA) и иерархические [19,20]. Пример островной ПЛИС (Altera Cyclone, Stratix) изображён на рисунке 1.13.

В островной ПЛИС все КЛБ «равны», в иерархических ПЛИС установлена иерархия связей, «ближние» КЛБ соединяются проще, чем «дальние» - рисунок 1.14.

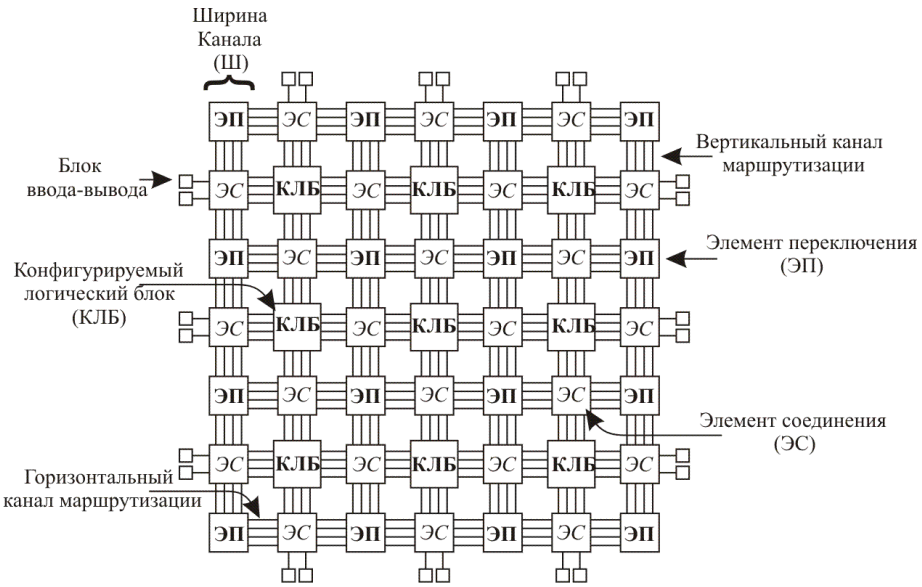


Рисунок 1.13. Пример островной ПЛИС

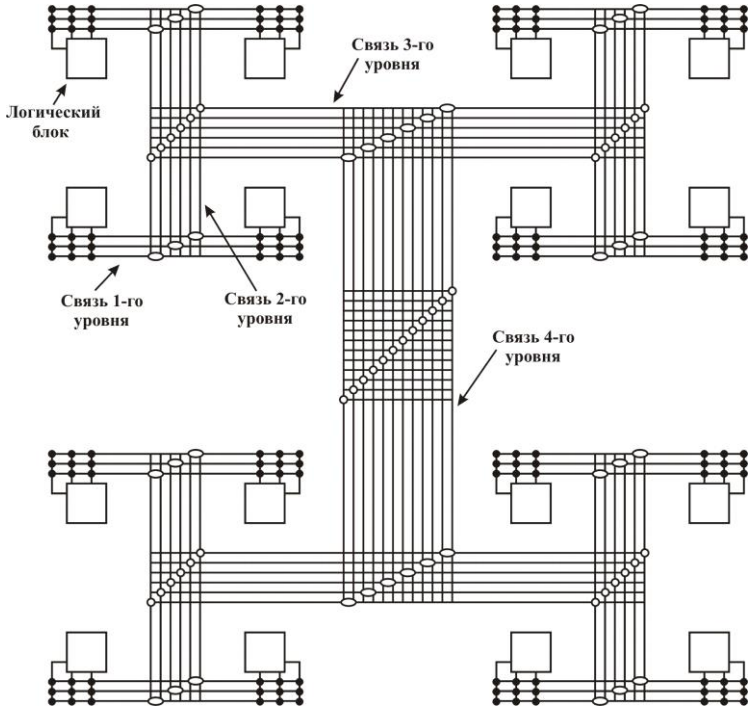


Рисунок 1.14. Пример иерархической ПЛИС (Flex10K, APEX фирмы Altera (Intel))

По типу конфигурации памяти ПЛИС различают SRAM - based, Flash - based, Antifuse – based [19,20]. SRAM – based основаны на ячейках SRAM при этом после выключения питания программа, записанная в ячейки, стирается и существует необходимость при следующем включении питания заново записывать прошивку в ПЛИС. Для хранения конфигурации на ПЛИС устанавливают блок ПЗУ (Flash или другой энергонезависимой (Nonvolatile)

памяти), с которого осуществляется загрузка. Такая технология весьма уязвима при воздействии радиации [19,20].

Flash-based дороже SRAM-based, так как совмещение технологии (MOS) МОП и Flash весьма сложно, к тому же общее возможное число циклов перезаписи Flash значительно меньше, чем у SRAM –based.

Antifuse - based программируются однократно, как БМК, но зато она менее подвержена радиации.

1.2. Анализ адаптивных многоразрядных логических элементов FPGA

Если первые ПЛИС содержали сначала сотни ЛЭ, потом их уже были тысячи, десятки тысяч, сотни тысяч, то сейчас современные ПЛИС имеют миллионы ЛЭ, при этом общее количество транзисторов – несколько миллиардов [21-23]. Это уже не просто «гибкая логика», появилось понятие «Гиперфлекс» (The Intel HyperFlex FPGA Architecture). В ПЛИС реализуются уже конвейерные вычисления, «Гиперфлекс коммутация», «Гиперрегисты», «Гипероптимизация». Архитектура ПЛИС (например, Stratix III Altera (Intel)) включает целые «лаборатории» логики - блоки LAB (logic array block), они состоят из адаптивных логических модулей ALM (adaptive logic modules) в количестве 10 штук – рисунок 1.15:

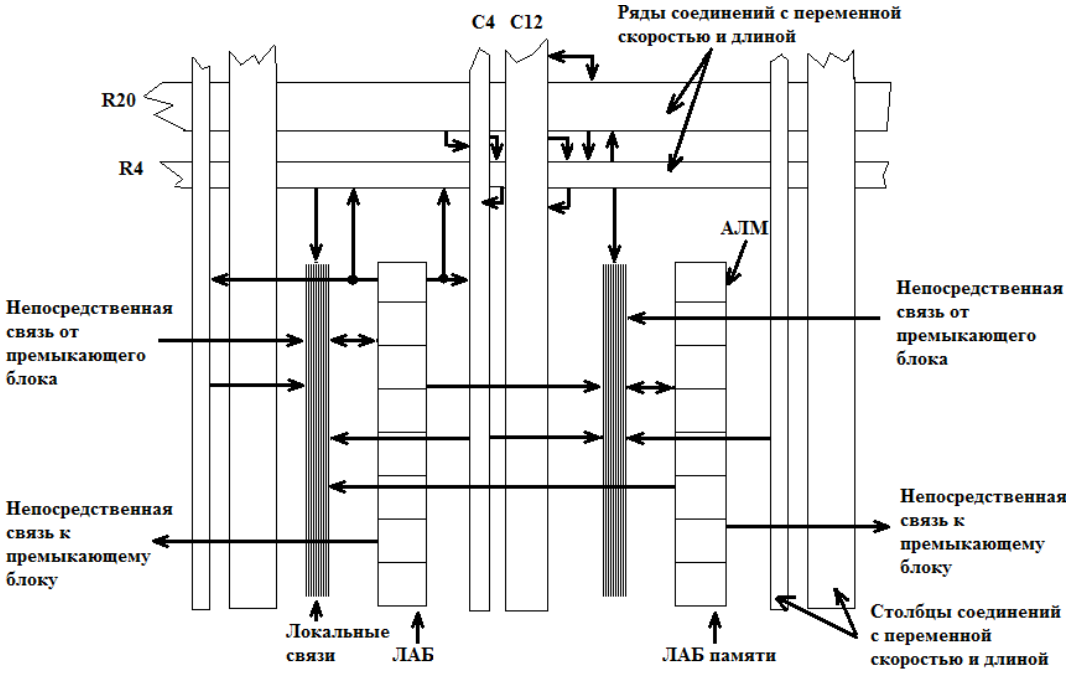


Рисунок 1.15. Архитектура ПЛИС Stratix III

Имеются «лаборатории памяти» - Memory LAB (MLAB) - LAB, с возможной реконфигурацией в ОЗУ - рисунок 1.16:

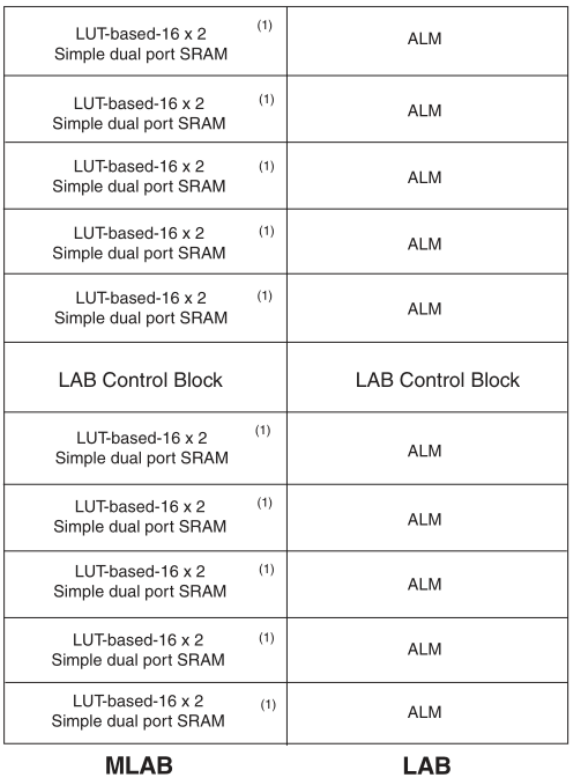


Рисунок 1.16. Состав MLAB и LAB

Непосредственная связь через ALM реализуется следующим образом – рисунок 1.17:

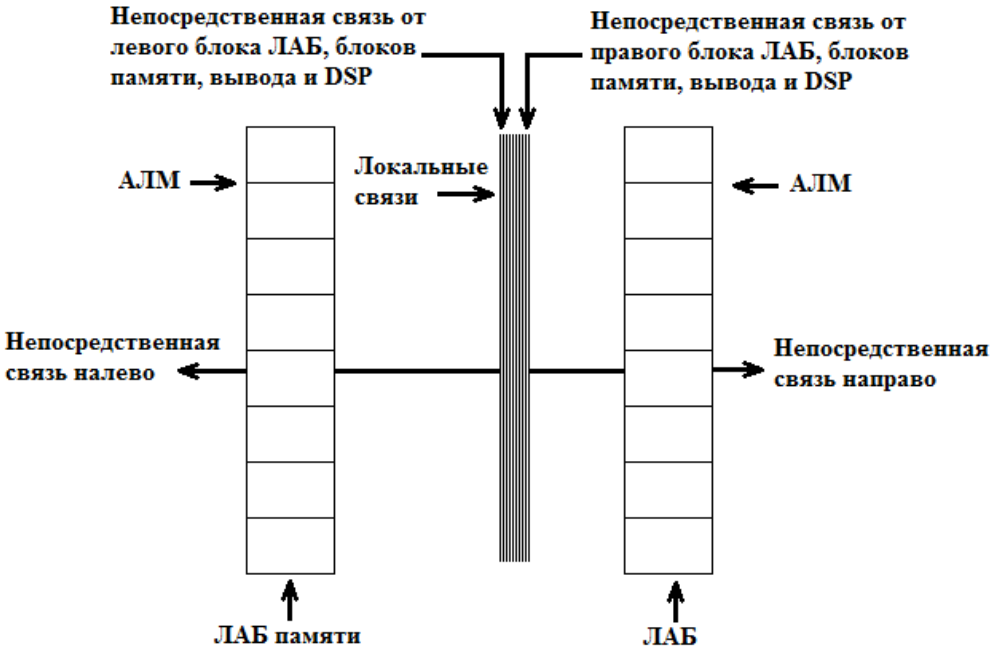


Рисунок 1.17. АЛМ может «пропускать» сигнал через себя -direct link connection

Это позволяет создать эффективную трассировочную архитектуру ЛЭ - MultiTrack – рисунок 1.18.

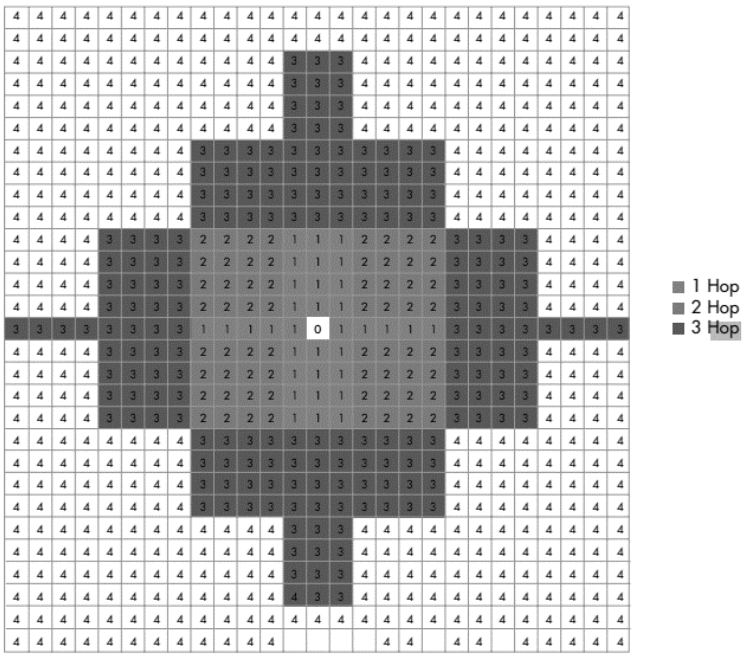


Рисунок 1.18. Трассировочная архитектура MultiTrack компании Altera (Intel), для элементарного шага используется термин hop (шаг).

В одной из последних разработок ПЛИС Intel Stratix 10 используется технология «Триггеры повсюду» [22], что обеспечивает конвейеризацию вычисления логических функций при соответствующей «гиперсинхронизации» – рисунок 1.19.

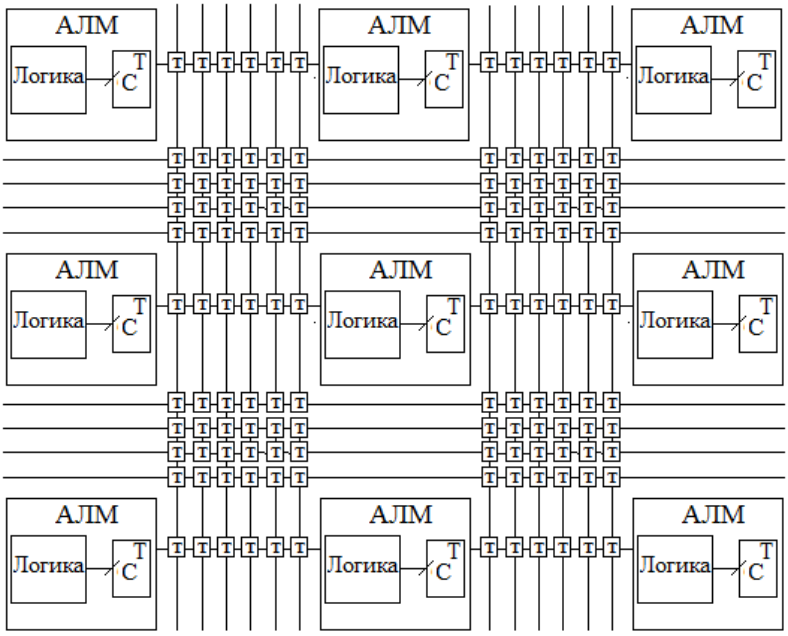


Рисунок 1.19 Технология «Триггеры повсюду»

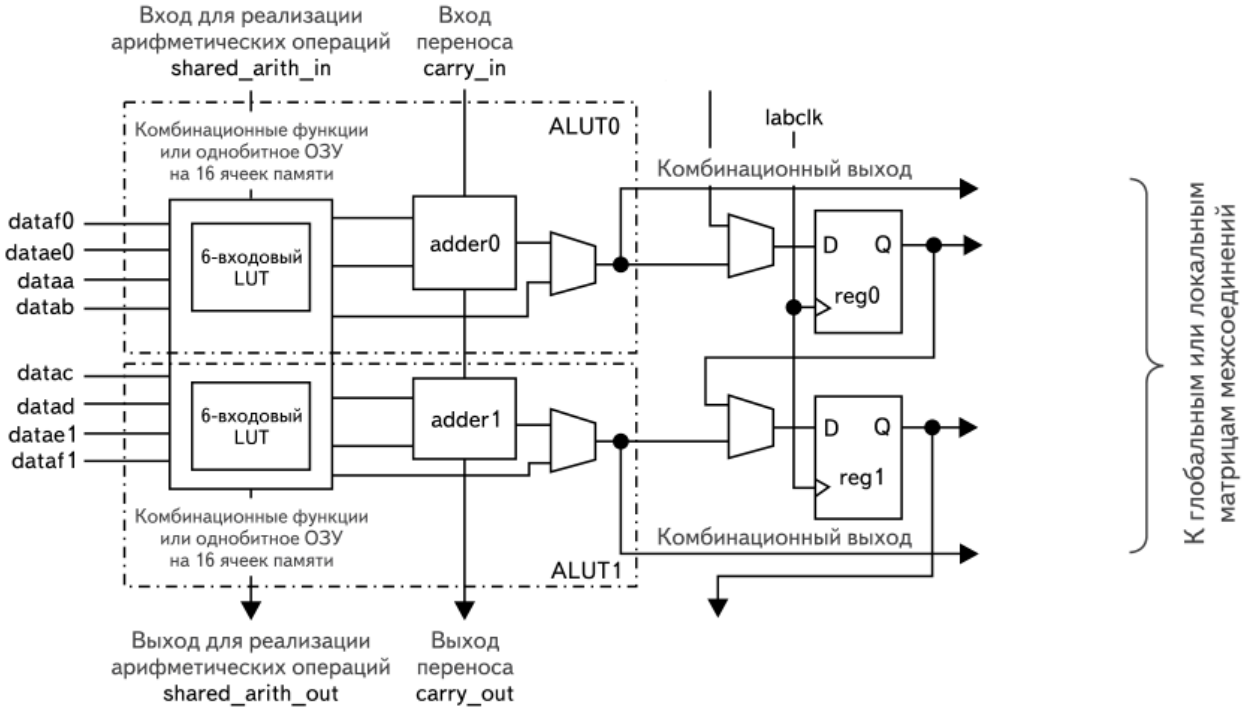
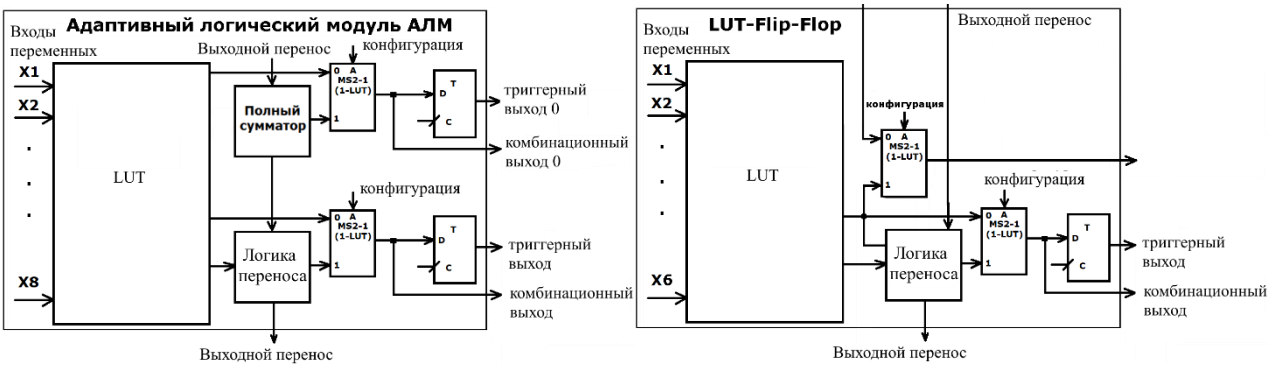


Рисунок 1.20. АЛМ ПЛИС Stratix III

Архитектура АЛМ Stratix III [21,22] имеет вид – рисунок 1.20.

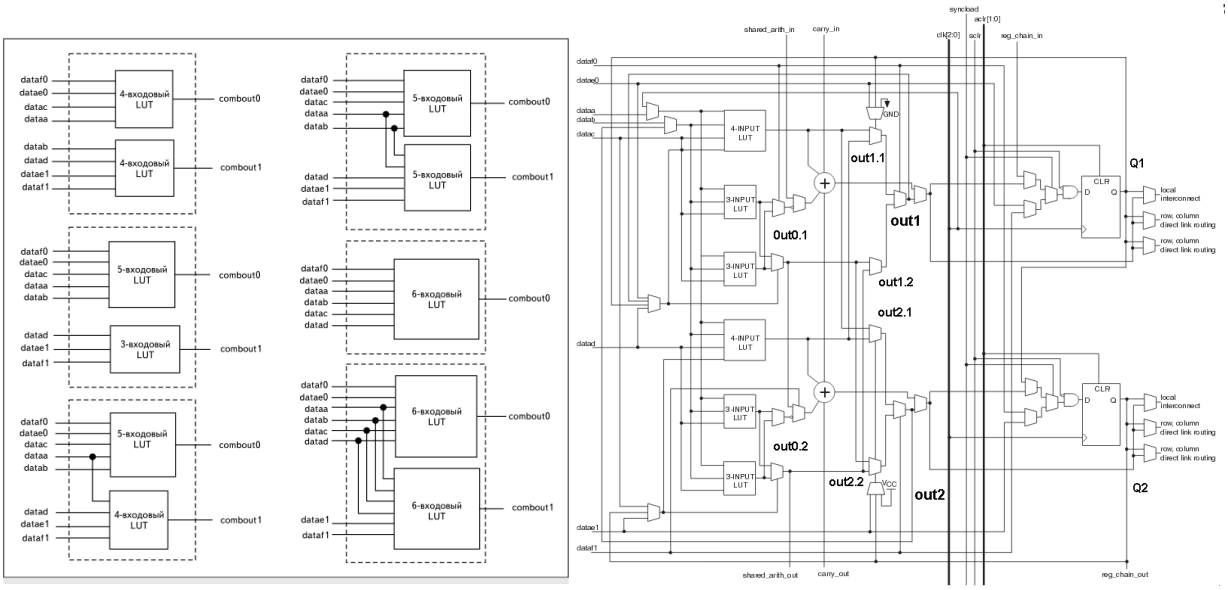
Устройства, подобные АЛМ включаются и в ПЛИС фирмы Xilinx [23] - рисунок 1.21.



а) б)

Рисунок 1.21. АЛМ: а) фирмы Altera (Intel): б) фирмы Xilinx

Хотя число входов АЛМ лежит в диапазоне 6-8 переменных, но детальный анализ показывает, что имеется фактически только 8 блоков 3-LUT (рисунок 1.22), то есть могут реализовать любые функции только 6 переменных.



а) б)

Рисунок 1.22. Архитектура АЛМ: а) упрощенная структура; б) детализация АЛМ

Реализация любых функций 7,8 переменных, то есть реализуются только некоторые функции, такие как:

$$4LUT_{1.1} = [a \vee (clk)];$$
$$4LUT_{1.2} = [b \vee (out2)];$$
$$4LUT_{1.3} = c;$$
$$4LUT_{1.4} = [e0 \vee (out1) \vee Q1 \vee d].$$

$$3LUT_{1.1} = 3LUT_{2.1} = 4LUT_{1.1};$$
$$3LUT_{1.2} = 3LUT_{2.2} = 4LUT_{1.2};$$
$$3LUT_{1.3} = 3LUT_{2.3} = 4LUT_{1.3}.$$

$$4LUT_{2.1} = 4LUT_{1.1} = [a \vee (clk)];$$
$$4LUT_{1.2} = 4LUT_{1.1} = [b \vee (out2)];$$
$$4LUT_{1.3} = d;$$
$$4LUT_{1.4} = [c \vee Q2 \vee e1].$$

$$3LUT_{3.1} = 3LUT_{4.1} = 4LUT_{2.1};$$
$$3LUT_{3.2} = 3LUT_{4.2} = 4LUT_{2.2};$$
$$3LUT_{3.3} = 3LUT_{4.3} = 4LUT_{2.3}.$$

$$out0.1 = 3LUT_1 \vee 3LUT_2; \quad out0.2 = 3LUT_3 \vee 3LUT_4;$$
$$out1.1 = 4LUT_1 \vee 4LUT_2; \quad out2.1 = 4LUT_1 \vee 4LUT_2;$$
$$out1 = out1.1 \vee out1.2; \quad out2 = out1.1 \vee out1.2;$$
$$out1.2 = out0.1 \vee out2.2. \quad out1.2 = out0.1 \vee out2.2. \qquad (1.2)$$

1.3. Анализ научно-методического аппарата оптимизации логики ПЛИС

Анализ научно-технических источников показывает, что «оптимизация» программируемой логики заключается в основном в «правильном» размещении проекта в ПЛИС [24]. Эти возможности имеются в САПР: Quartus II (Altera (Intel)), ISE Suite, Vivaldo Design Suite (Xilinx), Libero IDE, Libero SoC (Microsemi). Выделяются ключевые параметры: потребляемая мощность, размер проекта и его быстродействие. Например, для уменьшения размера проекта в САПР Xilinx: Project -> Design Goals & Strategies, Area Reduction [25].

В САПР Quartus II фирмы Altera (Intel) можно минимизировать память автомата (State Machine) – установить вместо унитарного кодирования (One-hot) минимальное – (Minimal Bits), имеются опции анализа энергопотребления power analysis.

Работ, посвященных оптимизации при проектировании самой ПЛИС немного. В [26] описываются ALM и исследуется требуемая для них размерность логического элемента (генератора логических функций) LUT FPGA. Наиболее предпочтительным по задержке с учетом площади кристалла в качестве базового устанавливается 4-LUT [21,26] – рисунок 1.23.

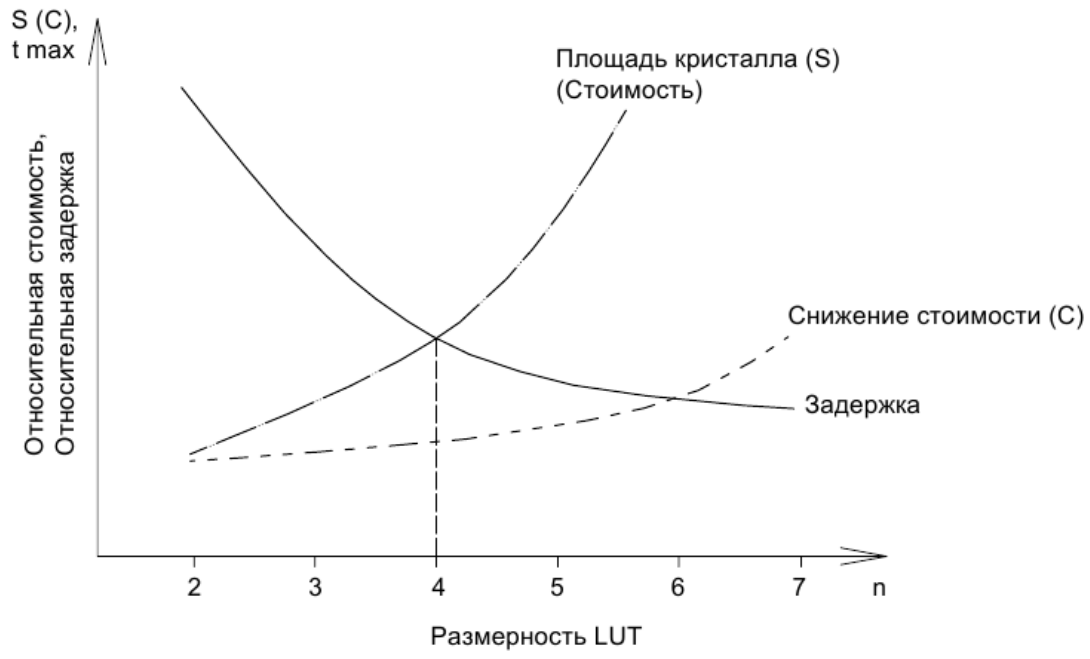


Рисунок 1.23. Оптимизация разрядности LUT

Причём процент использования LUT разной разрядности показывает целесообразность использования адаптивных LUT [26] – рисунок 1.24.

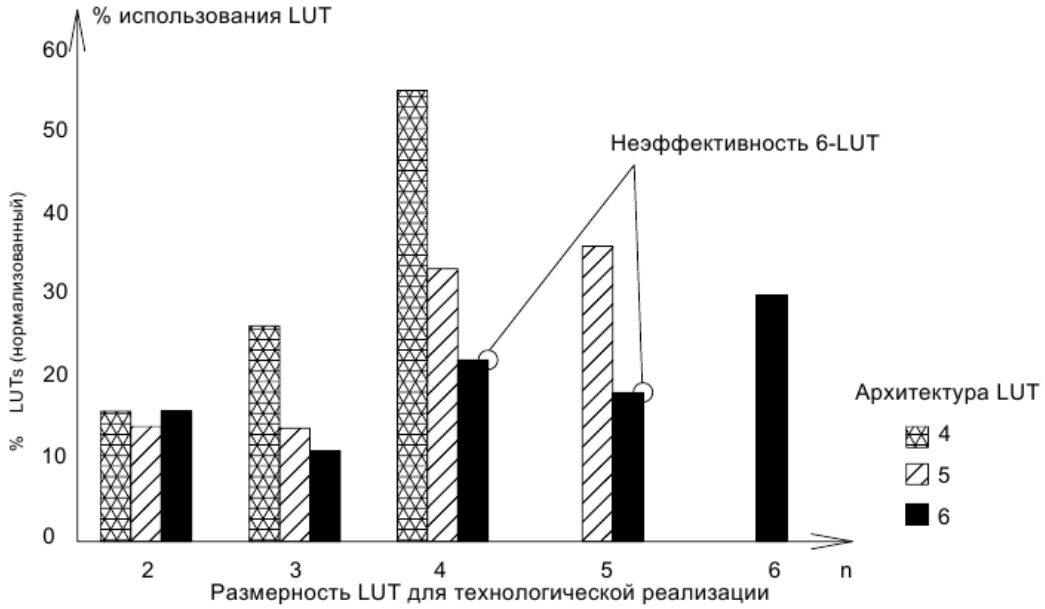


Рисунок 1.24. Распределение потребности в LUT по результатам синтеза

Такое распределение различается у разных производителей, использующих разные САПР [26] – рисунок 1.25.

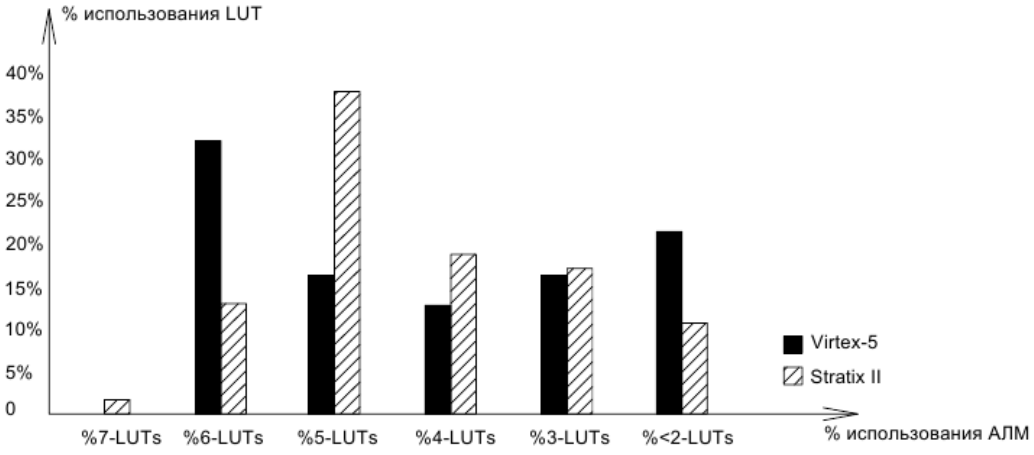


Рисунок 1.25. Распределение потребности в LUT

Кроме того, выполняется сравнительный анализ гибкости логики различных производителей, очевидно в рекламных целях [26] – рисунки 1.26, 1.27.

АЛМ Stratix III		LUT Virtex-5	Минимальное число общих входов АЛМ
Выход 1	Выход 2		
5-вх. LUT	5-вх. LUT	5	2
5-вх. LUT	4-вх. LUT	4	1
5-вх. LUT	3-вх. LUT	3	0
4-вх. LUT	4-вх. LUT		
4-вх. LUT	3-вх. LUT	2	
3-вх. LUT		1	

Рисунок 1.26. Сравнение гибкости логических элементов FPGA Stratix III и Virtex-5

Число шагов	Число достижимых LAB/CLB		Число достижимых ЛЭ		Отношение числа ЛЭ Stratix III к ЛЭ Virtex-5
	Stratix III	Virtex-5	Stratix III	Virtex-5	
1	34	12	850	132	6,4
2	96	96	2400	1056	2,3
3	160	180	4000	1980	2,0
Всего	290	288	7250	3168	2,3

Рисунок 1.27. Сравнение возможностей межсоединений FPGA Stratix III и Virtex-5

Так, АЛМ компании Altera (Интел) способен реализовать две независимые 4-входовые функции (без общих входов), тогда как LUT Virtex-5 требует использования трех общих входов [21].

Имеются работы исследователей из университета Торонто по созданию комбинированной ПЛИС FPGA-CPLD в плане использования ПМЛ (PAL), что, как утверждается, уменьшает площадь кристалла вдвое [27,28] – рисунок 1.28.

BM	4-LUT (area)	HFA (area)	PALBs	4-LUTs	Area gain
5xp1	78	28	4	12	2.8
9sym	172	69	11	25	2.5
count	55	39	8	7	1.4
C499	74	74	0	74	1
9symml	159	58	9	22	2.7
misex1	21	16	3	4	1.3
s298	1970	271	62	23	7.3
z4ml	18	10	1	6	1.8
vg2	69	32	4	16	2.2
alu2	213	105	14	49	2
Average					2.5
Total:	2829	702	116	238	4

Рисунок 1.28. Оценка комбинированного подхода ПЛИС FPGA-CPLD - Hybrid FPGA Architecture (HFA)

Здесь количество конъюнкций (Product terms), так и входов исследуемых функций оценивается от 1 до 50. Оценивается работа ПЛИС FPGA на частоте 10 до 50 МГц, в то же время CPLD - 100 МГц. Используется случайный синтез схем (Benchmark Circuit – BenchMark - BM) и три статистических параметра: 1) среднее число входов, 2) среднее число конъюнкций (Pterms), 3) отношение количества входов к числу Pterms для каждого комбинационного узла. На рисунке 1.29 приведены значения и дисперсия этих параметров [27,28].

Параметр случайного синтеза	**Входы (μ, σ)**	**Среднее число конъюнкций (μ, σ)**	**Входы / Среднее число конъюнкций (μ, σ)**	**Среднее число конъюнкций (отфильтровано) (μ, σ)**	**Входы / Среднее число конъюнкций (отфильтровано) (μ, σ)**
K=6	12.5, 6.56	9.32, 7.29	1.55, 1.02	6.64, 7.48	1.81, 1.32
K=5	11.57, 6.48	8.47, 7.08	1.61, 1.01	6.25, 7	1.77, 1.21
K=4	10.04, 6.32	7.23, 6.62	1.68, 1.03	5.53, 6.3	1.7, 1.07
K=3	8.59, 6.09	6.02, 6.17	1.78, 1.01	4.89, 5.62	1.62, 0.94
K=2	6.35, 5.45	4.36, 5.21	1.83, 0.94	3.77, 4.58	1.5, 0.76
K=1	4.54, 4.69	3.12, 4.26	1.74, 0.79	2.96, 3.65	1.29, 0.66
K=0	4.46, 4.66	3.07, 4.22	1.72, 0.79	2.91, 3.62	1.29, 0.66

Рисунок 1.29. Оценка параметров комбинационных схем методом случайного синтеза

Оценка энергопотребления и задержки единичного LUT дана, например, в [29] в связи со сравнением технологии MOS LUT с технологией CNTFET LUT (Carbon nanotube field-effect transistor) – рисунок 1.30.

Vdd	CMOS LUT power(nW)	CNFET LUT power(nW)
0.2	10.4	0.5
0.25	17	0.75
0.3	27.7	0.69
0.35	43.5	1.04
0.4	68.6	1.34

а)

Vdd	CMOS LUT delay(ns)	CNFET LUT delay(ns)
0.2	201	2.02
0.25	92	0.9
0.3	51.1	0.54
0.35	30.5	0.19
0.4	20	0.086

б)

Рисунок 1.30. Оценка MOS LUT при сравнении с технологией CNTFET LUT: а) потребляемая мощность; б) задержка

1.4. Постановка задачи исследования

Анализ объекта исследования [15-23] позволил установить востребованную тенденцию усложнения логических элементов ПЛИС и создания адаптивных логических модулей с увеличением числа переменных реализуемых логических функций на основе варьирования ЛЭ 3-LUT,4-LUT.

Анализ предмета исследования [30-45] показал, что дальнейшее увеличение числа переменных с целью повышения быстродействия LUT наталкивается на экспоненциальный рост сложности СДНФ, при этом подход реализации систем функций в ДНФ рассматривается только в контексте CPLD [23-29]. Совершенствование логики ПЛИС в РФ особенно актуально в связи с известными проблемами импортозамещения электронной компонентной базы [46-50]. Поэтому предлагается исследовать возможность модификации существующего адаптивного логического модуля – АЛМ в направлении реализации систем функций с возможной реализацией их в ДНФ.

Формальная постановка задачи следующая:

Дано: архитектура существующих ЛЭ и адаптивного логического модуля – АЛМ на их основе, множество элементов АЛМ: $\bigcup_{i=1}^{e} \Psi_i$, их количество:

$\sum_{i=1}^{e}|\Psi_i|$, значения сложности существующих логических элементов FPGA и CPLD L - $\sum_{i=1}^{e}\sum_{\xi=1}^{|\Psi_\xi|} L(n,m,v,k,r)_{i\xi}$, энергопотребления (потребляемой мощности) E- $\sum_{i=1}^{e}\sum_{\mu=1}^{|\Psi_\mu|} \mathrm{E}(n,m,v,k,r)_{i\mu}$ и максимальной задержки $\max \tau_{\Psi_i}$ относительно максимального количества n переменных, m функций, v конъюнкций в реализуемых логических функциях, а также k декомпозиции дерева транзисторов, r ограничения Мида-Конвей на число последовательно включенных передающих транзисторов (Pass Transistors).

Требуется: 1. Усовершенствовать методы реализации существующих АЛМ путём комплексирования подходов к реализации логики в FPGA и CPLD, и создания гибридного АЛМ - АЛМ*. Выбрать оптимальный набор логических элементов для реализации систем логических функций в АЛМ*, для чего:

2. Получить для АЛМ*: $\bigcup_{j=1}^{q}\Psi_i^*$ множество новых элементов на основе усовершенствованных методов реализации систем логических функций.

3. Для сравнения их между собой и с существующими элементами с целью выбора наиболее предпочтительных вариантов $\bigcup_{j=1}^{q}\Psi_i^*$ * выполнить моделирование в среде Multisim и Microwind с оценкой площади топологии S, энергопотребления E и максимальной задержки $\max \tau_{\Psi_i}$.

4. Относительно полученных вариантов $\bigcup_{j=1}^{q}\Psi_i^*$ * осуществить оптимизацию (минимизацию) по количеству транзисторов и площади S на основе получения их оценок сложности $L_s^*(n,m,v,k,r)_{j\gamma}$ и модифицирования венгерского метода (алгоритма) с целью получения оптимального набора (вектора) элементов $z_1 z_2 .... z_\pi$:

$$\{z_1 z_2 .... z_\pi\}(n,m,v,k,r); z_\lambda = \sum_{\sigma=1}^{\nu}\psi_\sigma; \psi_\sigma \in \Psi_i^*; \nu \le j; \sum_{\kappa=1}^{\pi} z = \sum_{i=1}^{e}|\Psi_i| \qquad (1.3)$$

для заданного набора параметров n,m,v,k,r :

$$L_{s(z_1 z_2 \ldots . z_\pi)}(n,m,v,k,r) \rightarrow \min. \qquad (1.4)$$

5. С использованием полученных наборов $\{z_1 z_2 \ldots . z_\pi\}(n,m,v,k,r)$ определить множество Парето, учитывающее полученные в результате моделирования оценки площади топологии, энергопотребления и задержки и их аппроксимацию для реальных значений параметров систем логических функций.

В результате получить требуемое количество новых элементов в одном АЛМ*: $\sum_{j=1}^{q}\left|\Psi_j^*\right|$ * так, чтобы с меньшими затратами в количестве транзисторов и площади топологии реализовать заданные системы логических функций

$$\sum_{j=1}^{q}\sum_{\gamma=1}^{|\Psi^*_\gamma|} L_s{}^*(n,m,v,k,r)_{j\gamma} << \sum_{i=1}^{e}\sum_{\xi=1}^{|\Psi_\xi|} L_s(n,m,v,k,r)_{i\xi}; \qquad (1.5)$$

при этом не ухудшая энергопотребление и задержку:

$$\sum_{j=1}^{q}\sum_{\gamma=1}^{|\Psi^*_\gamma|} \mathrm{E}(n,m,v,k,r)_{j\gamma} \le \sum_{i=1}^{e}\sum_{\xi=1}^{|\Psi_\xi|} \mathrm{E}(n,m,v,k,r)_{i\xi};$$
$$\forall \tau_{\Psi^*_j} \forall \tau_{\Psi_i} (\max \tau_{\Psi^*_j} \le \max \tau_{\Psi_i}). \qquad (1.6)$$

1.5. Выводы по главе 1

В главе представлены следующие основные результаты:

1. Выполнен анализ существующих БИС программируемой логики, который показал, что в настоящее время используются ПЛИС с логическими элементами, имеющими число переменных n=4,5,6. Дальнейшее увеличение числа переменных с целью повышения быстродействия LUT затруднительно в связи с экспоненциальным ростом сложности. Принцип программирования существующих ЛЭ ориентирован на раздельную реализацию логических функции в СДНФ, что делает невозможным реализацию систем функций или одновременное выполнение нескольких функций на одном ЛЭ относительно одних входных переменных. Для реализации систем функций приходиться

задействовать дополнительные ЛЭ, что ведет к дополнительным затратам площади кристалла.

2. Выполнен анализ адаптивных многоразрядных ЛЭ FPGA. В адаптивных логических модулях имеется возможность изменения размерности логического элемента до 7,8, но при этом реализуются не все возможные функции. Имеется большое количество публикаций посвященных комбинированию технологий различных типов ПЛИС, но нет исследований, посвященных реализации в FPGA систем логических функций от большого числа переменных, в том числе в ДНФ, с выбором оптимальных параметров для реализации типовых проектов. Реализации систем функций в ДНФ рассматривается только в контексте CPLD.

3. Выполнен анализ научно-методического аппарата оптимизации логики ПЛИС, который позволил выявить ключевые параметры такие, как энергопотребление, сложность, задержка. Существующие методы оптимизации программируемой логики в основном направлены на оптимальное размещение проекта в ПЛИС, не затрагивая самой структуры ПЛИС и ЛЭ.

4. Выполненный анализ показывает актуальность проведения исследований в области создания методов и средств реализации систем логических функций ПЛИС FPGA, так как особенности комплексирования подходов FPGA и CPLD не решены.

Глава 2. Разработка усовершенствованных методов реализации в FPGA систем логических функций

2.1. Усовершенствованный метод реализации в FPGA систем логических функций, заданных в СДНФ

Для реализации системы из m логических функций в существующих ПЛИС FPGA используются m деревьев передающих транзисторов [51,52], которые могут быть выражены следующим образом:

$$z_{out} = \bigvee_{i=1}^{2^n}\left(\mathop{\&}_{j=1}^{n} x_j^{\sigma(i-1,j)} \cdot d_i \right), \qquad (2.1)$$

где $\sigma(i,j)$ - показатель инверсирования переменной в соответствующей ветви дерева передающих транзисторов, его значение противоположно значению j-го разряда в двоичной записи числа i, $d_i \in \{0,1\}$ – настройка, значение i-й конфигурационной ячейки SRAM.

Выражение (2.1) реализует n-уровневое дерево передающих транзисторов. Представим его следующим образом:

$$z_{out}(x_n x_{n-1} \ldots x_2 x_1) = \vee\bullet \begin{array}{l} x_n \leftarrow \begin{array}{l} \bar{x}_{n-1} \leftarrow \begin{array}{l} \bar{x}_{n-2} \leftarrow \ldots\ldots \bar{x}_2 \leftarrow \begin{array}{l} \bar{x}_1 \cdot d_0 \\ x_1 \cdot d_1 \end{array} \\ x_{n-2} \leftarrow \end{array} \\ x_{n-1} \leftarrow \begin{array}{l} \bar{x}_{n-2} \leftarrow \\ x_{n-2} \leftarrow \end{array} \end{array} \\ \\ \bar{x}_n \leftarrow \begin{array}{l} \bar{x}_{n-1} \leftarrow \begin{array}{l} \bar{x}_{n-2} \leftarrow \\ x_{n-2} \leftarrow \end{array} \\ x_{n-1} \leftarrow \begin{array}{l} \bar{x}_{n-2} \leftarrow \\ x_{n-2} \leftarrow \ldots\ldots x_2 \leftarrow \begin{array}{l} \bar{x}_1 \cdot d_{2^n-2} \\ x_1 \cdot d_{2^n-1} \end{array} \end{array} \end{array} \end{array}, \qquad (2.1)$$

где $\vee\bullet$-корень дерева, монтажное ИЛИ (wired OR).

Предлагается реализация систем логических функций в СДНФ на основе DC LUT (DC-дешифратор, обратное дерево) [53-61]:

$$d_{out.i} = \mathop{\&}_{j=1}^{n} (x_j^{\sigma(i-1,j)}); i = 1,2^n, \qquad (2.2)$$

где $d_{out.i}$ - выход i-ой ветви дерева «наоборот», соответствующий истинности i-ой конъюнкции из 2^n конъюнкций n переменных.

$$d_{in} \bullet \begin{matrix} x_n \rightarrow \begin{matrix} \bar{x}_{n-1} \rightarrow \begin{matrix} \bar{x}_{n-2} \rightarrow \ldots\ldots \bar{x}_2 \rightarrow \begin{matrix} \bar{x}_1 \rightarrow d_{out\,0} \\ x_1 \rightarrow d_{out\,1} \end{matrix} \\ x_{n-2} \rightarrow \end{matrix} \\ x_{n-1} \rightarrow \begin{matrix} \bar{x}_{n-2} \rightarrow \\ x_{n-2} \rightarrow \end{matrix} \end{matrix} \\ \bar{x}_n \rightarrow \begin{matrix} \bar{x}_{n-1} \rightarrow \begin{matrix} \bar{x}_{n-2} \rightarrow \\ x_{n-2} \rightarrow \end{matrix} \\ x_{n-1} \rightarrow \begin{matrix} \bar{x}_{n-2} \rightarrow \\ x \end{matrix} \end{matrix} \end{matrix} \begin{matrix} \bar{x}_1 \rightarrow d_{out\,2^n-2} \\ n-2 \rightarrow \ldots\ldots x_2 \rightarrow \begin{matrix} \bar{x}_1 \rightarrow d_{out\,2^n-1} \end{matrix} \end{matrix}, \quad (2.3)$$

где $d_{in} \in 0,1$ определяет активный уровень при реализации соответствующей цепочки в обратном дереве.

Программирование значений m логических функций *z* осуществляется следующим образом:

$$z_l = \bigvee_{i=1}^{2^n} (d_{out.i} \cdot h_{l.i}); l = 1,m, \quad (2.4)$$

где *h* – настройка вхождения конституент *i* в данную функцию из *m* функций системы.

Рассмотрим подробно простейший LUT на одну переменную X (1-LUT) логическую функцию которого, можно выразить следующим образом:

$$\bar{z}_{out} = d_0 \cdot \bar{x} \vee d_1 \cdot \bar{\bar{x}}. \quad (2.5)$$

Причём двойная инверсия по входу X необходима для усиления сигнала, поступающего с матриц локальных и/или глобальных коммутаций. В дальнейшем будем указывать только один инвертор для получения сигнала не X. На выходе LUT устанавливается инвертор (буфер) для этих же целей, поэтому выходной сигнал (функция) инверсный. На входе настройки также имеются инверторы, а настройка (конфигурационные биты) также инверсна. Поэтому, получим:

$$z_{out} = d_0 \cdot \bar{x} \vee d_1 \cdot x. \quad (2.6)$$

В дереве передающих транзисторов, реализующих (2.4) в зависимости от сигнала X на входе инвертора функции, сигналы всегда ортогональны, то есть со входов настройки всегда подаётся либо 0, либо 1 и нет ситуации, когда оба передающих транзистора по X и не X не активированы. Выполним реверс (2.4) для дешифрации входного набора:

$$\overline{d}_i = \mathop{\&}_{j=1}^{n} x_j^{\sigma(i-1,j)} \cdot \overline{z}_{out}; i = 1,2^n, \tag{2.7}$$

Введём входной сигнал d вместо z, получим:

$$\overline{d}_{out.i} = \mathop{\&}_{j=1}^{n} x_j^{\sigma(i-1,j)} \cdot \overline{d}_{in}; i = 1,2^n, \tag{2.8}$$

Но при реализации (2.6) в виде дерева передающих транзисторов нарушается условие ортогональности сигналов, так как в случае не активации одного из передающих транзисторов вход одного из инверторов получается «оборванным». Ортогональность обеспечивается ($\overline{d}_{in} = 1$) в случае:

$$d_{out.i} = \mathop{\&}_{j=1}^{n} (x_j^{\sigma(i-1,j)} \vee x_j^{\overline{\sigma}(i-1,j)}); i = 1,2^n. \tag{2.9}$$

Причём, в отличие от обеспечения ортогональности в известном LUT, где сигналы со всех ветвей дерева «собираются» на одном выходе, что привело бы к выражению обеспечения ортогональности по выходу всей цепочки:

$$d_{out.i} = \mathop{\&}_{j=1}^{n} (x_j^{\sigma(i-1,j)}) \vee \mathop{\&}_{j=1}^{n} (x_j^{\overline{\sigma}(i-1,j)}); i = 1,2^n, \tag{2.10}$$

Выражение (2.7) описывает обеспечение ортогональности по каждой переменной в каждой ветви дерева.

Выражение (2.5) представляет собой мультиплексор 2-1 и может быть реализовано в виде элементарного дерева с управляемыми переменной X ветвями - рисунок 2.1:

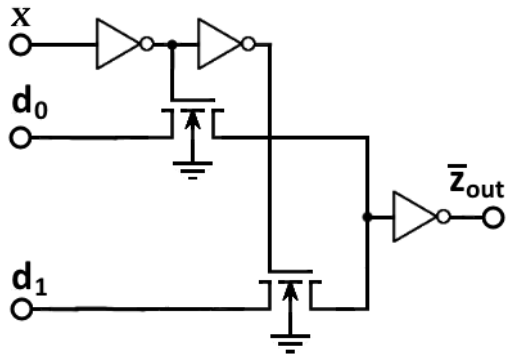


Рисунок 2.1. Элементарный LUT на одну переменную (1-LUT), настроенный на вычисление функции «не X»

Элементарный DC LUT - на одну переменную (1- DC LUT) получается путём передачи сигнала в элементарном 1- LUT в другом направлении – получаем дешифратор (DC)- рисунок 2.2:

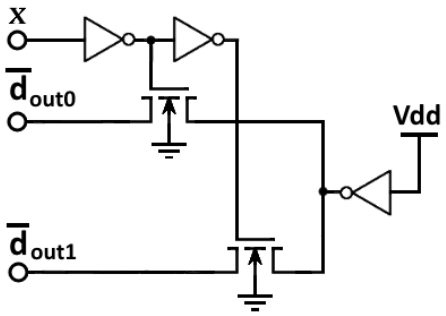


Рисунок 2.2. Элементарный DC LUT - на одну переменную (1- DC LUT)

Для рисунка 2.2 получим:

$$\bar{d}_{out.0}(x) = \bar{x};$$
$$\bar{d}_{out.1}(x) = x. \qquad (2.11)$$

Если исключить инвертор на входе, получим- рисунок 2.3:

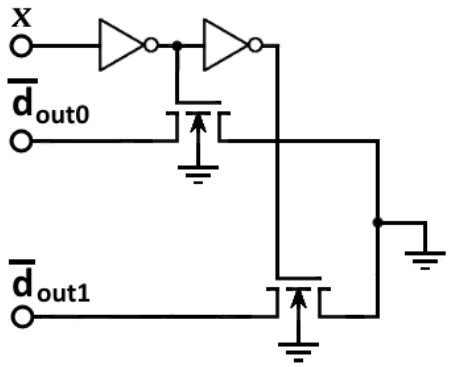


Рисунок 2.3. Элементарный 1-DC 1LUT - на одну переменную без входного инвертора

Тогда ноль передаётся либо на выход 0 (набор 0, X=0), либо на выход 1 (набор 1, X=1). Как окажется в дальнейшем, удобней активная единица на выходе, поэтому вводим инверторы- рисунок 2.4

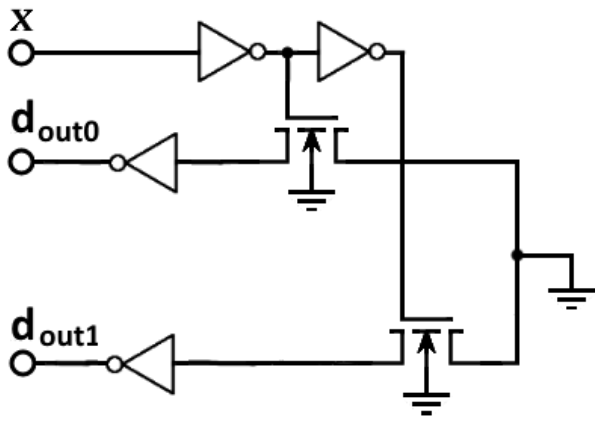


Рисунок 2.4. Элементарный 1-DC LUT - на одну переменную с выходными инверторами, активная единица

Выражению (2.9) обеспечения ортогональности при n=1 соответствует схема на рисунке 2.5:

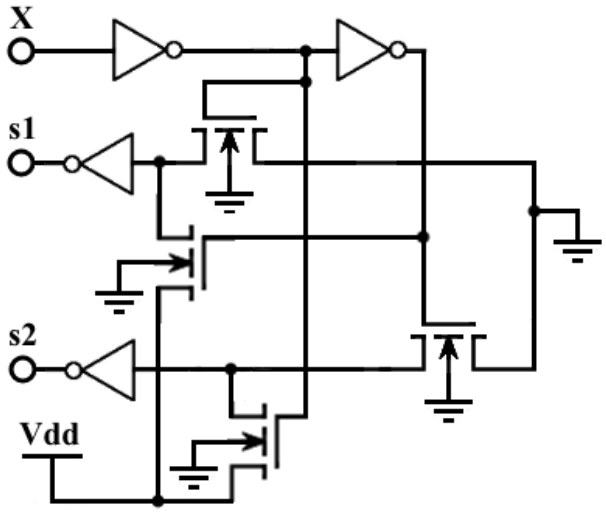


Рисунок 2.5. Элементарный дешифратор без инверторов на выходах и с обеспечением ортогональности сигналов на выходах s1, s2

Рассмотрим предлагаемую реализацию DC LUT [56] от большего числа переменных и особенности отказоустойчивой их реализации с учётом ограничений Мида-Конвей на число транзисторов в последовательной цепочке [62].

С учётом схемы на рисунке 2.5 дешифратор 2- DC LUT будет реализован следующим образом [56] – рисунок 2.6:

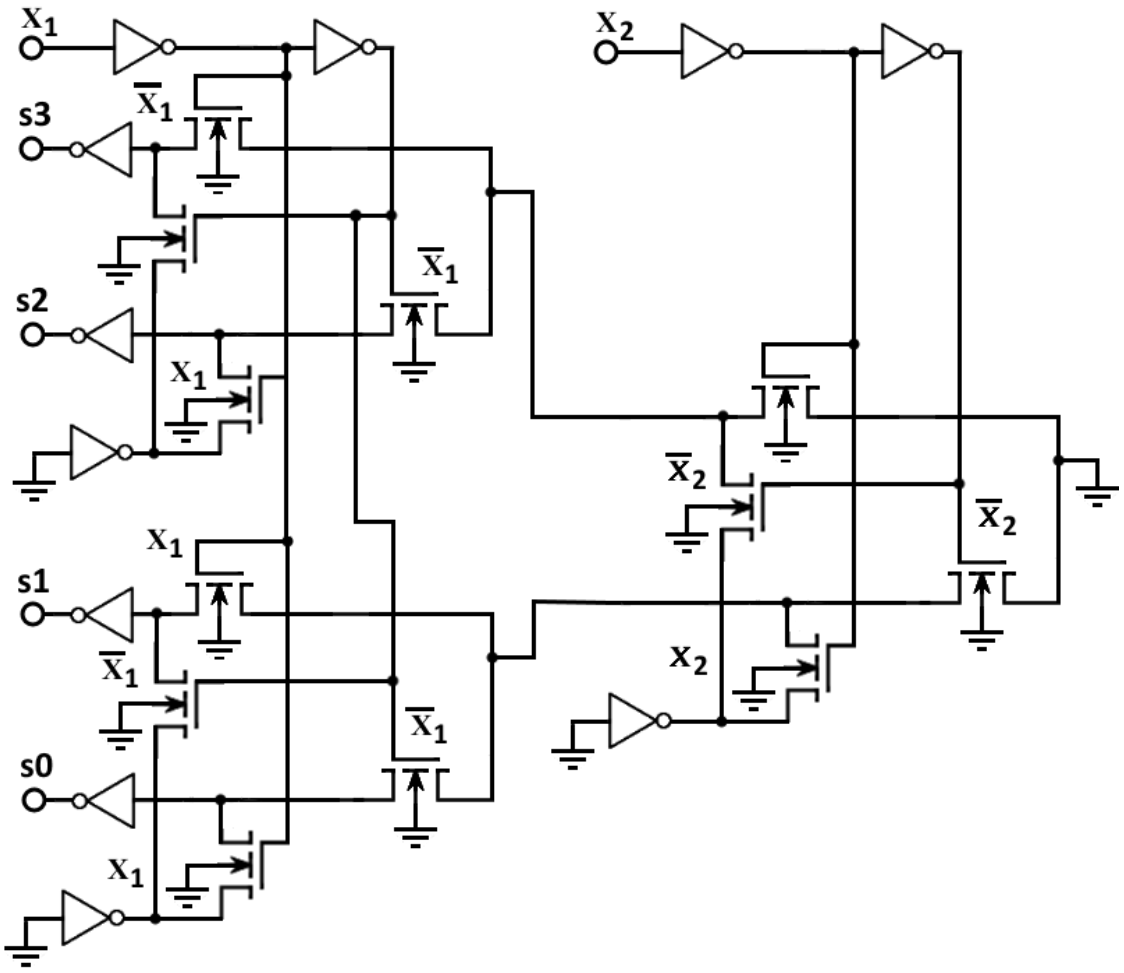


Рисунок 2.6. Дешифратор на две переменные с обеспечением ортогональности сигналов по выходам 0,1,2,3

Здесь, в отличие от LUT, используется локальная ортогональность – в каждом элементарном дешифраторе. Оба дополнительных транзистора активированы, но на вход инвертора по выходу 3 поступает логическая единица только через $\overline{x_1}$ поскольку x_1 закрыт.

На наборе 01 на вход инвертора по выходу 3 поступает логическая единица через $\overline{x_1}$ поскольку x_1 закрыт, а через x_2 идёт ноль, но не доходит.

На наборе 10 на вход инвертора по выходу 3 поступает логическая единица через $\overline{x_2}$ и x_1 поскольку x_1 открыт, а x_2 закрыт. Для «ортогонализации» по одной переменной можно использовать один инвертор, в отличие от того, что представлено на рисунке 2.7:

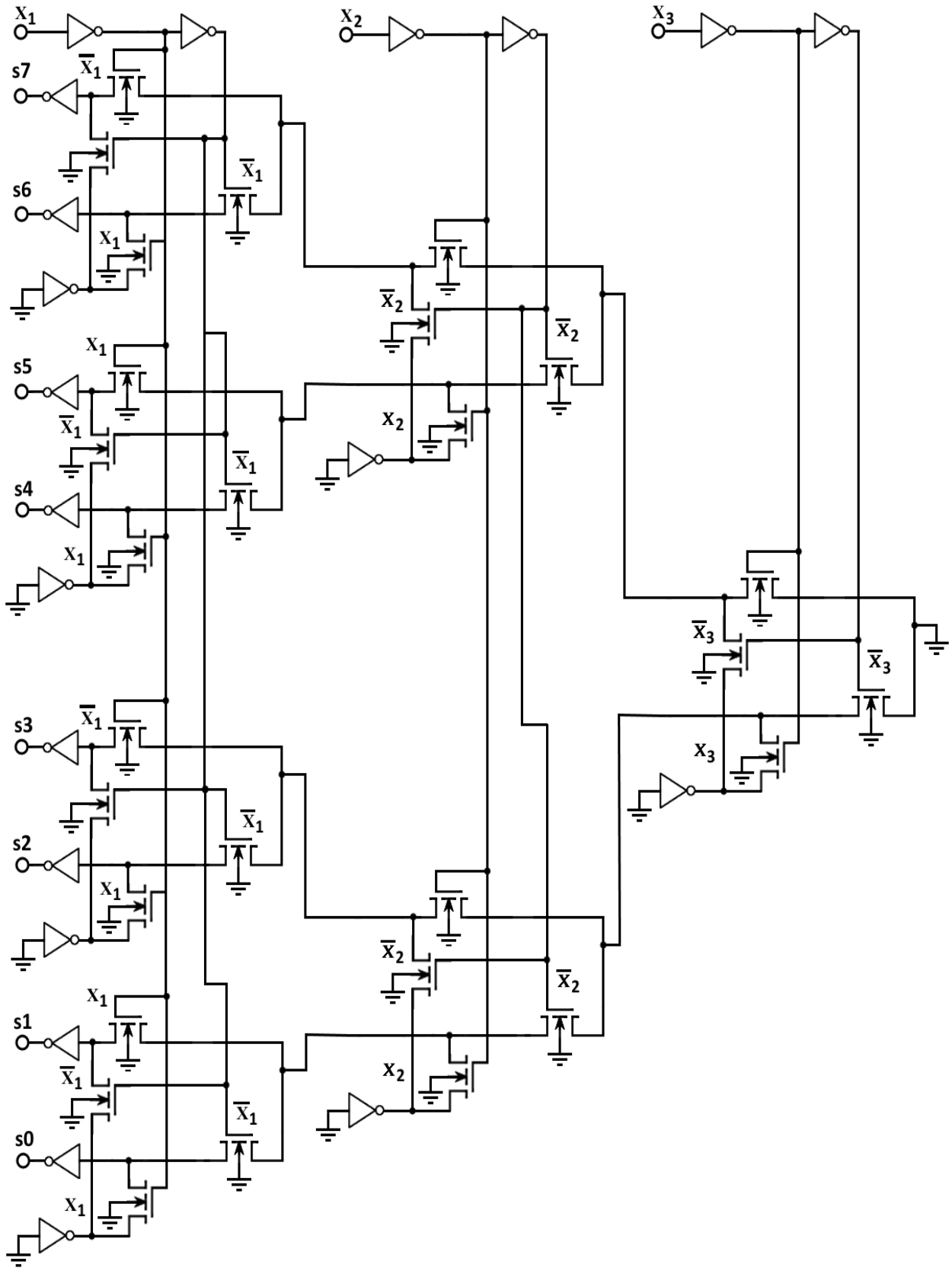


Рисунок 2.7. Дешифратор на три переменные с обеспечением ортогональности сигналов по каждому передающему транзистору

Если отказаться от инверторов по цепям ортогональности, получим схему на рисунке 2.8:

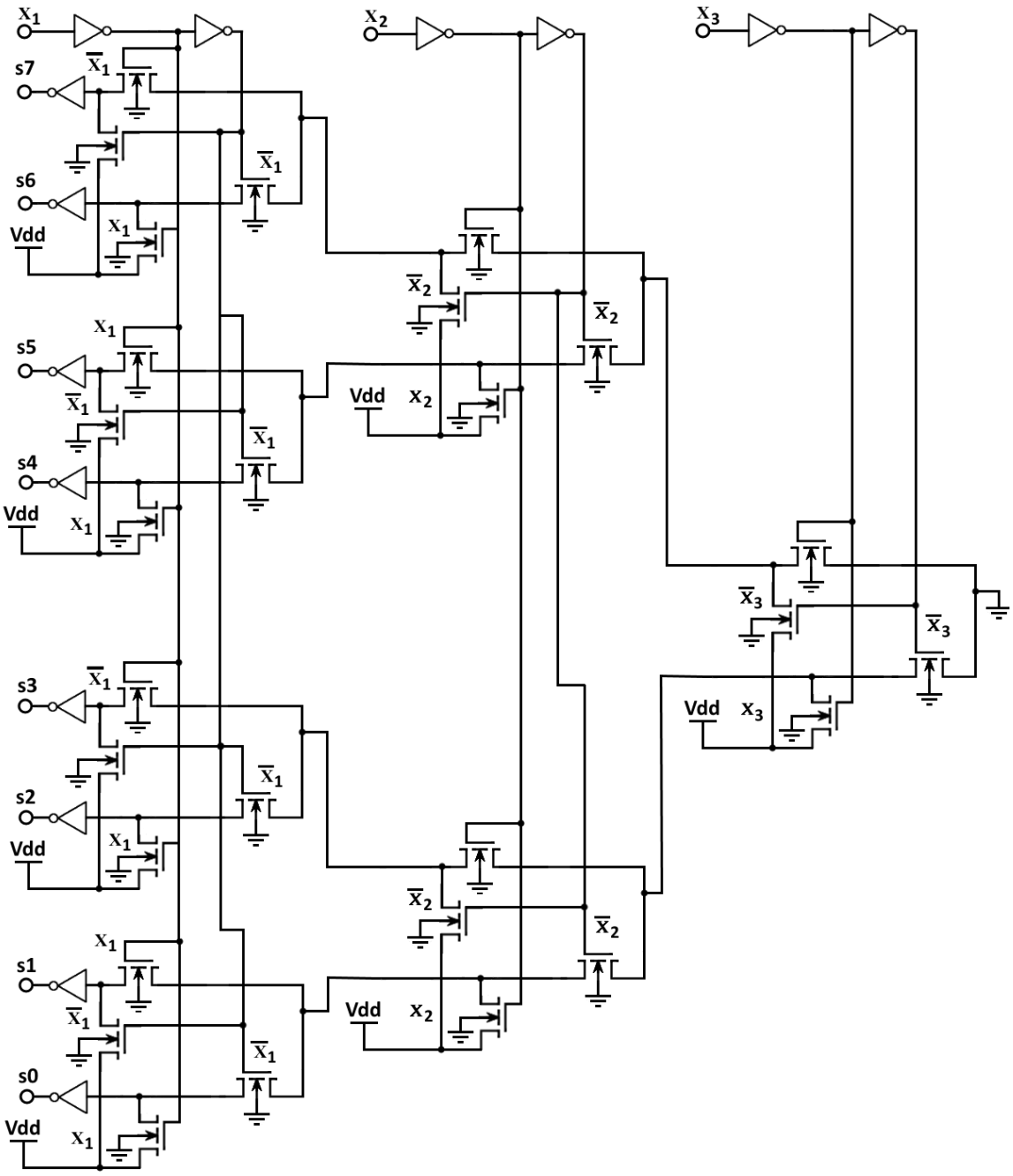


Рисунок 2.8. Дешифратор на три переменные с обеспечением ортогональности сигналов по выходам 0,1,2,3,4,5,6,7 по каждому транзистору с коммутацией шин «Ноль вольт» и «Vdd»

Дешифратор на три переменные с обеспечением ортогональности сигналов на выходах 0,1,2,3,4,5,6,7 изображён на рисунок 2.9:

Предлагается блок дизъюнкций конституент логической функции - рисунок 2.10:

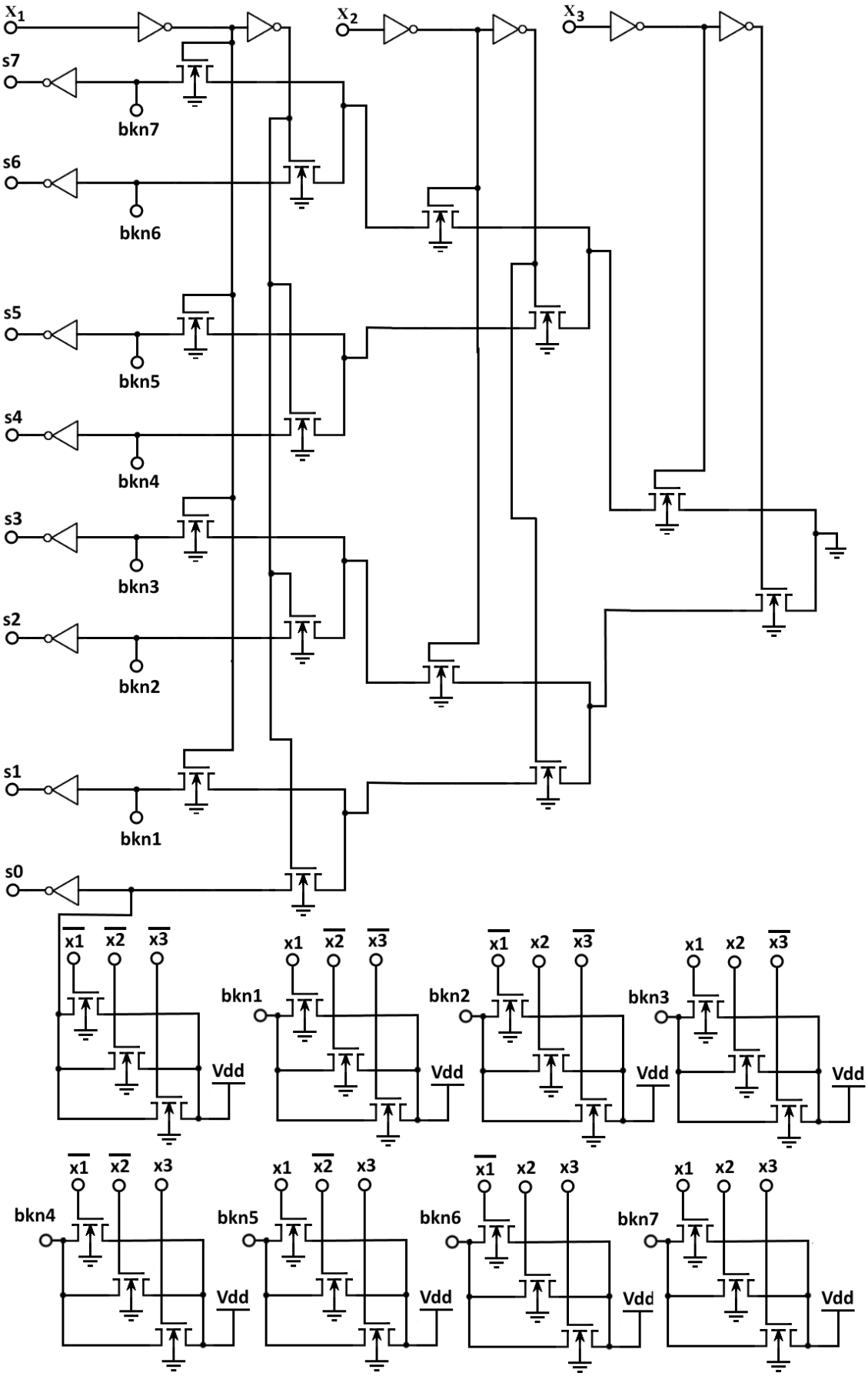


Рисунок 2.9. Дешифратор на три переменные с обеспечением ортогональности сигналов на выходах S 0,1,2,3,4,5,6,7.

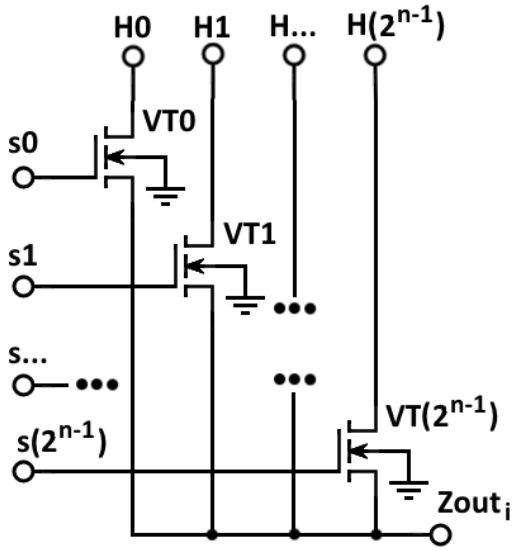


Рисунок 2.10. Схема блока дизъюнкций конституент логической функции

Активные единицы с выхода DC LUT открывают один из транзисторов блока на рисунке 2.10, а настройка H – инверсная должна быть записана? в соответствующих ячейках ОЗУ (SRAM). Таким образом обеспечивается ортогональность и путь прохождения сигнала по сравнению с LUT увеличивается на два транзистора в блоке рисунок 2.10 - один передающий, другой в выходном инверторе.

2.2. Разработка адаптивного логического элемента

Вариант DC LUT (для n=1) с $d_{in} = 0$ и активной логической единицей на двух выходах без транзисторов ортогональности, с резисторами по выходам изображён на рисунке 2.11.

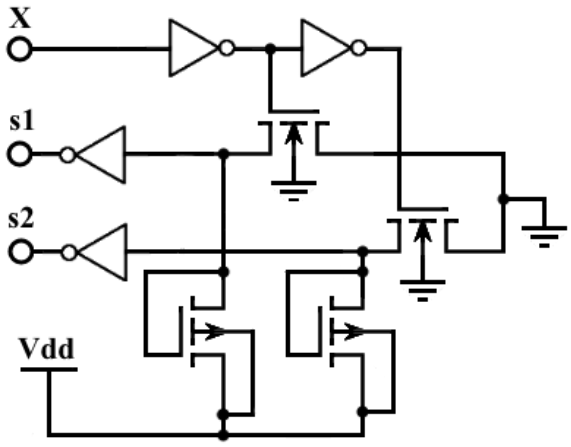


Рисунок 2.11 Элементарный DC LUT с восстановителями сигнала по выходам цепочки передающих транзисторов

На основе схемы на рисунке 2.11 построим адаптивный элемент [63]. Такой предлагаемый элемент может в зависимости от настройки реализовать либо LUT, либо DC LUT. Настройка дерева на реализацию дешифратора осуществляется конфигурационной константой s:

$$(d_{in}\cdot s \vee z_{out}(x_n \dots x_1)\cdot \bar{s})\cdot \begin{array}{l} x_n \quad \begin{array}{l} \bar{x}_{n-2} \\ x_{n-1}x_{n-2} \end{array} \dots \begin{array}{l} \bar{x}_1\cdot(\bar{s}\cdot d_{2^{n-1}} \vee s\cdot[=z_0]) \\ x_1\cdot(\bar{s}\cdot d_{2^{n-1}+1} \vee s\cdot[=z_1]) \end{array} \\ \overline{x_n} \quad \begin{array}{l} \bar{x}_{n-2} \\ x_{n-1}x_{n-2} \end{array} \dots \begin{array}{l} \bar{x}_1\cdot(\bar{s}\cdot d_{2^{n-1}-2} \vee s\cdot[=z_{2^{n-2}}]) \\ x_1\cdot(\bar{s}\cdot d_{2^{n-1}-1} \vee s\cdot[=z_{2^{n-1}}]). \end{array} \end{array}$$

(2.12)

Реализация дизъюнкций (функций) осуществляется подключением выходов (2.12) к блокам, реализующим выражения (2.4). Схема ADC LUT на одну переменную изображена на рисунке 2.12.

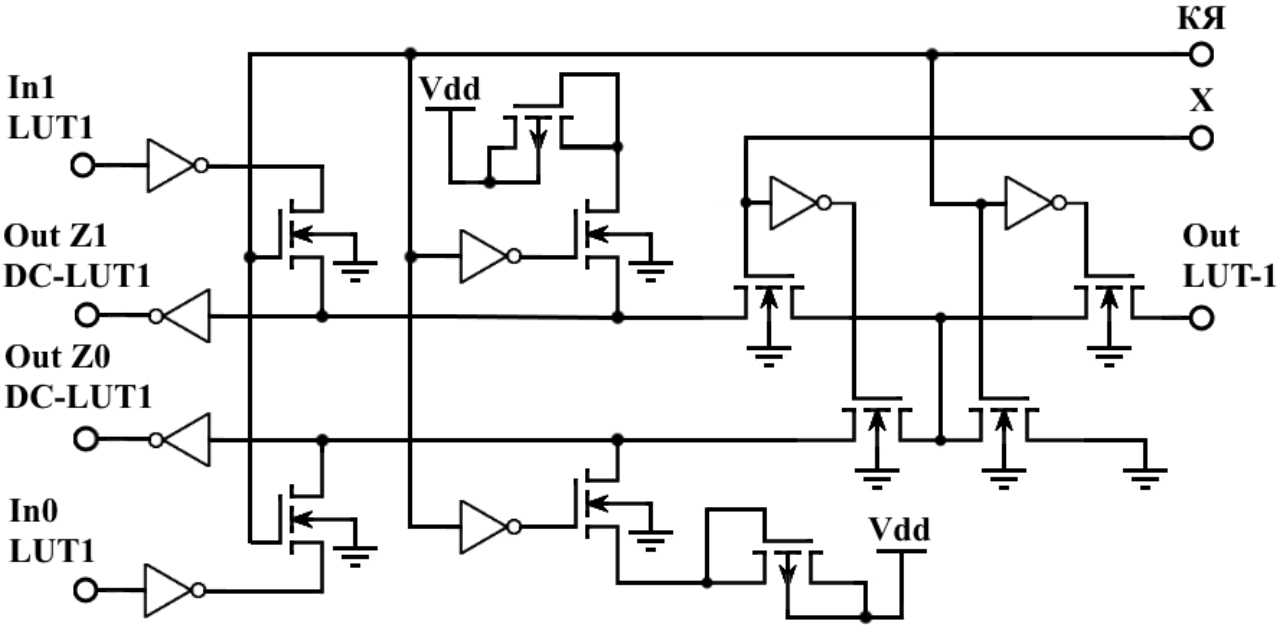


Рисунок 2.12. Схема ADC 1-LUT (на одну переменную), информация из КЯ-конфигурационной ячейки

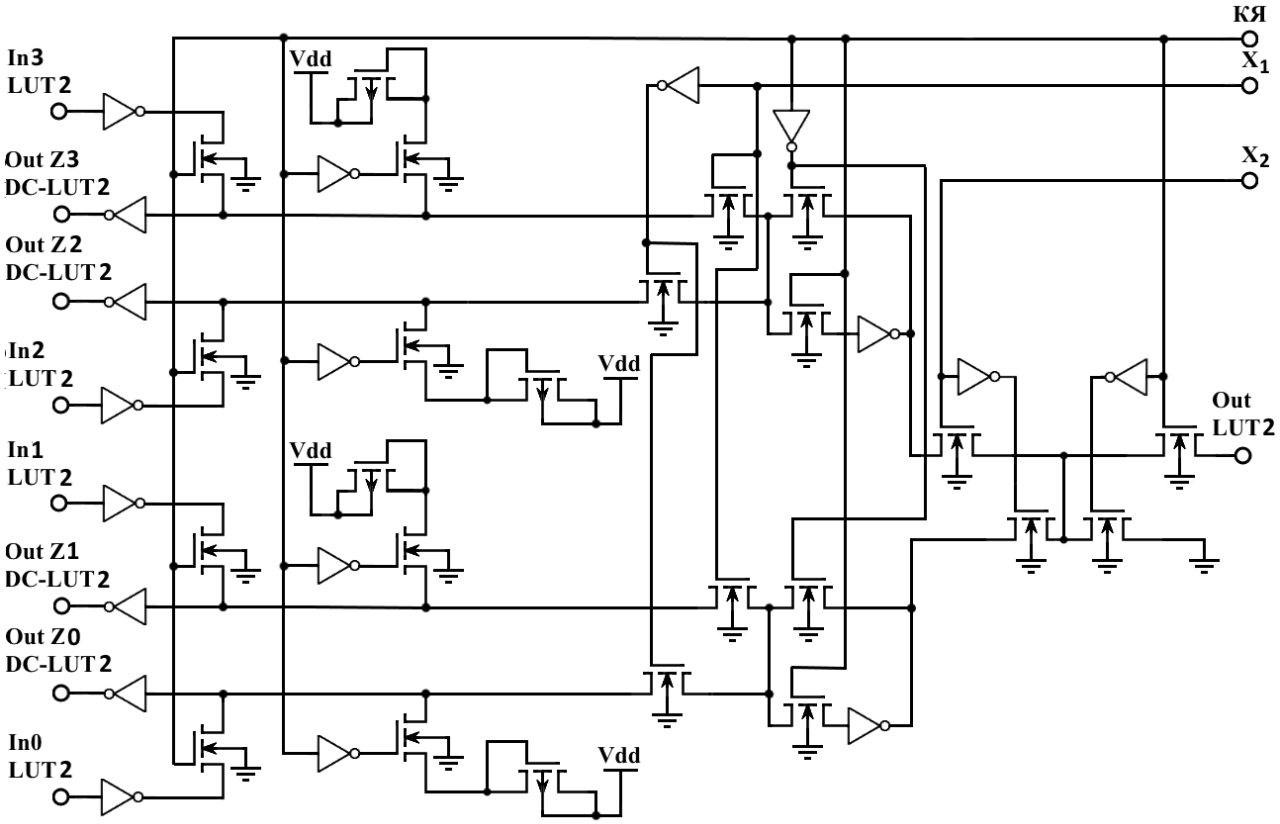


Рисунок 2.13. Схема ADC 2-LUT (на две переменных), информация из КЯ-конфигурационной ячейки

Таким образом, дополнительные коммутации возникают в «точках сочленения» дерева, если выполняется декомпозиция. Если её нет (на рисунке 2.13, её не будет, если исключить транзисторы, соединяющие поддеревья по x1 с поддеревом по x2), то дополнительные коммутации не нужны, остаются коммутации по входу и по выходам дерева. Это возможно до n=3. Эти результаты следует учесть при исследовании оценок сложности адаптации LUT для последующей оценки технико-экономической эффективности. Конкретная реализация коммутаций возможна на основе известных подходов к построению коммутаторов локальных матриц связей. То есть используются штатные коммутаторы, дополнительные инверторы и транзисторы. Кроме того, полученные сигналы значений конъюнкций могут быть использованы для настройки входов других LUT для реализации более сложных конъюнкций. Иначе, их можно подключать ко входам переменных других

LUT, например, для реализации дизъюнкций, в противном случае необходима реализация дополнительных блоков ИЛИ.

2.3. Усовершенствованный метод реализации в FPGA систем логических функций, заданных в ДНФ

Рассмотрим реализацию вычисления логических функций в ДНФ на основе программируемых логических матриц (ПЛМ) и в ПЛИС типа CPLD (Complex Programmable Logic Device) [16, 64]. Программирование по *i*-ой переменной *j*-ой конъюнкции, зависящей от одной, переменной, с учётом её возможного инверсного значения, может быть представлено следующим образом:

$$y_j(x_n \ldots x_2 x_1) = (x_i \vee s_{j.i})(\bar{x}_i \vee \bar{s}_{j.i}) \quad (2.13)$$

Для j-ой конъюнкции от *n* переменных, получим:

$$y_j = \mathop{\&}_{i=1}^{n} (x_i \vee s_{j.i})(\bar{x}_i \vee \bar{s}_{j.i}), j = 1,k \quad (2.14)$$

Причём, для настройки *s* выполняются следующие условия: $s_{j.i}\bar{s}_{j.i} = 0; s_{j.i} \vee \bar{s}_{j.i} = 1$ для существенной переменной *j*-ой конъюнкции, $s_{j.i}\bar{s}_{j.i} = 1$ для не существенной переменной.

Программирование дизъюнкций (значений функций) может быть описано выражением аналогичным (2.13). Тогда матрица И (AND array) может быть представлена следующим образом:

$$\begin{array}{l}
y_1(x_n x_{n-1} \ldots x_2 x_1) = \frac{x_n}{s_{1.n}} \cdot \frac{\bar{x}_n}{\bar{s}_{1.n}} \bullet \frac{x_{n-1}}{s_{1.n-1}} \cdot \frac{\bar{x}_{n-1}}{\bar{s}_{1.n-1}} \bullet \ldots . \frac{x_2}{s_{1.2}} \cdot \frac{\bar{x}_2}{\bar{s}_{1.2}} \bullet \frac{x_1}{s_{1.1}} \cdot \frac{\bar{x}_1}{\bar{s}_{1.1}} \\
y_2(x_n x_{n-1} \ldots x_2 x_1) = \frac{x_n}{s_{2.n}} \cdot \frac{\bar{x}_n}{\bar{s}_{2.n}} \bullet \frac{x_{n-1}}{s_{2.n-1}} \cdot \frac{\bar{x}_{n-1}}{\bar{s}_{2.n-1}} \bullet \ldots . \frac{x_2}{s_{2.2}} \cdot \frac{\bar{x}_2}{\bar{s}_{2.2}} \bullet \frac{x_1}{s_{2.1}} \cdot \frac{\bar{x}_1}{\bar{s}_{2.1}} \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
y_j(x_n x_{n-1} \ldots x_2 x_1) = \frac{x_n}{s_{j.n}} \cdot \frac{\bar{x}_n}{\bar{s}_{j.n}} \bullet \frac{x_{n-1}}{s_{j.n-1}} \cdot \frac{\bar{x}_{n-1}}{\bar{s}_{j.n-1}} \bullet \ldots . \frac{x_2}{s_{j.2}} \cdot \frac{\bar{x}_2}{\bar{s}_{j.2}} \bullet \frac{x_1}{s_{j.1}} \cdot \frac{\bar{x}_1}{\bar{s}_{j.1}} \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
y_k(x_n x_{n-1} \ldots x_2 x_1) = \frac{x_n}{s_{k.n}} \cdot \frac{\bar{x}_n}{\bar{s}_{k.n}} \bullet \frac{x_{n-1}}{s_{k.n-1}} \cdot \frac{\bar{x}_{n-1}}{\bar{s}_{k.n-1}} \bullet \ldots . \frac{x_2}{s_{k.2}} \cdot \frac{\bar{x}_2}{\bar{s}_{k.2}} \bullet \frac{x_1}{s_{k.1}} \cdot \frac{\bar{x}_1}{\bar{s}_{k.1}},
\end{array} \quad (2.15)$$

где $\cdot$-последовательное соединений по данной переменной; $\bullet$-последовательное соединений по разным переменным; $-$-параллельное соединение переменной и настройки.

Матрица ИЛИ (OR array) может быть описана следующим образом:

$$f_1(x_n x_{n-1}...x_2 x_1) = y_1(x_n x_{n-1}...x_2 x_1)q_{1.1} \vee y_2(x_n x_{n-1}...x_2 x_1)q_{1.2} \vee .... \vee y_k(x_n x_{n-1}...x_2 x_1)q_{1.k}$$
$$f_2(x_n x_{n-1}...x_2 x_1) = y_1(x_n x_{n-1}...x_2 x_1)q_{2.1} \vee y_2(x_n x_{n-1}...x_2 x_1)q_{2.2} \vee .... \vee y_k(x_n x_{n-1}...x_2 x_1)q_{2.k}$$
$$.....$$
$$f_\mu(x_n x_{n-1}...x_2 x_1) = y_1(x_n x_{n-1}...x_2 x_1)q_{\mu.1} \vee y_2(x_n x_{n-1}...x_2 x_1)q_{\mu.2} \vee .... \vee y_k(x_n x_{n-1}...x_2 x_1)q_{\mu.k} \quad (2.16)$$
$$.....$$
$$f_m(x_n x_{n-1}...x_2 x_1) = y_1(x_n x_{n-1}...x_2 x_1)q_{m.1} \vee y_2(x_n x_{n-1}...x_2 x_1)q_{m.2} \vee .... \vee y_k(x_n x_{n-1}...x_2 x_1)q_{m.k},$$

где $q_{\mu.j}; \mu = \overline{1,m}$-настройка вхождения конъюнкций в m функций.

В качестве альтернативы ПЛМ CPLD (рисунок 2.14), где ортогональность по переменным не обеспечивается, а используется «подтягивающий» резистор, предлагается программирование с обеспечением ортогональности на основе выражения:

$$s_i = (d_{SRAM.i} \;\; d_{in} \vee \bar{d}_{SRAM.i} \;\; d_0) \;\; x_i \vee (d_{SRAM.\bar{i}} \;\; d_{in} \vee \bar{d}_{SRAM.\bar{i}} \;\; d_0) \;\; \bar{x}_i; i = n. \quad (2.17)$$

В выражении (2.17) $d_{SRAM.i}, d_{SRAM.\bar{i}}$, - настройка вхождения i-ой переменной из n переменных в j-ую конъюнкцию системы из m функций; d_{in} -входная константа, которая должна формироваться при «правильном» значении переменной, либо её несущественности; d_0 -признак того, что заданная переменная имеет «неправильное» значение. Этот признак передаётся дальше, он участвует в формировании значения конъюнкции.

Суть в том, что в случае несущественности переменной, значение выражения (2.17) было бы эквивалентно значению выражения при правильном, заданном значении переменной. То есть, если $d_{SRAM.i} = 1; d_{SRAM.\bar{i}} = 0,$ переменная существенна и должна быть равна 1 (без инверсии), при

$d_{SRAM.i}=0; d_{SRAM.\bar{i}}=1,$ переменная существенна и должна быть равна 0 (с инверсией), при $d_{SRAM.i}=1; d_{SRAM.\bar{i}}=1,$ переменная не существенна. Значения $d_{SRAM.i}=0; \bar{d}_{SRAM.i}=0,$ запрещены. Рассмотрим пример. Пусть i-я переменная должна быть 1. Настройка для этого случая приведена в Таблице 2.1:

Таблица 2.1. Задание значения 1 для i-ой переменной

$d_{SRAM.i}$	$d_{SRAM.\bar{i}}$	d_{in}	d_0	x_i	$\bar{x}_i$	s_i
1	0	1	0	1	0	1
1	0	1	0	0	1	0

Рассмотрим подробно. Пусть заданная переменная равна 1:

$$s_{i=1}=[(d_{SRAM.i}=1)\cdot(d_{in}=1)\vee(\bar{d}_{SRAM.i}=0)\cdot(d_0=0))]\cdot(x_i=1)\vee \vee[(d_{SRAM.\bar{i}}=0)\cdot(d_{in}=1)\vee(\bar{d}_{SRAM.\bar{i}}=1)\cdot(d_0=0))]\cdot(\bar{x}_i=0)=1. \quad (2.18)$$

Пусть заданная переменная равна 0:

$$s_{i=0}=[(d_{SRAM.i}=1)\cdot(d_{in}=1)\vee(\bar{d}_{SRAM.i}=0)\cdot(d_0=0))]\cdot(x_i=0)\vee \vee[(d_{SRAM.\bar{i}}=0)\cdot(d_{in}=1)\vee(\bar{d}_{SRAM.\bar{i}}=1)\cdot(d_0=0))]\cdot(\bar{x}_i=1)=0. \quad (2.19)$$

Пусть i-я переменная должна быть 0. Настройка для этого случая приведена в Таблице 2.2:

Таблица 2.2. Задание значения 0 для i-ой переменной

$d_{SRAM.i}$	$d_{SRAM.\bar{i}}$	d_{in}	d_0	x_i	$\bar{x}_i$	s_i
0	1	1	0	0	1	1
0	1	1	0	1	0	0

Рассмотрим подробно. Пусть заданная переменная равна 0:

$$s_{i=1}=[(d_{SRAM.i}=0)\cdot(d_{in}=1)\vee(\bar{d}_{SRAM.i}=1)\cdot(d_0=0))]\cdot(x_i=0)\vee \vee[(d_{SRAM.\bar{i}}=1)\cdot(d_{in}=1)\vee(\bar{d}_{SRAM.\bar{i}}=0)\cdot(d_0=0))]\cdot(\bar{x}_i=1)=1. \quad (2.20)$$

Пусть заданная переменная равна 1:

$$s_{i=0}=[(d_{SRAM.i}=0)\cdot(d_{in}=1)\vee(\bar{d}_{SRAM.i}=1)\cdot(d_0=0))]\cdot(x_i=1)\vee \vee[(d_{SRAM.\bar{i}}=1)\cdot(d_{in}=1)\vee(\bar{d}_{SRAM.\bar{i}}=0)\cdot(d_0=0))]\cdot(\bar{x}_i=0)=0. \quad (2.21)$$

Если заданная переменная несущественна получим Табл.2.3.

Таблица 2.3. Задание несущественной i-ой переменной

$d_{SRAM.i}$	$d_{SRAM.\bar{i}}$	d_{in}	d_0	x_i	$\bar{x}_i$	s_i
1	1	1	0	1	0	1
1	1	1	0	0	1	1

Пусть заданная несущественная переменная равна 1:

$$s_{i=1} = [(d_{SRAM.i} = 1) \cdot (d_{in} = 1) \vee (\bar{d}_{SRAM.i} = 1) \cdot (d_0 = 1))] \cdot (x_i = 1) \vee \\ \vee [(d_{SRAM.\bar{i}} = 1) \cdot (d_{in} = 1) \vee (\bar{d}_{SRAM.\bar{i}} = 1) \cdot (d_0 = 0))] \cdot (\bar{x}_i = 0) = 1. \quad (2.22)$$

Пусть заданная несущественная переменная равна 0:

$$s_{i=0} = [(d_{SRAM.i} = 1) \cdot (d_{in} = 1) \vee (\bar{d}_{SRAM.i} = 1) \cdot (d_0 = 1))] \cdot (x_i = 0) \vee \\ \vee [(d_{SRAM.\bar{i}} = 1) \cdot (d_{in} = 1) \vee (\bar{d}_{SRAM.\bar{i}} = 1) \cdot (d_0 = 0))] \cdot (\bar{x}_i = 1) = 1. \quad (2.23)$$

Выражения (2.11) для всех переменных данной конъюнкции могут быть объединены последовательно:

$$\&_{i=1}^{n} \{(d_{SRAM.i} \;\; d_{in} \vee \bar{d}_{SRAM.i} \;\; d_0) \;\; x_i \vee (d_{SRAM.\bar{i}} \;\; d_{in} \vee \bar{d}_{SRAM.\bar{i}} \;\; d_0) \;\; \bar{x}_i\}. \quad (2.24)$$

Причём $d_{in} = 1$ (2.11) для старшей переменной и передаёт логическую единицу, которая транслируется на выход всей цепочки.

При параллельной реализации для всей заданной конъюнкции:

$$\bigvee_{i=1}^{n} \{(d_{SRAM.i} \cdot d_{in} \vee \bar{d}_{SRAM.i} \cdot d_0) \cdot x_i \vee (d_{SRAM.\bar{i}} \cdot d_{in} \vee \bar{d}_{SRAM.\bar{i}} \cdot d_0) \cdot \bar{x}_i\}. \quad (2.25)$$

В первом случае (2.20) конъюнкция «набирается», то есть активируется соответствующая цепочка, если все переменные «правильные» - либо несущественны, либо равны заданному значению, то есть значение сигнала на её выходе равно логической единице.

Если хотя бы одна переменная «неправильная», цепочка не активируется, то есть значение сигнала на её выходе равно логическому нулю. Разрыв цепочки не допустим. Во втором случае конъюнкция не активируется, если хотя бы она параллельная цепочка «неправильная».

Выражения (2.24) или (2.25) описывают условия активирования соответствующей j-ой конъюнкции системы из m функций.

Для программирования значений m логических функций в ДНФ-LUT:

$$z_l = \bigvee_{j=1}^{k}(d_j \cdot h_{l.j}); l = 1, m. \qquad (2.26)$$

Таким образом конфигурирование осуществляется не по 2^n наборам функций n переменных, а более компактно, в случае, если количество конъюнкций в системе функций гораздо меньше 2^n.

ДНФ-LUT конфигурирование предполагается таким же, как и в обычном LUT, путём загрузки оперативной памяти ОЗУ (SRAM). На рисунке 2.15 представлена структура ДНФ-LUT, настройка конъюнкций загружается в SRAM, не указанную на рисунке 2.14:

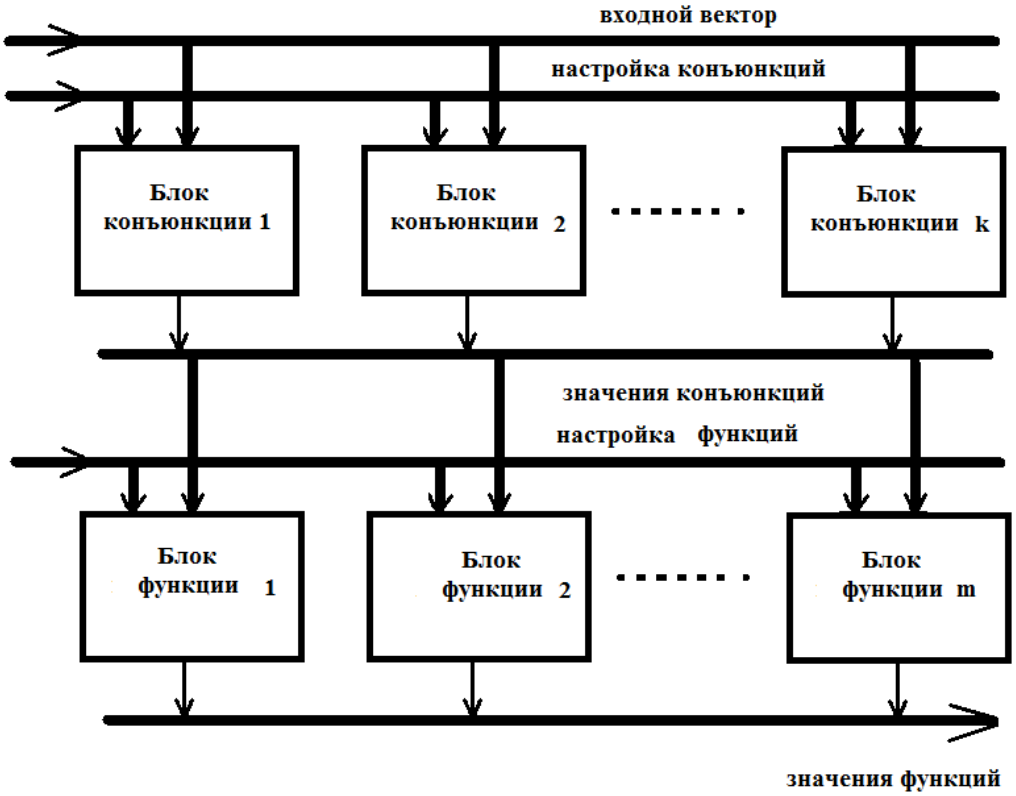


Рисунок 2.14. Логический элемент ДНФ-LUT

Таким образом, вместо загрузки значений таблицы истинности, программируются лишь значения конъюнкций длиной n, где n – число переменных m логических функций. Вхождения k конъюнкций в m функций также программируется настройкой функций. По заданному входному набору (вектору) n переменных блоки конъюнкций вычисляют значения k

конъюнкций, которые далее формируют «по ИЛИ» значения m логических функций. Рассмотрим внутреннюю структуру блоков конъюнкций и функций.

Предлагаемая структура блока переменной ДНФ-LUT [65,66] изображена на рисунке 2.15:

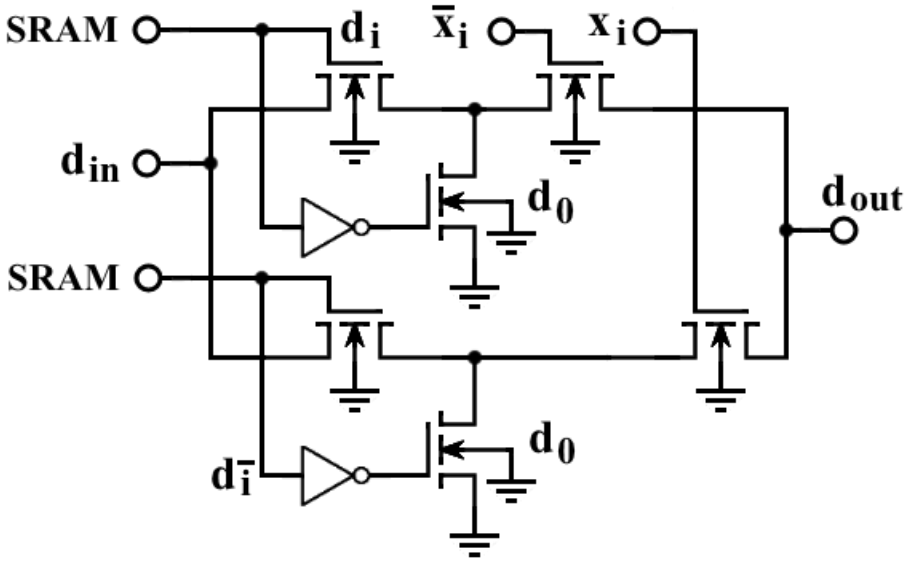


Рисунок 2.15. Структура блока одной переменной DNF-LUT

В ячейки статической оперативной памяти SRAM загружается настройка переменной. Если переменная входит в конъюнкцию без инверсии, то единица загружается в верхнюю ячейку SRAM, если переменная входит в конъюнкцию с инверсией, то в нижнюю. Если переменная несущественна, то единицы загружаются в обе ячейки SRAM. Таким образом, если переменная активирована «правильно», блок, изображённый на рисунке 2.15 пропускает сигнал логической единицы со входа (слева) на выход (справа). То же происходит при несущественности переменной, то есть при любом значении переменной. Если же активируется не та переменная, которая задана настройкой, с помощью инверторов и дополнительных передающих транзисторов обеспечивается подача на выход логического нуля – рисунок 2.16:

SRAM X	SRAM не X	На выходе 1	На выходе 0
1	0	При X	При не X
0	1	При не X	При X
1	1	В любом случае	-
0	0	Запрещено	Запрещено

Рисунок 2.16. Сигналы управления блоком переменной

На рисунке 2.17 изображена структура блока конъюнкции переменных ДНФ-LUT на основе последовательной цепочки блоков переменных:

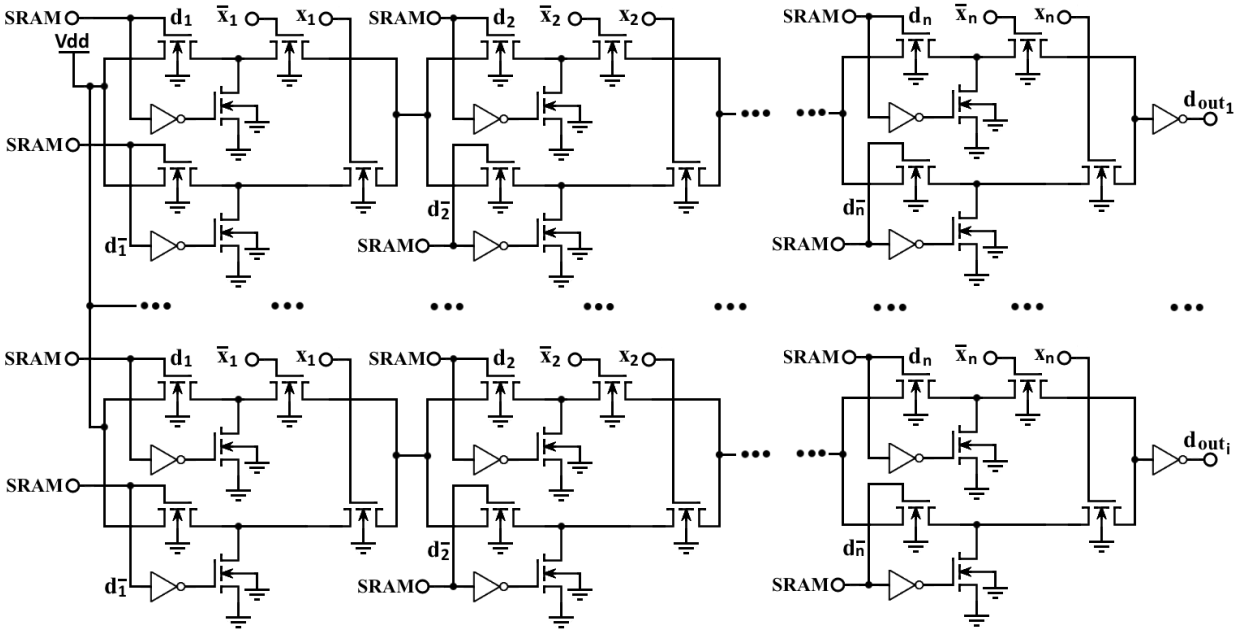


Рисунок 2.17 Структура блока конъюнкции на основе последовательной цепочки блоков переменных ДНФ-LUT

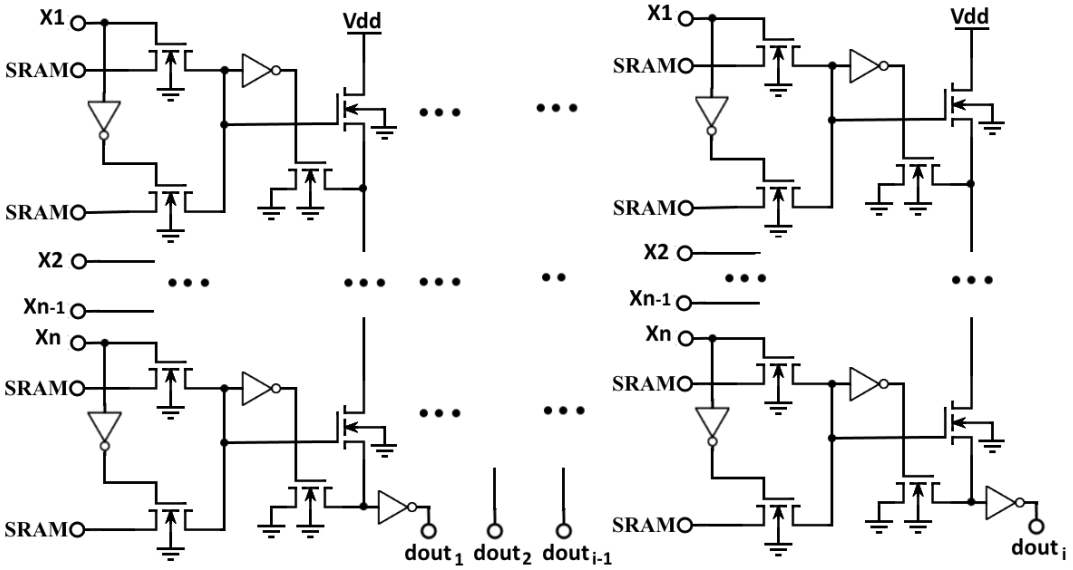


Рисунок 2.18 Структура блока конъюнкции на основе параллельной цепочки блоков переменных ДНФ-LUT

Если заданная конъюнкция равна единице, то на выходе z устанавливается логический ноль, иначе – единица. Здесь последовательная цепочка передающих транзисторов не управляет затворами других передающих транзисторов. Для соблюдения ограничений на число последовательно соединённых передающих транзисторов необходимо использовать инверторы в цепочке, изображённой на рисунке 2.18.

2.4 Выводы по главе 2

В главе представлены следующие основные научные и практические результаты.

1. Предложен усовершенствованный метод реализации систем логических функций в СДНФ отличающийся тем, что выполнен реверс дерева передающих транзисторов LUT, при этом ортогональность сигналов в ветвях дерева передающих транзисторов обеспечивается тремя способами: по каждому транзистору ветви, по выходам дерева, с помощью нагрузочных транзисторов. Предложенный метод позволяет реализовывать системы логических функций на одном ЛЭ с помощью необходимого количества блоков дизъюнкций конституент логической функции.

2. Предложен усовершенствованный метод построения адаптивного логического элемента ADC-LUT отличающийся тем, что в зависимости от настройки возможна реализация либо стандартного логического элемента, либо дешифратора для реализации систем логических функций.

3. Предложен усовершенствованный метод реализации систем логических функций в ДНФ на базе ПЛМ отличающийся тем, что выполнена адаптация к уровню передающих транзисторов, используемых в LUT и применено оригинальное кодирование конъюнкций ДНФ, обеспечивающее формирование значения конъюнкции.

4. Предлагаемые технические решения DC–LUT и ДНФ-LUT защищены патентами РФ. Для дальнейшего исследования работоспособности необходимо разработать модели и выполнить моделирование.

Глава 3. Моделирование разработанных устройств для реализации систем логических функций в ПЛИС - FPGA

3.1. Моделирование логического элемента – LUT

3.1.1. Статическое моделирование логического элемента - LUT

Разработаем статическую модель существующего логического элемента LUT на две входные переменные (рисунок 3.1). Выполним моделирование в системе схемотехнического моделирования NI Multisim фирмы National Instruments Electronics Workbench Group [67]. Дерево передающих транзисторов формируют транзисторы VT1-VT6. Ключи SA1-SA2 моделируют ячейки памяти SRAM, в которые загружаются значения таблицы истинности логической функции. Для примера ячейки SRAM сконфигурированы на реализацию логической функции «исключающее ИЛИ». Транзисторы VT7-VT14 образуют 4 инвертора, которые реализуют инвертирование сигналов с ячеек памяти SRAM. Ключи SA5 и SA6 моделируют входные переменные X1 и X2 ЛЭ LUT. Транзисторы VT15-VT18 формируют 2 инвертора входных переменных, которые реализуют инверсию входного набора. Транзисторы VT19-VT20 формируют выходной инвертор LUT.

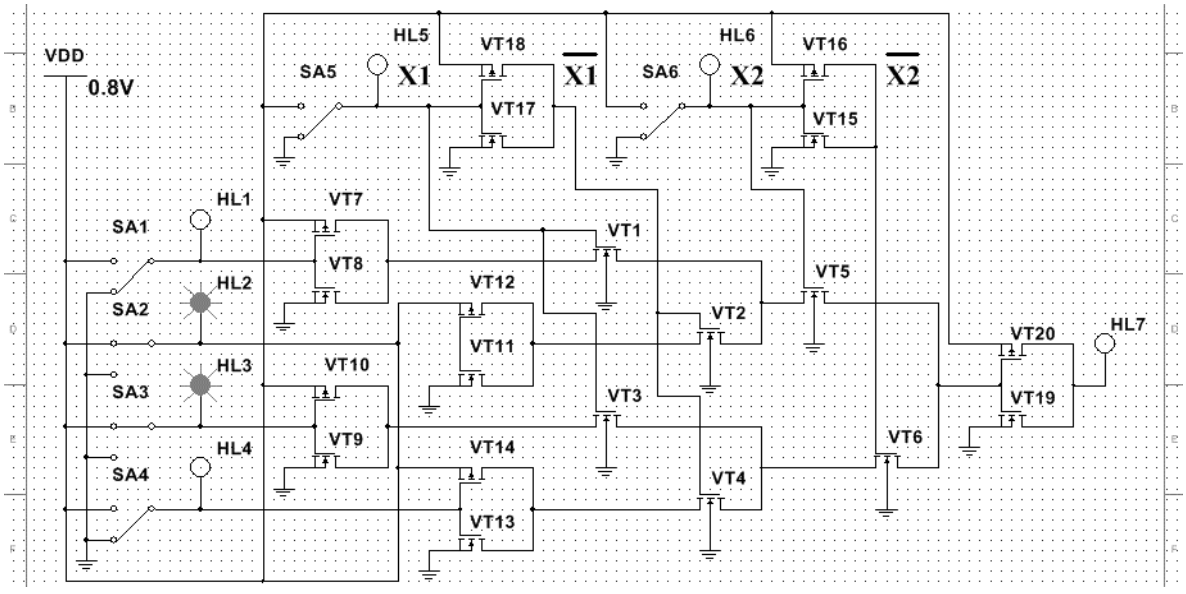


Рисунок 3.1. Моделирование LUT на две переменные реализующего функцию «исключающее ИЛИ», входной набор X1=0, X2=0

Пусть подается входной набор X1=0, X2=0, который активирует транзисторы VT4 и VT6 и тем самым происходит передача логического нуля с ключа SA4 на выход LUT. Транзисторы VT3 и VT5 закрыты, так как на затворы с X1 и X2 приходит логический ноль, и тем самым активация соответствующих ветвей не происходит.

При входном наборе X1=1; X1=1 происходит активация ветви, состоящей из транзисторов VT1 и VT5 и передачи логического нуля с ключа SA1 – рисунок 3.2.

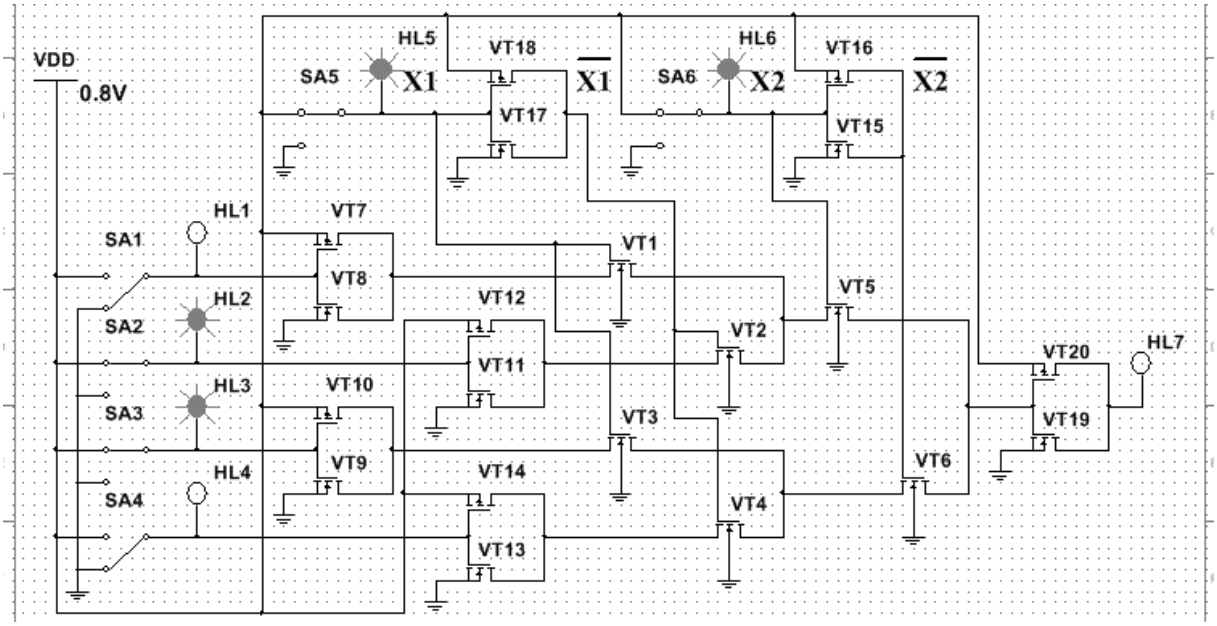


Рисунок 3.2. Моделирование LUT на две переменные реализующего функцию «исключающее ИЛИ», входной набор X1=1, X2=1

В случае входного набора X1=1, X2=0 происходит активация ветви, состоящей из транзисторов VT3 и VT6, и передача логической единицы с ключа SA3, поэтому светодиод на выходе инвертора светится – рисунок 3.3.

Аналогично схема работает при X1=0; X1=1 происходит активация ветви, состоящей из транзисторов VT2 и VT5, и передача логической единицы с ключа SA2 – рисунок 3.4.

Таким образом, моделирование LUT на две входные переменные показывает [68], что при подаче соответствующего входного набора X1 и X2 происходит активация одной ветви дерева и передача по ней соответствующего значения таблицы истинности из ячеек памяти SRAM на

выход LUT и тем самым реализуется одна логическая функция. Моделирование подтверждает не способность существующего LUT реализовывать системы функций, так как для реализации уже 2 логических функций относительно одного входного набора потребуется дополнительный LUT [68]. Для дальнейшей оценки характеристик быстродействия и потребления существующего LUT целесообразно провести динамическое моделирование.

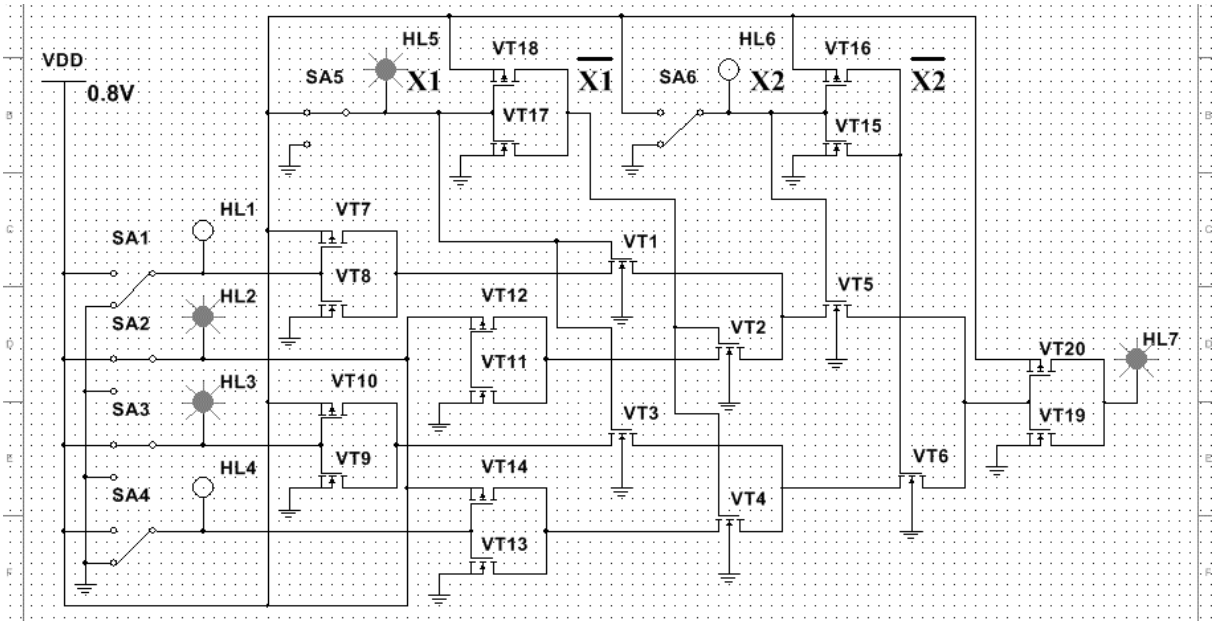


Рисунок 3.3 Моделирование LUT на две переменные реализующего функцию «исключающее ИЛИ», входной набор X1=1, X2=0

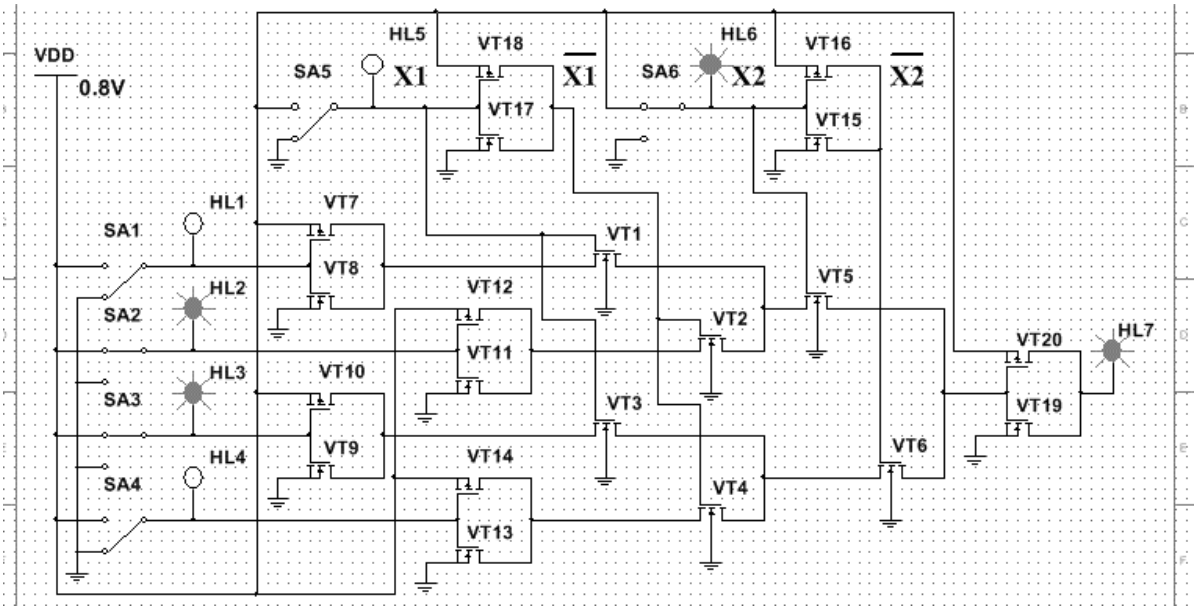


Рисунок 3.4 Моделирование LUT на две переменные реализующего функцию «исключающее ИЛИ», входной набор X1=0, X2=1

3.1.2. Динамическое моделирование логического элемента - LUT

Разработаем модель LUT на 2 входные переменные для оценки динамических параметров [68] – рисунок 3.5.

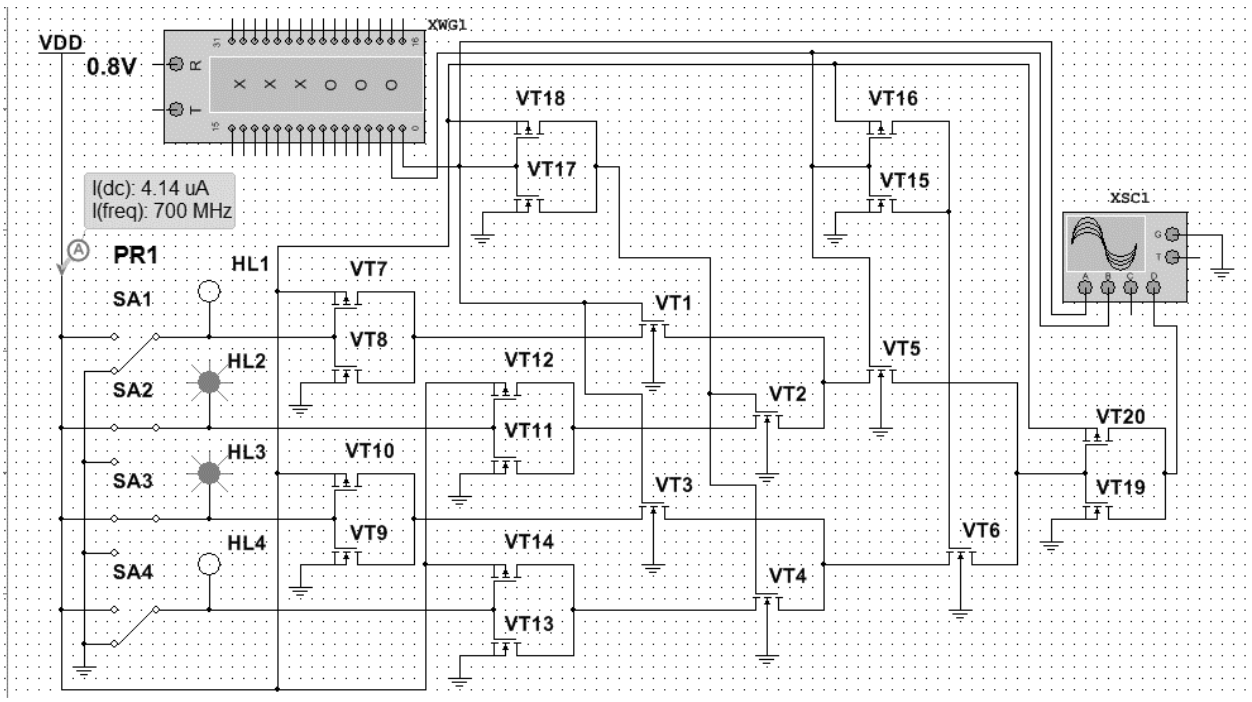


Рисунок 3.5. Модель 2-LUT (на 2 переменные)

Сигналы значений входных переменных подаются с помощью генератора наборов XWG1 в коде Грея. Для оценки тока потребления к источнику питания подключаем зонд PR1. Для анализа выхода LUT используем четырехканальный осциллограф XSC1.

Получим осциллограмму работы 2-LUT на частоте 700 МГц (Spice модель транзисторов BSIM4.8 технология 65нм), конфигурированного ключами SA1-SA4 на вычисление логической функции «исключающее ИЛИ» («Сумма по модулю два») - рисунок 3.6. Наблюдаем формирование логической единицы (нижняя осциллограмма) в случае разницы логических уровней входов (два верхних сигнала осциллограммы на рисунке 3.6). С помощью осциллографа получим значения задержки сигнала - рисунок 3.7. Аналогично получим значения задержки и ток потребления для LUT от 1 до 8 входных переменных, которые целесообразно использовать для нахождения решения поставленной задачи диссертационного исследования.

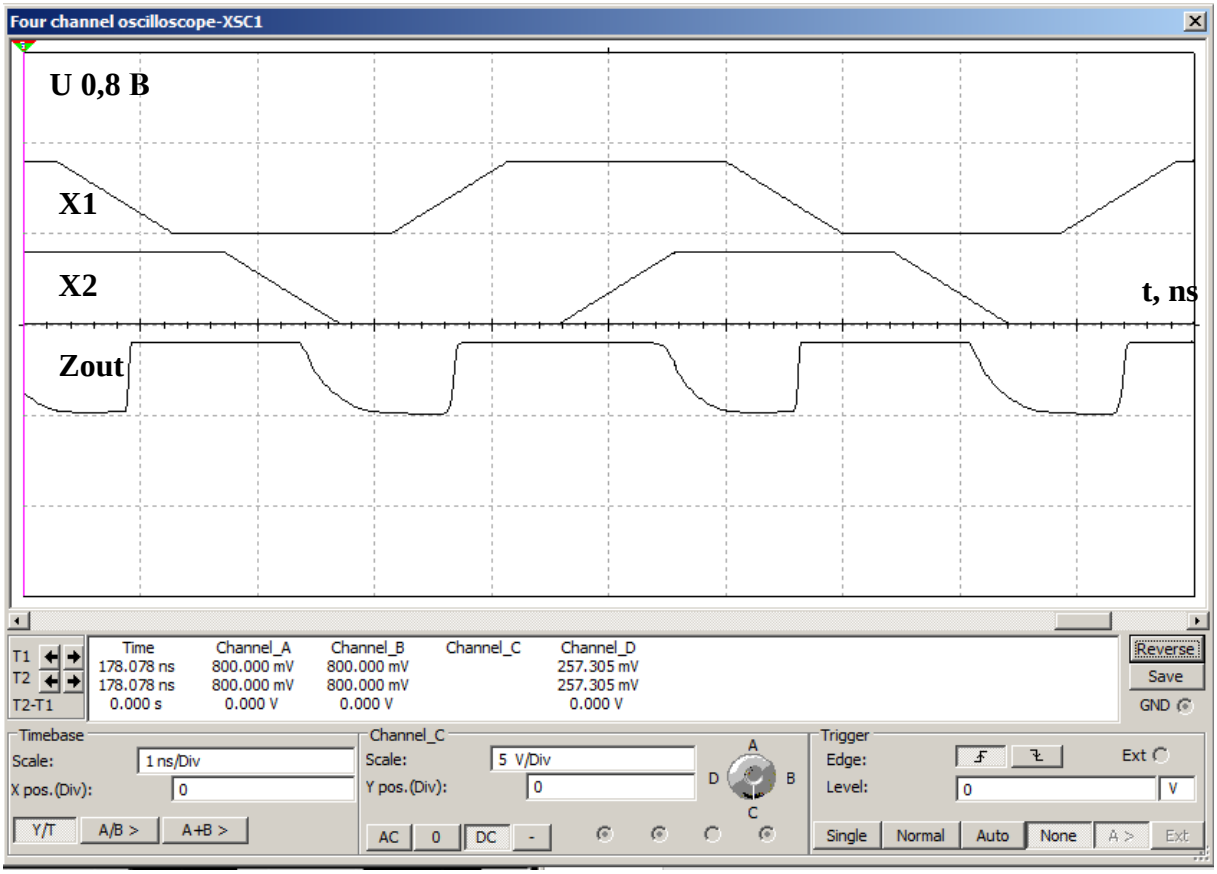


Рисунок 3.6. Осциллограмма работы модели 2-LUT (на 2 переменные)

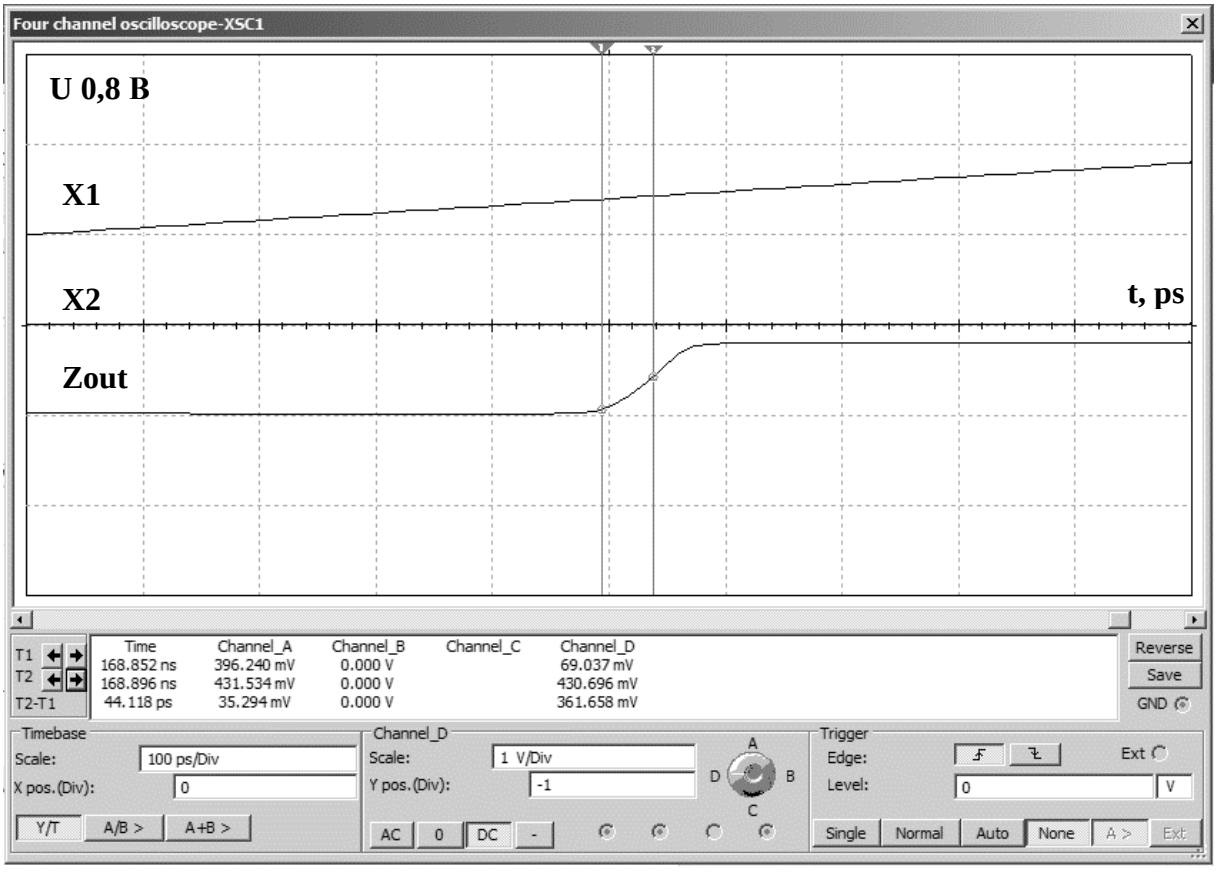


Рисунок 3.7. Значение задержки с осциллограммы модели 2-LUT (на 2 переменные) на напряжении 0,8 В при частоте 700МГц

3.2. Моделирование логического элемента – дешифратора DC-LUT

3.2.1 Статическое моделирование логического элемента – DC-LUT-O

Разработаем статическую модель обратного дерева передающих транзисторов, используемого в LUT [68]. Выполним моделирование DC-LUT-O на 2 входные переменные с цепочками ортогональности по каждой ветви (рисунок 3.8) в системе схемотехнического моделирования NI Multisim фирмы National Instruments Electronics Workbench Group [67].

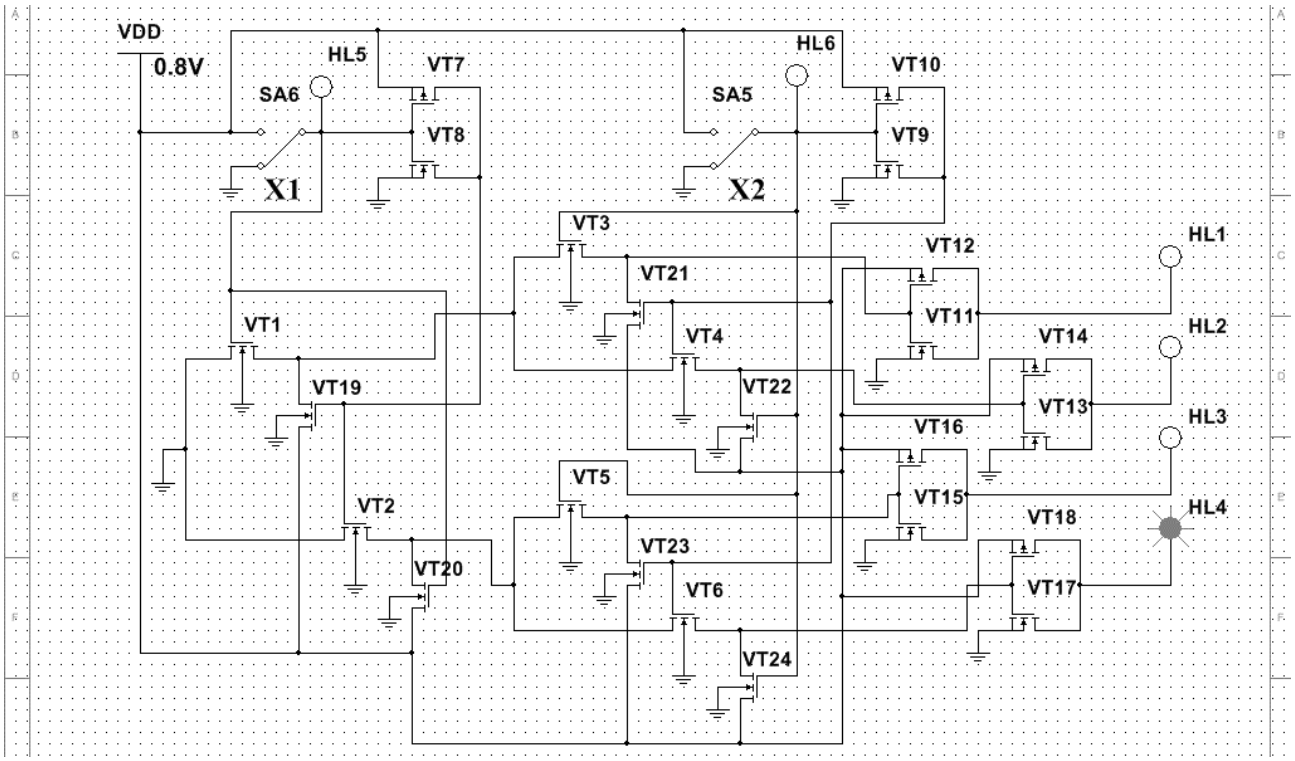


Рисунок 3.8 Моделирование DC-LUT-O на две переменные, входной набор X1=0, X2=0

Пусть подается входной набор X1=0; X2=0, который активирует транзисторы VT2, VT6, и на вход инвертора VT17-VT18 передается логический ноль, с выхода которого формируется логическая единица, светодиод HL4 на выходе инвертора цепочки светится.

Ортогональность сигналов обеспечивается цепочками ортогональности, транзисторы VT19–VT24. При входном наборе X1=0; X2=0 активируется цепочка из транзисторов VT2, VT6 так как на затворы этих транзисторов приходит логическая единица и по активированной ветви передается

логический ноль, а транзисторы ортогональности VT20 и VT24 которые так же подключены к ветви не активируются, так как на затворы приходит логический ноль. Другие транзисторы ортогональности VT19, VT22 активированы, так как на затворы подается логическая единица и на входы соответствующих инверторов VT11-VT16 гарантированно поступает логическая единица и тем самым с выходов инверторов ветвей формируется логический ноль, светодиоды HL1 – HL3 не светятся.

Аналогично моделируется другие входные наборы: X1=0, X2=1 рисунок 3.9 ; X1=1; X2=0 рисунок 3.10; X1=1, X2=1 рисунок 3.11.

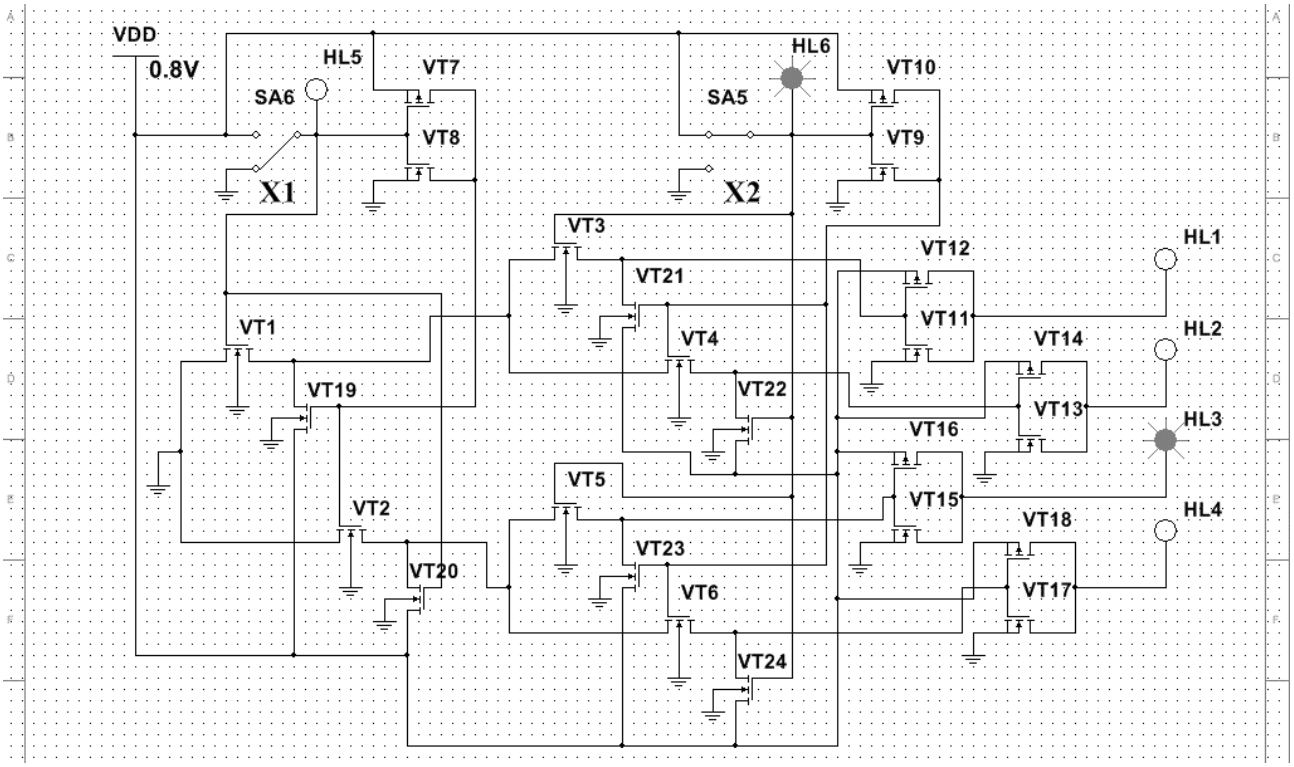


Рисунок 3.9 Моделирование DC-LUT-O на две переменные, входной набор X1=0, X2=1

Моделирование показывает, что дешифрация сигнала производится, предлагаемый метод DC-LUT-O работоспособен. Аналогично можно убедиться в том, что в статическом режиме правильно функционирует 2-DC LUT-O, 3-DC LUT-O, 4-DC LUT-O, построенный из двух 3-DC LUT и одного 1-DC LUT.

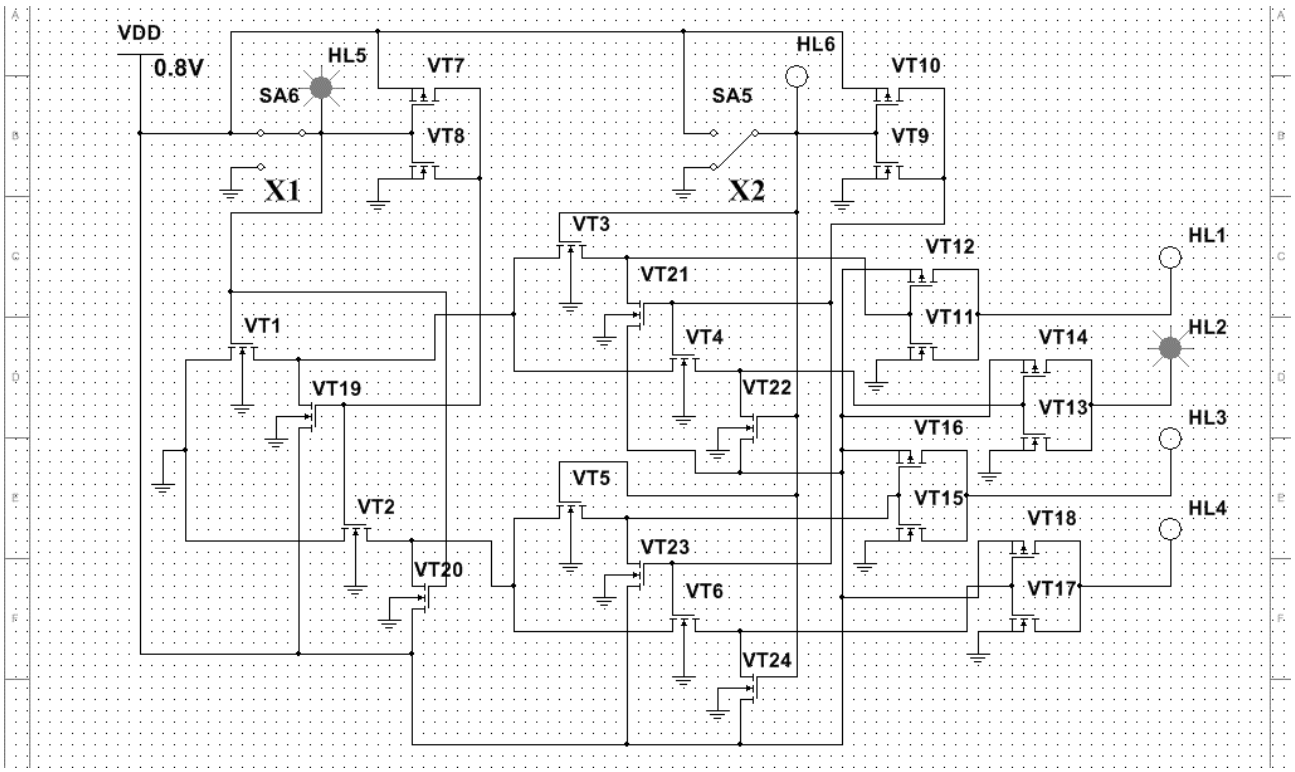


Рисунок 3.10 Моделирование DC-LUT-O на две переменные, входной набор X1=1, X2=0

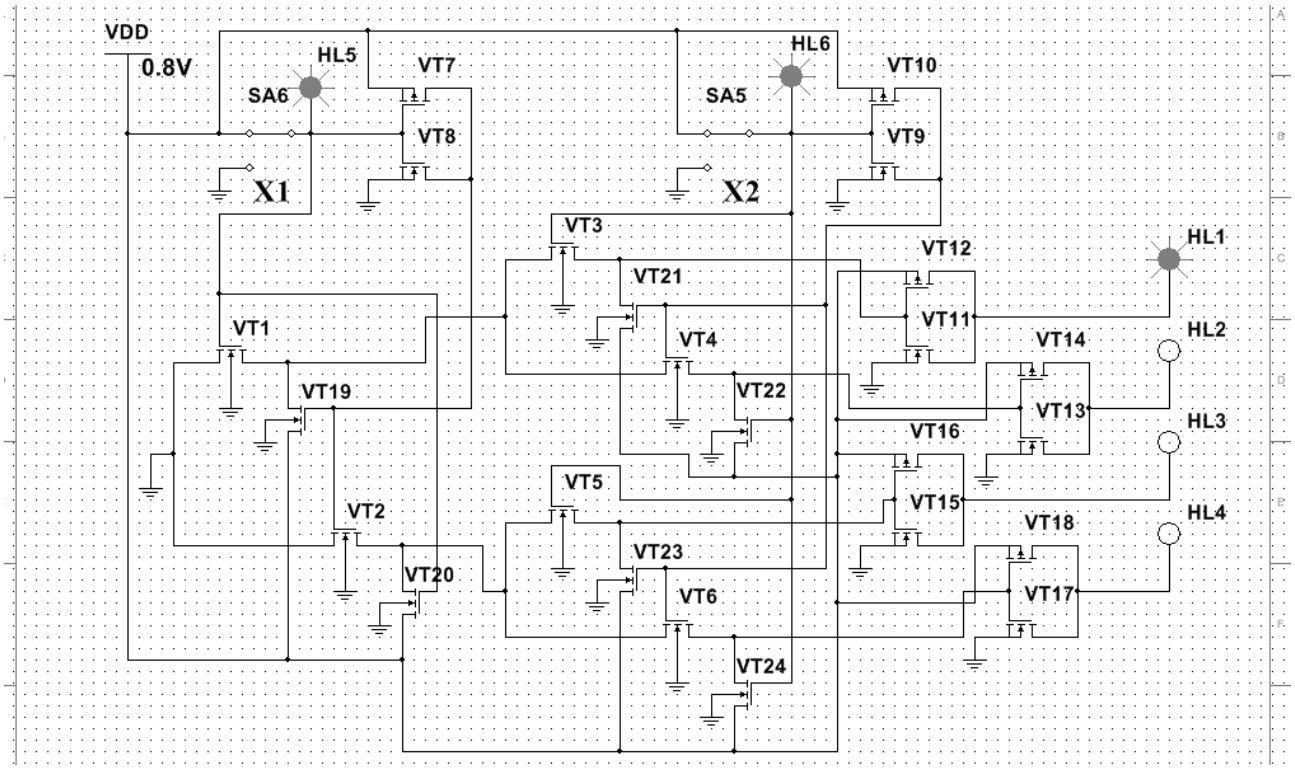


Рисунок 3.11 Моделирование DC-LUT-O на две переменные, входной набор X1=1, X2=1

3.2.2 Статическое моделирование логического элемента – DC-LUT-R

Разработает модель DC-LUT-R на 2 входные переменные с нагрузочными транзисторами, подключенными по выходам цепей (рисунок 3.12) в системе схемотехнического моделирования NI Multisim фирмы National Instruments Electronics Workbench Group [67]. Выполним моделирование для разных входных наборов [68]. Дешифратор DC-LUT-R работает аналогично, как DC-LUT-O для входного набора X1=0, X2=0 рисунок 3.12; для входного набора X1=0, X2=1 рисунок 3.13; для входного набора X1=1; X2=0 рисунок 3.14; для входного набора X1=1, X2=1 рисунок 3.15. Дешифратор DC-LUT-R отличается от DC-LUT-O реализацией ортогональности, путем подключения по выходам ветвей нагрузочных транзисторов VT19-VT22, которые в случае не активации ветви дерева «обрыва» подтягивают на вход инвертора логическую единицу.

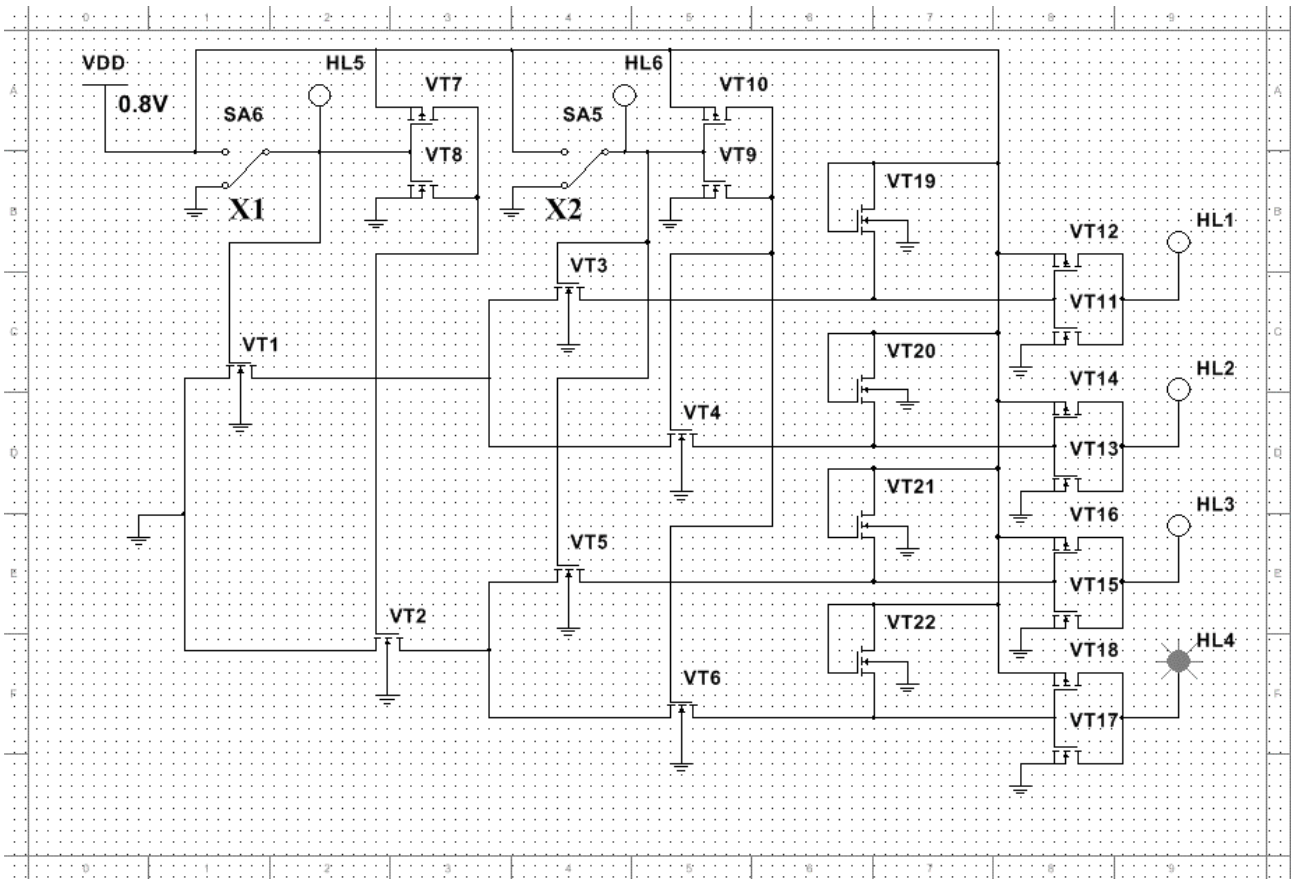


Рисунок 3.12 Моделирование DC-LUT-R на две переменные, входной набор X1=0, X2=0

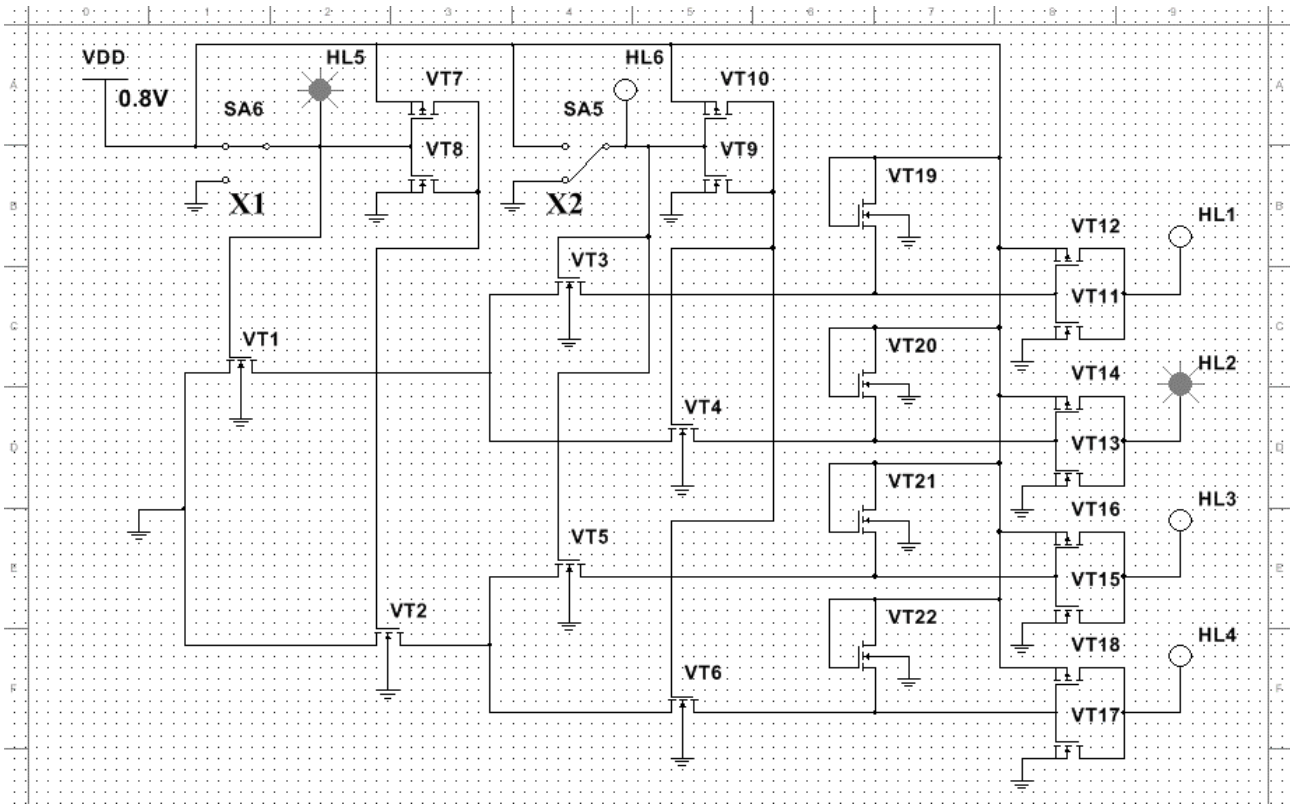


Рисунок 3.13 Моделирование DC-LUT-R на две переменные, входной набор X1=1, X2=0

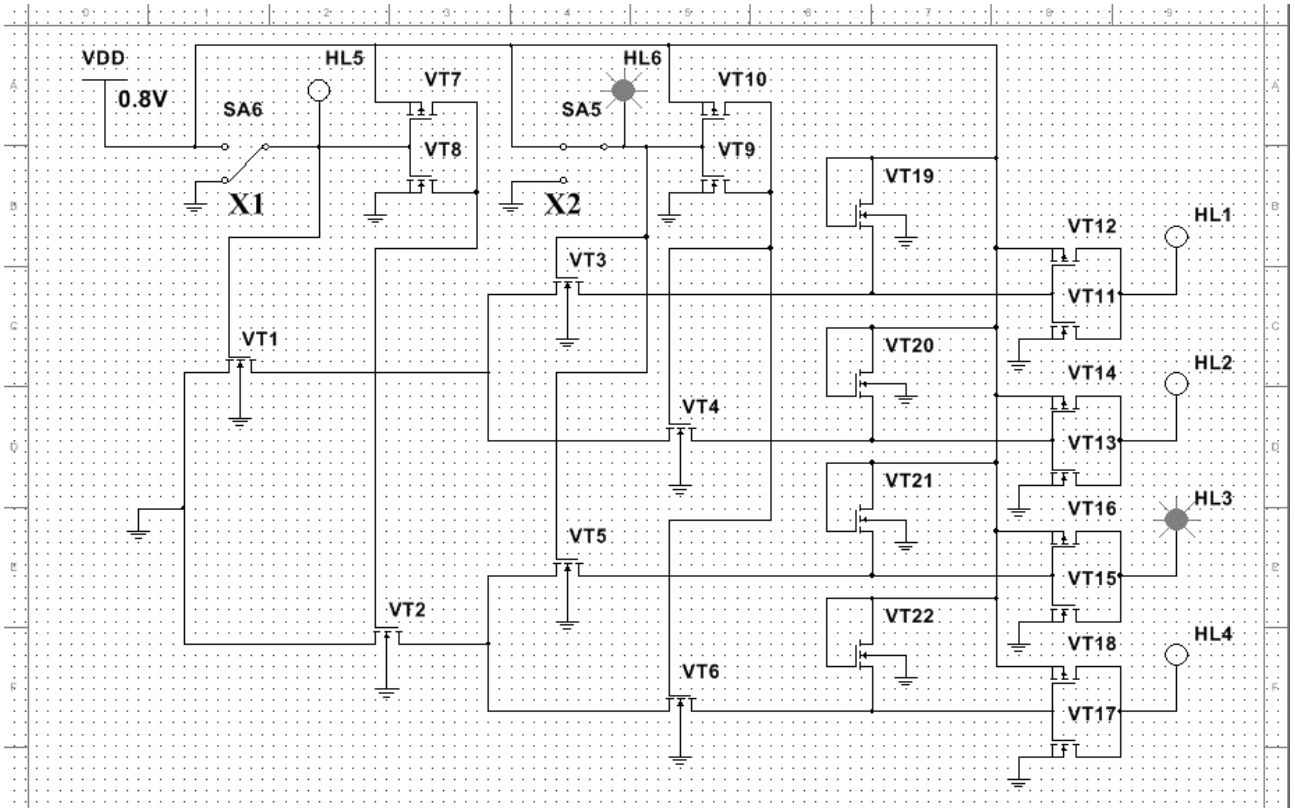


Рисунок 3.14 Моделирование DC-LUT-R на две переменные, входной набор X1=0, X2=1

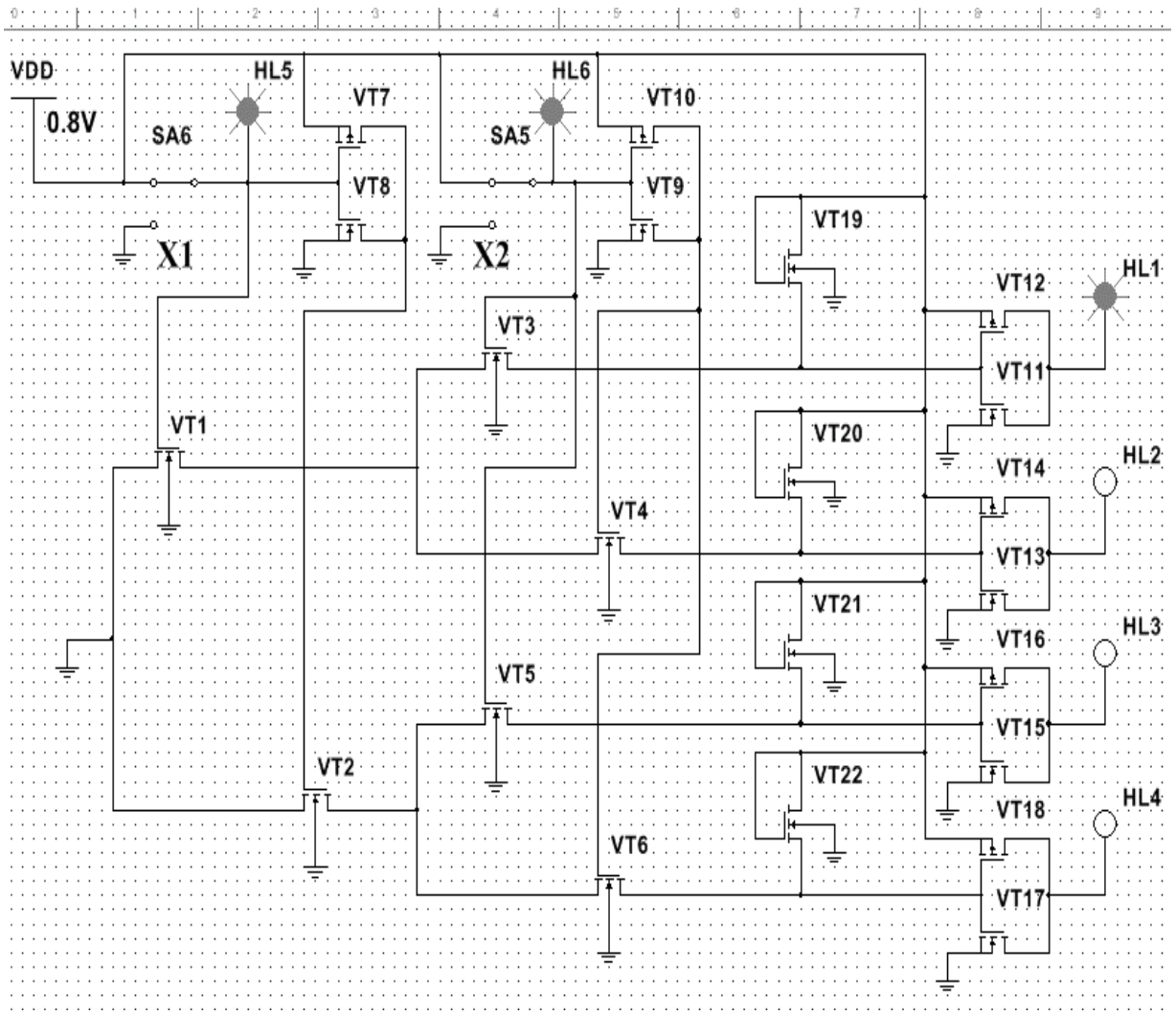


Рисунок 3.15 Моделирование DC-LUT-R на две переменные, входной набор X1=1, X2=1

3.2.3 Статическое моделирование логического элемента – DC-LUT-BKN

Разработаем модель DC-LUT-BKN на 2 входные переменные с блоками конституент нуля подключенными по выходам цепей (рисунок 3.16) в системе схемотехнического моделирования NI Multisim фирмы National Instruments Electronics Workbench Group [67]. Выполним моделирование для разных входных наборов [68]. Дешифратор DC-LUT-DKN работает аналогично, как DC-LUT-O и DC-LUT-R для входного набора X1=0, X2=0 рисунок 3.16; X1=0, X2=1 рисунок 3.17; X1=1; X2=0 рисунок 3.18; X1=1, X2=1 рисунок 3.19. Дешифратор DC-LUT-BKN отличается от DC-LUT-O и DC-LUT-R реализацией ортогональности сигналов. В DC-LUT-BKN по выходам ветвей, подключаются блоки конституент нуля, транзисторы VT19-VT26, которые

активируются при соответствующем наборе и подтягивают логическую единицу к не активированной ветви. Для примера разберем работу одного блока конституент нуля, подключенного к цепочке транзисторов VT2, VT6. При входном наборе X1=0, X2=0 с инверторов VT7-VT10 на затворы транзисторов (VT2, VT6) поступает логическая единица и тем самым активируется цепочка этих транзисторов, а на затворы подключенного к этой ветви блока конституент нуля (VT19, VT20) поступает логический ноль и тем самым блок конституент нуля не подтягивает логическую единицу на выход ветви. При других входных наборах (X1=1, X2=0; X1=0, X2=1; X1=1, X2=1) цепочка транзисторов VT2, VT6 не активируется, но один из транзисторов VT19 или VT20, блока конституент нуля будет активен и ко входу инвертора VT18-VT17 подтягивается логическая единица - рисунок 3.16.

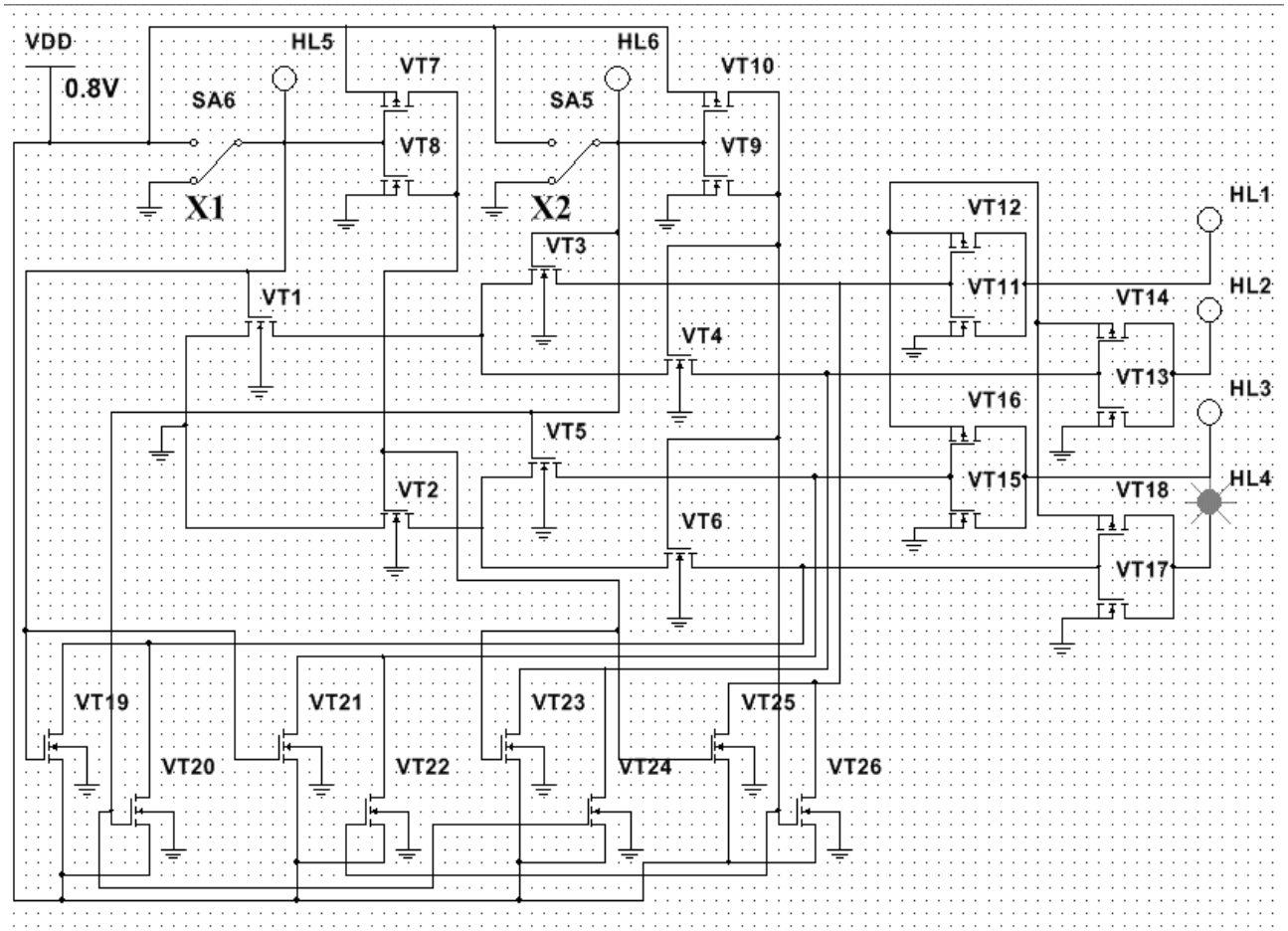


Рисунок 3.16 Моделирование DC-LUT-BKN на две переменные, входной набор X1=0, X2=0

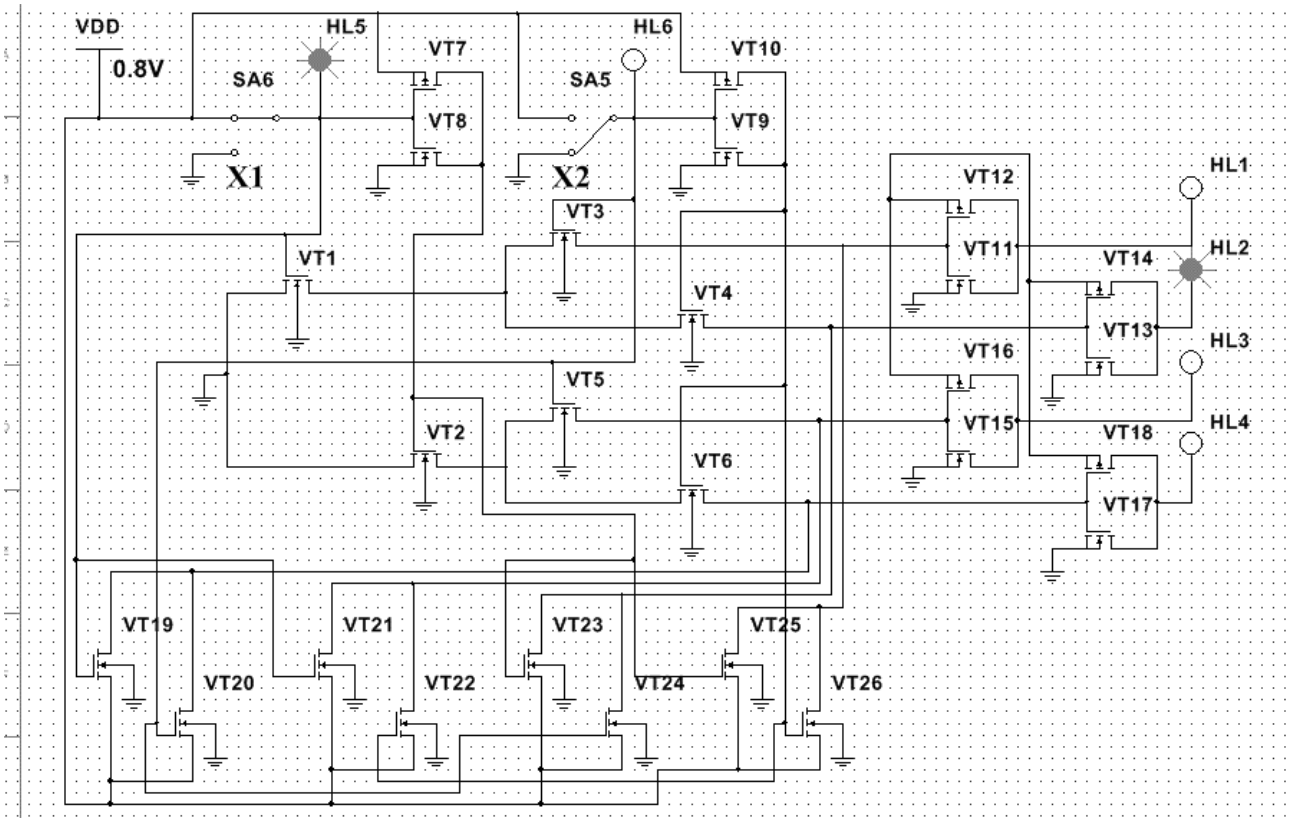


Рисунок 3.17 Моделирование DC-LUT-BKN на две переменные, входной набор X1=1, X2=0

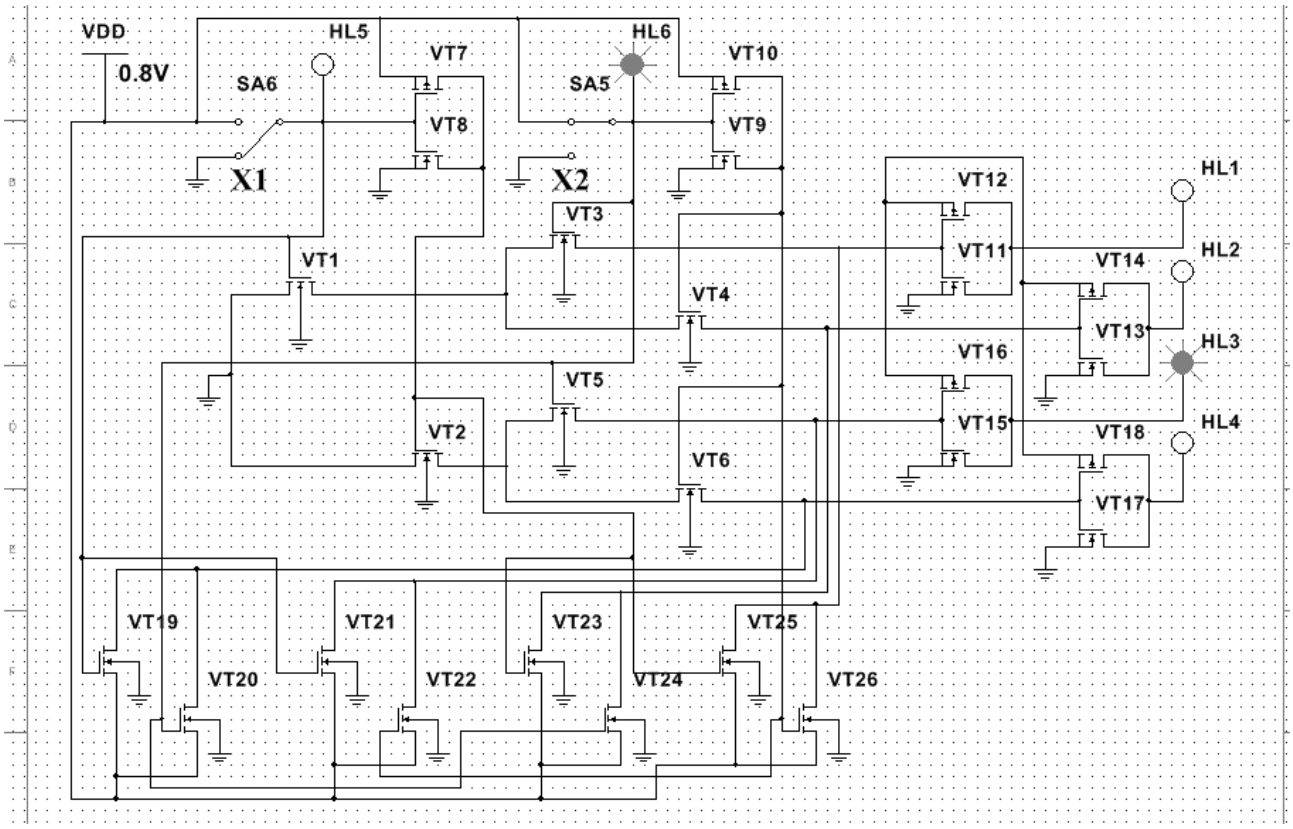


Рисунок 3.18 Моделирование DC-LUT-BKN на две переменные, входной набор X1=0, X2=1

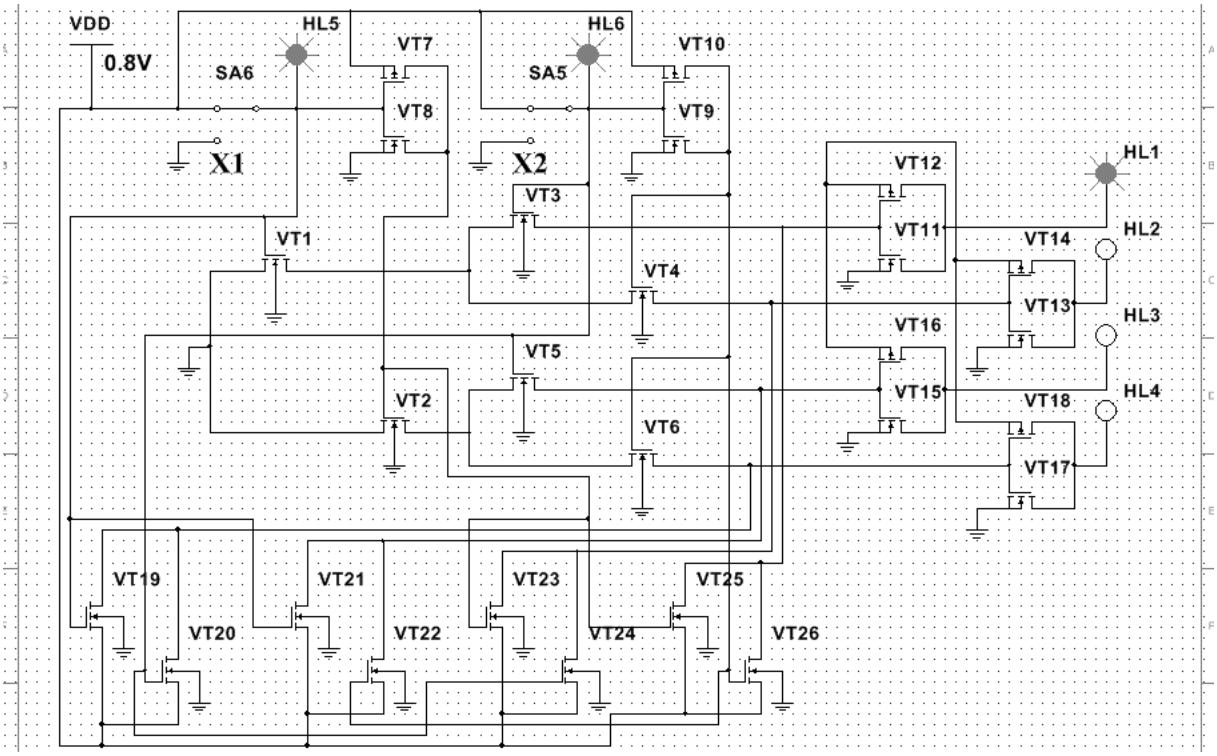


Рисунок 3.19 Моделирование DC-LUT-BKN на две переменные, входной набор X1=1, X2=1

3.2.4. Динамическое моделирование логического элемента – DC-LUT-O

Разработаем модель предлагаемого ЛЭ DC-LUT-O на 2 входные переменные – рисунок 3.20 и оценим динамические параметры [68].

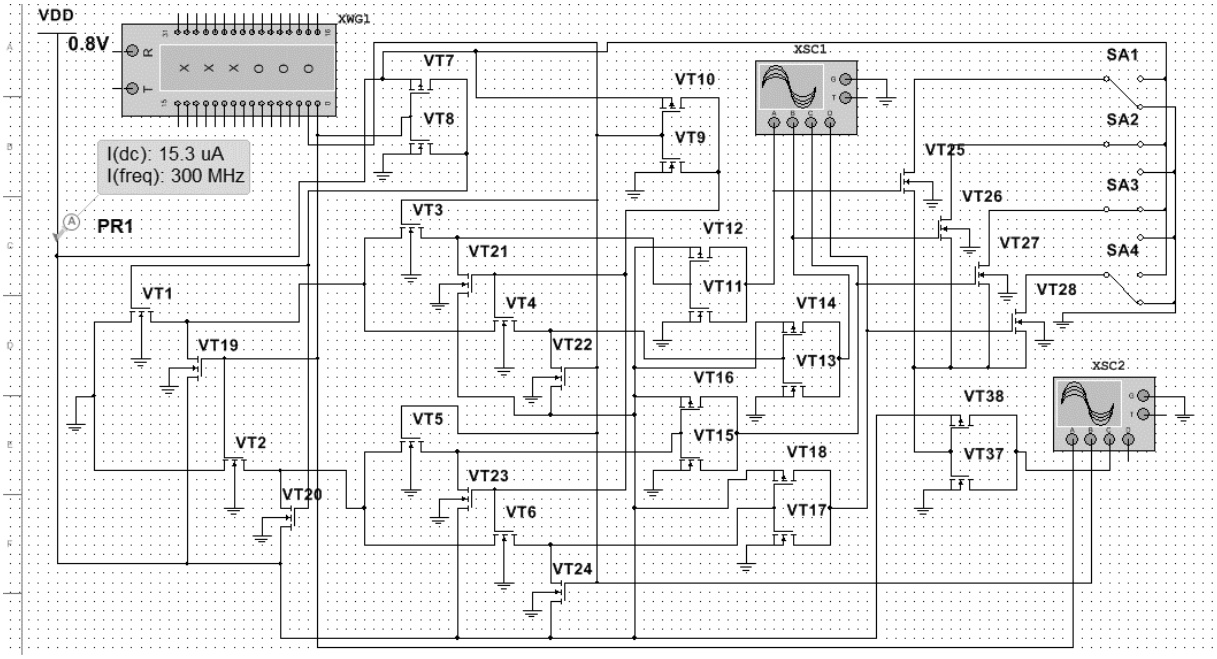


Рисунок 3.20. Модель 2-DC-LUT-O (на 2 переменные)

Сигналы значений входных переменных подаются с помощью генератора наборов XWG1 в коде Грея. Для оценки тока потребления к источнику питания подключаем зонд PR1. Для анализа выхода DC-LUT-O используем четырехканальный осциллограф XSC2. Реализация логических функций осуществляется с помощью блока настройки, транзисторы VT25-VT28, который с помощью конфигурации ключами SA1-SA4 можно настроить на вычисление любой логической функции, в данном случае показана настройка на реализацию логической функции «исключающее ИЛИ» («Сумма по модулю два»). Комбинируя ЛЭ DC-LUT-O необходимым количеством блоков настройки и конфигурируя их на реализацию различных логических функций тем самым реализуем системы логических функций на одном ЛЭ DC-LUT-O.

Получим осциллограмму работы DC-LUT-O на частоте 300 МГц (Spice модель транзисторов BSIM4.8 технологии 65нм) - рисунок 3.21.

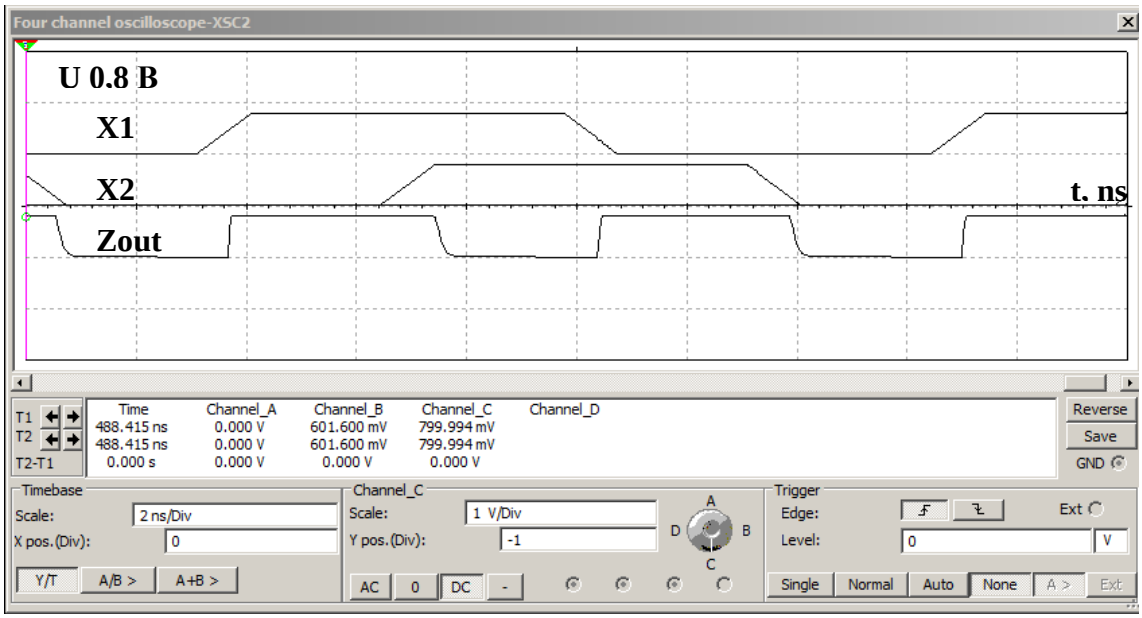


Рисунок 3.21. Осциллограмма работы модели 2-DC-LUT-O (на 2 переменные)

Наблюдаем формирование логической единицы (нижняя осциллограмма) в случае разницы логических уровней входов (два верхних сигнала осциллограммы на рисунке 3.21). С помощью осциллографа получим значения задержки сигнала – рисунок 3.22. Аналогично получим значения

задержки и ток потребления для DC-LUT-O от 1 до 8 входных переменных, полученные значения целесообразно использовать для нахождения решения поставленной задачи диссертационного исследования.

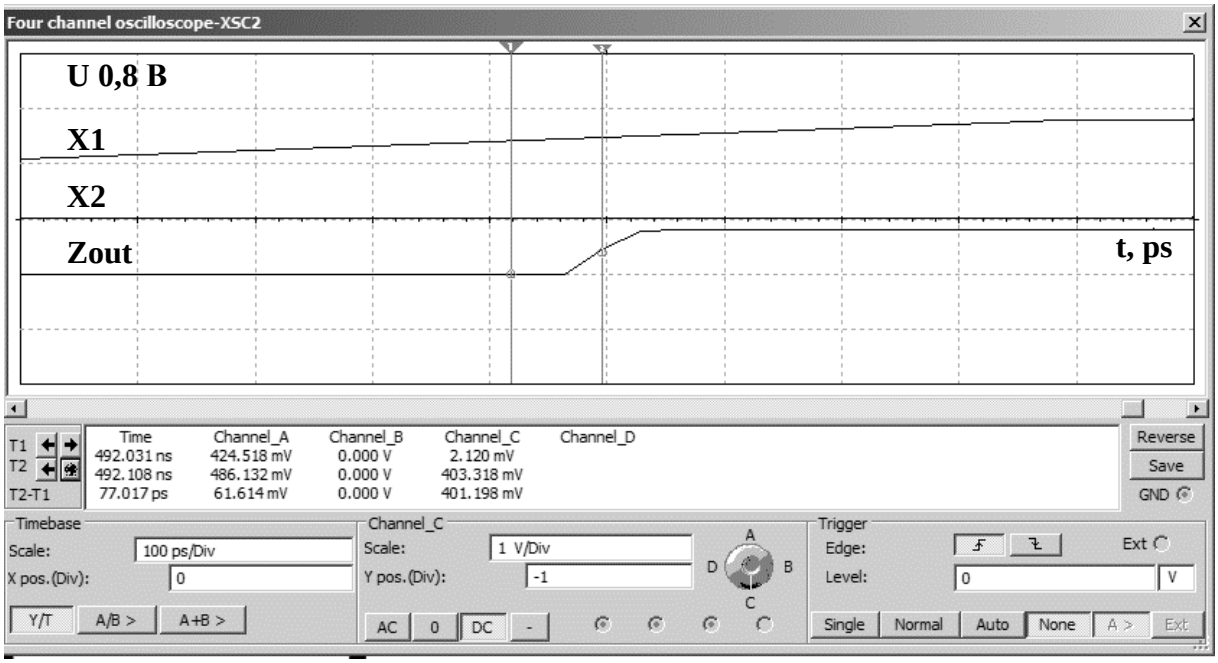


Рисунок 3.22. Значение задержки с осциллограммы модели 2-DC-LUT-O (на 2 переменные) на напряжении питания 0,8 В при частоте 300МГц

3.2.5. Динамическое моделирование логического элемента – DC-LUT-R

Разработаем модель предлагаемого ЛЭ на 2 входные переменные 2-DC-LUT-R – рисунок 3.23 и оценим динамические параметры [68].

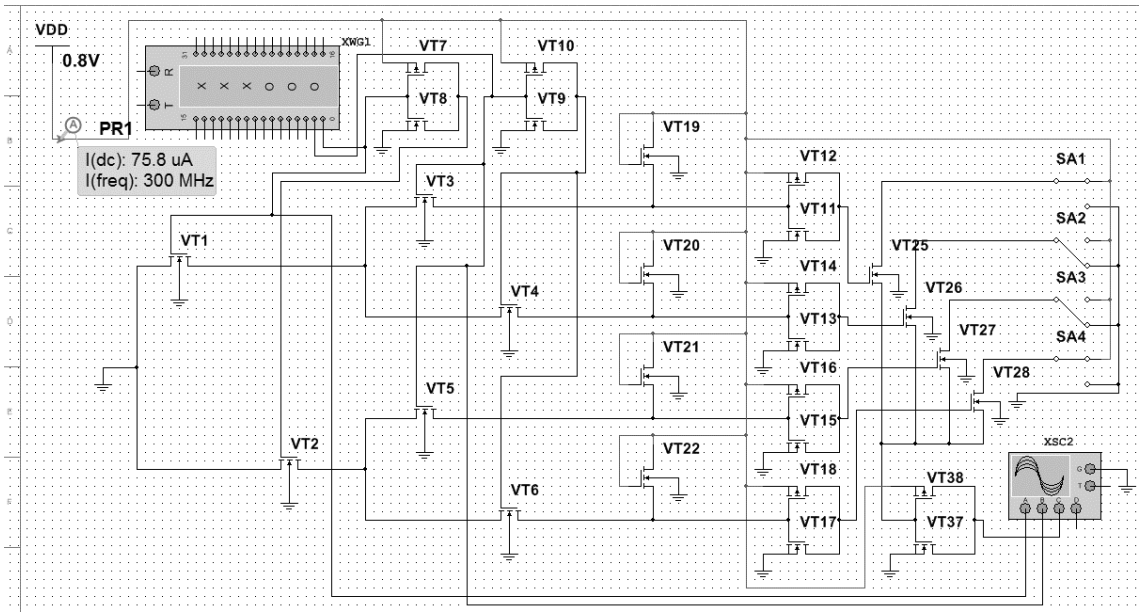


Рисунок 3.23. Модель DC-LUT-R (на 2 переменные)

Сигналы значений входных переменных подаются с помощью генератора наборов XWG1 в коде Грея. Для оценки тока потребления к источнику питания подключаем зонд PR1. Для анализа выхода DC-LUT-R используем четырехканальный осциллограф XSC2. Реализация логических функций осуществляется с помощью блока настройки, транзисторы VT25-VT28, который с помощью конфигурации ключей SA1-SA4 можно настроить на вычисление любой логической функции, в данном случае показана настройка на реализацию логической функции «исключающее ИЛИ» («Сумма по модулю два»). Комбинируя ЛЭ DC-LUT-R необходимым количеством блоков настройки и конфигурируя их на реализацию различных логических функций, реализуем системы логических функций на одном ЛЭ DC-LUT-R.

Получим осциллограмму работы DC-LUT-R на частоте 300 МГц (Spice модель транзисторов BSIM4.8 технологии 65нм) - рисунок 3.24.

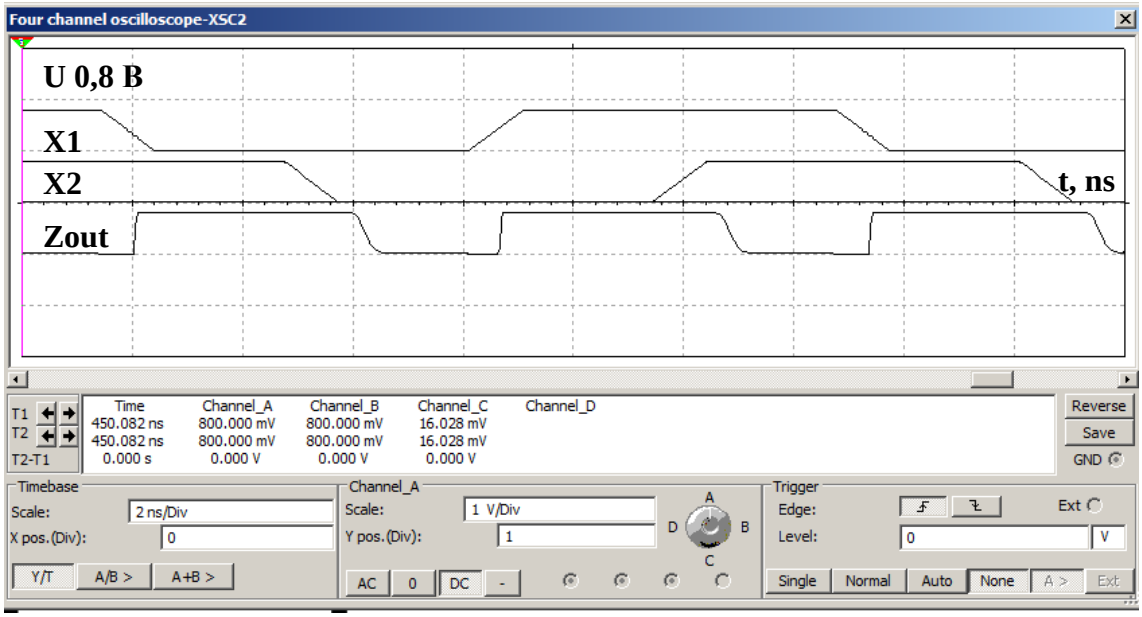


Рисунок 3.24. Осциллограмма работы модели 2-DC-LUT-R (на 2 переменные)

Наблюдаем формирование логической единицы (нижняя осциллограмма) в случае разницы логических уровней входов (два верхних сигнала осциллограммы на рисунке 3.24). С помощью осциллографа получим значения задержки сигнала – рисунок 3.25. Аналогично получим значения задержки и ток потребления для DC-LUT-R от 1 до 8 входных переменных,

полученные значения будут использованы для нахождения решения поставленной задачи диссертационного исследования.

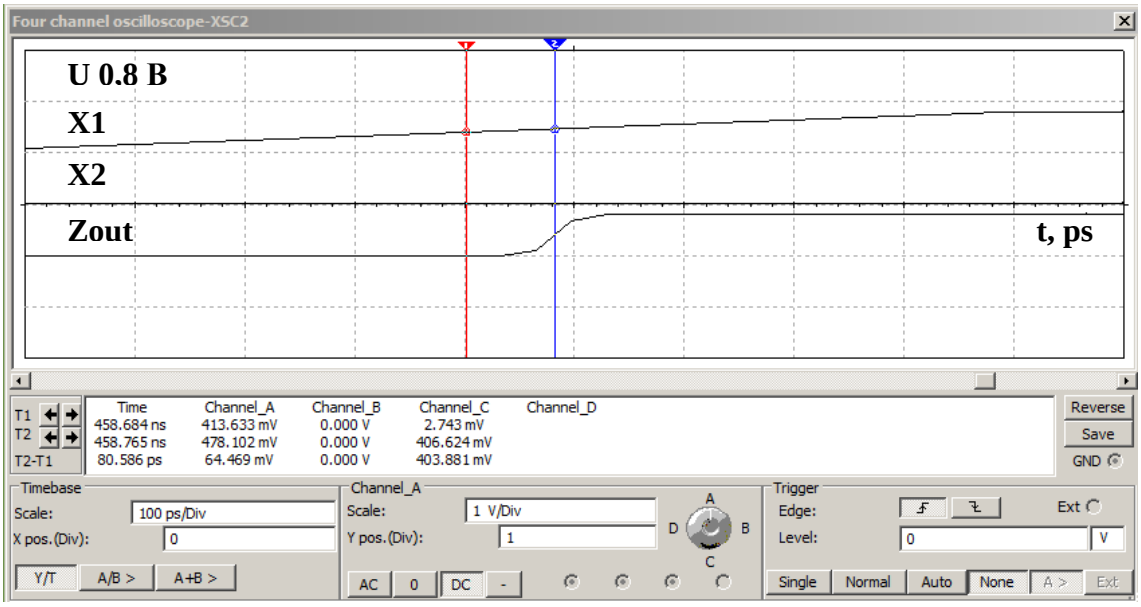


Рисунок 3.25. Значение задержки с осциллограммы модели 2-DC-LUT-R (на 2 переменные) на напряжении питания 0,8 В при частоте 700МГц

3.2.6. Динамическое моделирование логического элемента – DC-LUT-BKN

Разработаем модель предлагаемого ЛЭ на 2 входные переменные 2-DC-LUT-BKN – рисунок 3.26 и оценим динамические параметры [68].

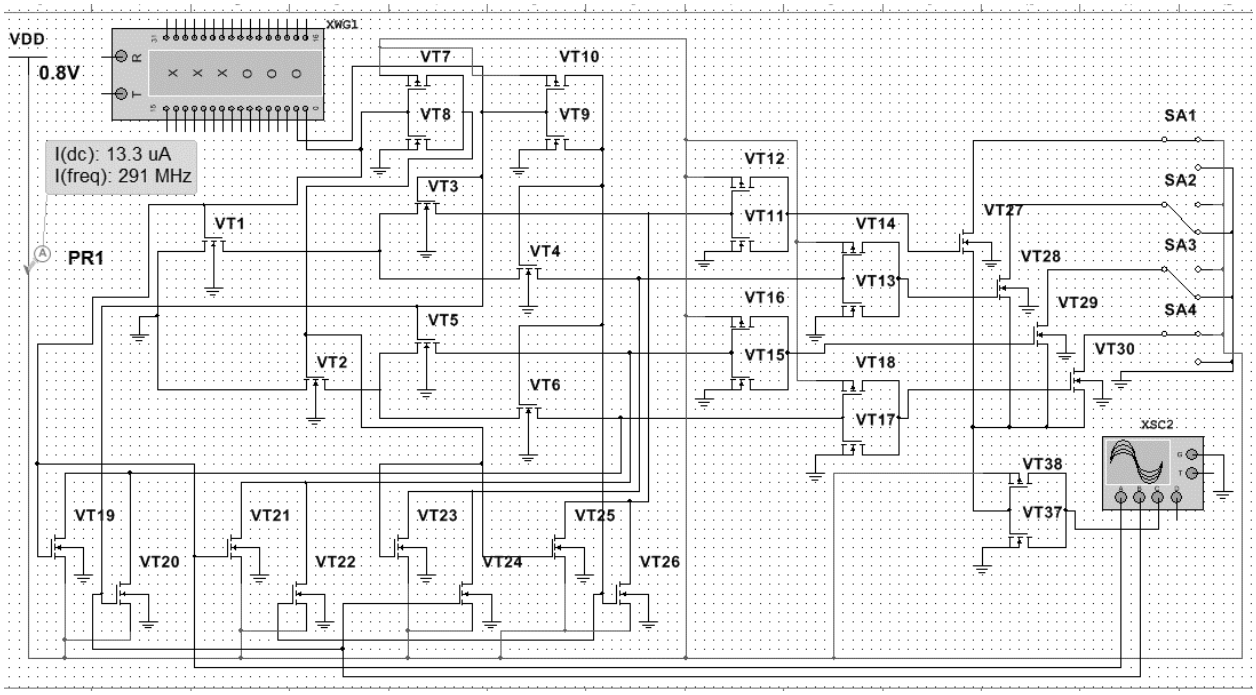


Рисунок 3.26. Модель DC-LUT-BKN (на 2 переменные)

Сигналы значений входных переменных подаются с помощью генератора наборов XWG1 в коде Грея. Для оценки тока потребления к источнику питания подключаем зонд PR1. Для анализа выхода DC-LUT-BKN используем четырехканальный осциллограф XSC2. Реализация логических функций осуществляется с помощью блока настройки, транзисторы VT27-VT30, который с помощью конфигурации ключей SA1-SA4 можно настроить на вычисление любой логической функции, в данном случае показана настройка на реализацию логической функции «исключающее ИЛИ» («Сумма по модулю два»). Комбинируя ЛЭ DC-LUT-BKN необходимым количеством блоков настройки и конфигурируя их на реализацию различных логических функций, реализуем системы логических функций на одном ЛЭ DC-LUT-BKN.

Получим осциллограмму работы DC-LUT-BKN на частоте 300 МГц (Spice модель транзисторов BSIM4.8 технологии 65нм) - рисунок 3.27. Наблюдаем формирование логической единицы (нижняя осциллограмма) в случае разницы логических уровней входов (два верхних сигнала осциллограммы на рисунке 3.27). С помощью осциллографа получим значения задержки сигнала – рисунок 3.28. Аналогично получим значения задержки и ток потребления для DC-LUT-BKN от 1 до 8 входных переменных, полученные значения будут использованы для нахождения решения поставленной задачи диссертационного исследования.

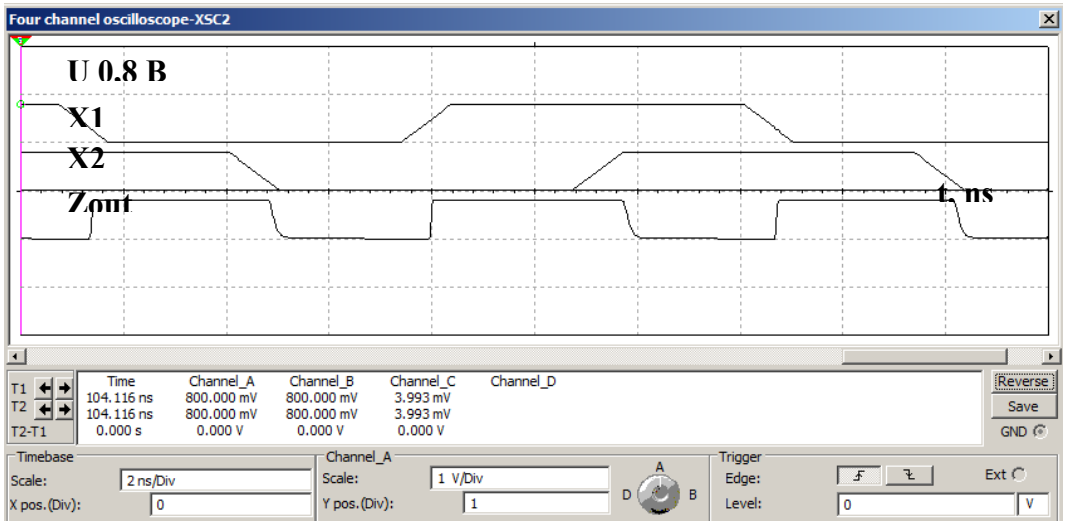


Рисунок 3.27. Осциллограмма модели 2-DC-LUT-BKN (на 2 переменные)

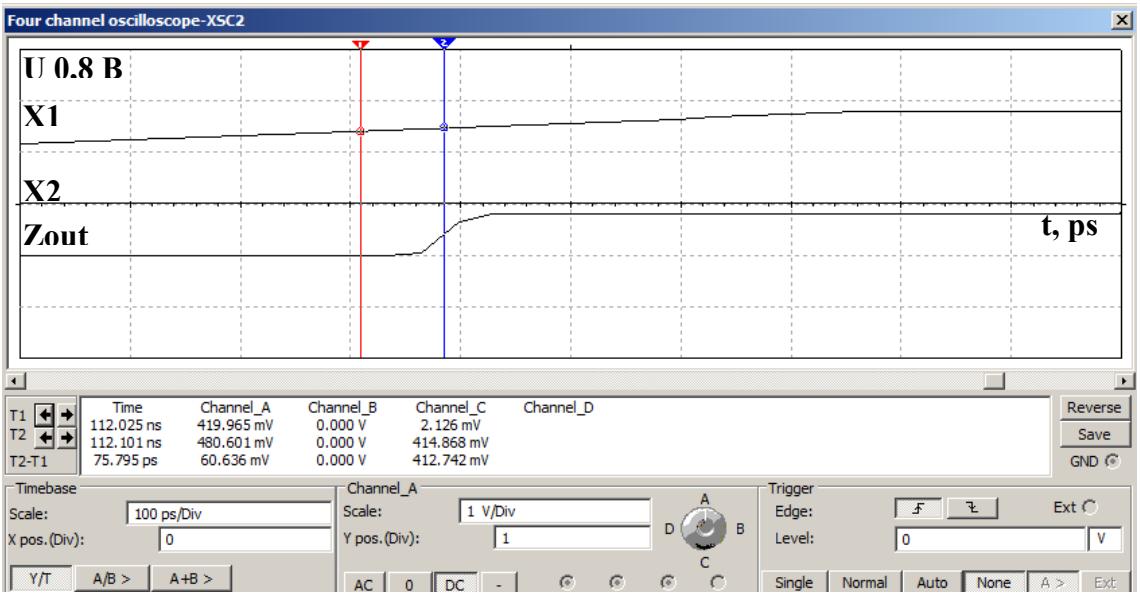


Рисунок 3.28. Значение задержки с осциллограммы модели 2-DC-LUT-BKN (на 2 переменные) на напряжении питания 0,8 В при частоте 300МГц

3.3. Моделирование логического элемента ADC-LUT

3.3.1 Статическое моделирование логического элемента ADC-LUT

Разработаем статическую модель [63,68] адаптивного логического элемента, использующий преимущества LUT и DC-LUT – ADC-LUT. Выполним моделирование ADC-LUT на 2 входные переменные (рисунок 3.29) в системе схемотехнического моделирования NI Multisim фирмы National Instruments Electronics Workbench Group [67].

С помощью ключа SA1 (КЯ – конфигурационная ячейка) настраивается режим работы ADC-LUT при подаче логической единицы реализуется LUT при подаче логического нуля DC-LUT. Транзисторы VT44-VT51 образуют входные инверторы LUT к которым подключаются ячейки памяти, конфигурирующие логическую функцию. Транзисторы VT52-VT59 образуют выходные инверторы с обратного дерева транзисторов DC-LUT, которые подключаются к затворам транзисторов блоков настройки. Транзисторы VT64-VT67 формируют один блок настройки, в данном случае настроенный на реализацию логической функции «эквиваленция» («логическая равнозначность»). Комбинируя ЛЭ ADC-LUT необходимым количеством

блоков настройки и конфигурируя их на реализацию различных логических функций, мы можем реализовать системы логических функций на одном ЛЭ ADC-LUT.

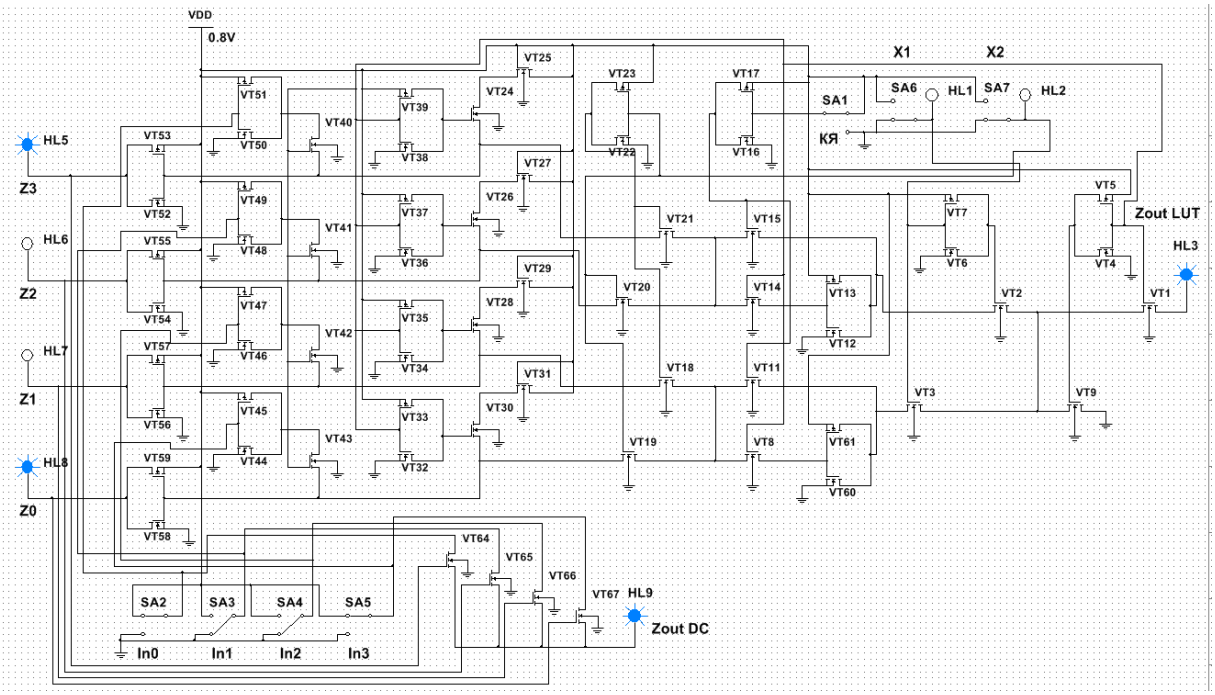


Рисунок 3.29. Модель ADC-LUT (на 2 переменные) в режиме LUT, входной набор X1=0, X2=0

Транзисторы VT2-VT3 и VT18-VT20 формируют дерево передающих транзисторов, как для LUT, так и для DC-LUT. Транзисторы VT25, VT27-VT29 подтягивающие транзисторы для режима DC. Когда на затворы транзисторов VT1, VT8, VT14, VT40-VT43 подается логическая единица с ключа SA1 активируется режим LUT. Режим DC-LUT активируется, когда логическая единица приходит на затворы транзисторов VT9, VT11, VT15, VT24, VT26, VT28, VT30. Инверторы VT4-VT5, VT16-VT17, VT32-VT39 обеспечивают конфигурирование режима работы. Работа ADC-LUT в режиме LUT аналогична работе ЛЭ LUT. Реализация логической функции «эквиваленция», для входного набора X1=0, X2=0 показана на рисунке 3.26, для входного набора X1=1, X2=0 показана на рисунке 3.27, для входного набора X1=0, X2=1 показана на рисунке 3.30, для входного набора X1=1, X2=1 показана на рисунке 3.31. Наблюдаем формирование логической нуля в случае разницы логических уровней входов рисунок 3.32 и рисунок 3.33.

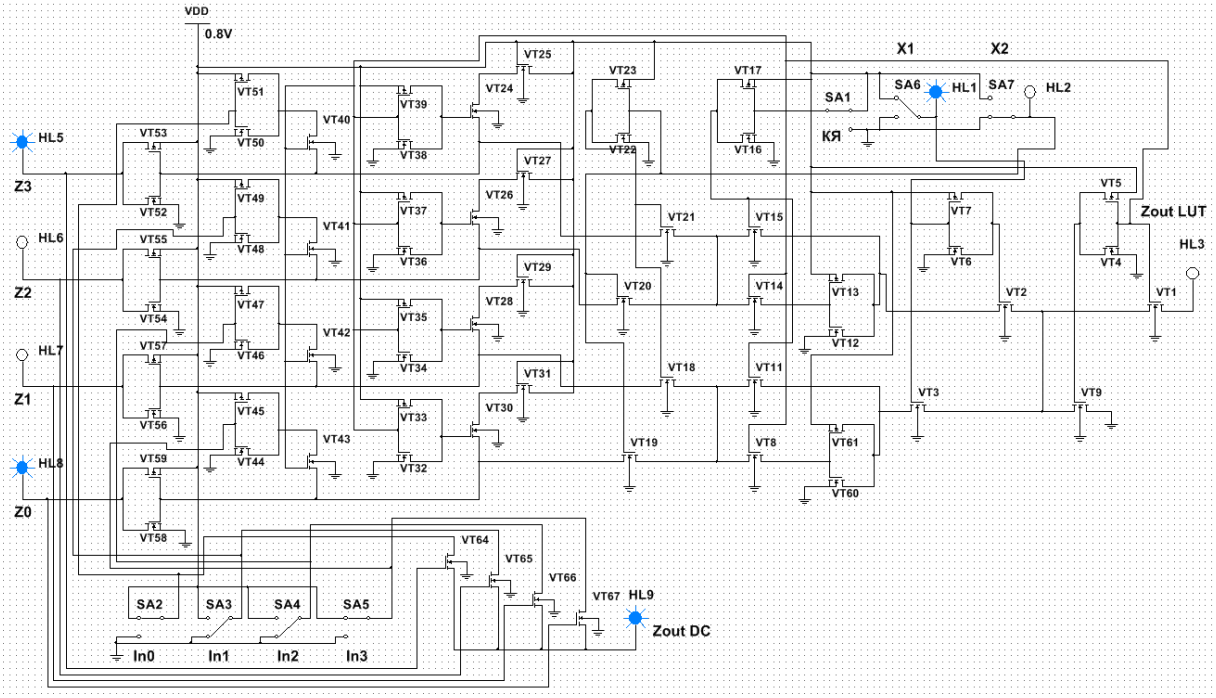


Рисунок 3.30. Модель ADC-LUT (на 2 переменные) в режиме LUT, входной набор X1=1, X2=0

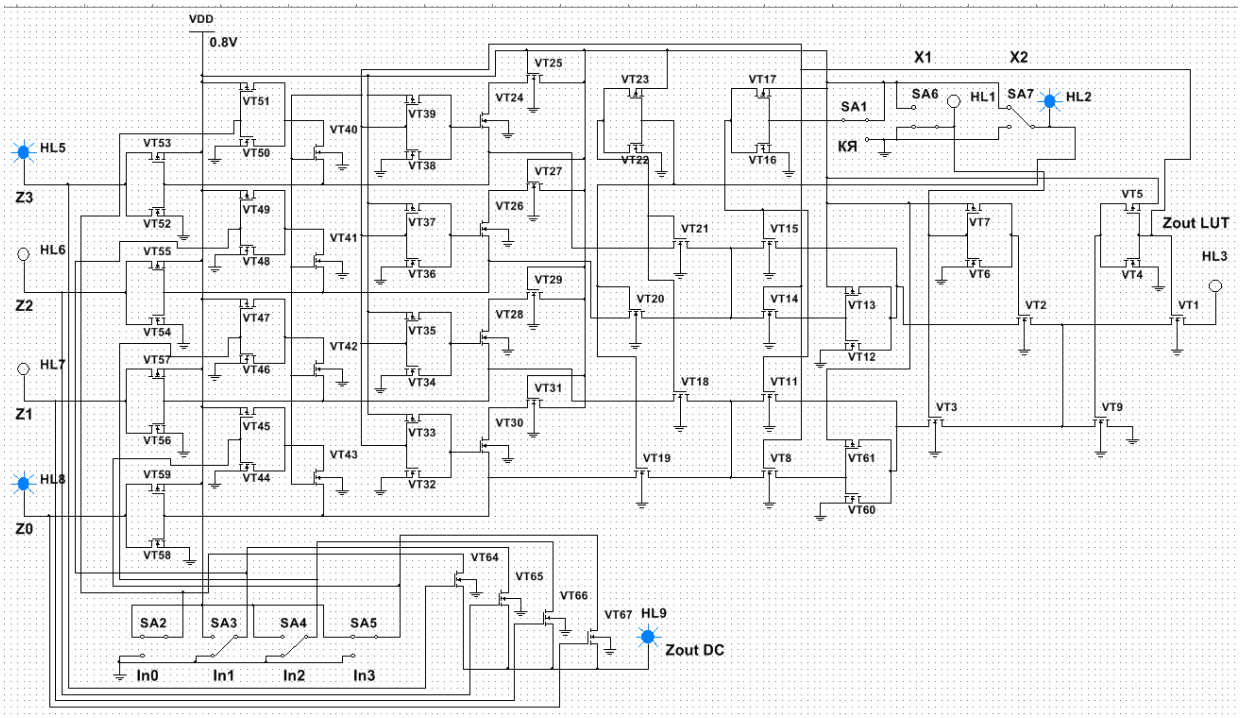


Рисунок 3.31. Модель ADC-LUT (на 2 переменные) в режиме LUT, входной набор X1=0, X2=1

Работа ADC-LUT в режиме DC-LUT аналогична работе ЛЭ DC-LUT-R. Реализация логической функции «эквиваленция» для входного набора X1=0, X2=0 показана на рисунке 3.34, для входного набора X1=1, X2=0 показана на рисунке 3.35, для входного набора X1=0, X2=1 показана на рисунке 3.36, для

входного набора X1=1, X2=1 показана на рисунке 3.37 Наблюдаем формирование логической нуля в случае разницы логических уровней входов рисунок 3.38 и рисунок 3.39.

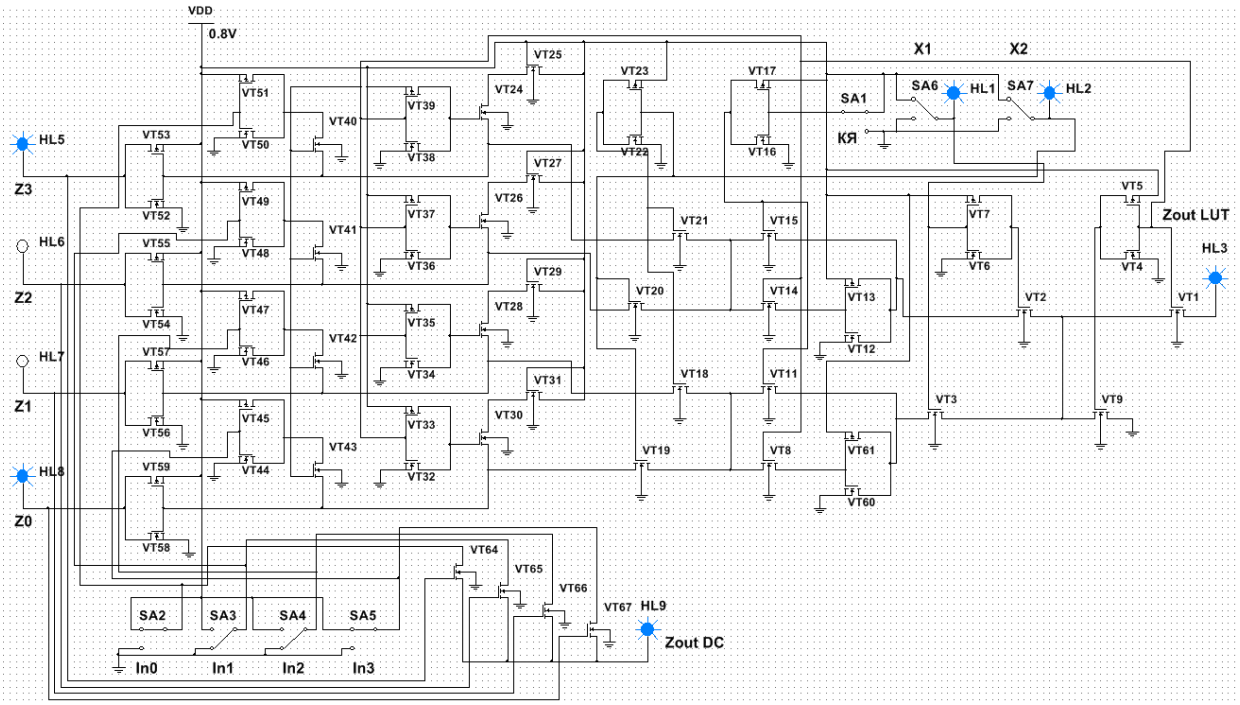


Рисунок 3.32. Модель ADC-LUT (на 2 переменные) в режиме LUT, входной набор X1=1, X2=1

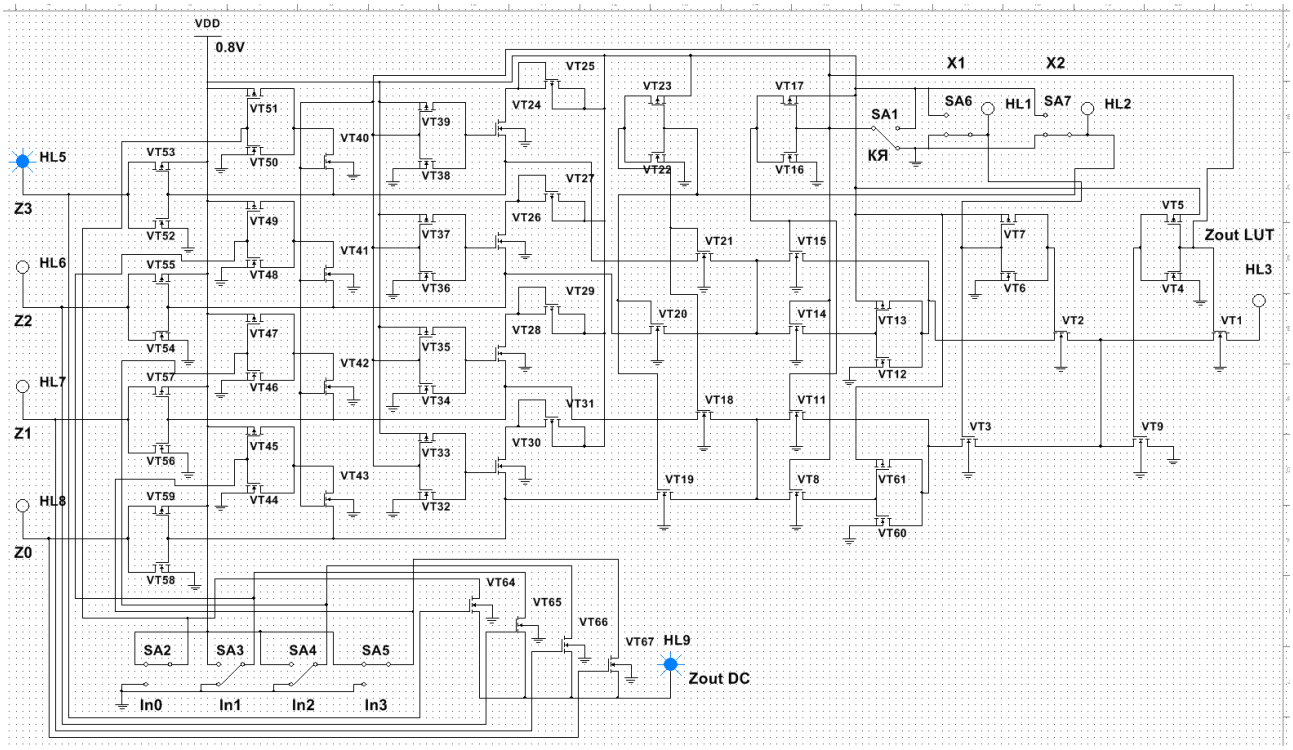


Рисунок 3.33. Модель ADC-LUT (на 2 переменные) в режиме DC, входной набор X1=0, X2=0

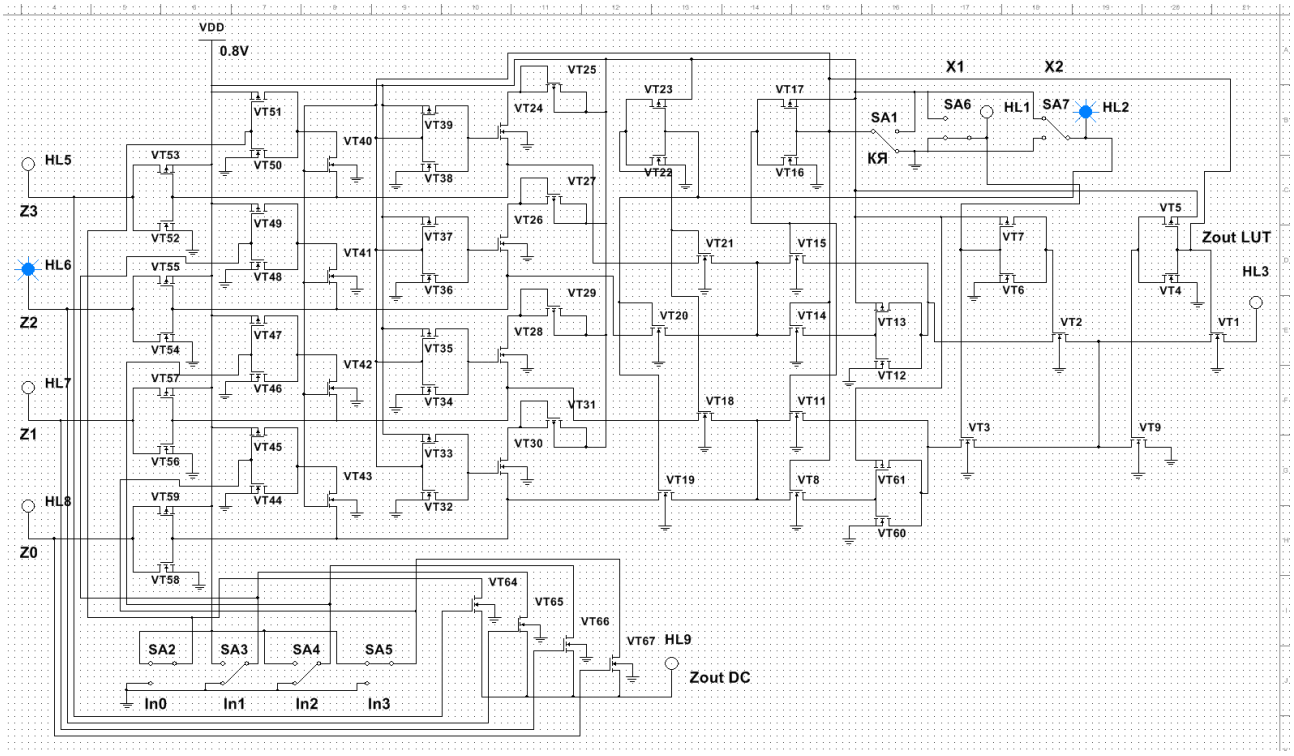


Рисунок 3.34. Модель ADC-LUT (на 2 переменные) в режиме DC, входной набор X1=0, X2=1

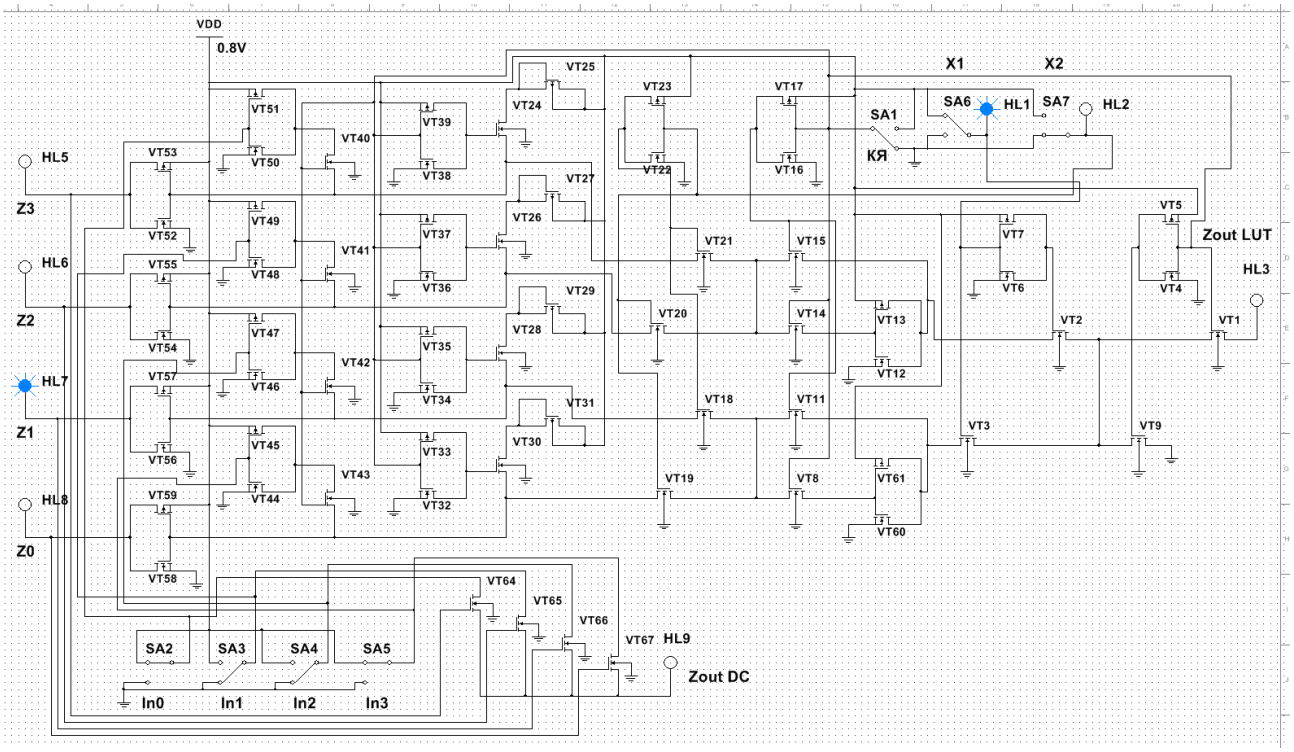


Рисунок 3.35. Модель ADC-LUT (на 2 переменные) в режиме DC, входной набор X1=1, X2=0

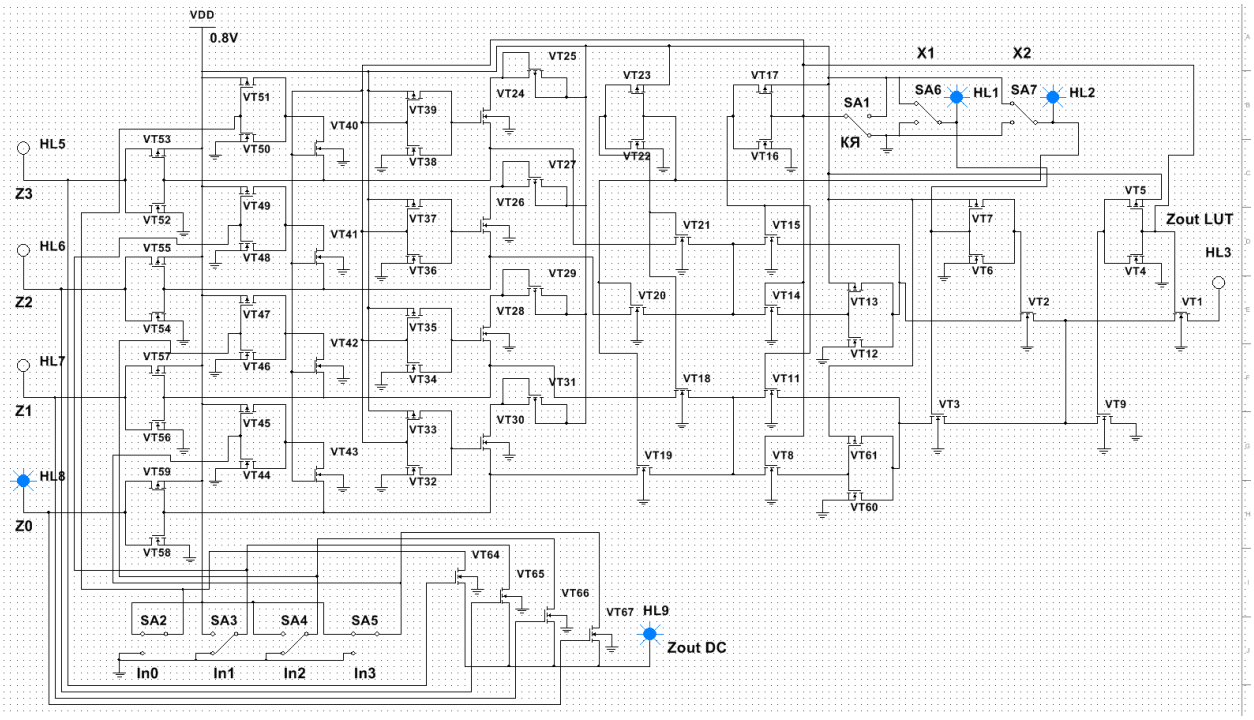


Рисунок 3.36. Модель ADC-LUT (на 2 переменные) в режиме DC, входной набор X1=1, X2=1

3.3.2. Динамическое моделирование логического элемента – ADC-LUT

Разработаем модель предлагаемого ЛЭ на 2 входные переменные ADC-LUT – рисунок 3.37 и оценим динамические параметры[63,68].

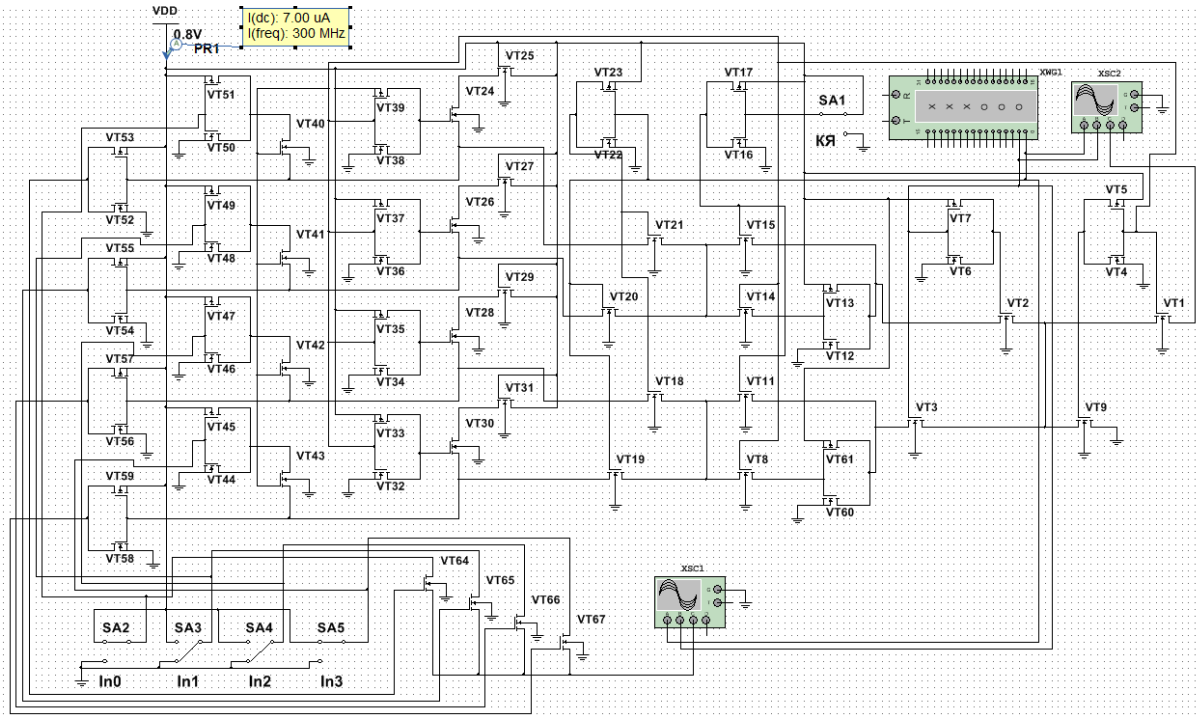


Рисунок 3.37. Модель ADC-LUT (на 2 переменные)

Сигналы значений входных переменных подаются с помощью генератора наборов XWG1 в коде Грея. Для оценки тока потребления к источнику питания подключаем зонд PR1. Для анализа выхода ADC-LUT в режиме DC используем четырехканальный осциллограф XSC1, а в режиме LUT используем четырехканальный осциллограф XSC2. Режим ADC-LUT настраивается ключом SA1. Реализация логических функций осуществляется с помощью блока настройки, транзисторы VT64-VT67, который с помощью конфигурации ключей SA2-SA5 можно настроить на вычисление любой логической функции, в данном случае показана настройка на реализацию логической функции «эквиваленция» («логическая равнозначность»). Комбинируя ЛЭ ADC-LUT необходимым количеством блоков настройки и конфигурируя их на реализацию различных логических функций, реализуем системы логических функций на одном ЛЭ ADC-LUT.

Получим осциллограмму работы ADC-LUT в режиме LUT на частоте 300 МГц (Spice модель транзисторов BSIM4.8 технологии 65нм) - рисунок 3.35.

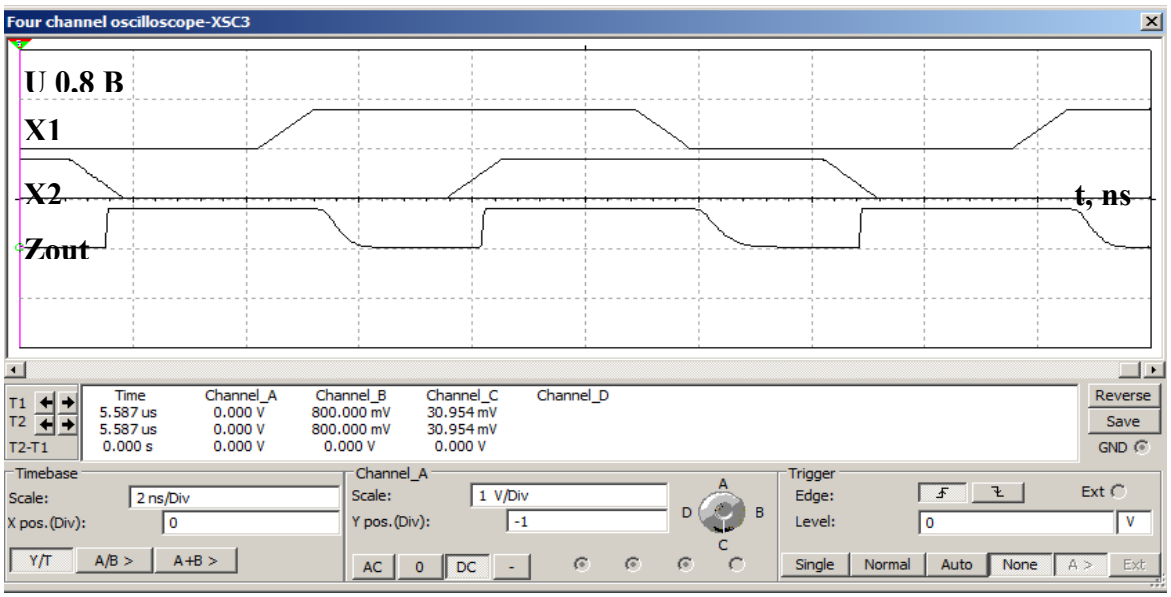


Рисунок 3.38. Осциллограмма модели 2-ADC-LUT в режиме LUT (на 2 переменные)

Получим осциллограмму работы ADC-LUT в режиме DC на частоте 300 МГц (Spice модель транзисторов BSIM4.8 технологии 65нм) - рисунок 3.36. Наблюдаем формирование логической единицы (нижняя осциллограмма) в

случае разницы логических уровней входов (два верхних сигнала осциллограммы на рисунке 3.35 и на рисунке 3.36). С помощью осциллографа получим значения задержки сигнала – рисунок 3.37 и рисунок 3.38. Аналогично получим значения задержки и ток потребления для ADC-LUT-R от 1 до 8 входных переменных, полученные значения будут использованы для нахождения решения поставленной задачи диссертационного исследования.

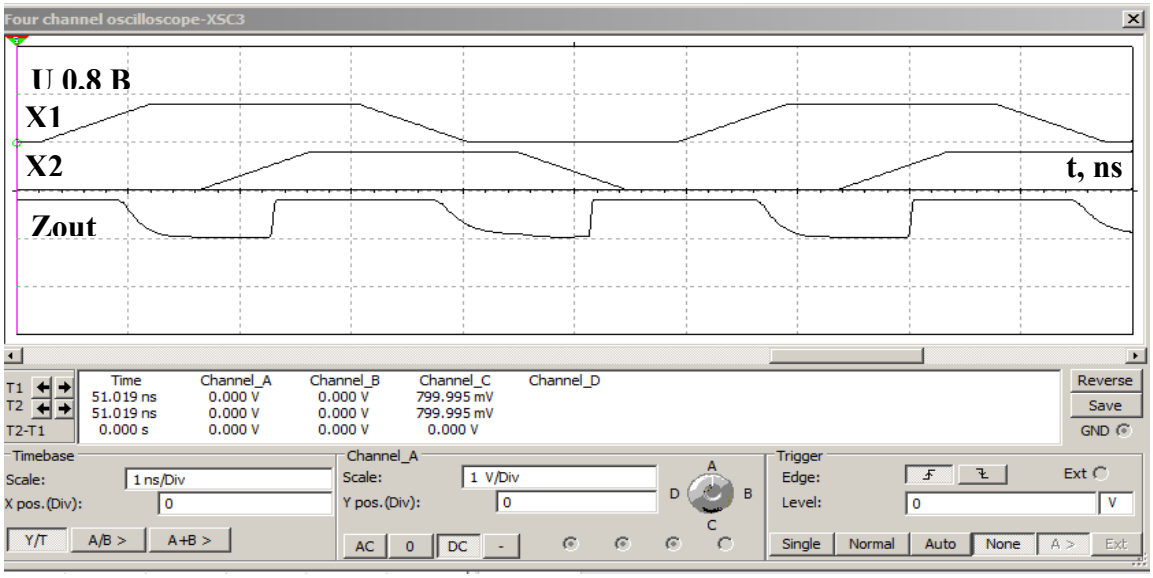


Рисунок 3.39. Осциллограмма модели 2-ADC-LUT в режиме DC (на 2 переменные)

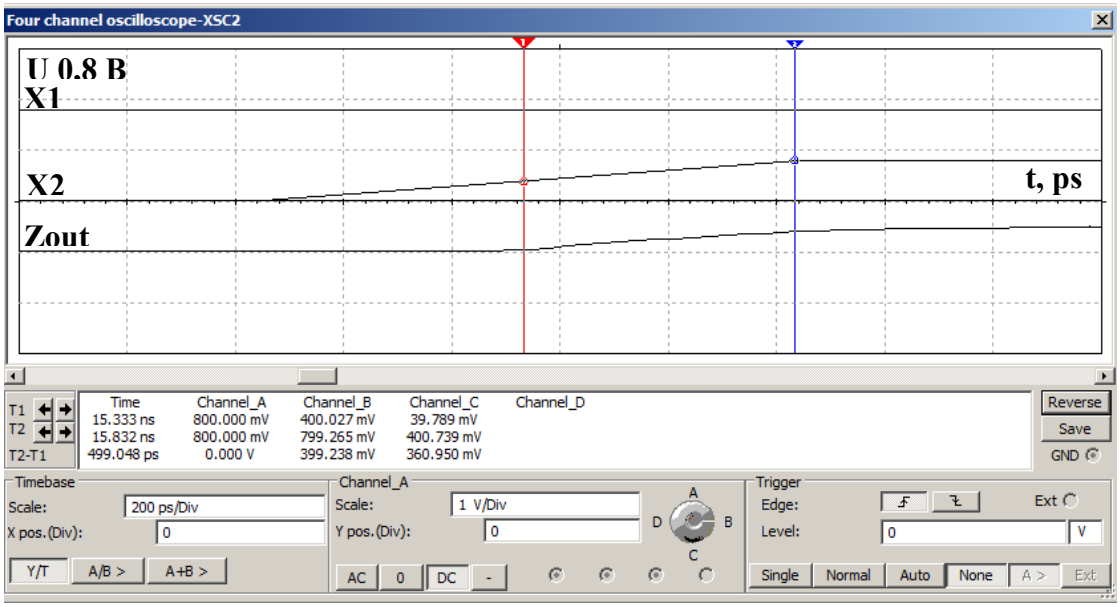


Рисунок 3.40. Значение задержки с осциллограммы модели 2-ADC-LUT в режиме LUT (на 2 переменные) на напряжении питания 0,8 В при частоте 300МГц

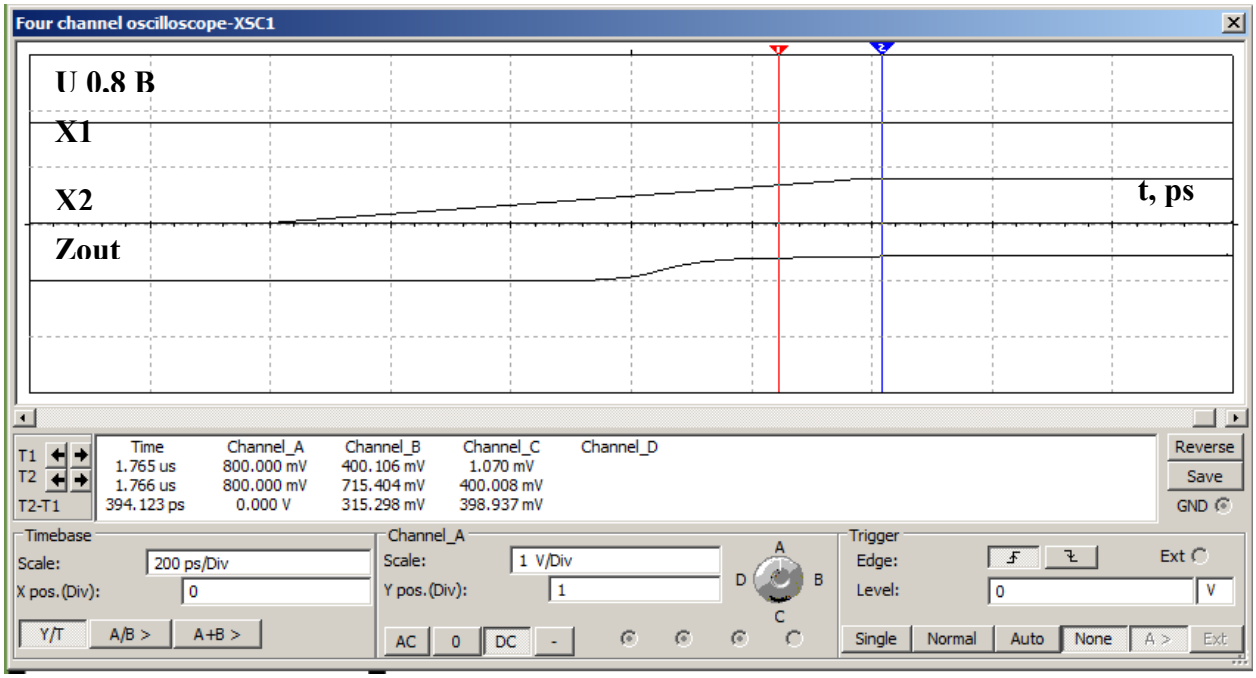


Рисунок 3.41. Значение задержки с осциллограммы модели 2-ADC-LUT в режиме DC (на 2 переменные) на напряжении питания 0,8 В при частоте 300МГц

3.4. Моделирование одного разряда блока конъюнкций для ДНФ реализации логических функций в ПЛИС

3.4.1 Статическое моделирование блока конъюнкций для ЛЭ DNF-R с нагрузочным транзистором для реализации систем логических функций в ПЛИС

Разработаем модель существующего блока конъюнкций на 2 входные переменные применяемого в ЛЭ DNF-R – рисунок 3.39 в системе схемотехнического моделирования NI Multisim фирмы National Instruments Electronics Workbench Group [68]. Выполним моделирование для разных входных наборов: для входного набора X1=0, X2=0 рисунок 3.39; для входного набора X1=0, X2=1 рисунок 3.40; для набора X1=1, X2=1 рисунок 3.41; для набора X1=1, X2=0 рисунок 3.42. Рисунок 3.42 показывается активацию блока конъюнкции, так как на выходе инвертора VT19-VT20 светодиод HL3 не светится. При входном наборе X1=1, X2=0 на затворы VT1, VT4, VT6, VT7 поступает логическая единица и тем самым активируется цепочка этих

транзисторов, которая передает логическую единицу, подключенную ко входу блока конъюнкций, на вход выходного инвертора VT19-VT20. Аналогично можно проверить функционирование на остальных наборах и для остальных настроек.

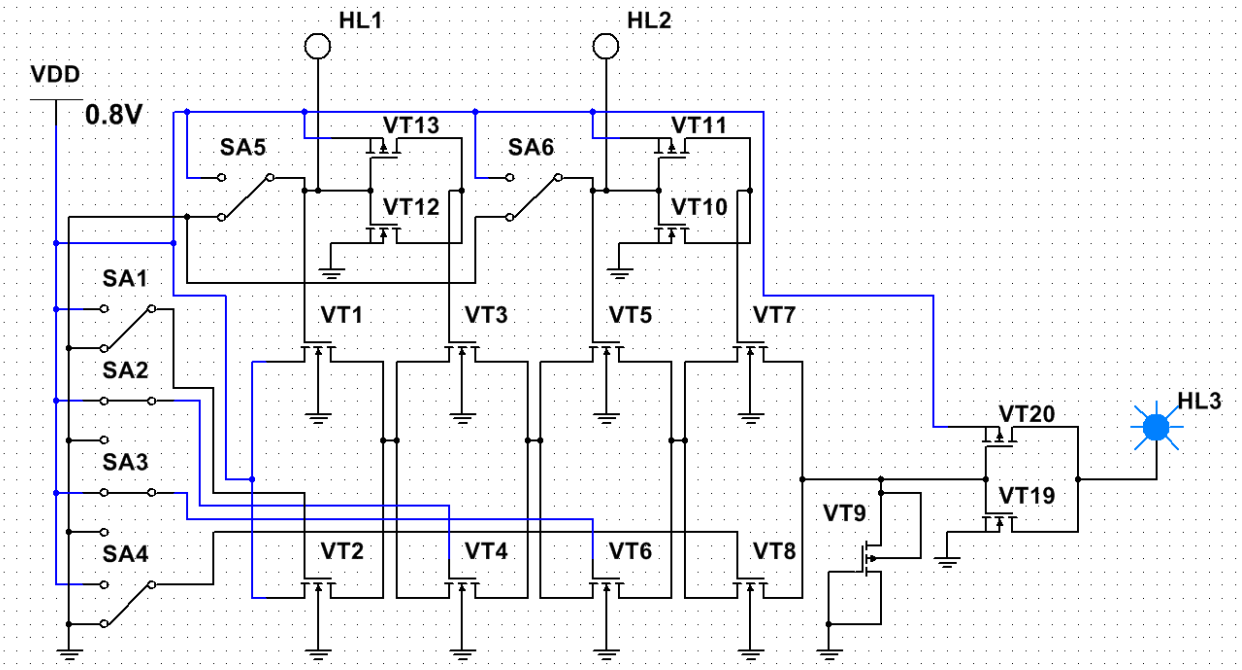


Рисунок 3.42. Модель DNF-R (на 2 переменные), входной набор X1=0, X2=0

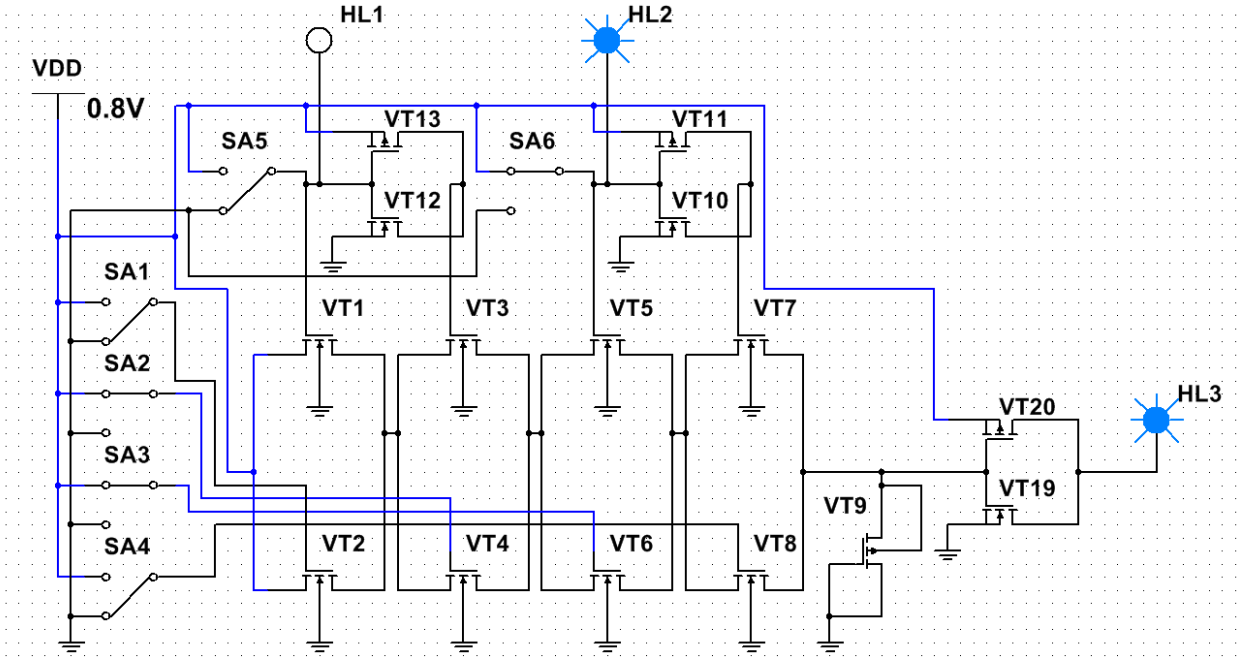


Рисунок 3.43. Модель DNF-R (на 2 переменные), входной набор X1=0, X2=1

Программирование на реализацию заданной конъюнкции осуществляется с помощью ключей SA1-SA4. Транзисторы VT10-VT13 формируют два инвертора входного набора. Ключи SA5-SA6 моделируют входные переменные X1 и X2. Нагрузочный транзистор VT9 в случае не активации блока подтягивает логический ноль на вход выходного инвертора VT19-VT20.

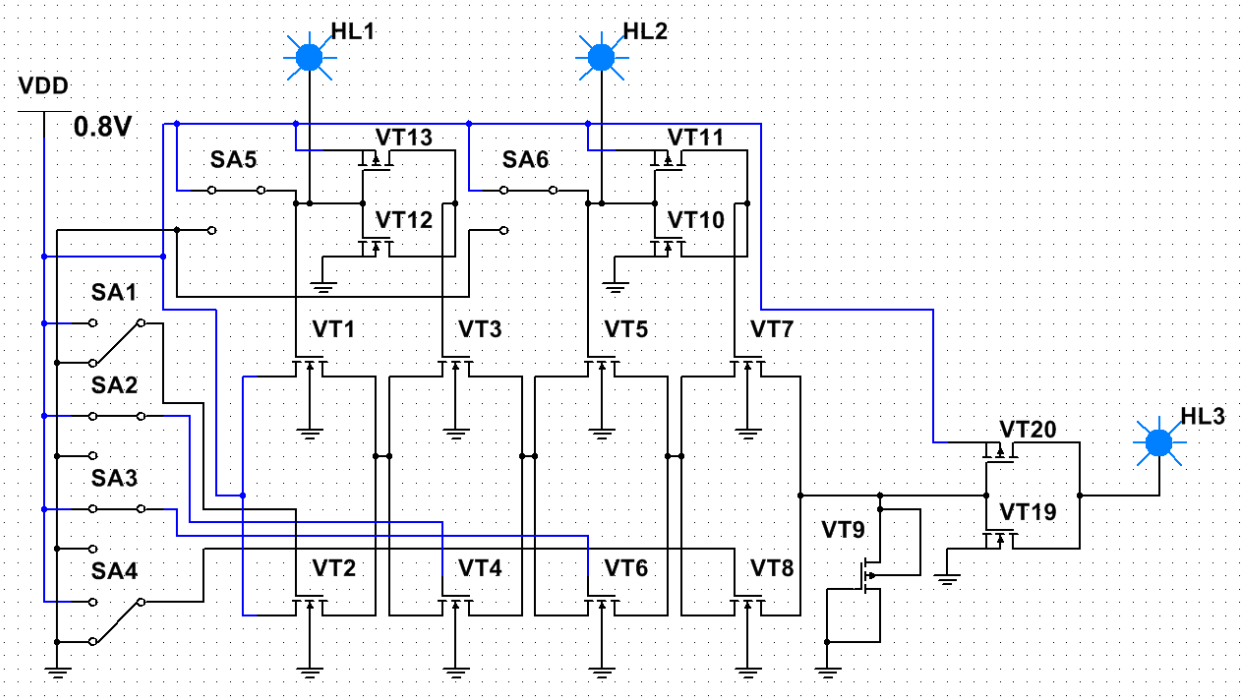


Рисунок 3.44. Модель DNF-R (на 2 переменные), входной набор X1=1, X2=1

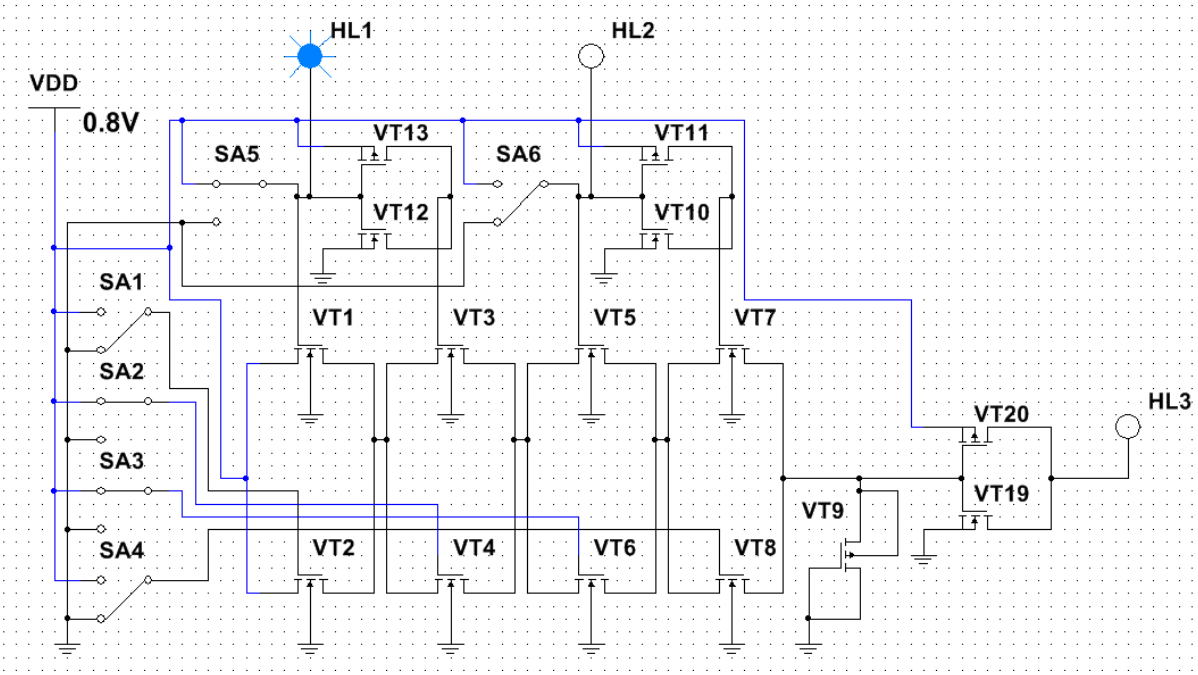


Рисунок 3.45. Модель DNF-R (на 2 переменные), входной набор X1=1, X2=0

3.4.2 Динамическое моделирование блока конъюнкций для ЛЭ DNF-R с нагрузочным транзистором для реализации систем логических функций в ПЛИС

Разработаем модель, состоящую из 2-х блоков конъюнкций на 2 входные переменные применяемых в ЛЭ DNF-R – рисунок 3.46 и оценим динамические параметры.

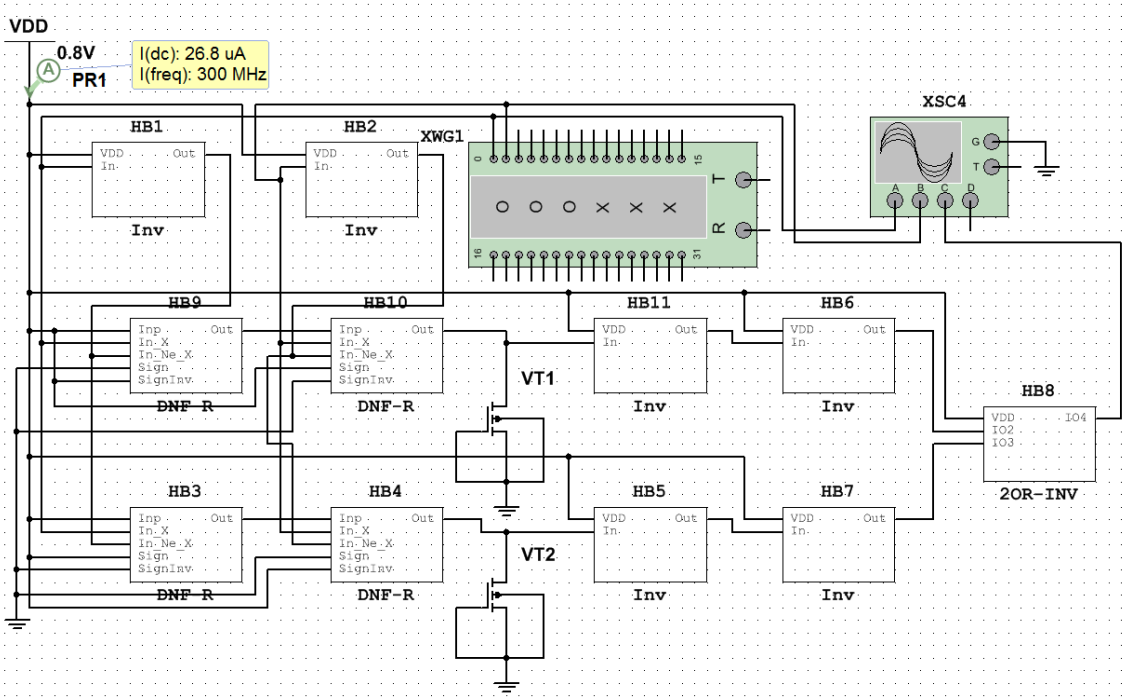


Рисунок 3.46. Модель ЛЭ DNF-R (на 2 переменные)

Схема блока DNF-R представлена на рисунке 3.47. Транзисторы VT3-VT6 образуют блок ДНФ одной переменной. Блок 2OR-INV представленный на рисунке 3.48 реализует схему 2ИЛИ-НЕ. Блок INV представленный на рисунке 3.49 реализует одноходовой инвертор.

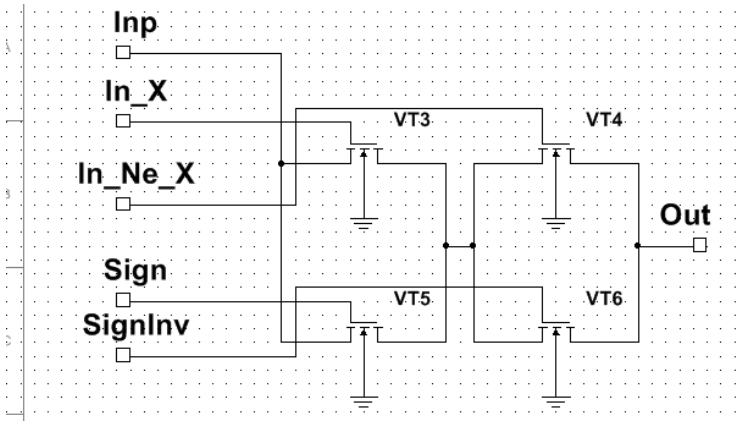


Рисунок 3.47. Модель блока DNF-R

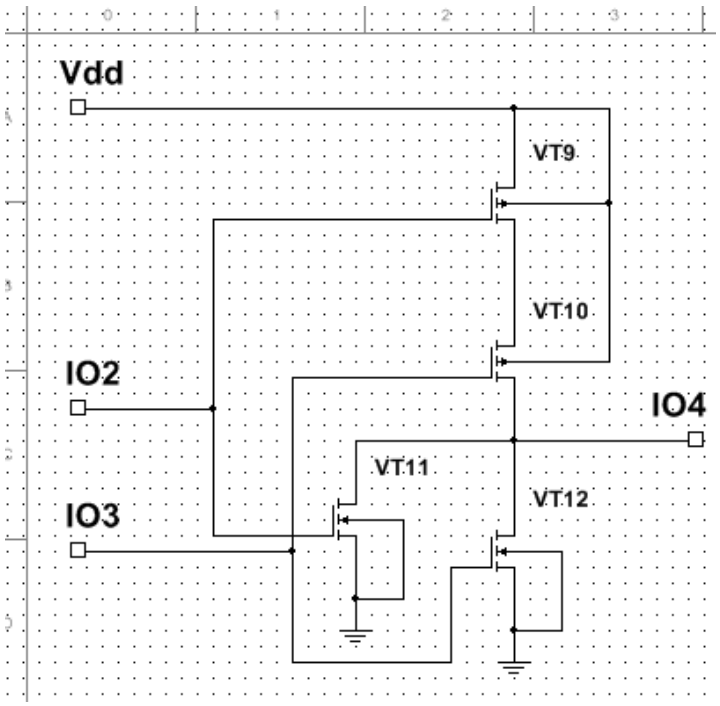


Рисунок 3.48. Модель блока 2OR-INV

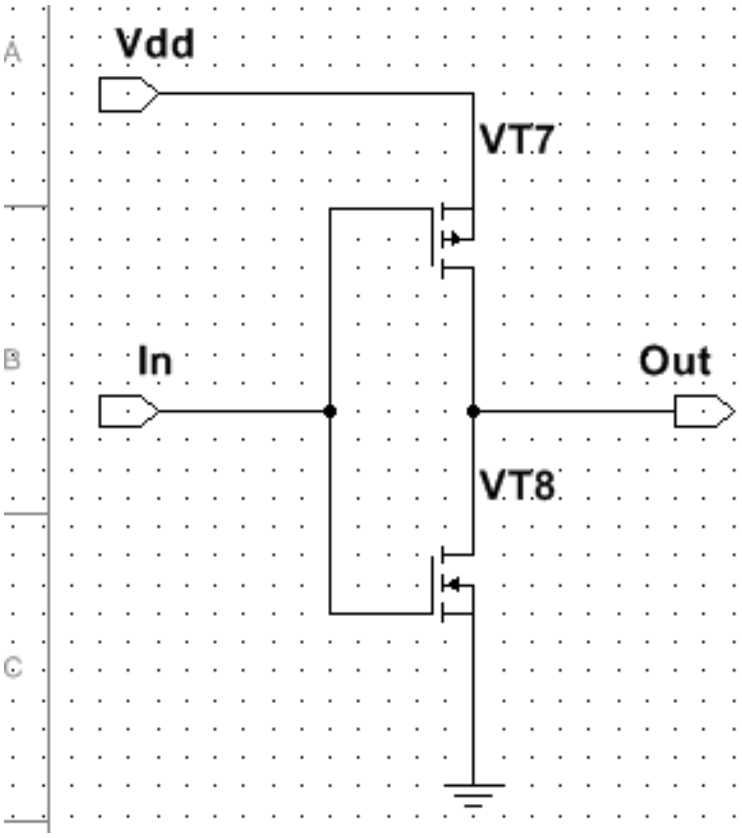


Рисунок 3.49. Модель блока INV

Сигналы значений входных переменных подаются с помощью генератора наборов XWG1 в коде Грея. Для оценки тока потребления к источнику питания подключаем зонд PR1. Для анализа выхода DNF-R используем четырехканальный осциллограф XSC4. Транзисторы VT1-VT2 -

это нагрузочные транзисторы, подключенные по выходу блока конъюнкции. Транзисторы VT3-VT6 формируют блок ДНФ. Транзисторы VT9-VT12 образуют блок 2-ИЛИ-НЕ. Транзисторы VT7-VT8 формируют инвертор.

Получим осциллограмму работы 2-DNF-R на частоте 300 МГц (Spice модель транзисторов BSIM4.8 технология 65нм), конфигурированного на вычисление логической функции «эквиваленция» («Логическая равнозначность») - рисунок 3.50.

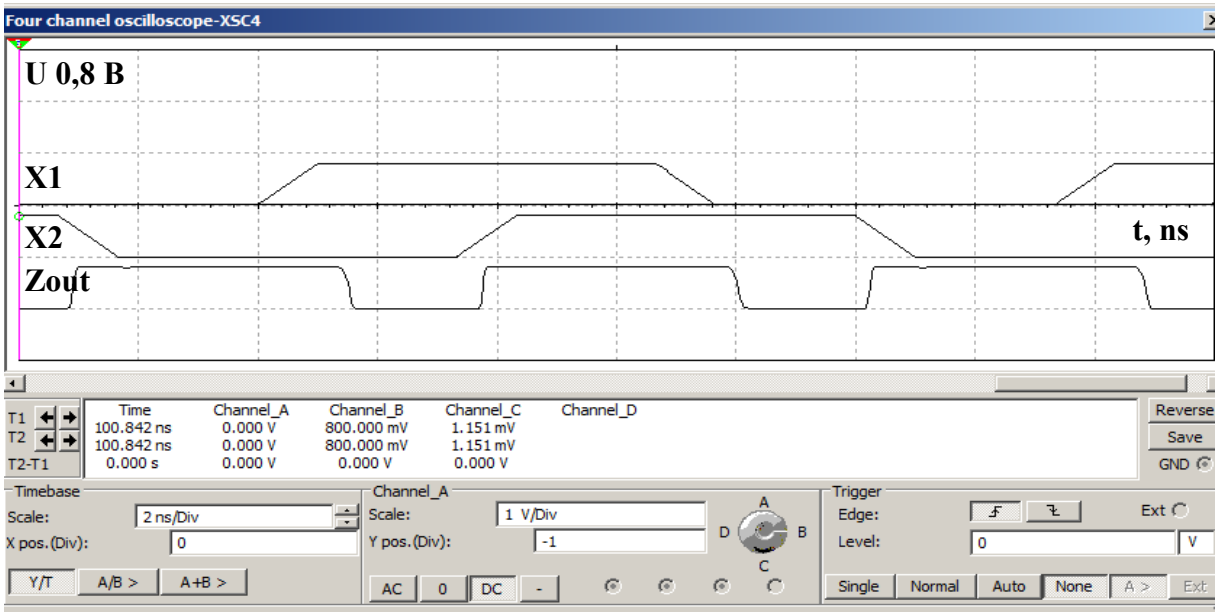


Рисунок 3.50. Осциллограмма модели 2-DNF-R (на 2 переменные)

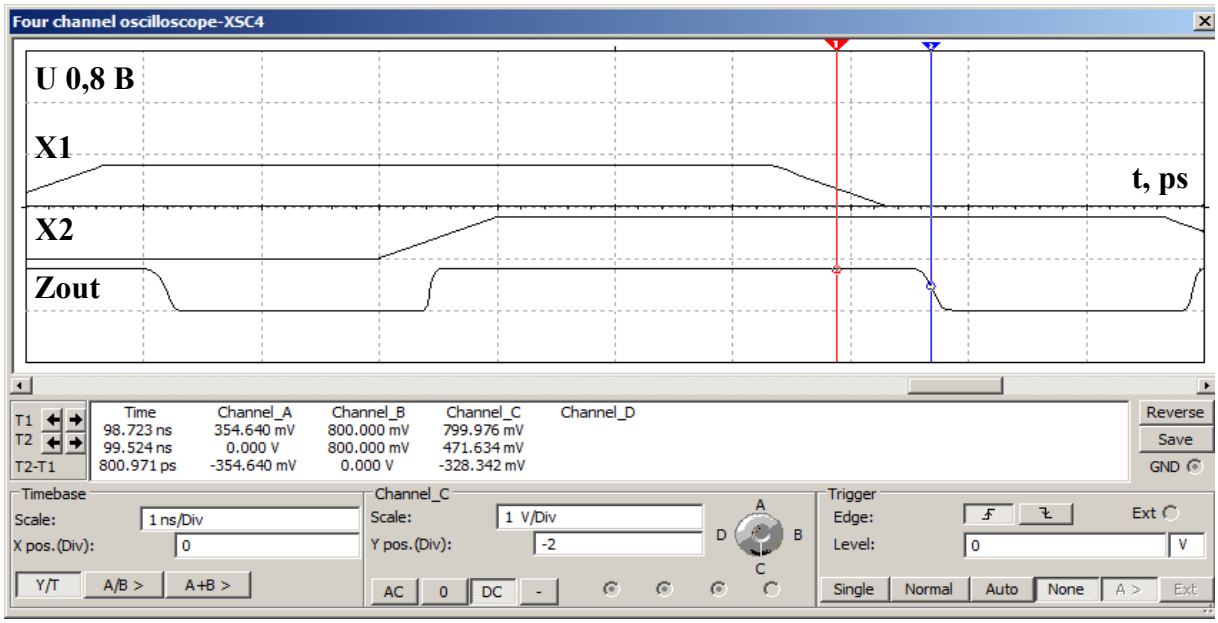


Рисунок 3.51. Значение задержки с осциллограммы модели 2-DNF-R (на 2 переменные) при напряжении питания 0,8 В на частоте 300МГц

Наблюдаем формирование логического нуля (нижняя осциллограмма) в случае разницы логических уровней входов (два верхних сигнала осциллограммы на рисунке 3.50). С помощью осциллографа получим значения задержки сигнала - рисунок 3.51. Аналогично получим значения задержки и ток потребления для DNF-R от 1 до 18 входных переменных, которые целесообразно использовать для нахождения решения поставленной задачи диссертационного исследования.

3.4.3. Статическое моделирование блока конъюнкций для ЛЭ DNF-P с параллельным подключением для реализации систем логических функций в ПЛИС

Разработаем модель предлагаемого блока конъюнкций ЛЭ DNF-P на 2 входные переменные с параллельным подключением – рисунок 3.43 в системе схемотехнического моделирования NI Multisim фирмы National Instruments Electronics Workbench Group [68]. Выполним моделирование для разных входных наборов: для входного набора X1=0, X2=0 рисунок 3.43; для входного набора X1=0, X2=1 рисунок 3.44; для набора X1=1, X2=1 рисунок 3.45; для набора X1=1, X2=0 рисунок 3.46. Рисунок 3.46 показывается активацию блока конъюнкции, так как на выходе инвертора VT18, VT21 светодиод HL3 не светится. При входном наборе X1=1, X2=0 на затворы VT1, VT4, VT6, VT7 поступает логическая единица и тем самым активируется цепочка этих транзисторов, которая передает логическую единицу, подключенную ко входу блока конъюнкций, на вход выходного инвертора VT19-VT20. Аналогично можно проверить функционирование на остальных наборах и для остальных настроек.

При входном наборе X1=1, X2=0 на затворы транзисторов VT1, VT6 поступает логическая единица и активирует их, поэтому с ключей настройки SA1, SA4 поступает логическая единица на затворы VT3, VT7 и логическая единица, подключенная к блоку конъюнкции, поступает на вход выходного инвертора VT18, VT21 светодиод HL3 не светиться. При других входных

наборах для заданной настройки активируются транзисторы VT4, VT8 которые передают на вход выходного инвертора логический ноль, светодиод HL3 светиться.

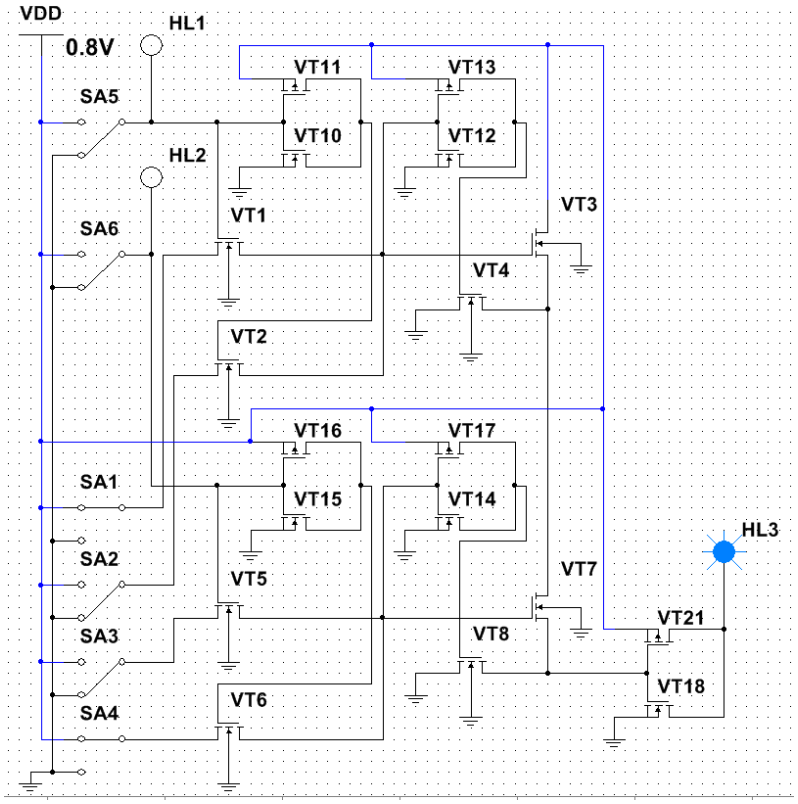


Рисунок 3.52. Модель DNF-P (на 2 переменные), входной набор X1=0, X2=0

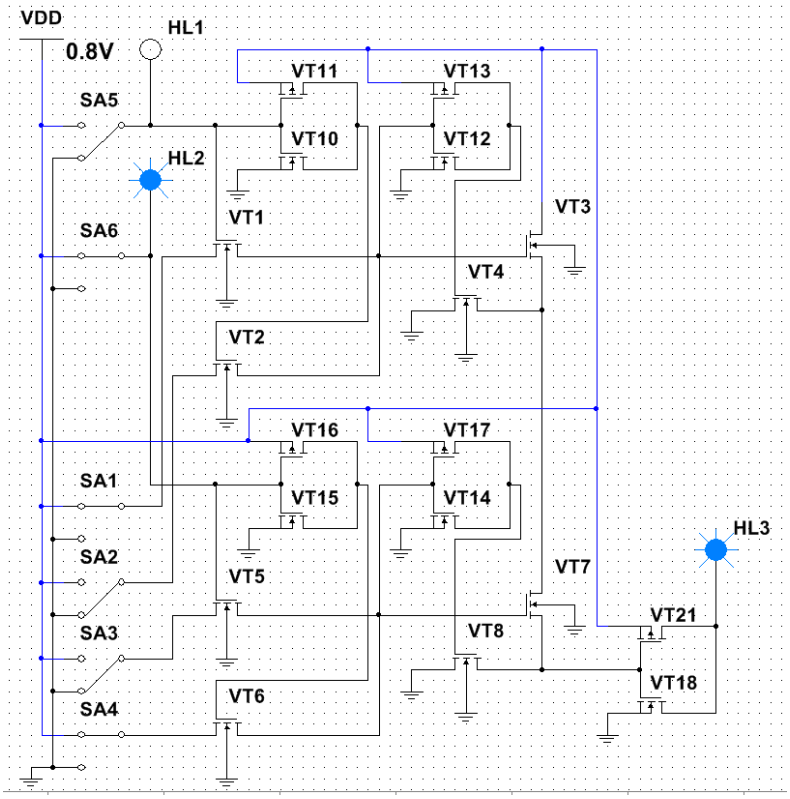


Рисунок 3.53. Модель DNF-P (на 2 переменные), входной набор X1=0, X2=1

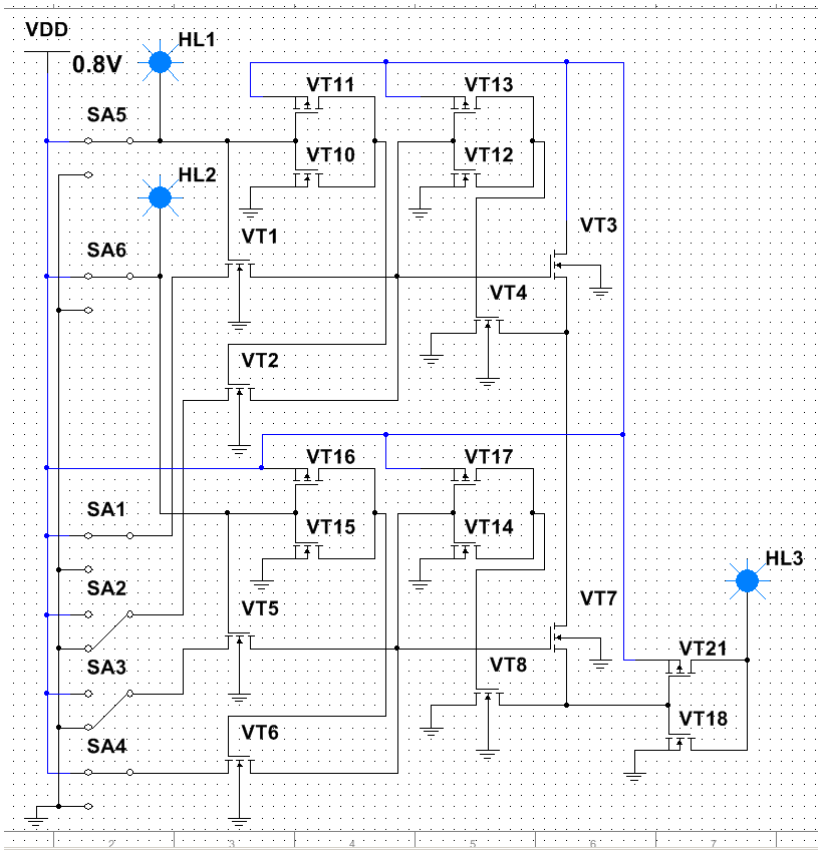


Рисунок 3.54. Модель DNF-P (на 2 переменные), входной набор X1=1, X2=1

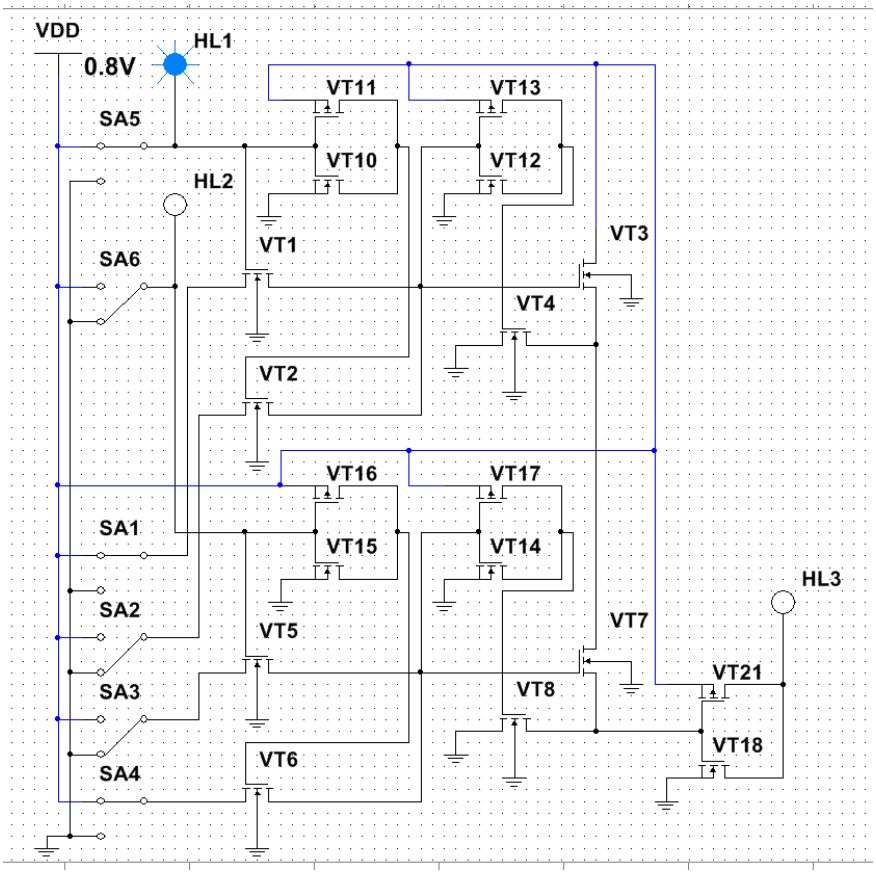


Рисунок 3.55. Модель DNF-P (на 2 переменные), входной набор X1=1, X2=0

3.4.4. Динамическое моделирование блока конъюнкций для ЛЭ DNF-P с параллельным подключением для реализации систем логических функций в ПЛИС

Разработаем модель, состоящую из 2-х блоков конъюнкций на 2 входные переменные предлагаемого DNF-P с параллельным подключением – рисунок 3.56 и оценим динамические параметры.

Сигналы значений входных переменных подаются с помощью генератора наборов XWG1 в коде Грея. Для оценки тока потребления к источнику питания подключаем зонд PR1. Для анализа выхода DNF-R используем четырехканальный осциллограф XSC4. VT1-VT2 - это нагрузочные транзисторы, подключенные по выходу блока конъюнкции.

Получим осциллограмму работы 2-DNF-R на частоте 300 МГц (Spice модель транзисторов BSIM4.8 технология 65нм), конфигурированного на вычисление логической функции «исключающее ИЛИ» («Сумма по модулю 2») - рисунок 3.57.

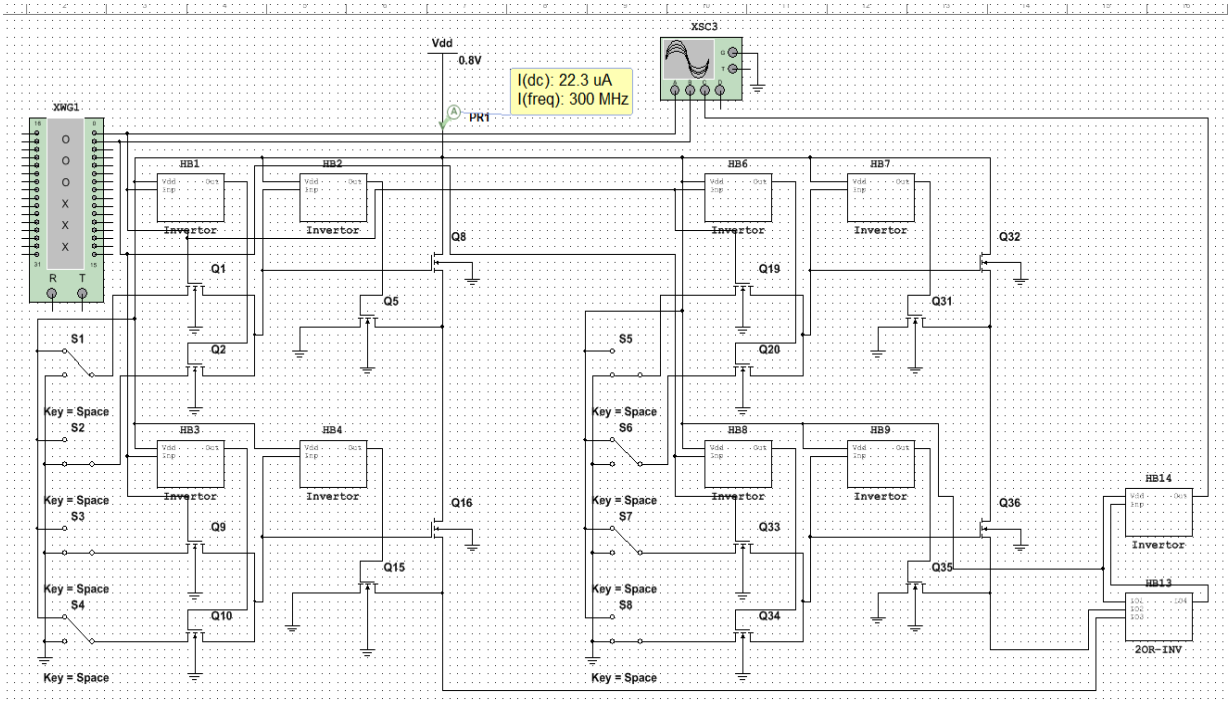


Рисунок 3.56. Модель ЛЭ DNF-P с параллельным подключением (на 2 переменные)

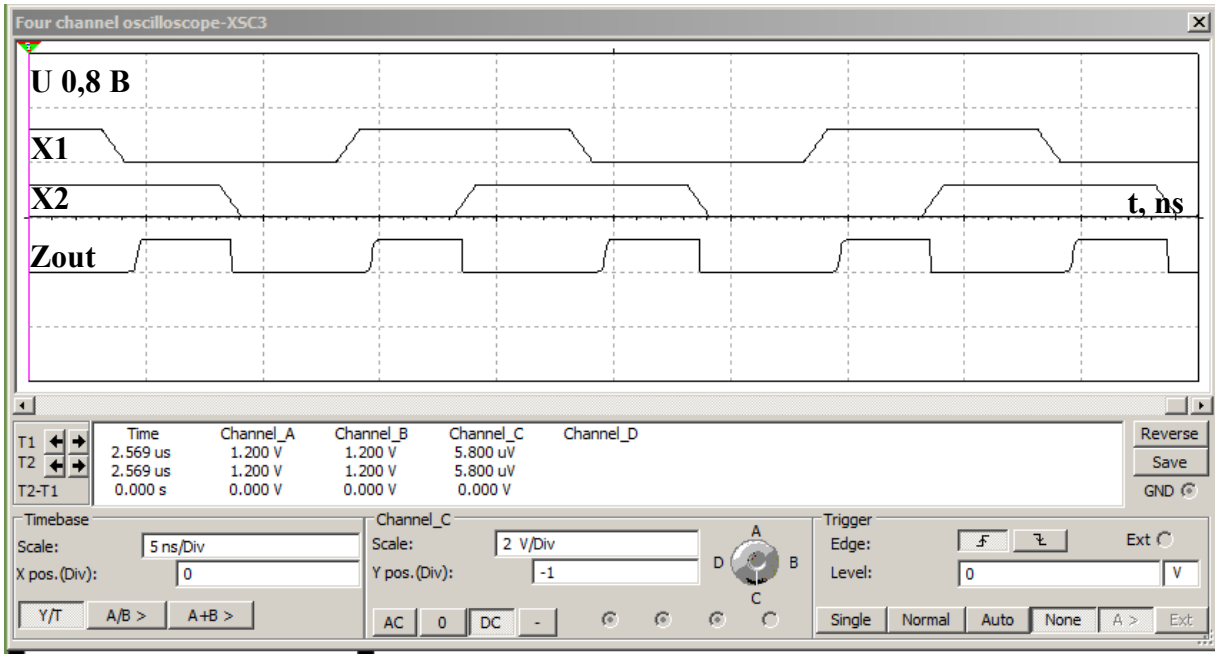


Рисунок 3.57. Значение задержки с осциллограммы модели 2-DNF-P (на 2 переменные) при напряжении питания 0,8 В на частоте 300МГц

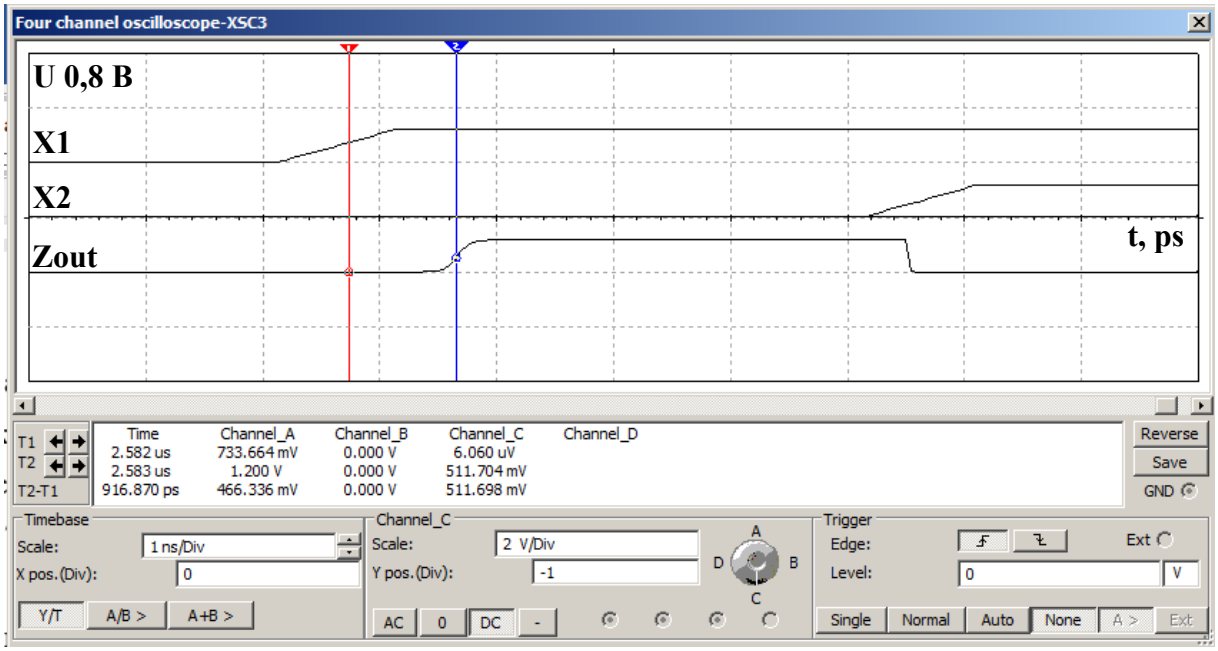


Рисунок 3.58. Значение задержки с осциллограммы модели 2-DNF-P (на 2 переменные) при напряжении питания 0,8 В на частоте 300МГц

Наблюдаем формирование логической единицы (нижняя осциллограмма) в случае разницы логических уровней входов (два верхних сигнала осциллограммы на рисунке 3.57). С помощью осциллографа получим значения задержки сигнала - рисунок 3.58. Аналогично получим значения задержки и ток потребления для DNF-P от 1 до 18 входных переменных,

которые целесообразно использовать для нахождения решения поставленной задачи диссертационного исследования.

3.4.5. Статическое моделирование блока конъюнкций для ЛЭ DNF-S с последовательным подключением для реализации систем логических функций в ПЛИС

Разработаем модель предлагаемого блока конъюнкций ЛЭ DNF-S на 2 входные переменные с последовательным подключением – рисунок 3.43 в системе схемотехнического моделирования NI Multisim фирмы National Instruments Electronics Workbench Group [68]. Выполним моделирование для разных входных наборов: для входного набора X1=0, X2=0 рисунок 3.47; для входного набора X1=1, X2=0 рисунок 3.48; для набора X1=1, X2=1 рисунок 3.49; для набора X1=0, X2=1 рисунок 3.50. Рисунок 3.48 показывается активацию блока конъюнкции, так как на выходе блока конъюнкций светодиод HL3 светится. При входном наборе X1=1, X2=0 на затворы VT1, VT2, VT16, VT17 поступает логическая единица и тем самым активируется цепочка этих транзисторов, которая передает логическую единицу, подключенную ко входу блока конъюнкций, на выход светодиод HL3 светится. Аналогично можно проверить функционирование на остальных наборах и для остальных настроек.

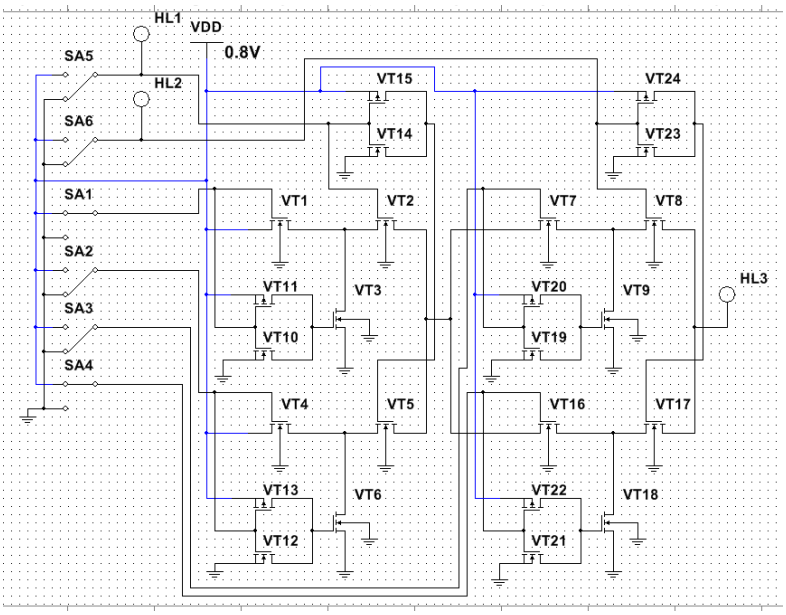


Рисунок 3.59. Модель DNF-S (на 2 переменные), входной набор X1=0, X2=0

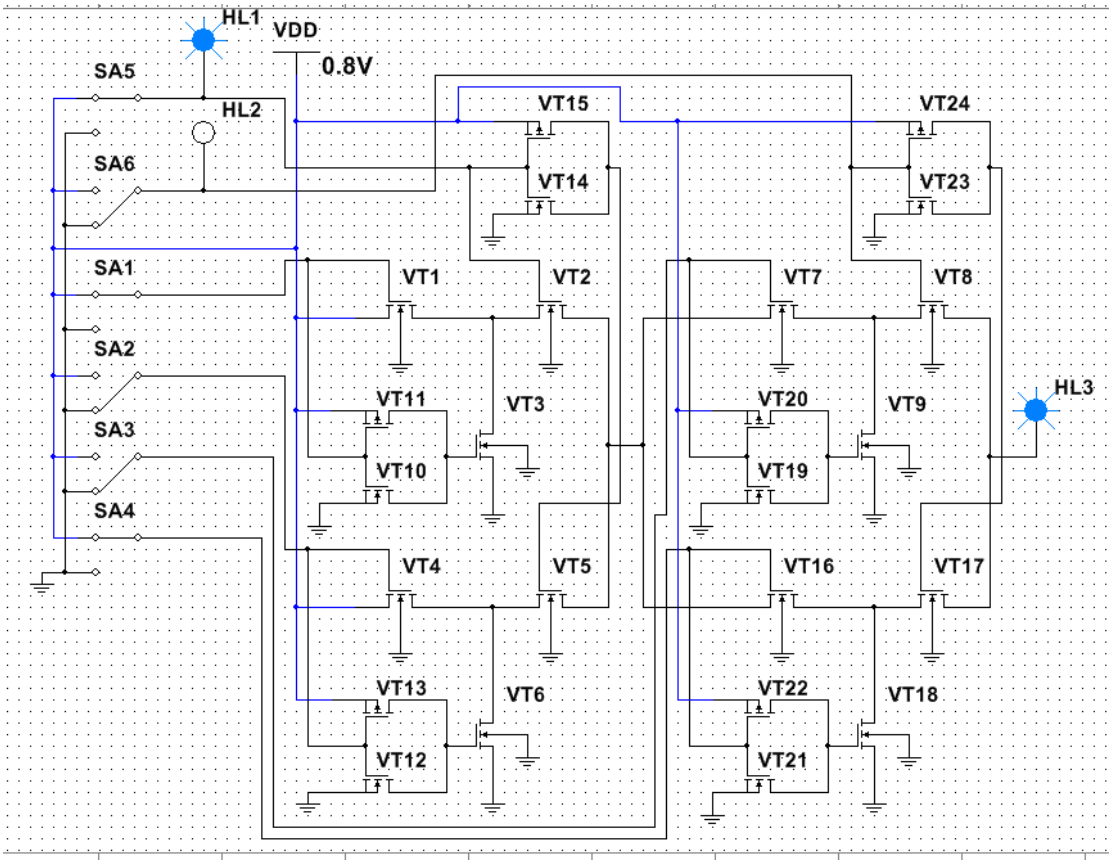


Рисунок 3.60. Модель DNF-S (на 2 переменные), входной набор X1=1, X2=0

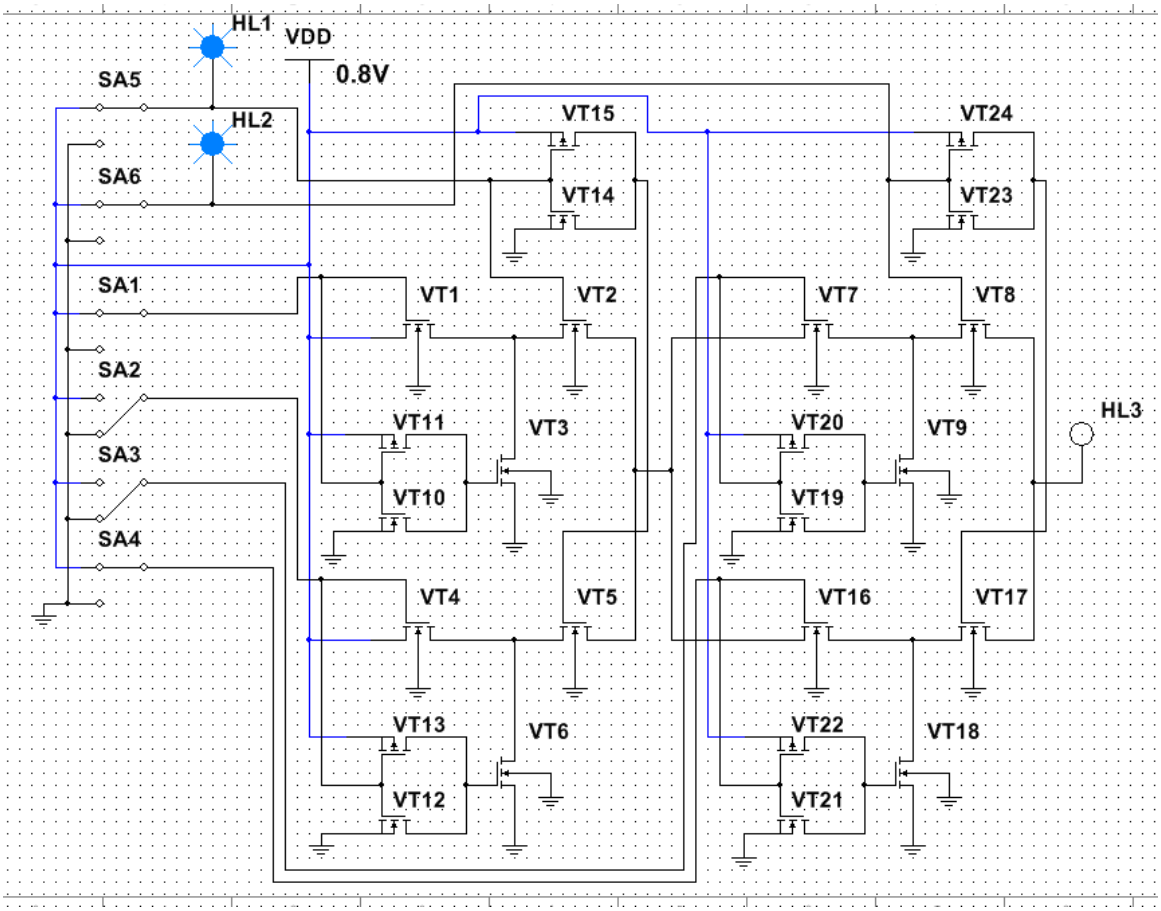


Рисунок 3.61. Модель DNF-S (на 2 переменные), входной набор X1=1, X2=1

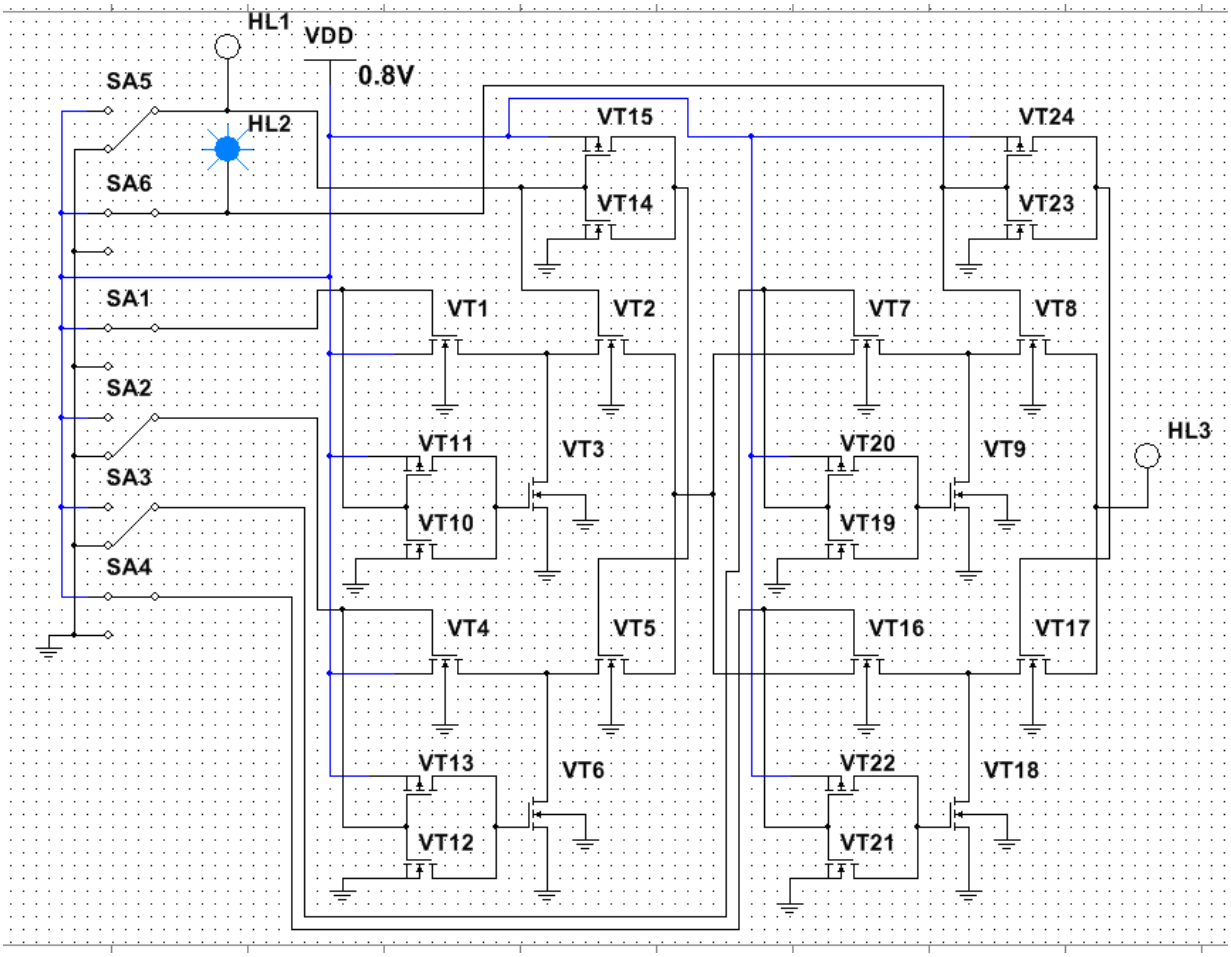


Рисунок 3.62. Модель DNF-S (на 2 переменные), входной набор X1=0, X2=1

3.4.6. Динамическое моделирование блока конъюнкций для ЛЭ DNF-S с последовательным подключением для реализации систем логических функций в ПЛИС

Разработаем модель, состоящую из 2-х блоков конъюнкций на 2 входные переменные предлагаемого ЛЭ DNF-S с последовательным подключением – рисунок 3.63 и оценим динамические параметры.

Сигналы значений входных переменных подаются с помощью генератора наборов XWG1 в коде Грея. Для оценки тока потребления к источнику питания подключаем зонд PR1. Для анализа выхода DNF-S используем четырехканальный осциллограф XSC1.

Получим осциллограмму работы 2-DNF-S на частоте 300 МГц (Spice модель транзисторов BSIM4.8 технология 65нм), конфигурированного на

вычисление логической функции «исключающее ИЛИ» («Сумма по модулю 2») - рисунок 3.64.

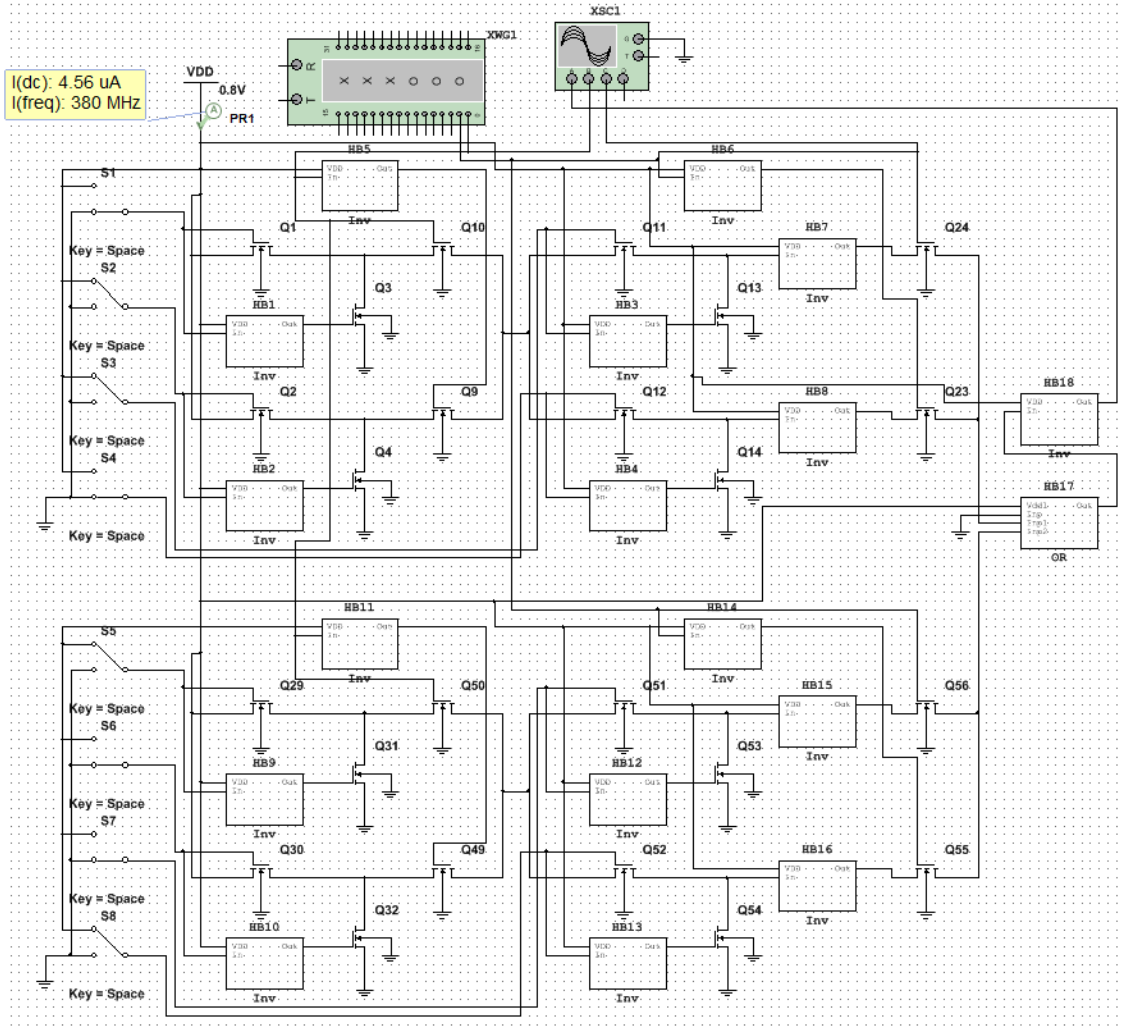


Рисунок 3.63 Модель DNF-S на 2 переменные

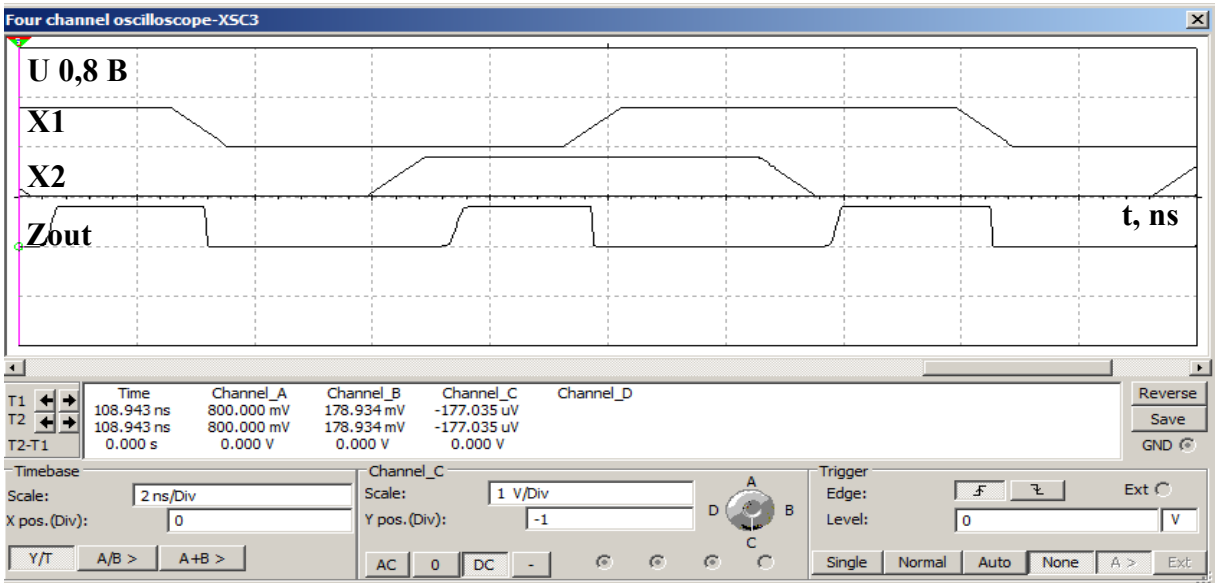


Рисунок 3.64 Осциллограмма работы DNF-S на 2 переменные

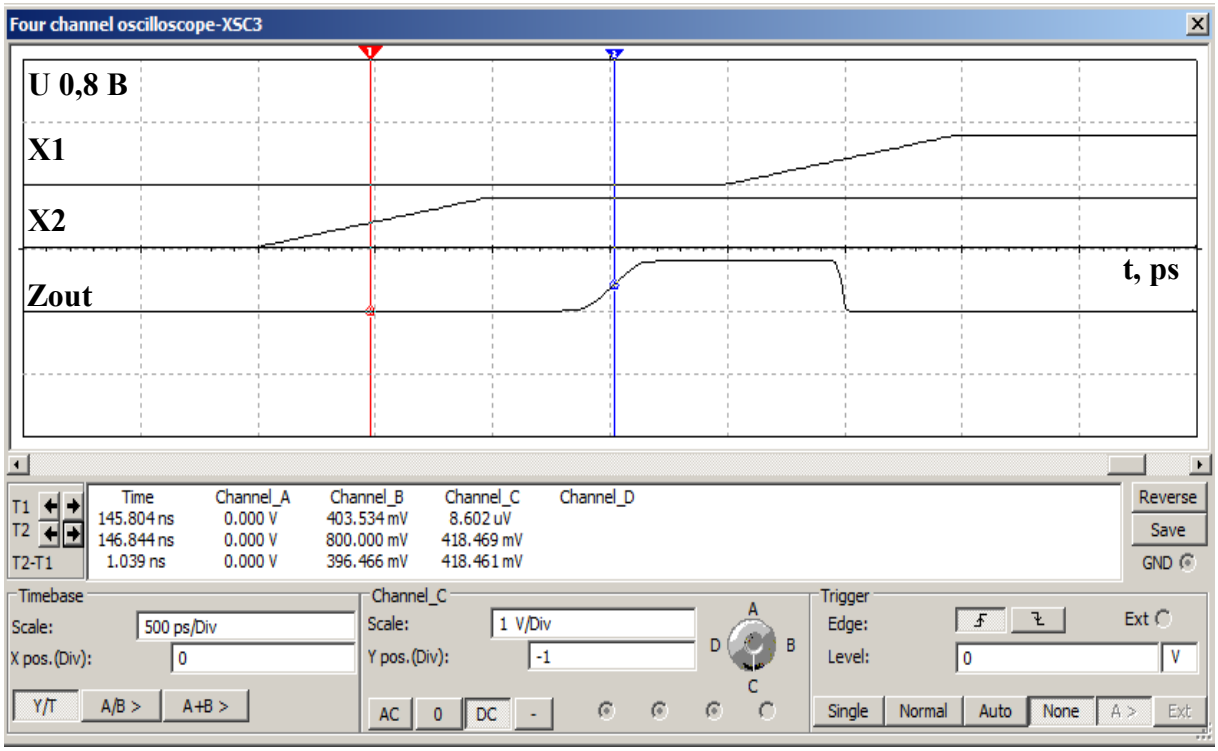


Рисунок 3.65 Значение задержки с осциллограммы модели 2-DNF-S (на 2 переменные) при напряжении питания 0,8 В на частоте 300МГц

Наблюдаем формирование логической единицы (нижняя осциллограмма) в случае разницы логических уровней входов (два верхних сигнала осциллограммы на рисунке 3.64). С помощью осциллографа получим значения задержки сигнала - рисунок 3.65. Аналогично получим значения задержки и ток потребления для DNF-S от 1 до 18 входных переменных, которые целесообразно использовать для нахождения решения поставленной задачи диссертационного исследования.

3.5. Сравнительная оценка энергопотребления ЛЭ DC–LUT, DNF-LUT и ADC

Используя модели существующих и предлагаемых логических элементов были получены значения энергопотребления.

Диаграмма значений энергопотребления для существующих и предлагаемых логических элементов СДНФ при разных входных переменных n при напряжении питания 0,8 В с реализацией m=1 логических функций представлена на рисунке 3.66.

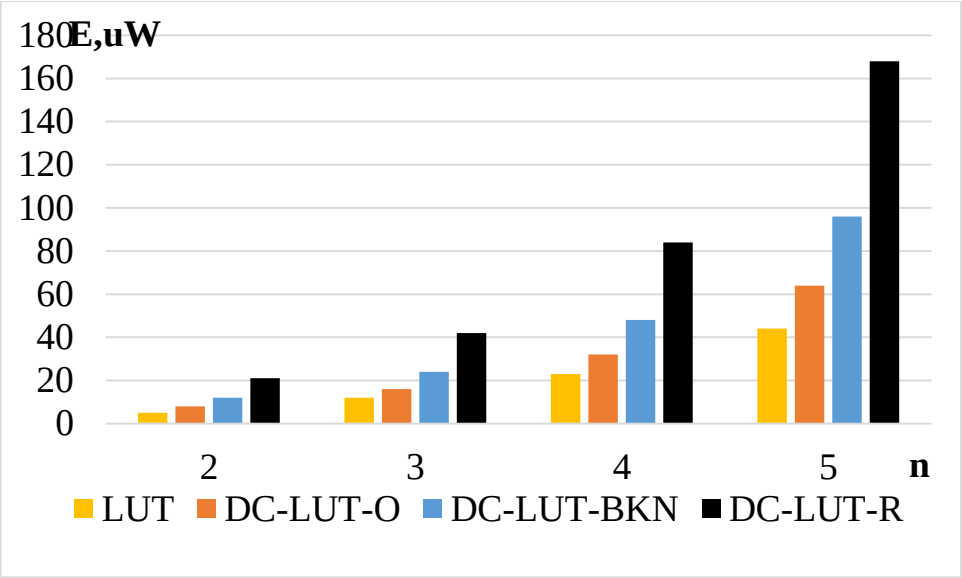


Рисунок 3.66 Энергопотребление логических элементов СДНФ при разных входных переменных n при напряжении 0,8 В; m=1.

Из диаграммы видно, что для реализации одной логической функции m=1 по потреблению приоритетным является LUT. Получим диаграмму значений энергопотребления для существующих и предлагаемых логических элементов СДНФ при разных входных переменных n при напряжении питания 0,5В с реализацией m=3 логических функций представлен на рисунке 3.67.

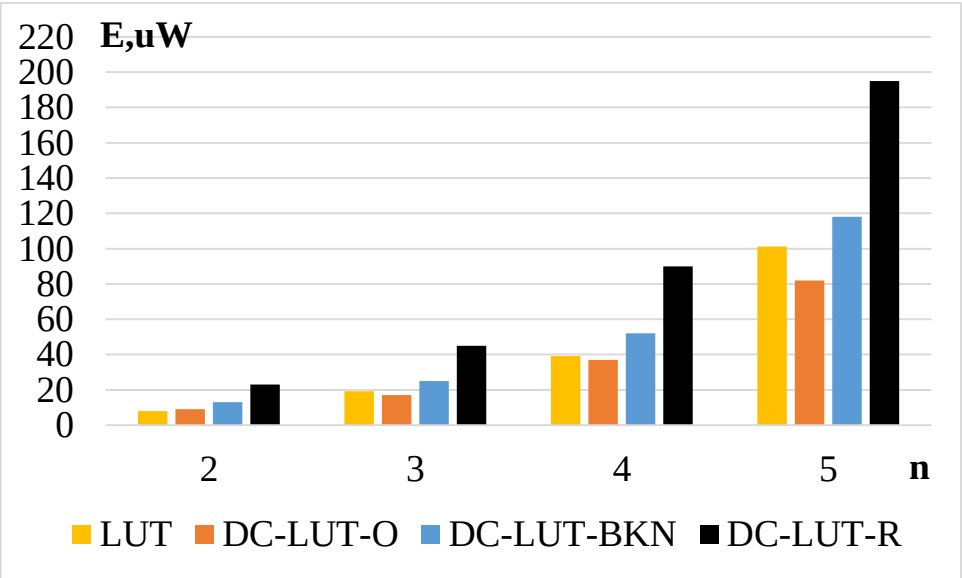


Рисунок 3.67 Энергопотребления логических элементов СДНФ при разных входных переменных n при напряжении 0,8 В; m=3.

Из диаграммы видно, что уже для реализации 3-х логических функции m=3 по потреблению приоритетным является предлагаемый DC-LUT-O, а при увеличении количества входных переменных разница в потребляемом токе по

сравнению с существующим LUT только увеличивается. Выраженно это тем, что для реализации 3-логических функций используется 3 ЛЭ LUT, а в случае с ЛЭ DC к дереву транзисторов добавляются 2 блока настройки который потребляет значительно меньше чем дерево транзисторов LUT.

Диаграмма значений энергопотребления для существующих и предлагаемых логических элементов ДНФ при разных входных переменных n, при напряжении питания 0,8В с реализацией m=3 логических функций представлена на рисунке 3.68.

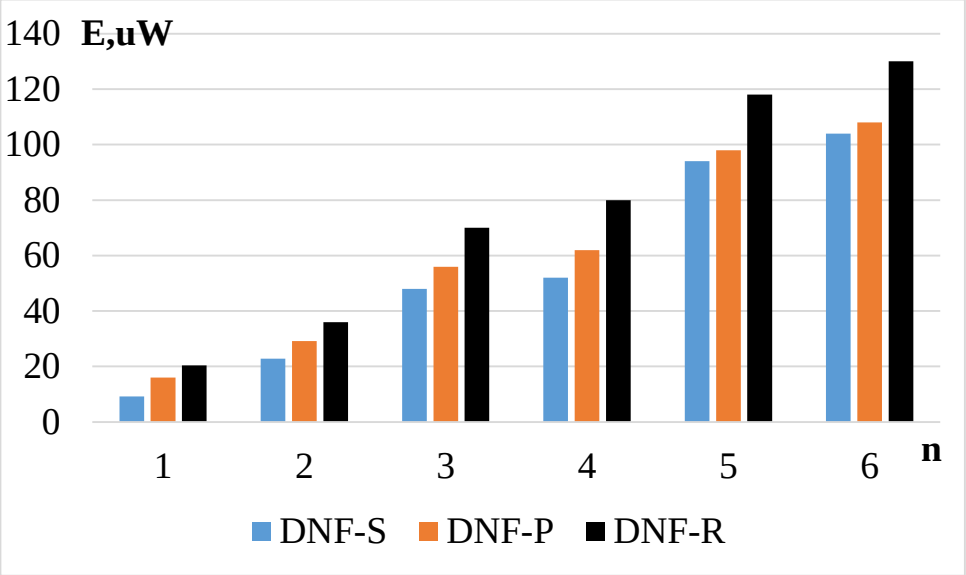


Рисунок 3.68 Энергопотребление логических элементов СДНФ при разных входных переменных n при напряжении 0,8 В; m=1.

Из диаграммы видно, что по потреблению приоритетным является предлагаемый DNF-S с последовательным подключением, а при увеличении количества входных переменных разница в потребляемом токе по сравнению с существующим LUT только увеличивается.

Полученные значения токов потребления и значения задержки в дальнейшем применим для решения поставленной задачи диссертационного исследования.

Диаграмма энергопотребления для DC-LUT-O при разных входных переменных n, при напряжении питания 0,8В с реализацией m=3 логических функций представлена на рисунке 3.69 и на рисунке 3.70.

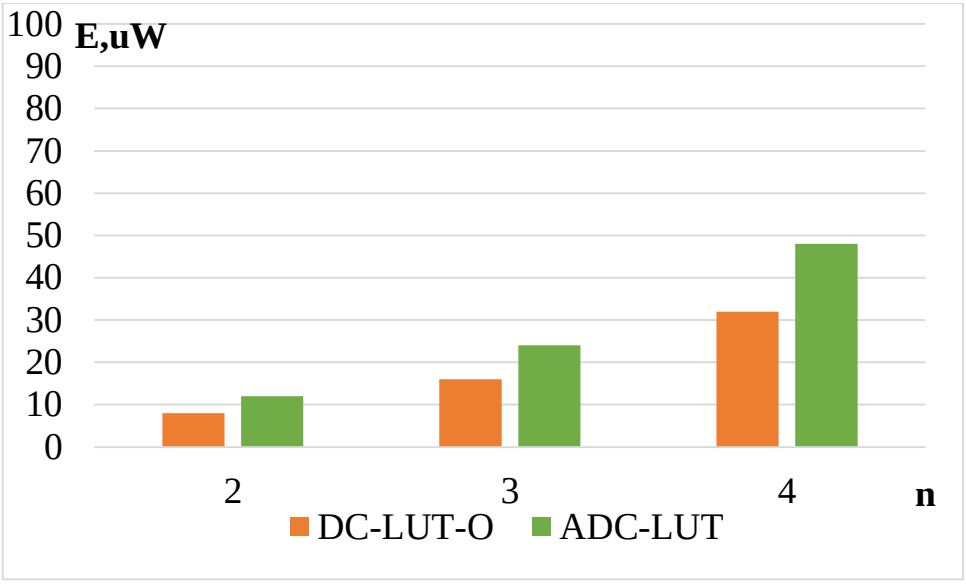


Рисунок 3.69 Энергопотребление логических элементов DC-LUT-O и ADC-LUT при входных переменных n от 1 до 4, при напряжении 0,8В; m=3.

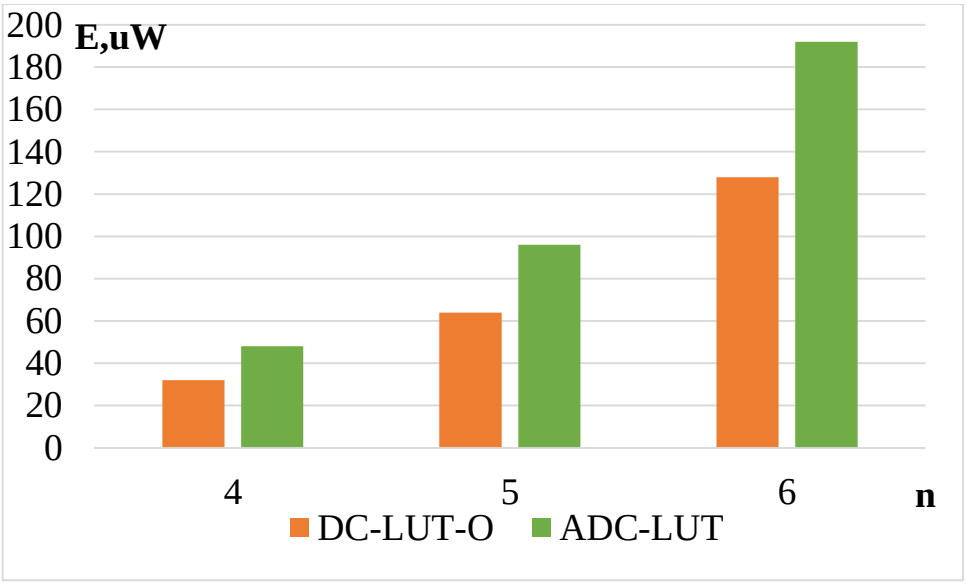


Рисунок 3.70 Энергопотребление логических элементов DC-LUT-O и ADC-LUT при входных переменных n от 5 до 8 при напряжении 0,8В; m=3.

3.6. Топологическое моделирование существующих и предлагаемых ЛЭ

3.6.1.Топологическое моделирование ЛЭ LUT

Для оценки значений занимаемой площади целесообразно выполнить разработку топологии с требованиями доступной технологии 32 нм в системе автоматизированного проектирования специализированных (заказных) интегральных схем MicroWind (версия, свободно распространяемая для обучения) [70,71]. Построение топологии представляет собой формирование слоев в относительных единицах λ (рисунок 3.70) параметры которой задаются технологическими нормами. Ограничения для технологии 32 нм представлены на рисунке 3.71, 3.72.

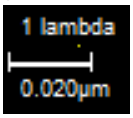


Рисунок 3.71. Расстояние между точками выражается в относительных единицах λ

Design rules for CMOS 32nm - 8 Metal

Design rules and electrical parameters

Layer	Width	Spacing	Surface	Surf capa	Lin capa	Ctk capa	Res	Unsalicid	Thickn	Height	Permitt
	lambda	lambda	lambda2	af/µm2	af/µm	af/µm	ohm	ohm	µm	µm	
nitride	0	0	0								
siOxNy	800	800	0								
metal6	8	15	128	30.00	25.00	12.00	0.10/sq	1.00/sq	0.50	4.00	2.20
via5	5	5	0				1.00/via		0.50	3.65	4.00
metal5	8	15	32	30.00	20.00	12.00	0.20/sq	1.00/sq	0.35	3.30	2.20
via4	3	4	0				1.00/via		0.50	2.95	4.00
metal4	3	4	32	30.00	20.00	12.00	0.20/sq	1.00/sq	0.35	2.60	2.20
via3	3	4	0				2.00/via		0.50	2.25	4.00
metal3	3	4	32	30.00	20.00	12.00	0.20/sq	1.00/sq	0.35	1.90	2.20
via2	3	4	0				2.00/via		0.50	1.55	4.00
metal2	3	4	32	30.00	20.00	12.00	0.20/sq	1.00/sq	0.35	1.20	2.20
via	3	4	0				1.00/via		0.50	0.85	4.00
metal	3	4	32	28.00	42.00	10.00	0.20/sq	1.00/sq	0.35	0.50	2.20
poly	2	3	8	80.00			4.00/sq	40.00/sq	0.10	0.15	4.00
poly2	2	2	8				4.00/sq	1.00/sq	0.00	1.00	4.00
contact	2	3	0				2.00/via		0.50	0.00	4.00
diffn	4	4	24	350.00	100.00		25.00/sq	250.00/sq	0.10	0.00	4.00
diffp	4	4	24	300.00	100.00		30.00/sq	300.00/sq	0.10	0.00	4.00
nwell	10	11	144	250.00			120.00/sq		0.50	0.00	4.00
oxide				28000.00					1.80nm	(4.00nm)	4.00

Рисунок 3.72. Геометрические ограничения построения топологии для технологии 32 нм.

Формирование транзистора n типа заключается в формировании 4 слоев [93]: Слой затвора, слой Сток-Исток, Слой метализации, слой контакных окон. Формирование транзистора p типа дополнительно включает формирование слоя кармана n-типа в котором строим p-канальный транзистор. Структура транзисторов NMOS и PMOS представлена на рисунке 3.73. Топология 3 входового LUT представлена на рисунке 3.74.

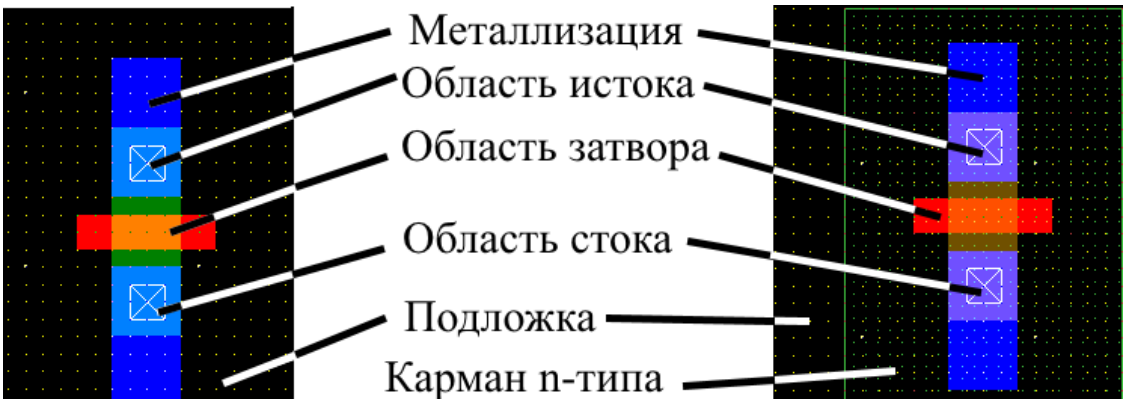


Рисунок 3.73. Структура топологии NMOS и PMOS

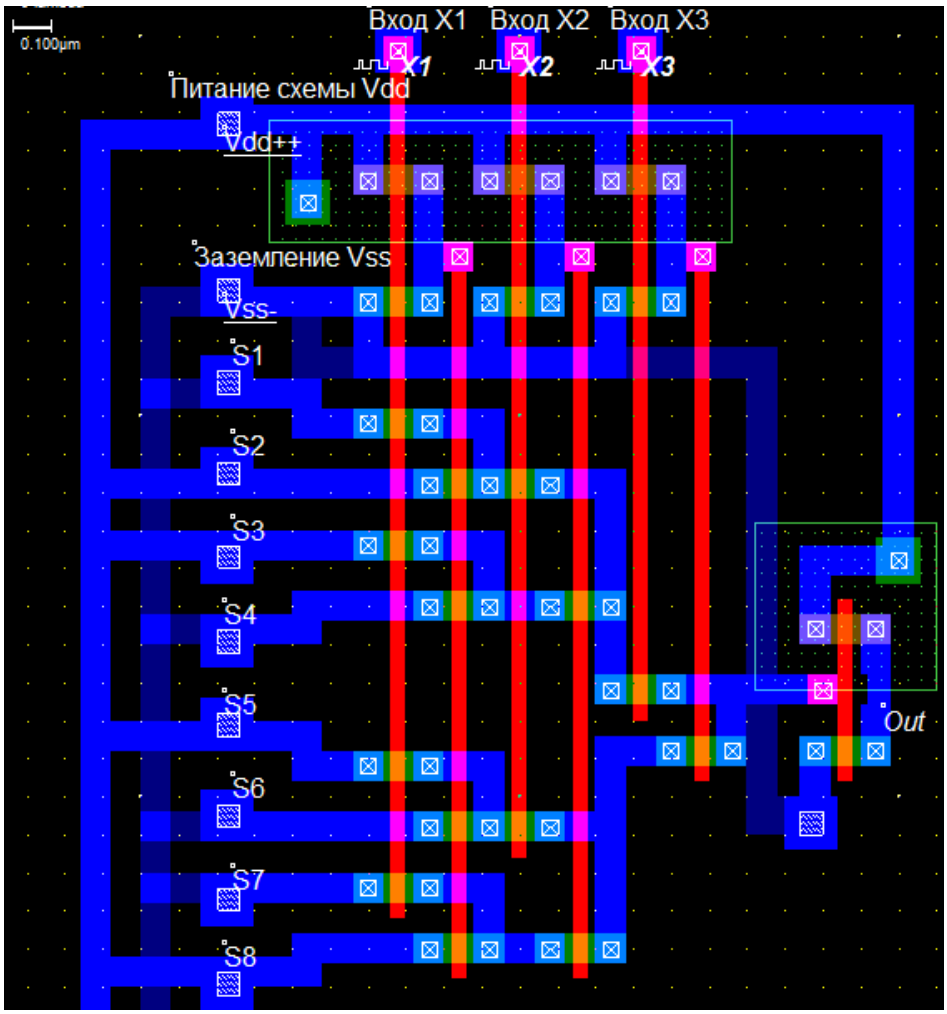


Рисунок 3.74. Топология логического элемента 3-входового LUT, 32 нм.

X1,X2,X3 - входы переменных логического элемента которые поступают на затворы транзисторов. Контакты S1-S8 подключаются к ячейкам SRAM в которых загружена таблица истинности логической функции. Область Out это выход инвертора и представляет собой вычисленный выходной сигнал логического элемента 3-LUT.

Моделирование на топологии представлено на рисунке 3.75.

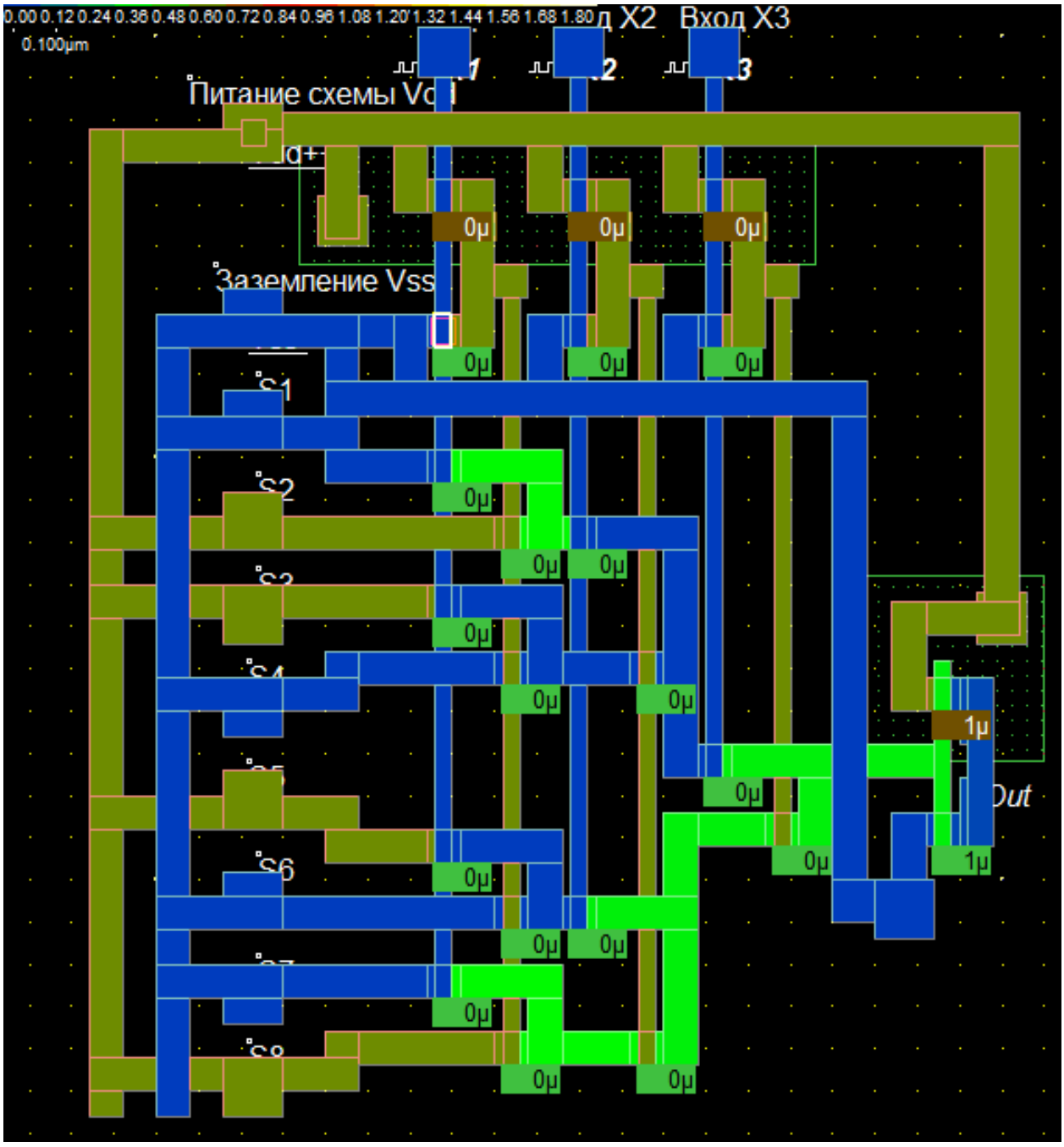


Рисунок 3.75. Моделирование топологии 3-LUT входной набор (000).

На вход подается набор X1=0, X2=0, X3=0 через инверторы входа происходит активация нижней цепочки дерева, которая в свою очередь

передает информацию из ячейки S8 (уровень 1) на вход выходного инвертора. На выходе наблюдаем инвертированный сигнал из ячейки S8 (уровень 1) Out=0. Рассмотрим работу логического элемента на рисунке 3.76, когда на вход подается набор X1=1, X2=1, X3=1. Происходит активация верхней цепочки дерева, которая в свою очередь передает информацию из ячейки S1 (уровень 0) на вход выходного инвертора. На выходе наблюдаем инвертированный сигнал из ячейки S1 (уровень 1) Out=1. Аналогично схема работает и при других наборах.

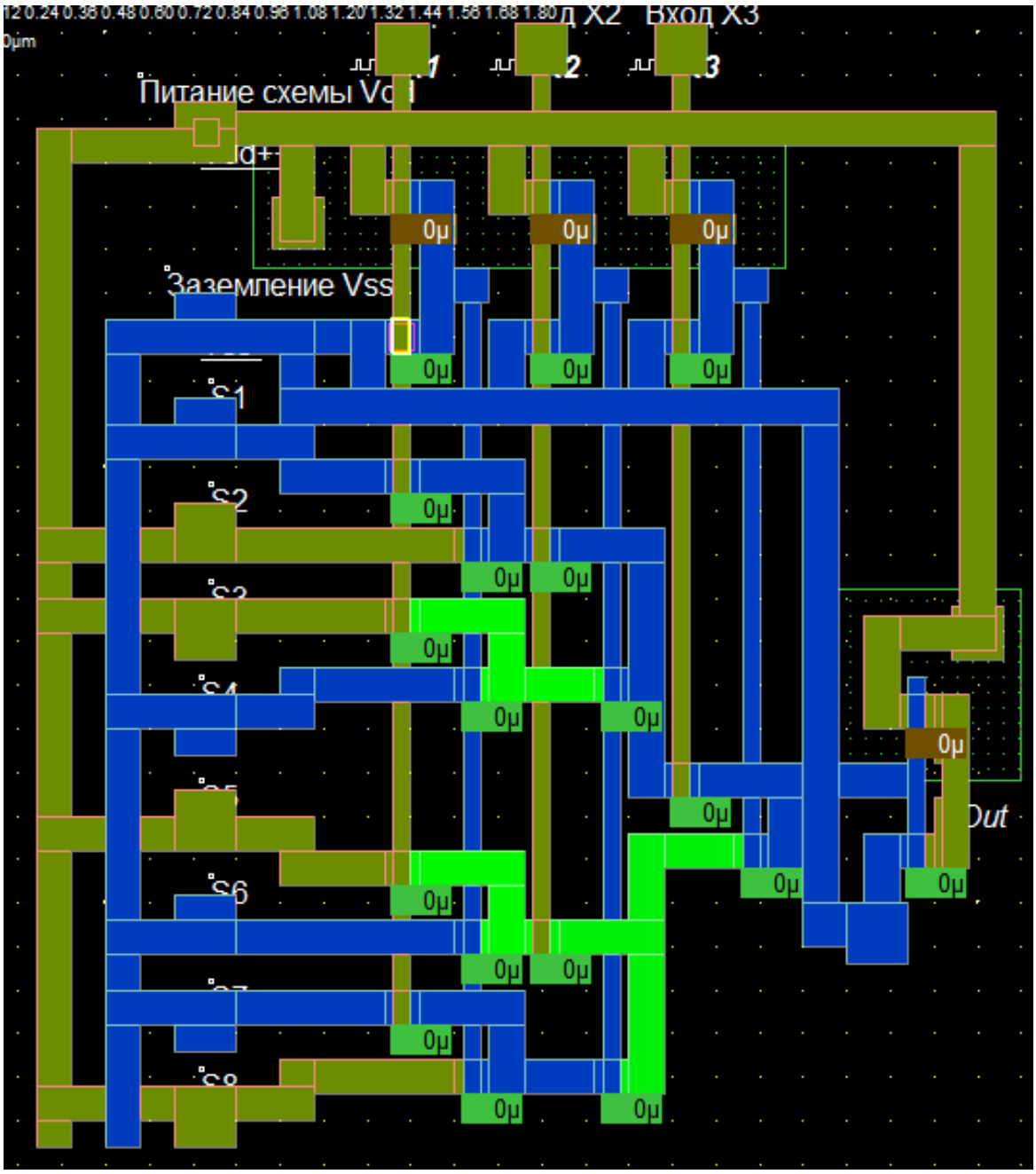


Рисунок 3.76. Моделирование топологии 3-LUT входной набор (111).

Осциллограмма выхода Out представленная на рисунке 3.77 подтверждает правильность работы известного ЛЭ.

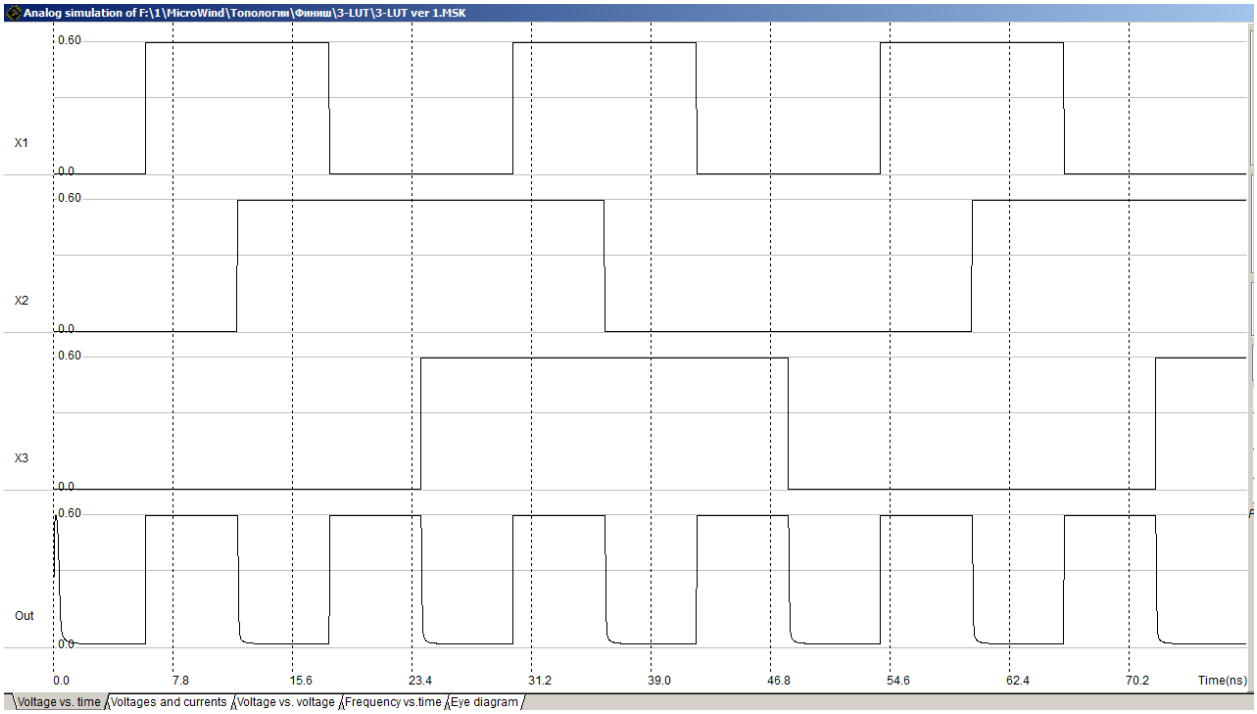


Рисунок 3.77. Осциллограмма работы логического элемента 3-LUT реализующего функцию «Исключающее ИЛИ».

При подаче всех вариантов входного набора X1, X2, X3 на выходе формируется значение логической функции «Исключающее ИЛИ». С помощью осциллограмм снимаются значения задержки по фронту и по срезу. Так же с помощью MicroWind снимаются значения занимаемой площади «Layout Size» представлено на рисунке 3.78.

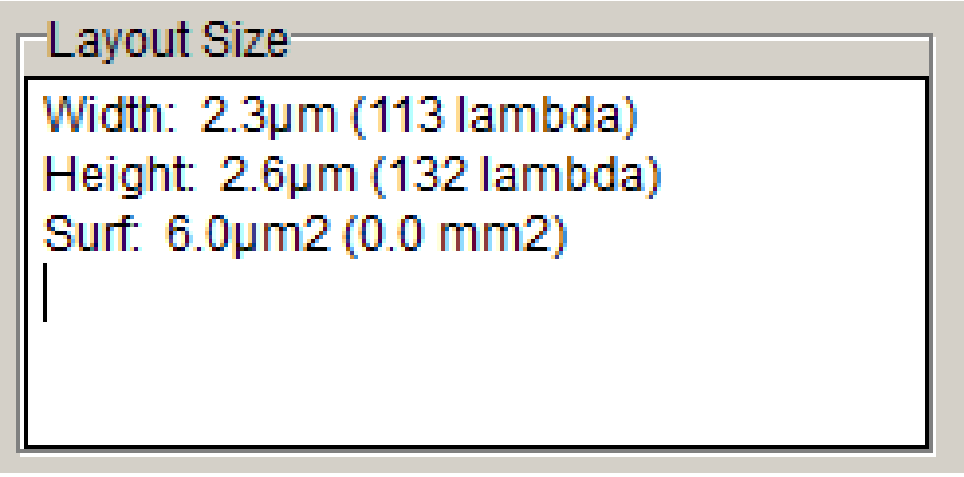


Рисунок 3.78. Параметры занимаемой площади логического элемента 3-LUT.

Построены топологии 1,2,4,5,6 входовых известных логических элементов LUT. LUT на 6 входов представлен на рисунке 3.79.

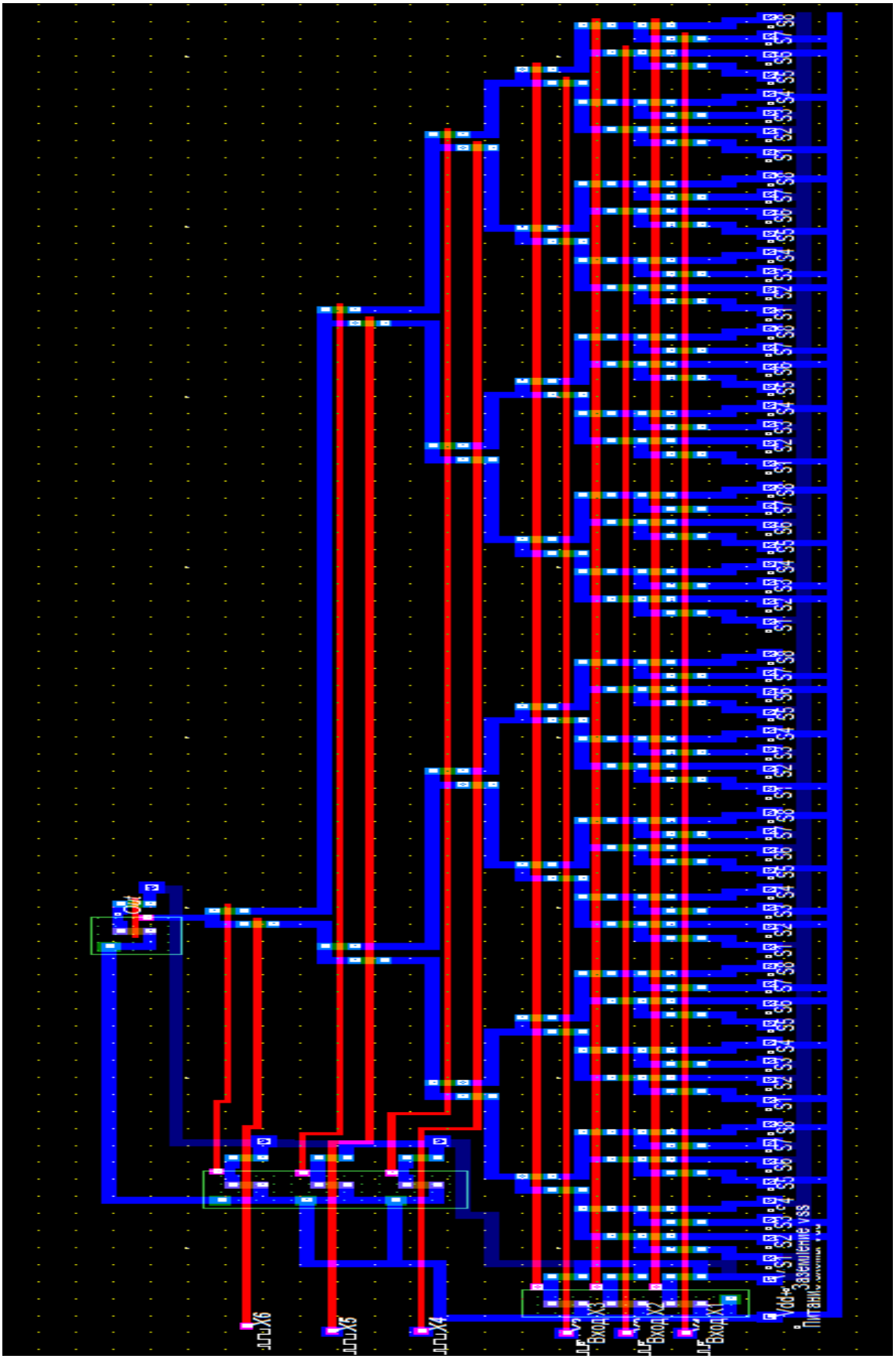


Рисунок 3.79. Топология логического элемента 6-LUT, 32 нм.

3.6.2. Топологическое моделирование логического элемента – DC-LUT-BKN

Для оценки занимаемой площади и других параметров аналогично была построена топология предлагаемого 3 входового DC-LUT-BKN [72], представленая на рисунке 3.80.

X1,X2,X3 это входы переменных логического элемента которые поступают на затворы транзисторов. Область Out это выход блока настройки «Setup» и представляет собой вычисленный выходной сигнал логического элемента 3-LUT.

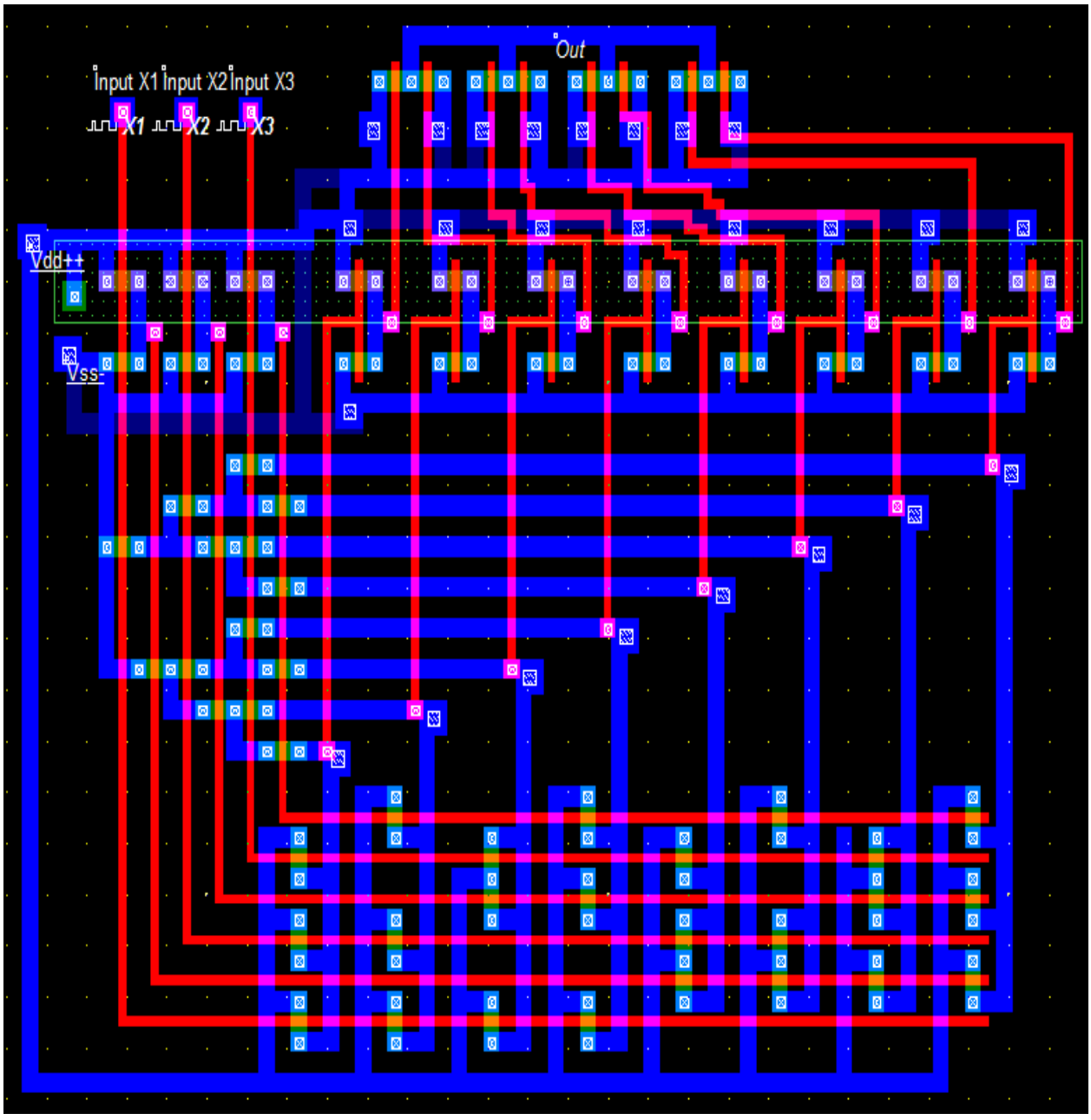


Рисунок 3.80. Топология 3-DC-LUT-BKN, 32 нм.

С помощью MicroWind сняты значения занимаемой площади «Layout Size» для 3-DC-LUT-BKN представлено на рисунке 3.81.

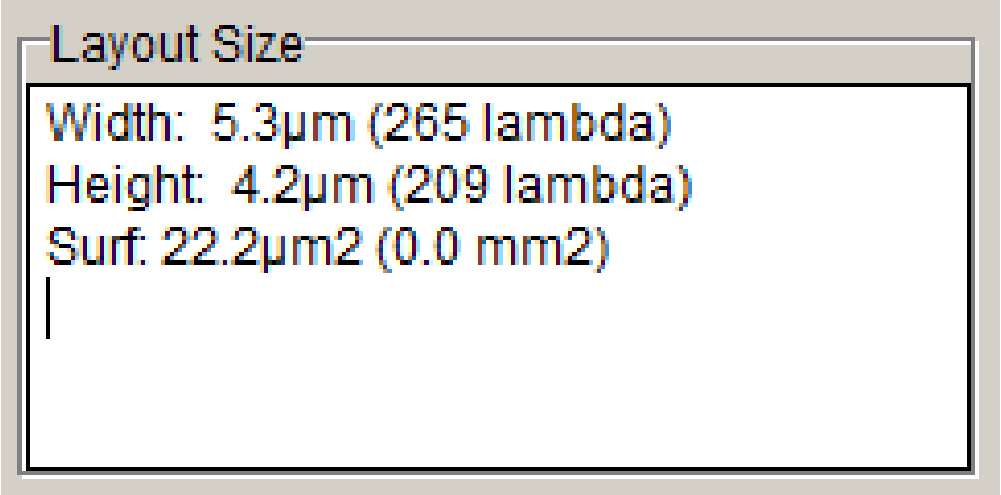


Рисунок 3.81. Параметры занимаемой площади логического элемента 3-DC-LUT-BKN.

Аналогично с помощью MicroWind разработаны топологии и получены значения площади и задержек для 4,5,6-DC-LUT-BKN. Осциллограмма работы DC-LUT-BKN представлена на рисунке 3.82 Топология 6-DC-LUT-BKN представлена на рисунке 3.83.

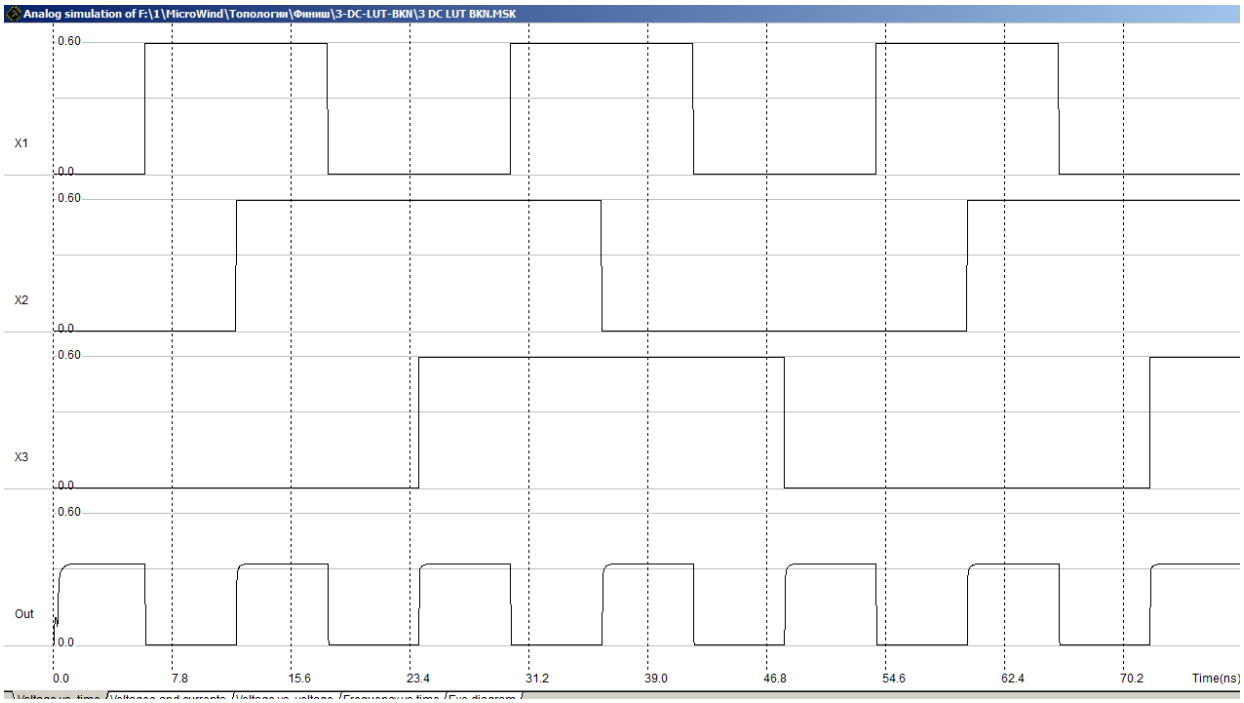


Рисунок 3.82. Осциллограмма работы логического элемента 3-DC-LUT–BKN реализующего функцию «Исключающее ИЛИ».

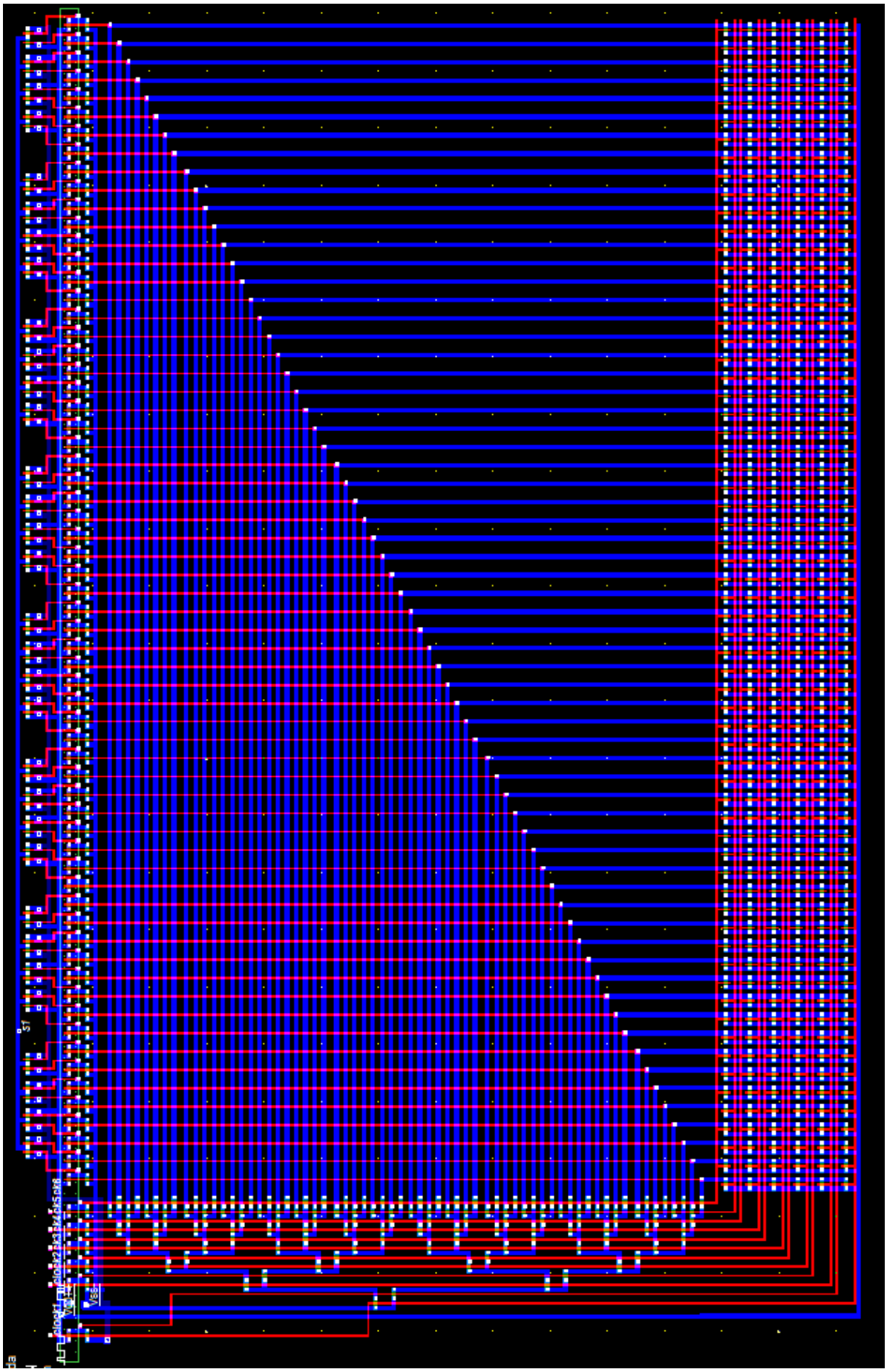


Рисунок 3.83. Топология 6-входового 6-DC-LUT-BKN, 32 нм.

3.6.3. Топологическое моделирование логического элемента – DC-LUT-O

Разработанная топология предлагаемого 3 входового DC-LUT-O [72] представлена на рисунке 3.84.

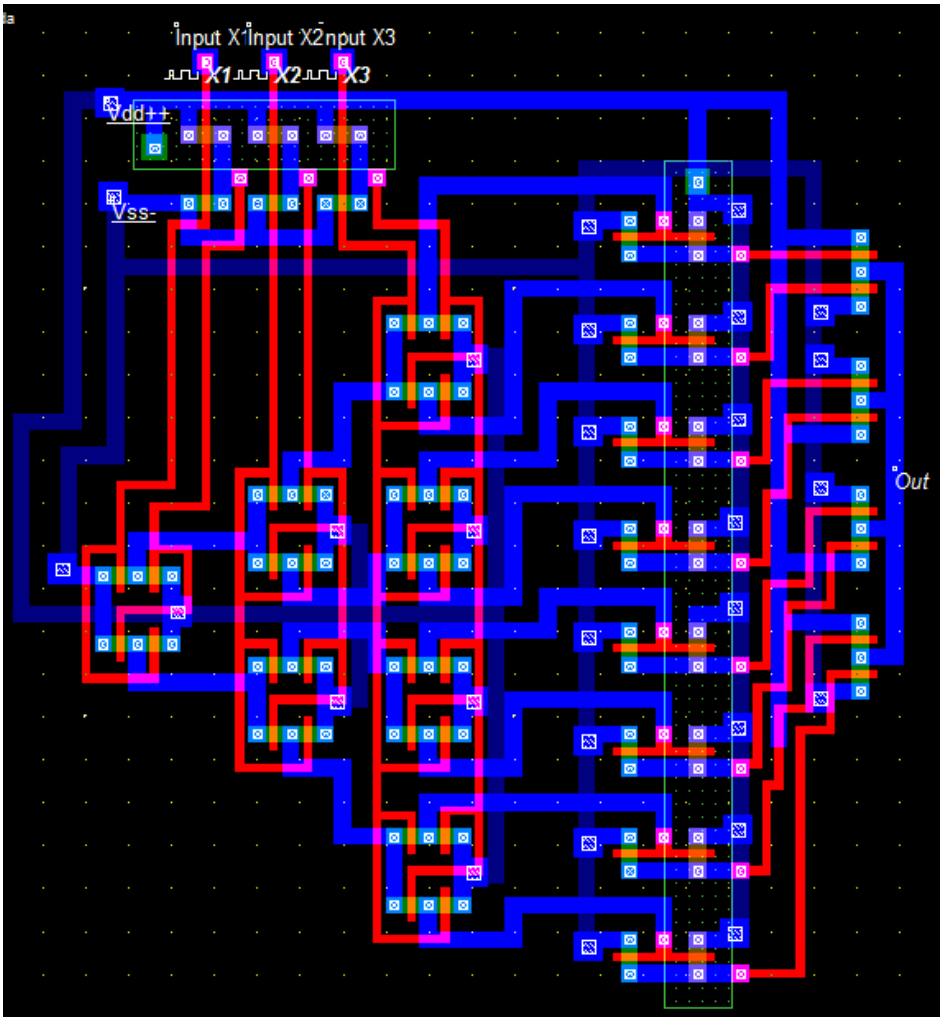


Рисунок 3.84. Топология 3-DC-LUT-O, 32 нм.

Параметры занимаемой площади логического элемента 3-DC-LUT-O представлены на рисунке 3.85.

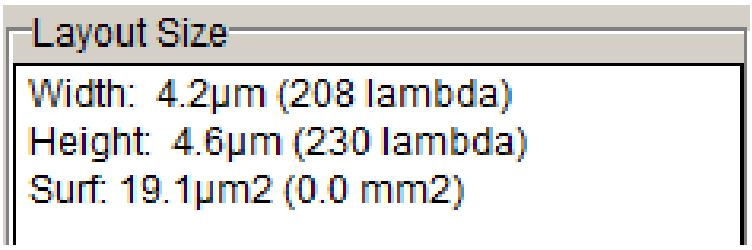


Рисунок 3.85. Параметры занимаемой площади логического элемента 3-DC-LUT-O.

Аналогично с помощью MicroWind разработаны топологии для 4,5,6-DC-LUT). Топология 6-DC-LUT-BKN представлена на рисунке 3.86.

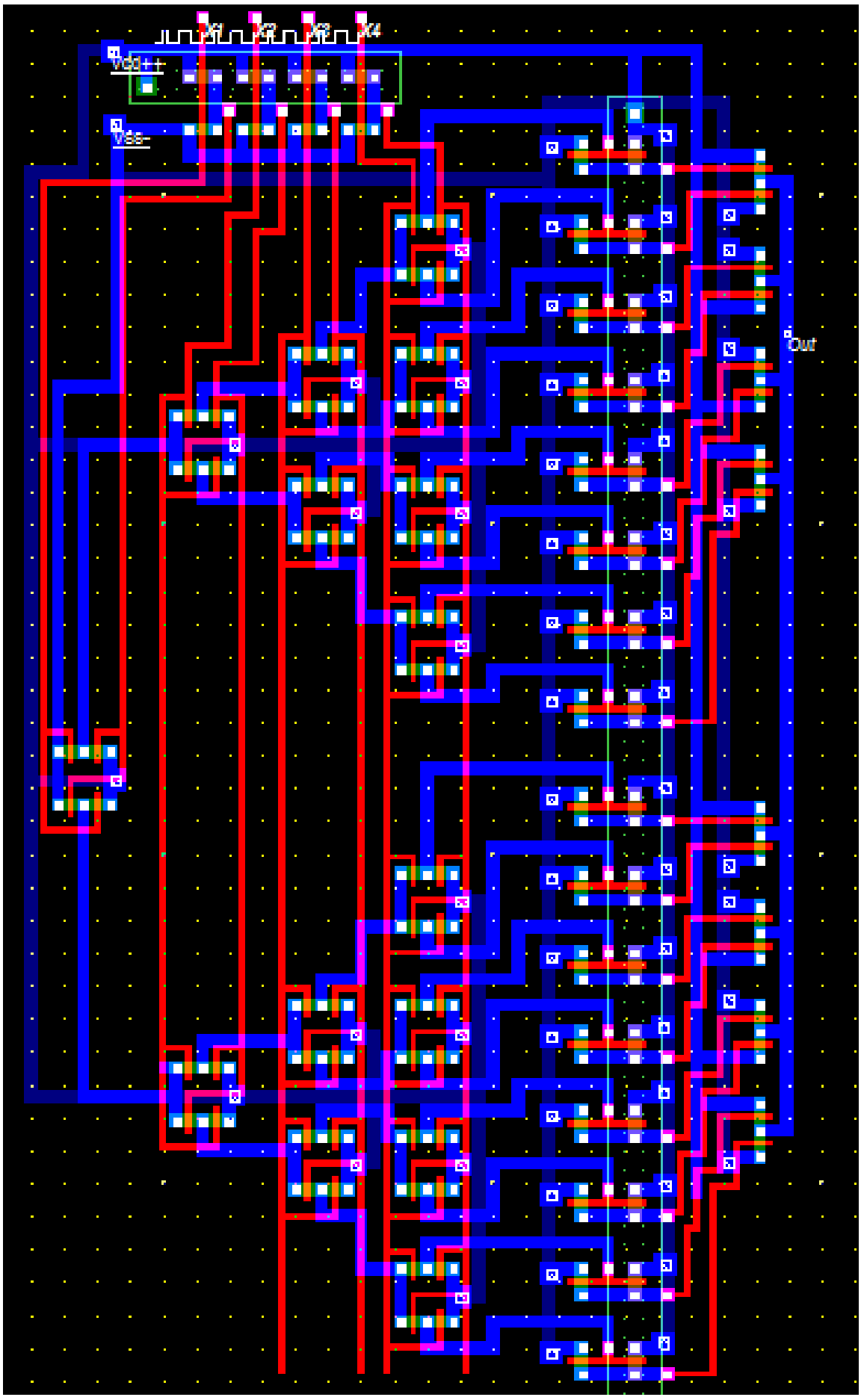


Рисунок 3.86. Топология 4-DC-LUT-O, 32 нм.

3.6.4. Топологическое моделирование логических элементов DNF

Разработанная топология известного 2-DNF-R, реализующего одну конъюнкцию двух переменных, представлена на рисунке 3.87.

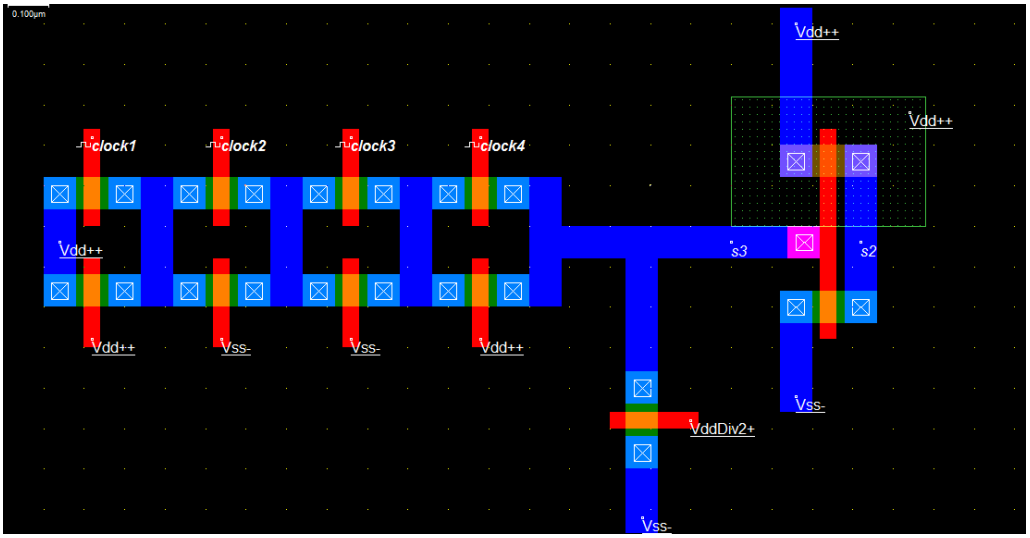


Рисунок 3.87. Топология 2-DNF-R, 32 нм.

Топологическое моделирование предлагаемого логического элемента – DNF-S представлено на рисунке 3.88.

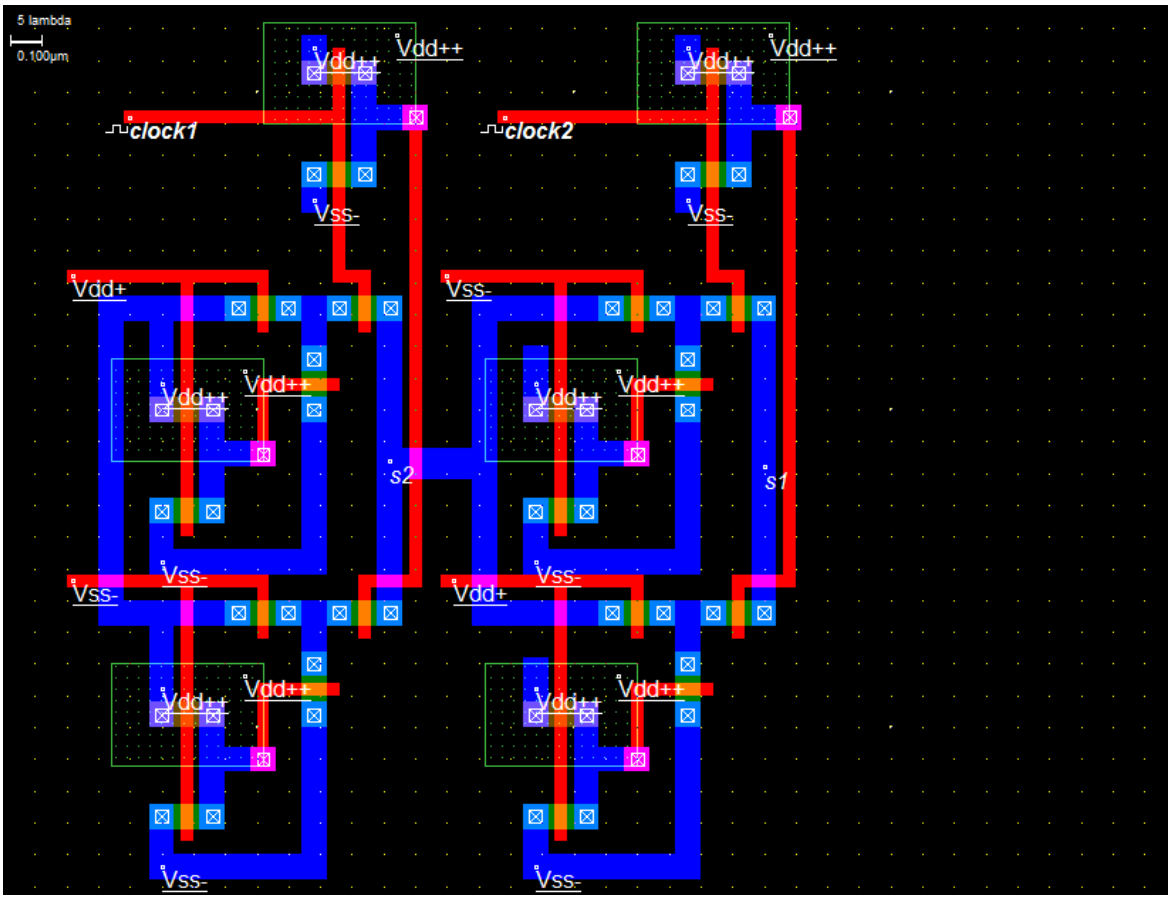


Рисунок 3.88. Топология 2-DNF-S, 32 нм.

Топологическое моделирование предлагаемого логического элемента – DNF-P представлено на рисунке 3.89.

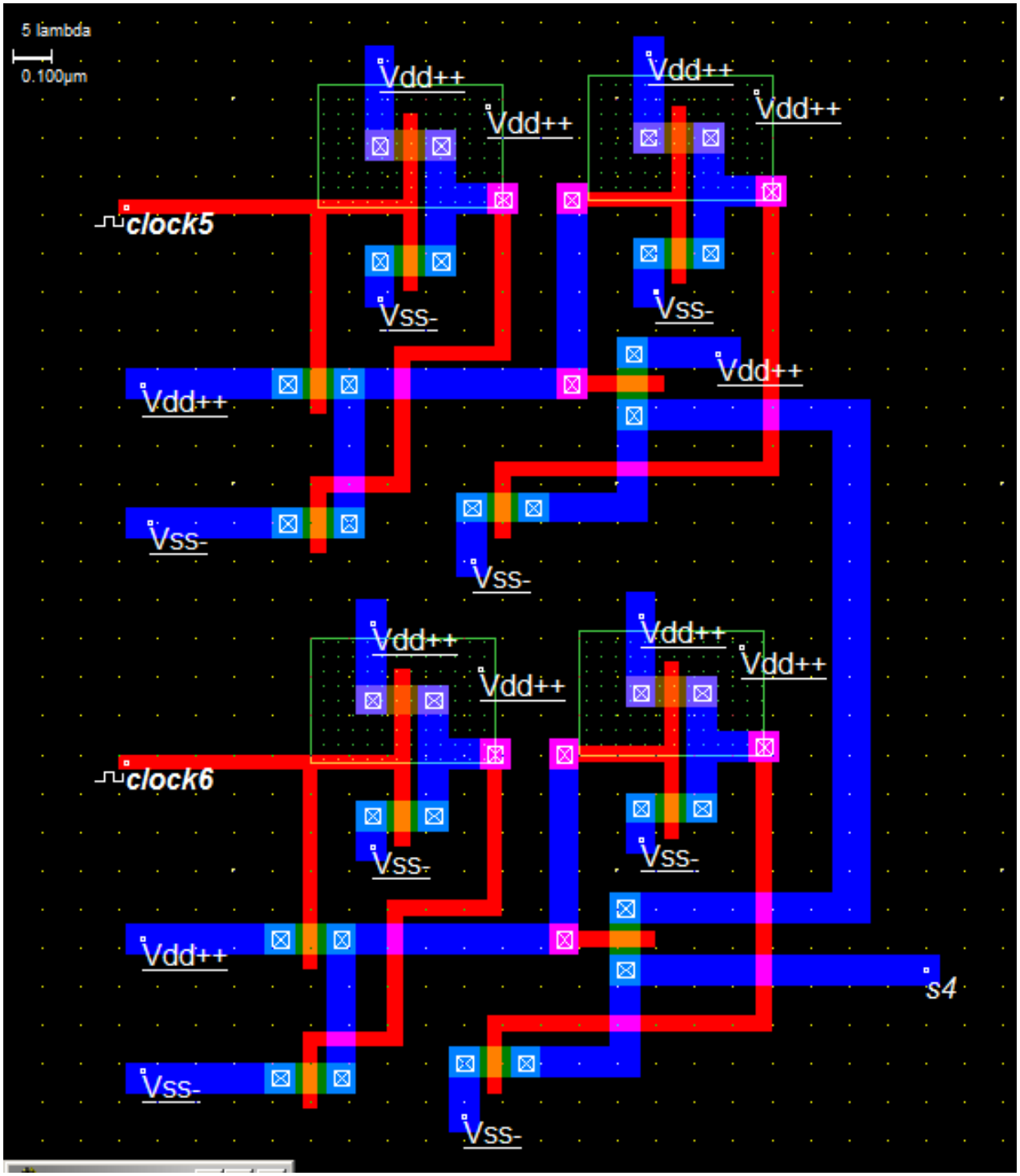
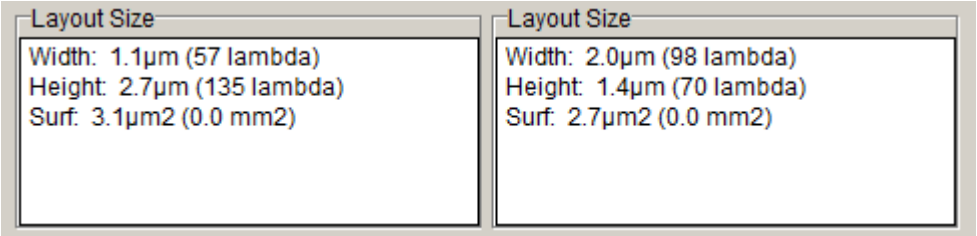


Рисунок 3.89. Топология 2-DNF-P, 32 нм.

Параметры занимаемой площади логического элемента DNF-P, DNF-S представлены на рисунке 3.90.



а) б)

Рисунок 3.90. Параметры занимаемой площади логических элементов а)DNF-P и б)DNF-S

3.6.5. Результаты топологического моделирования предложенных ЛЭ DC-LUT-O, DC-LUT-BKN, DNF-LUT-S, DNF-LUT-P

Выполненное топологическое моделирование [72,73] подтверждает работоспособность предложенных устройств DC-LUT-O, DC-LUT-BKN и DNF-S, DNF-P на различных частотах (до 2 гГц) и при напряжении питания (0,8В). Получены диаграммы значений площади топологии и задержки ЛЭ LUT, DC-LUT-O, DC-LUT-BKN - рисунок 3.91 и 3.92.

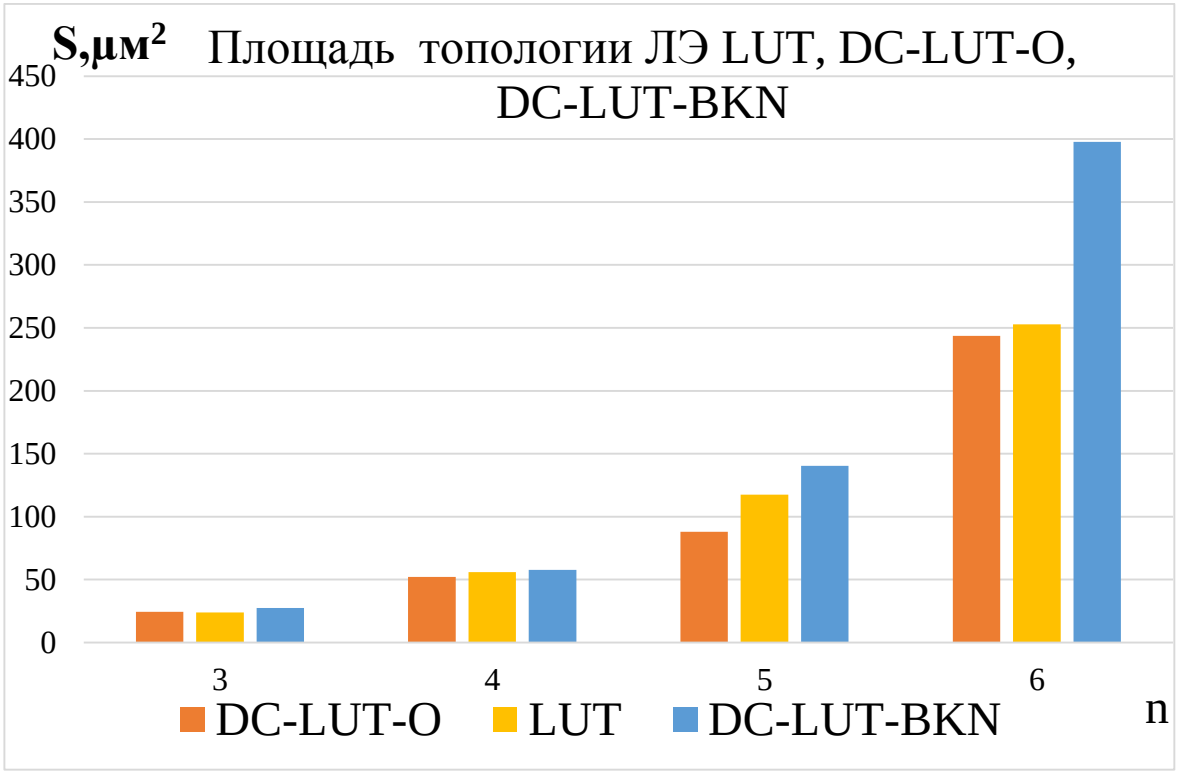


Рис. 3.91. Результаты моделирования топологии LUT и DC-LUT для разного количества входных переменных n, количество реализуемых логических

функций m=4: значения площади топологий ЛЭ LUT, DC-LUT-O, DC-LUT-BKN

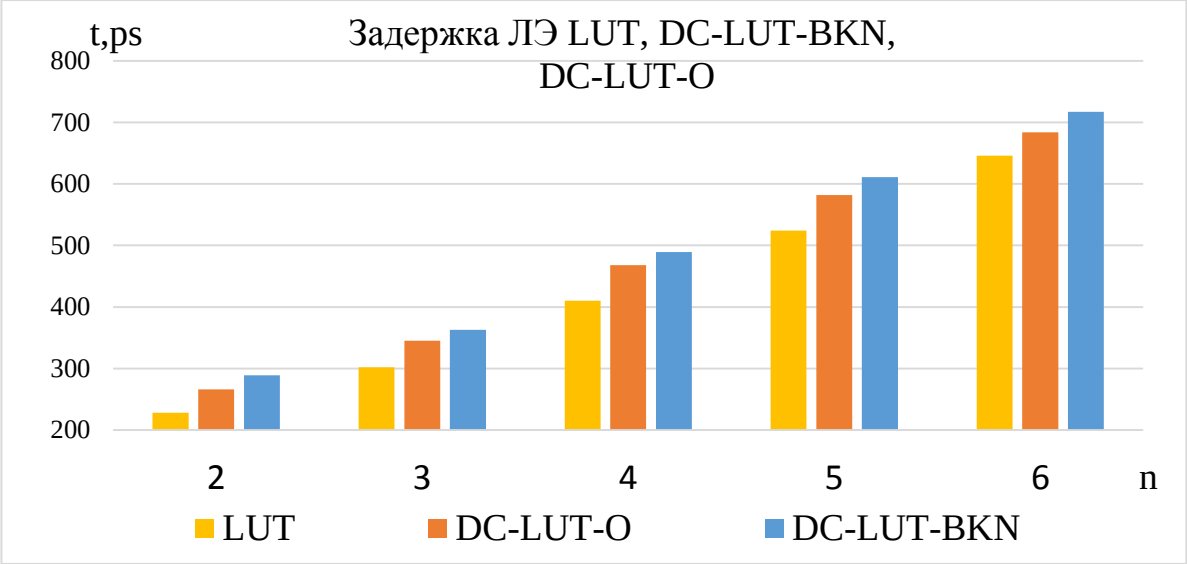


Рис. 3.92. Результаты моделирования топологии LUT и DC-LUT для разного количества входных переменных n, количество реализуемых логических функций m=4: значения максимальной задержки ЛЭ LUT, DC-LUT-O, DC-LUT-BKN

Численные значения занимаемой площади ЛЭ LUT, DC-LUT-O, DC-LUT-BKN для m=4 приведена в таблице 3.1

Таблица 3.1

n	DC-LUT-BKN	DC-LUT-O	LUT
1	6,6	7,8	11,2
2	13,8	11,8	15,2
3	27,4	24,3	24
4	57,7	52,1	56
5	140,3	88	117,6
6	397,7	243,6	252,8

Численные значения занимаемой площади ЛЭ DNF-S, DNF-P приведена в таблице 3.2

Получены диаграммы значений площади топологии и задержки ЛЭ DNF-S, DNF-P - рисунок 3.93, 3.94 и 3.95.

Таблица 3.2 – Значения занимаемой площади ЛЭ DNF от n переменных и v конъюнкций.

Таблица 3.2

Площадь топологии ЛЭ DNF μm^2			
n	v	DNF-P	DNF-S
1	1	2,7	3,1
2	2	10,8	12,4
3	4	32,4	37,2
4	8	86,4	99,2
5	16	216	248
6	32	518,4	595,2

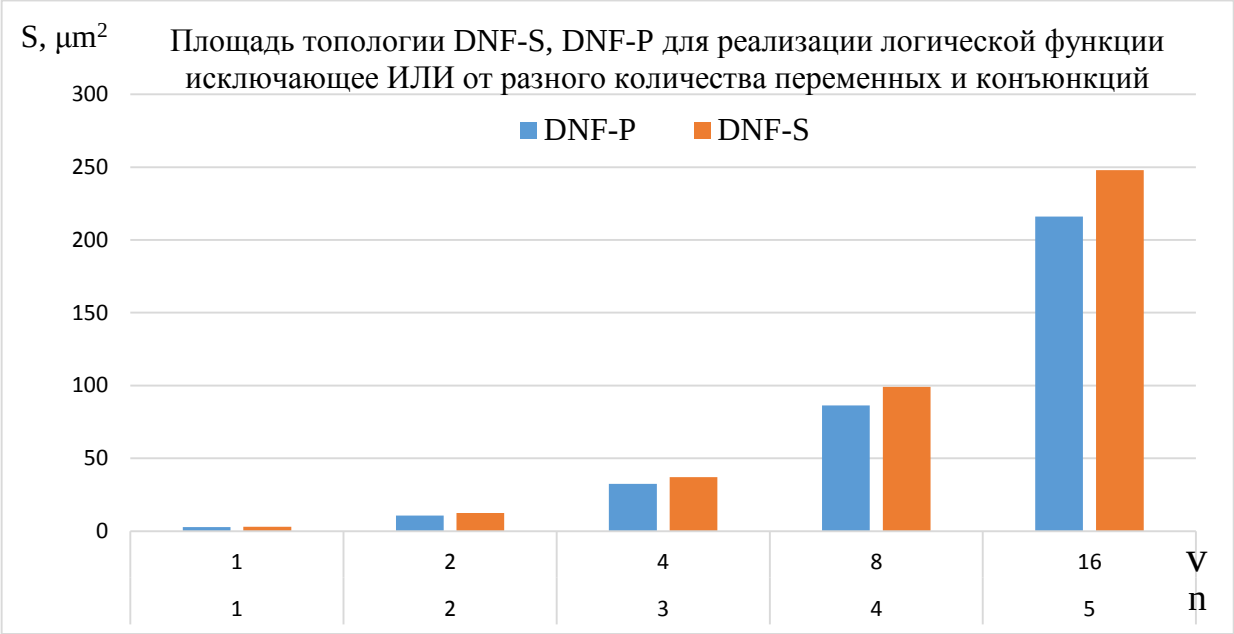


Рис. 3.93. Результаты моделирования топологии ЛЭ DNF-S, DNF-P для параметров n и v.

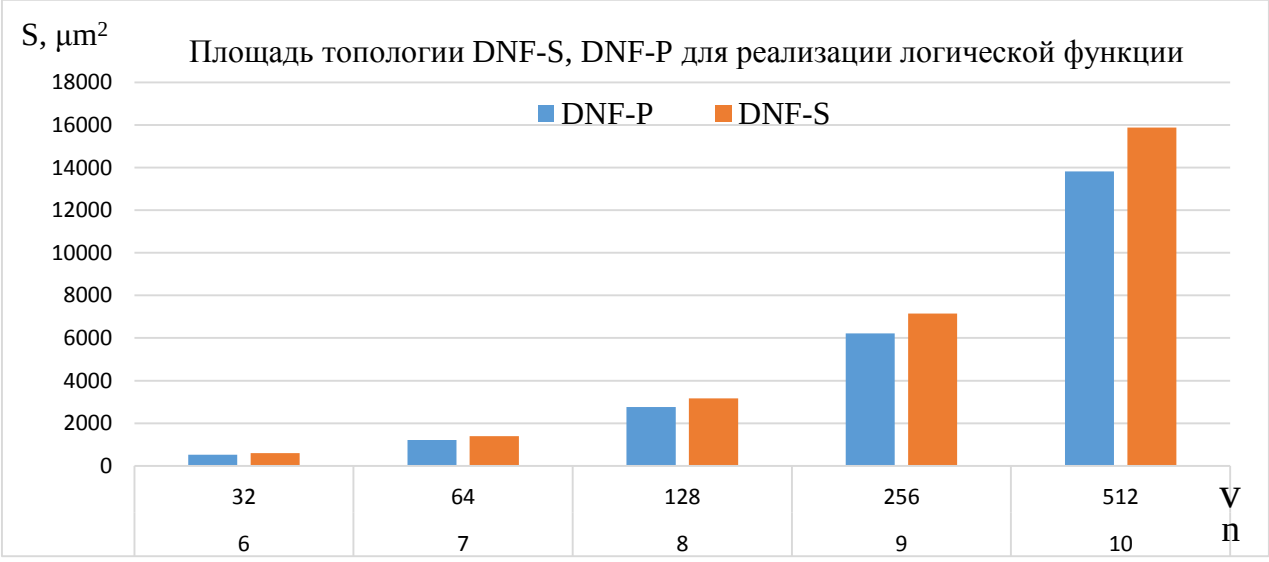


Рис. 3.94. Результаты моделирования топологии ЛЭ DNF-S, DNF-P для параметров n и v.

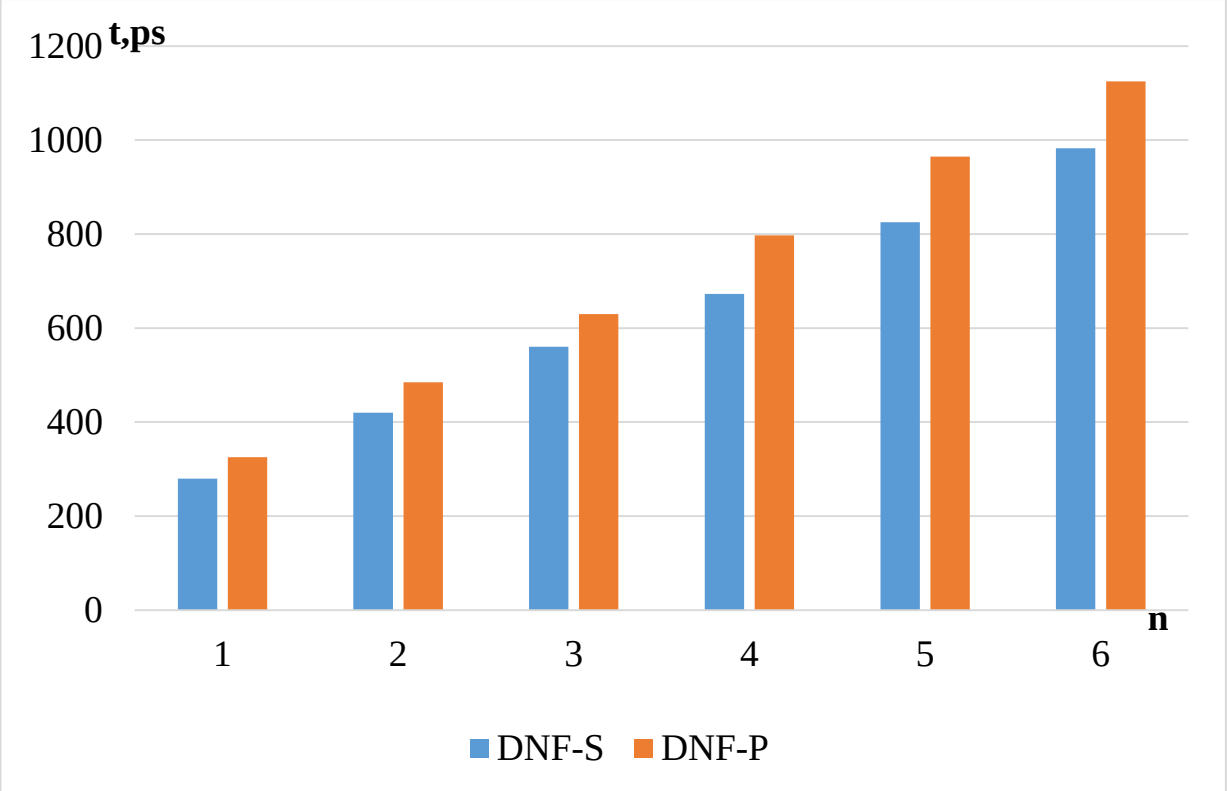


Рисунок 3.95 Результаты моделирования топологии LUT и DC-LUT для разного количества входных переменных n, количество реализуемых логических функций m=4, значения максимальной задержки

3.6.6. Топологическое моделирование логического элемента ADC-LUT

Модель топологии базового ADC-LUT на одну переменную [72,73] представлена на рисунке 3.96.

При наличии на контакте S уровня 1 происходит конфигурация ЛЭ ADC в режим работы LUT. X1 и X2 моделируют входные переменные, d1-d4 подключаются к ячейкам SRAM1- SRAM4 (не показаны) в которых загружена таблица истинности вычисляемой функции. Zout это выход ADC в режиме LUT. При наличии на контакте S уровня 0 происходит конфигурация ЛЭ ADC в режим работы DC-LUT выходы которого dout1-dout4.

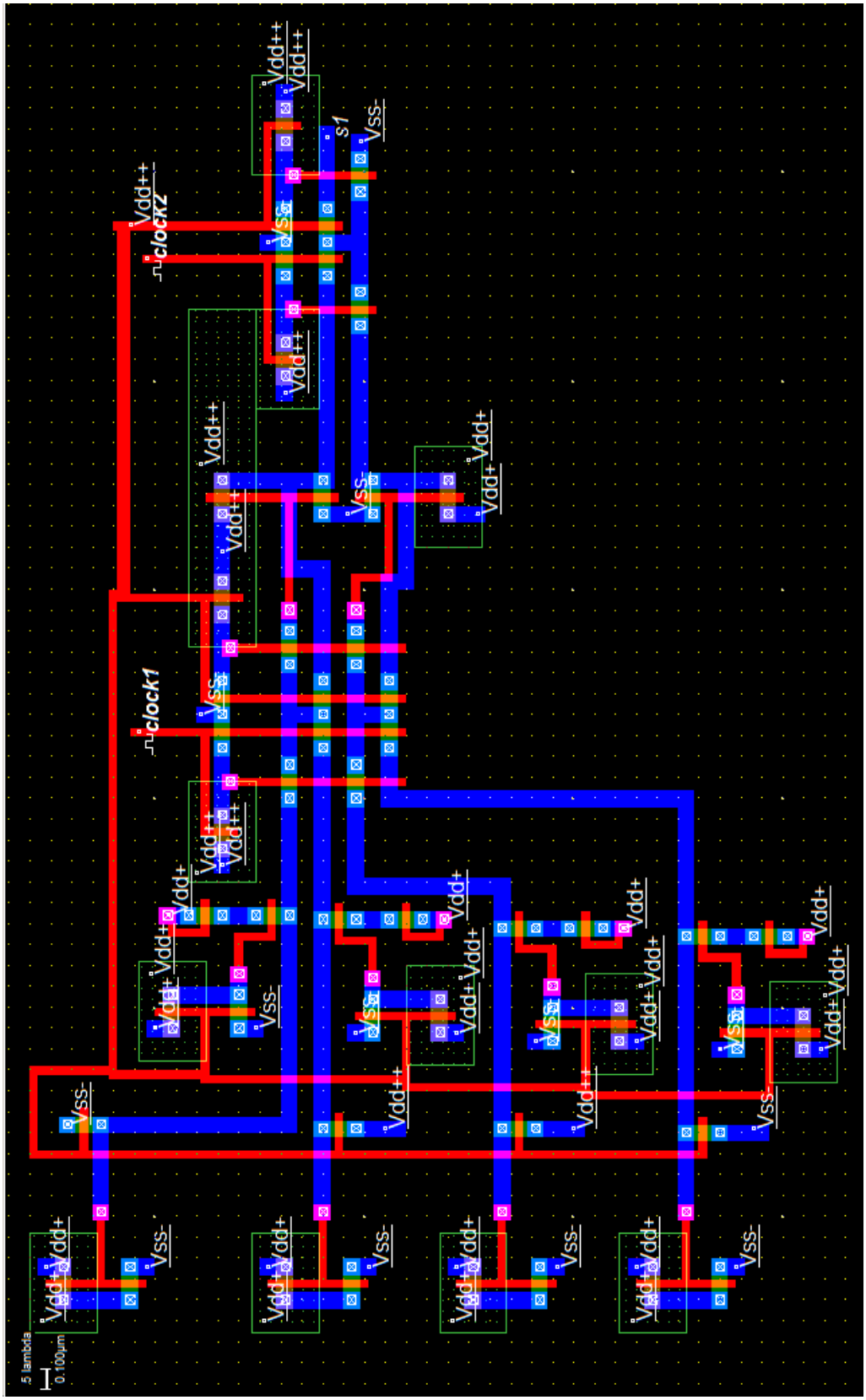


Рисунок 3.96. Топология базового 2-ADC-LUT для на две переменные, 32 нм.

С помощью MicroWind получены диаграммы значений занимаемой площади и задержки, которые представлены на рисунке 3.97 и рисунке 3.98 и рисунке 3.99.

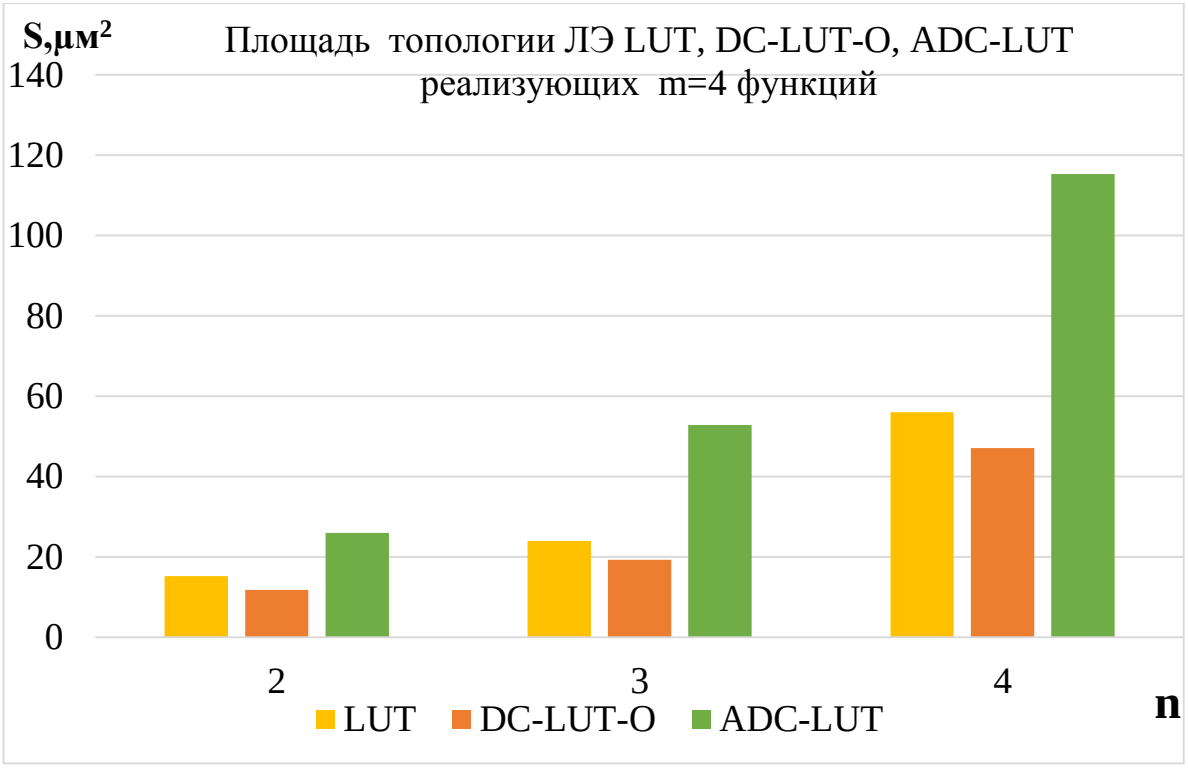


Рисунок 3.97. Занимаемая площадь ADC-LUT для различных n, в сравнении с ЛЭ LUT, DC-LUT-O для m=4.

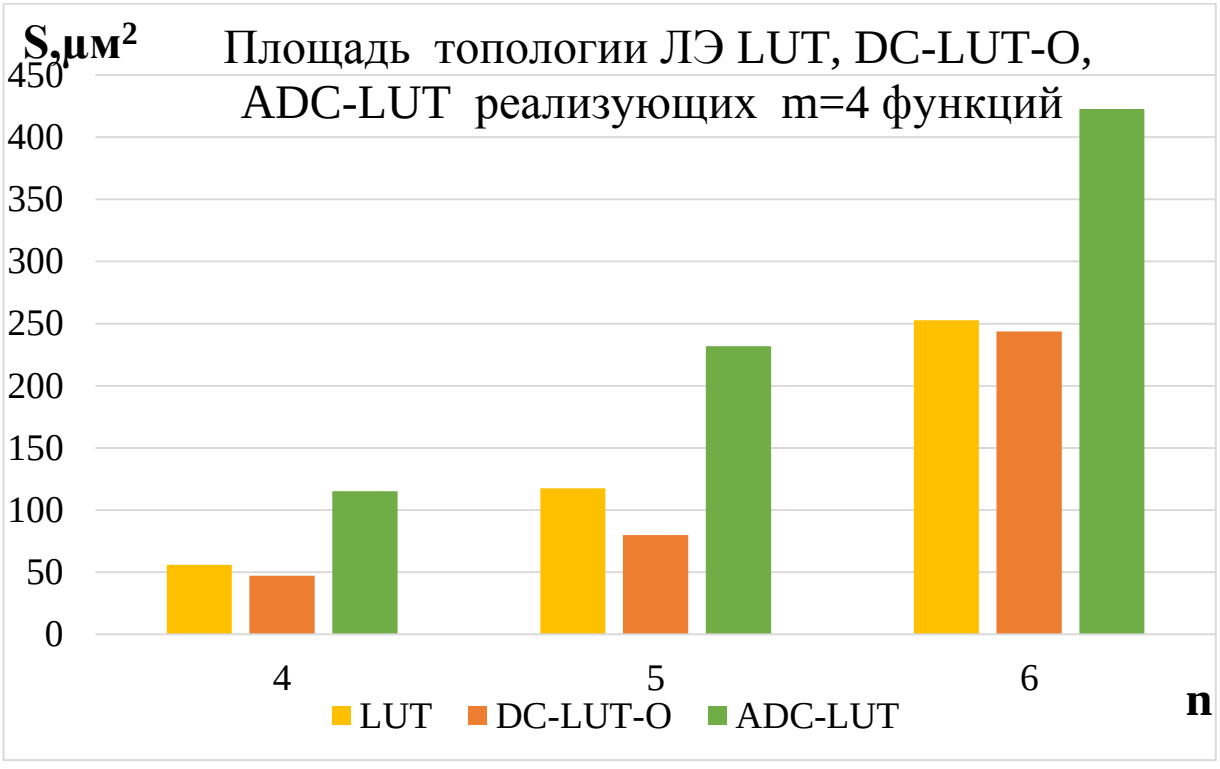


Рисунок 3.98. Занимаемая площадь ADC-LUT для различных n, в сравнении с ЛЭ LUT, DC-LUT-O для m=4.

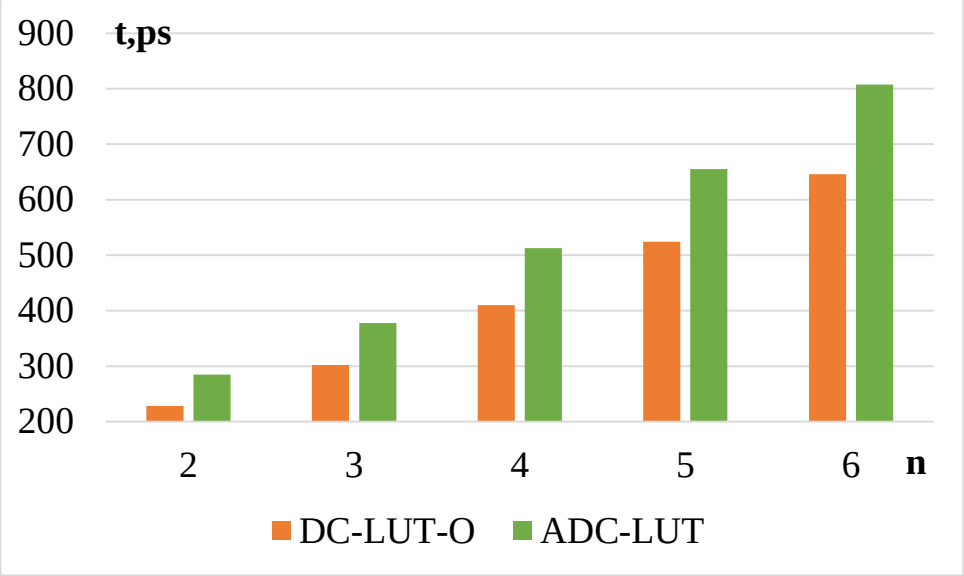


Рисунок 3.99. Значения максимальной задержки ADC-LUT для различных n, в сравнении с ЛЭ DC-LUT-O для m=4.

Выполненное топологическое моделирование подтверждает работоспособность предложенного ЛЭ ADC-LUT.

Другие результаты топологического моделирования представлены в приложении Б

3.7. Выводы по главе 3

1. Выполненное моделирование в системе схемотехнического моделирования NI Multisim фирмы National Instruments Electronics Workbench Group с помощью Spice моделей BSIM4.8 технологии 65нм, подтверждает работоспособность предложенных устройств DC-LUT-R, DC-LUT-O, DC-LUT-BKN, ADC-LUT и DNF-P, DNF-S FPGA на различных частотах (до 700МГц) и при минимальном напряжении питания (0,8В).

2. При моделировании на разных частотах и напряжении с существующего ЛЭ LUT так и с предлагаемых ЛЭ DC-LUT-R, DC-LUT-O, DC-LUT-BKN были сняты значения энергопотребления и задержки. Анализ энергопотребления показывает, что при реализации уже для 3-х входных переменных с реализацией 3-х логических функций DC-LUT-BKN по потреблению сравнялся с LUT, а при 4-х входных переменных ток потребления DC-LUT-BKN меньше тока потребления LUT на 23%,

дальнейшее увеличение числа входных переменных также показывает меньшее значение тока потребления у предлагаемого DC-LUT-BKN в сравнении со значениями тока потребления других предлагаемых реализаций ЛЭ DC-LUT, а также с существующим LUT.

3. Для ДНФ аналогично были промоделированы существующий ЛЭ DNF-R так и предлагаемые ЛЭ DNF-P (параллельная реализация) и DNF-S (последовательная реализация). Были сняты значения тока потребления и задержки. Характеристики как тока потребления, так и задержки для ЛЭ ДНФ показывают, что предпочтителен DNF-S (последовательная реализация) который потребляет меньше, чем DNF-R (с нагрузочным транзистором) используемый в CPLD и обладает меньшим значением задержки уже при 2-х входных переменных.

4. Выходы DC-LUT могут быть использованы либо для управления входами переменных других LUT, либо для их настройки, тем самым могут быть использованы вместо традиционных SRAM.

5. При реализации ДНФ LUT целесообразно обеспечивать на выходе блока программируемых конъюнкций логический ноль в случае истинности заданной конъюнкции.

6. Реализация ДНФ LUT требует использования восстановителей сигналов после каждого блока переменной, поскольку в двух таких блоках имеется цепь из 4-х последовательно соединенных транзисторов в силу ограничений Мида и Конвей тем самым увеличивая сложность на количество транзисторов в восстановителе (2 транзистора образующие инвертор).

7. Результаты моделирования показывают, что DC-LUT-BKN, ADC-LUT, DNF-S обладают большими функциональными возможностями, как в сравнении с существующими решениями LUT, DNF-R так и предлагаемыми DC-LUT-R, DC-LUT-O, DNF-P ввиду реализации системы логических функций меньших значений потребления и задержки.

8. Выполненное топологическое моделирование подтверждает работоспособность предложенных ЛЭ и позволяет определить показатели площади топологии, энергопотребления и задержки

Глава 4. Оценка технической эффективности усовершенствованных методов реализации в FPGA систем логических функций и выбор оптимального набора логических элементов

4.1. Исследование масштабирования разрядности LUT

В силу ограничений Мида и Конвей на число последовательно соединённых транзисторов [62], дерево передающих транзисторов не может содержать более четырёх МОП транзисторов в цепочке. Так LUT на четыре переменных (4-LUT) для входных переменных X4, X3, X2, X1 (настройка – 16 бит), строится на основе двух 3-LUT и одного 1-LUT, который выглядит так – рисунок 4.1:

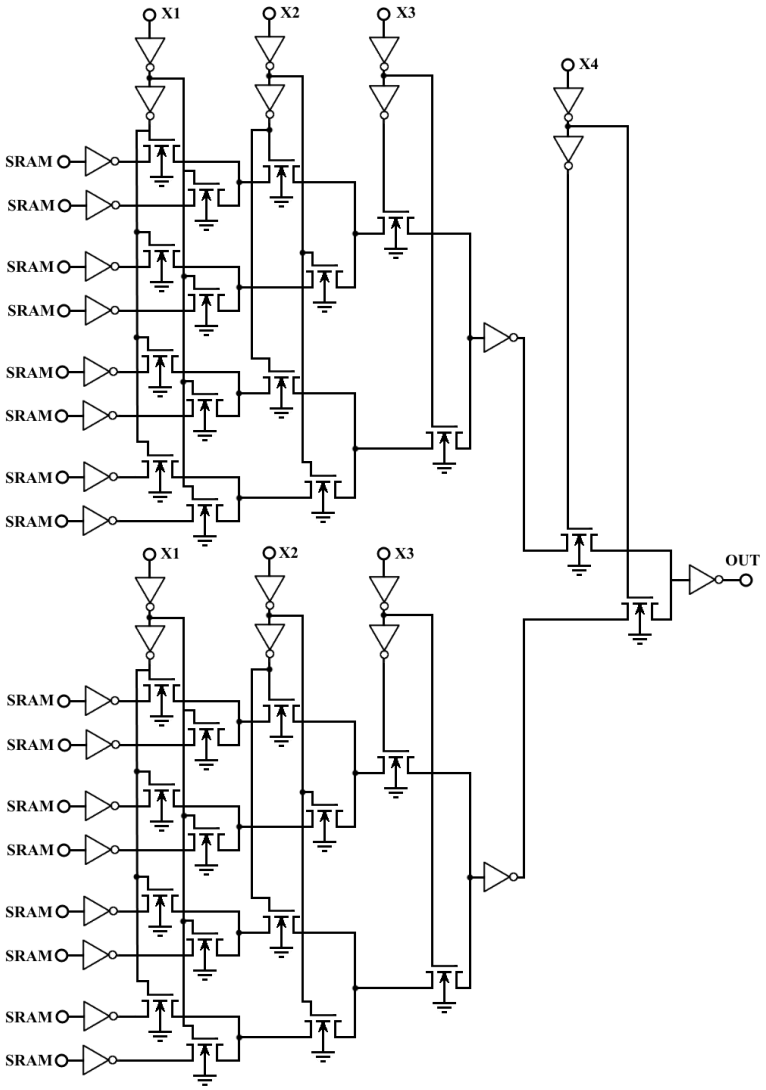


Рисунок 4.1. LUT на четыре переменные (4-LUT), построенный из двух 3-LUT и одного 1-LUT

В такой схеме настройка SRAM должна быть инверсной, поскольку имеется нечётное количество инверторов в цепи со входа на выход. Получаем 4-LUT с двумя восстановителями сигнала – рисунок 4.2:

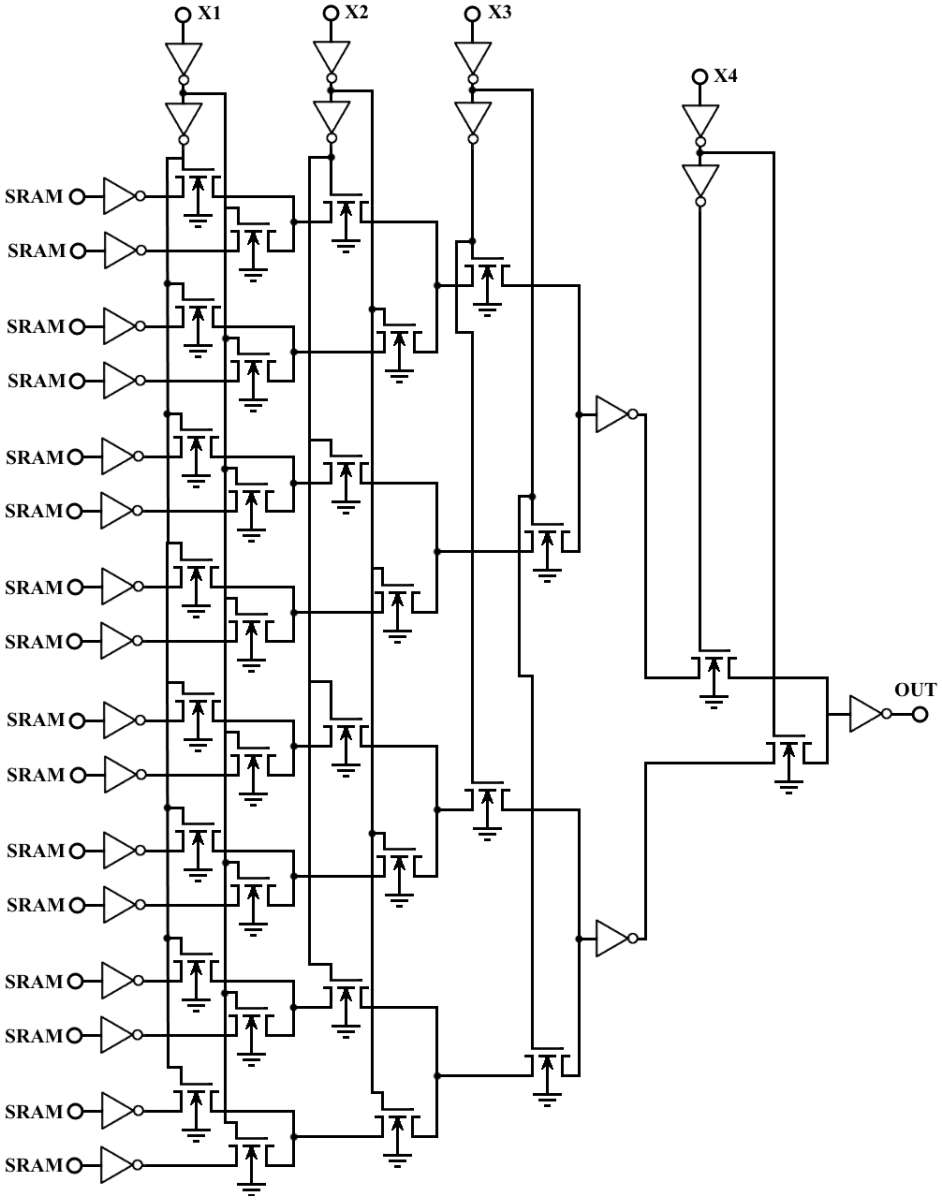


Рисунок 4.2. LUT на четыре переменные (4-LUT).

Такой 4-LUT описывается выражением:

$$z_{OUT}(x_4x_3x_2x_1) = a\bar{x}_4\bar{x}_3\bar{x}_2\bar{x}_1 \vee b\bar{x}_4\bar{x}_3\bar{x}_2x_1 \vee c\bar{x}_4\bar{x}_3x_2\bar{x}_1 \vee d\bar{x}_4\bar{x}_3x_2x_1 \vee$$
$$\vee e\bar{x}_4x_3\bar{x}_2\bar{x}_1 \vee f\bar{x}_4x_3\bar{x}_2x_1 \vee g\bar{x}_4x_3x_2\bar{x}_1 \vee h\bar{x}_4x_3x_2x_1 \vee$$
$$\vee jx_4\bar{x}_3\bar{x}_2\bar{x}_1 \vee jx_4\bar{x}_3\bar{x}_2x_1 \vee kx_4\bar{x}_3x_2\bar{x}_1 \vee lx_4\bar{x}_3x_2x_1 \vee$$
$$\vee mx_4x_3\bar{x}_2\bar{x}_1 \vee nx_4x_3\bar{x}_2x_1 \vee ox_4x_3x_2\bar{x}_1 \vee px_4x_3x_2x_1. \qquad (4.1)$$

Для построения многоразрядного LUT необходима декомпозиция на LUT меньшей разрядности [74], то есть построение дерева из поддеревьев.

LUT на пять переменных (5-LUT) из двух 4-LUT и 1-LUT изображён на рисунке 4.3:

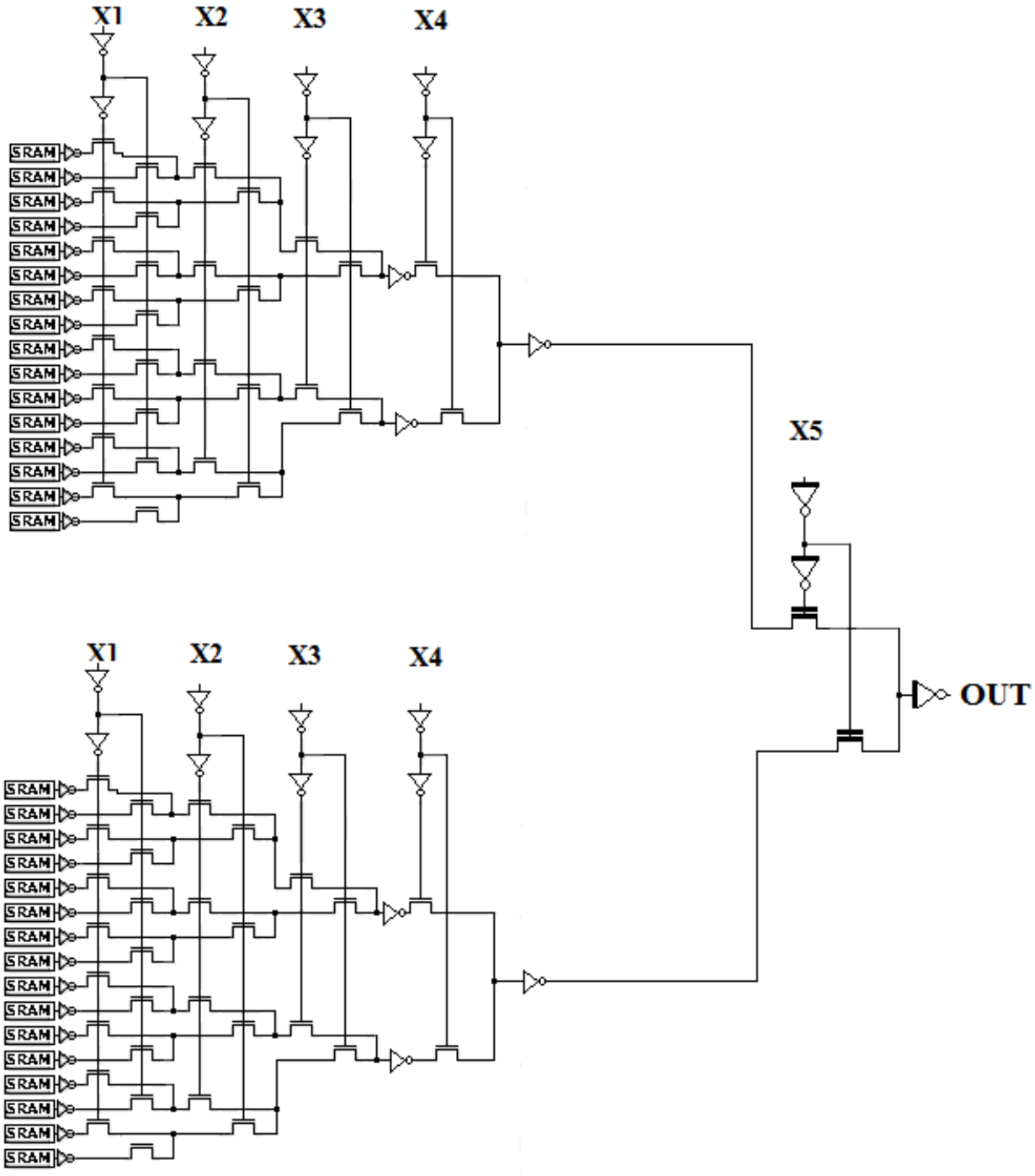


Рисунок 4.3. LUT на пять переменные из двух 4-LUT и 1-LUT

6-LUT из четырёх 4-LUT и одного 2-LUT изображён на рисунке 4.4:

На рисунке 4.4, поскольку число инверторов на пути сигнала чётное, настройки записываются без инверсии.

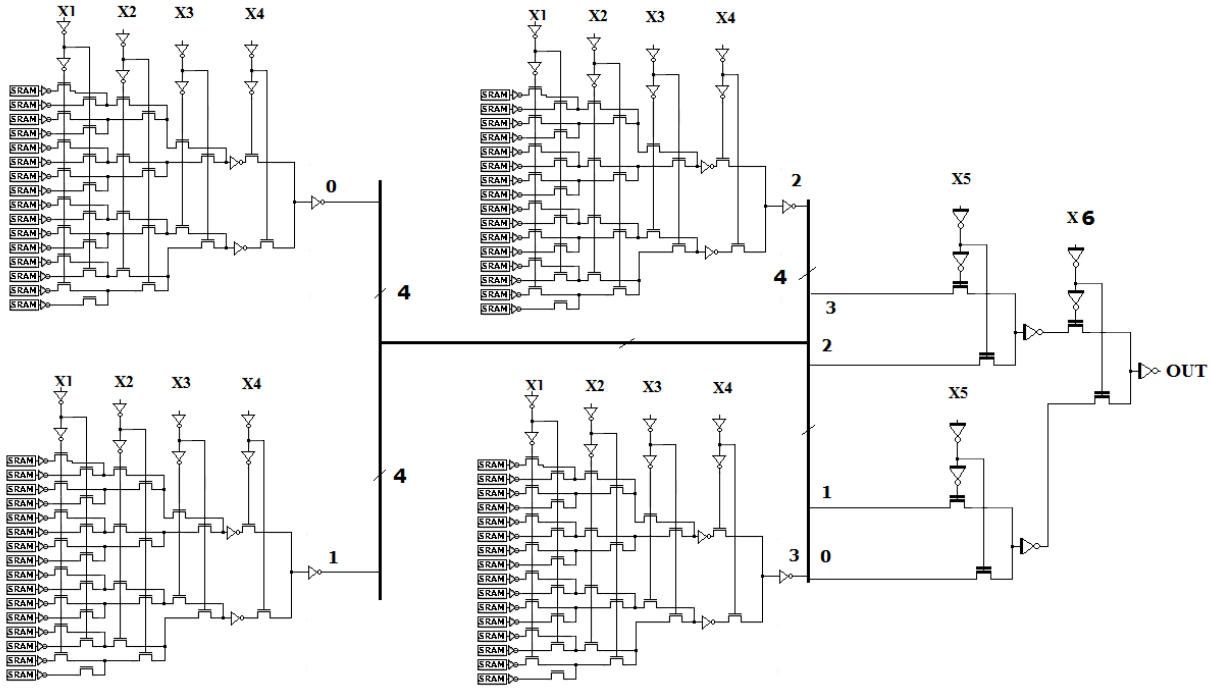


Рисунок 4.4. 6-LUT из четырёх 4-LUT и одного 2-LUT

Пусть k – это размерность основного (базового) LUT (k Є {1,2,3,4}). Для 1-LUT до n=4 нет необходимости в выходном инверторе. Больше 4 переменных по указанным ограничениям k в настоящий момент пока не практикуется.

Оценим сложность LUT без декомпозиции («идеальная» сложность, поскольку такое может быть только до n=4, не более):

$$L_n = 2^n \cdot 8 + 2^{n+1} + 2n, \tag{4.1}$$

где $2^n \cdot 8$ - сложность настройки (по каждому входу настройки необходимо 6 транзисторов памяти SRAM, 2 транзистора - инвертор на входе дерева передающих транзисторов); $2n$ - сложность инверторов по n переменным; 2^{n+1} - сложность дерева передающих транзисторов с выходным инвертором.

При декомпозиции n-дерева по k LUT, k Є{1,2,3,4}, n>= k, n<=8 [75]:

$$L_{n.k} = 2^n \cdot 8 + (2^{k+1} + 2k) \cdot 2^{n-k} + (2^{2^{n-k}+1} + 2^{n-k+1}) + 2n, \tag{4.2}$$

где 2^{k+1} - сложность дерева k LUT, 2k – число транзисторов в k инверторах, таких деревьев необходимо 2^{n-k} , для соединения получаемых при декомпозиции 2^{n-k} деревьев необходимы ещё LUT на 2^{n-k} входов (которые тоже можно декомпозировать), соответственно сложность

$2^{n-k+1}+2\cdot 2^{n-k}=2^{n-k+2}$ где 2^{n-k+1} - сложность дерева с выходным инвертором, $2\cdot 2^{n-k}=2^{n-k+1}$ - сложность входных инверторов. Здесь мы не пойдём дальше n=8, поэтому предполагается, что дополнительный LUT «уложится» в требуемые параметры декомпозиции по k LUT, k Є {1,2,3,4}. Сравнение сложности декомпозиции n LUT по k представлено на рисунке 4.5:

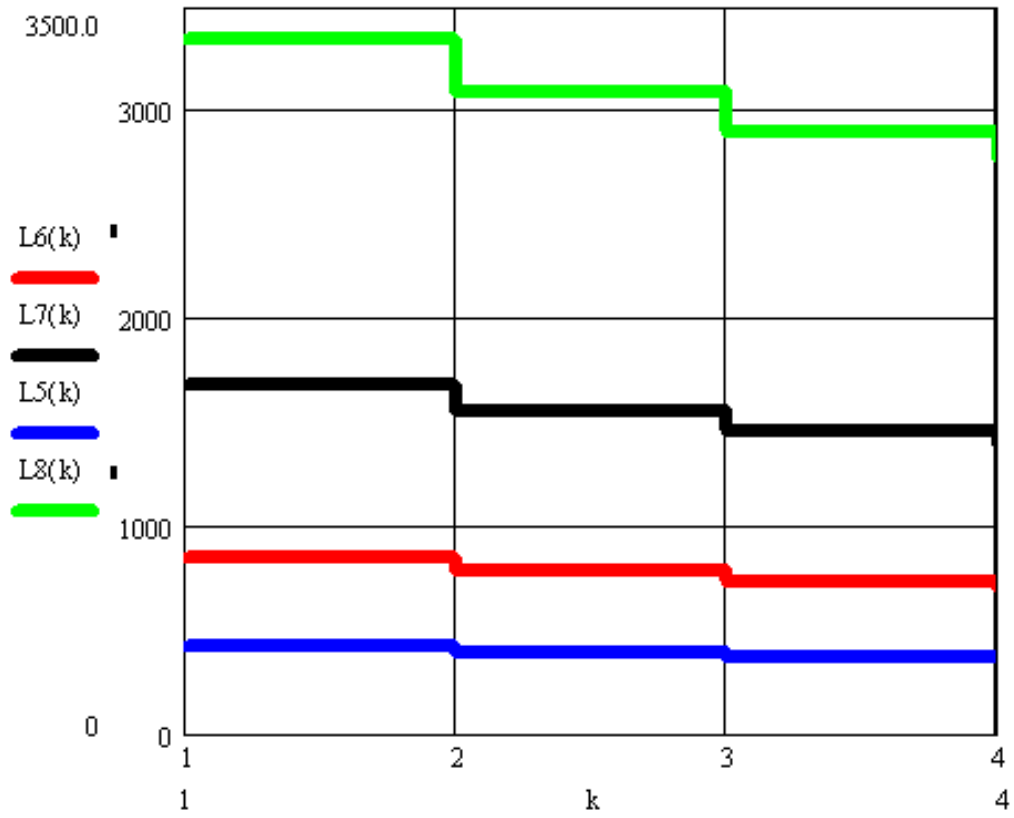


Рисунок 4.5. Сравнение сложности декомпозиции n LUT по k.

Таким образом, чем больше k, тем меньше затрат для реализации сложного LUT на 5,6,7 и 8 переменных.

Временная задержка оценивается длиной максимального пути в логическом элементе со входа на выход. При этом без декомпозиции – при «идеальном» варианте (рисунок 4.6) получим:

$$T_n = n + 2. \tag{4.3}$$

Путь в числе передающих транзисторов за счёт дополнительных инверторов на входе и выходе в цепочке LUT будет:

$$T_{n.k} = n + 2\left\lceil \frac{n}{\lfloor k \rfloor} \right\rceil. \tag{4.4}$$

Графики изменения (4.4) при n=5…8 изображены на рисунке 4.6:

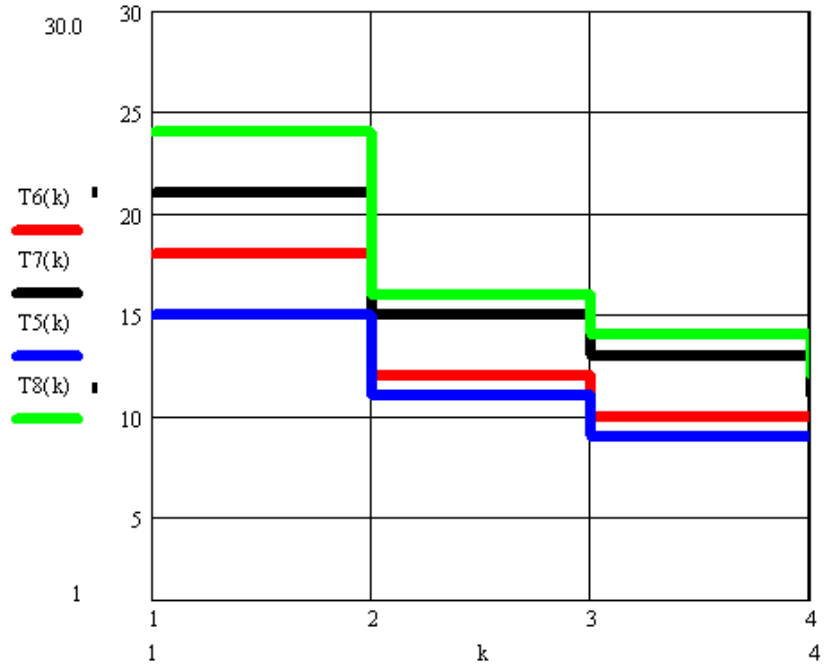


Рисунок 4.6. Сравнение временных затрат на декомпозицию n LUT по k при n=5…8 по k.

Графики изменения (4.4) при n=7…10 изображены на рисунке.4.7

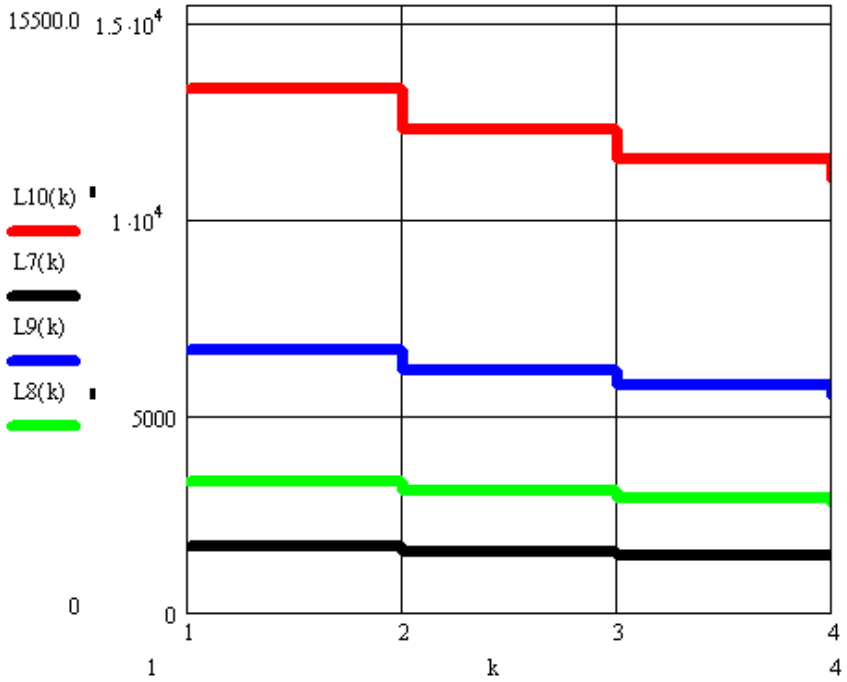


Рисунок 4.7. Сравнение временных затрат на декомпозицию n LUT по k при n=7…10 по k.

4.2. Оценка сложности предлагаемых ЛЭ DC-LUT, ADC-LUT ПЛИС FPGA

Сложность LUT в количестве транзисторов можно оценить выражением [75]:

$$L = 2^{n+1} + (2^n - 1) \cdot 4 + 2n. \quad (4.5)$$

Для задания m логических функций в СДНФ необходима соответствующая настройка + блоки дизъюнкций конституент, итого для всего DC n-LUT:

$$L_{DC-LUT} = 2^{n+1} + (2^n - 1) \cdot 4 + 2n + 6 \cdot m \cdot (2^n + 2). \quad (4.6)$$

Будем строить n-DC LUT из DC k=LUT, k=1,2,3,4:

n-LUT, n <5 для задания m логических функций оценивается выражением:

$$L_n = m(2^{n+1} + 8 \cdot 2^n + 2n). \quad (4.7)$$

Сравнение сложности реализации m логических функций в m LUT и в n-DC LUT при m=4 показано на рисунке 4.8 и рисунке 4.9.

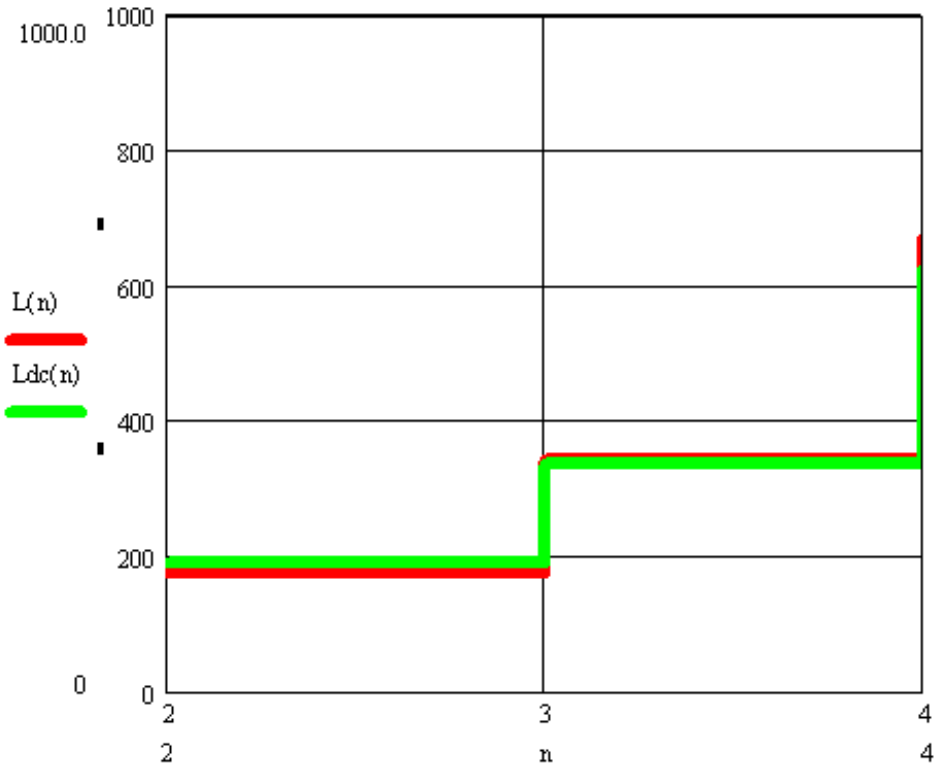
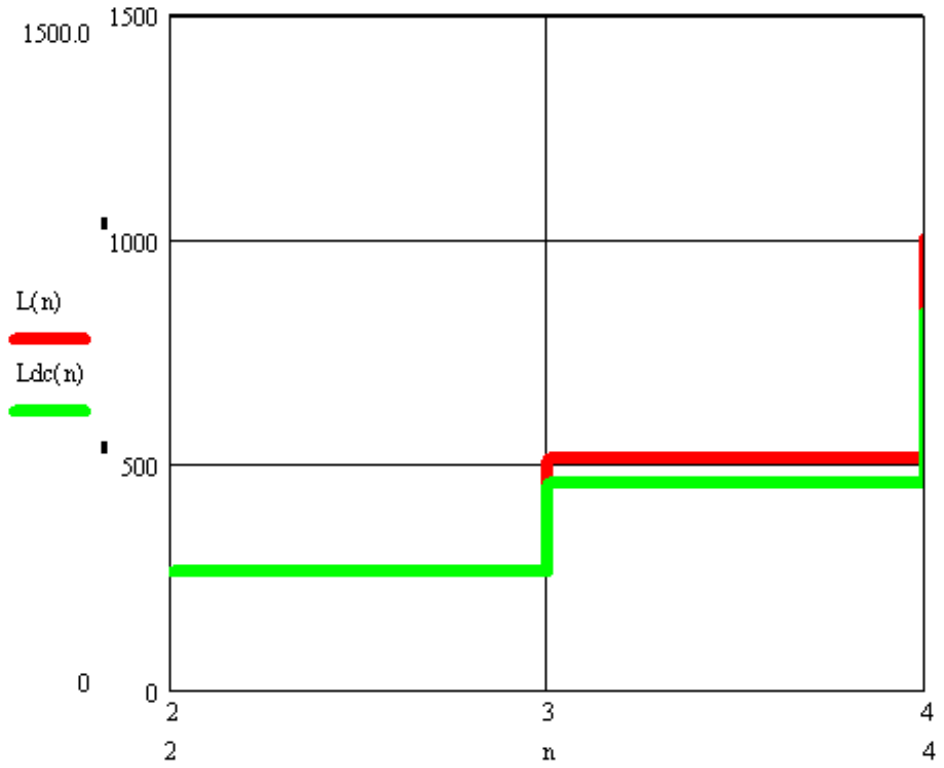
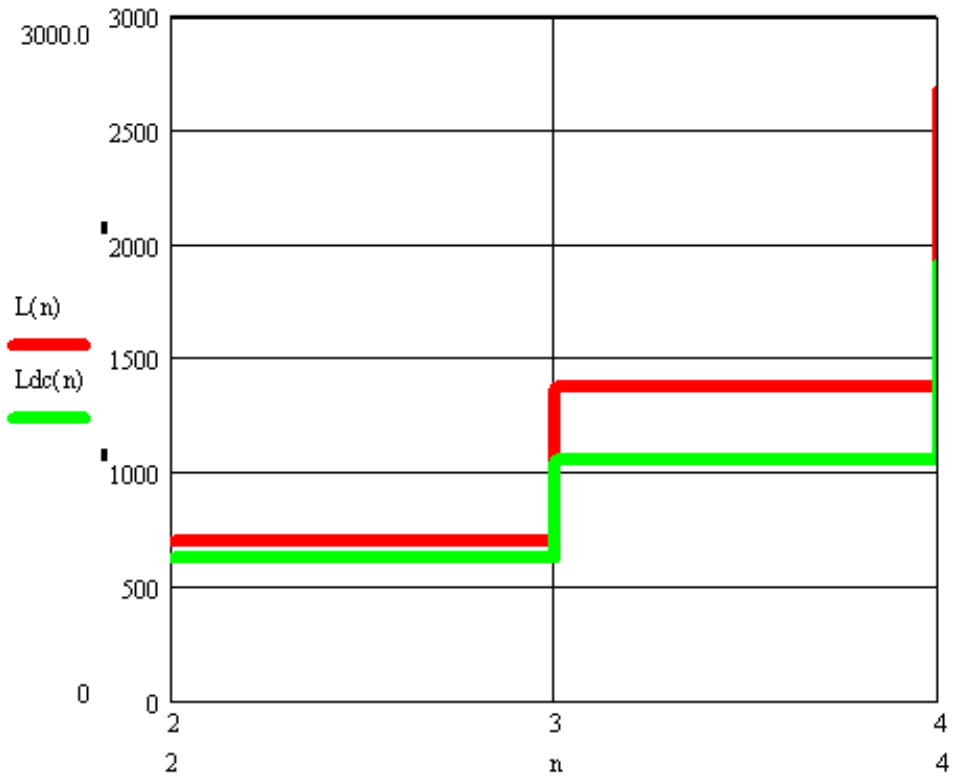


Рисунок 4.8. Сравнение сложности логических элементов 2,3,4 LUT - L(n) 2,3,4 DC LUT - Ldc(n) при: m=4



а)



б)

Рисунок 4.9. Сравнение сложности логических элементов 2,3,4 LUT - L(n) 2,3,4 DC LUT - Ldc(n) при: а) m=8; б) m=16

Декомпозиция n-LUT (при n>4) из элементарных мультиплексоров k=LUT, k=1,2,3,4 оценивается выражением:

$$L_{n.k} = 2^{n+3} + (2^{k+1} + 2k) \cdot 2^{n-k} + (2^{2^{n-k}+1} + 2^{n-k+1}) + 2n, \quad (4.8)$$

для задания m логических функций:

$$L_{n.k} = m \cdot [2^{n+3} + (2^{k+1} + 2k) \cdot 2^{n-k} + (2^{2^{n-k}+1} + 2^{n-k+1}) + 2n], \quad (4.9)$$

Декомпозиция DC n-LUT из элементарных дешифраторов k=LUT, k=1,2,3,4, определяется только деревом, причём это в два раза «дороже», чем при декомпозиции n-LUT, так как необходимо в два раза больше транзисторов в дереве для обеспечения ортогональности:

$$L_{n.k} = m(2^n \cdot 8 + 2 \cdot [2^{k+1} \cdot 2^{n-k} + 4 \cdot (2^{n-k} - 1)] + 2n). \quad (4.10)$$

$$L_{DC-LUT} = 2^{n+1} + [(2^{k+1} + 2k) \cdot 2^{n-k} + (2^{2^{n-k}+1} + 2^{n-k+1})]2 + 2n + 6 \cdot m \cdot (2^n + 2). \quad (4.11)$$

Сравним выражения 4.10 и 4.11 – рисунок 4.10, рисунок 4.11, рисунок 4.12, рисунок 4.13:

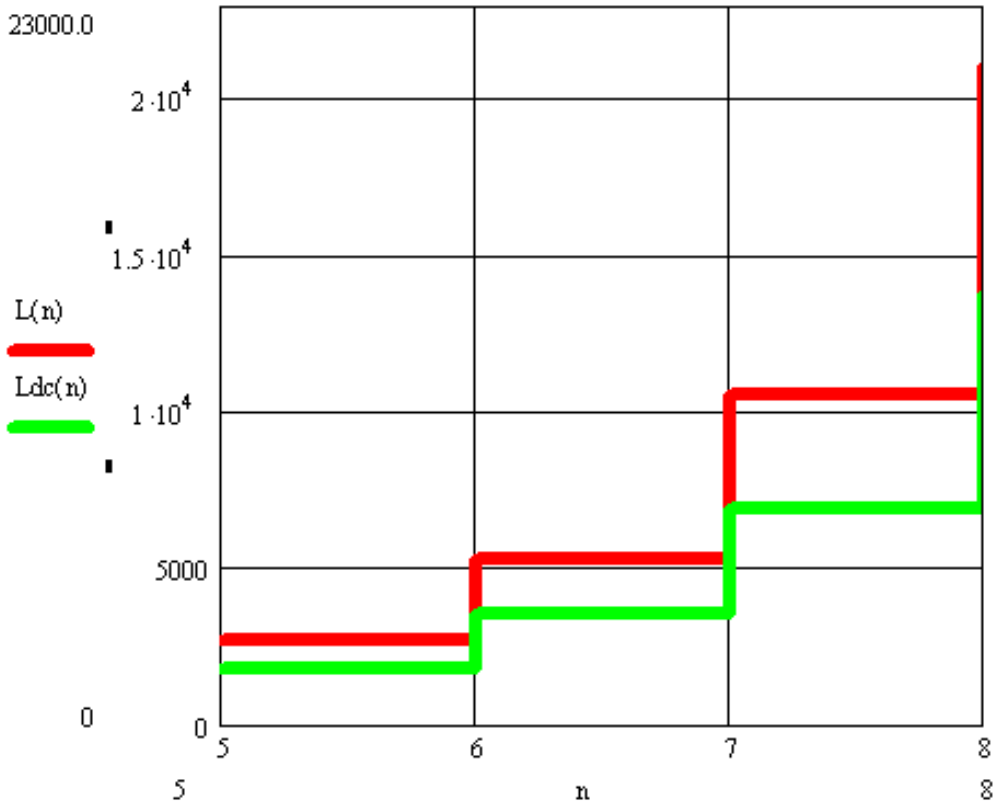
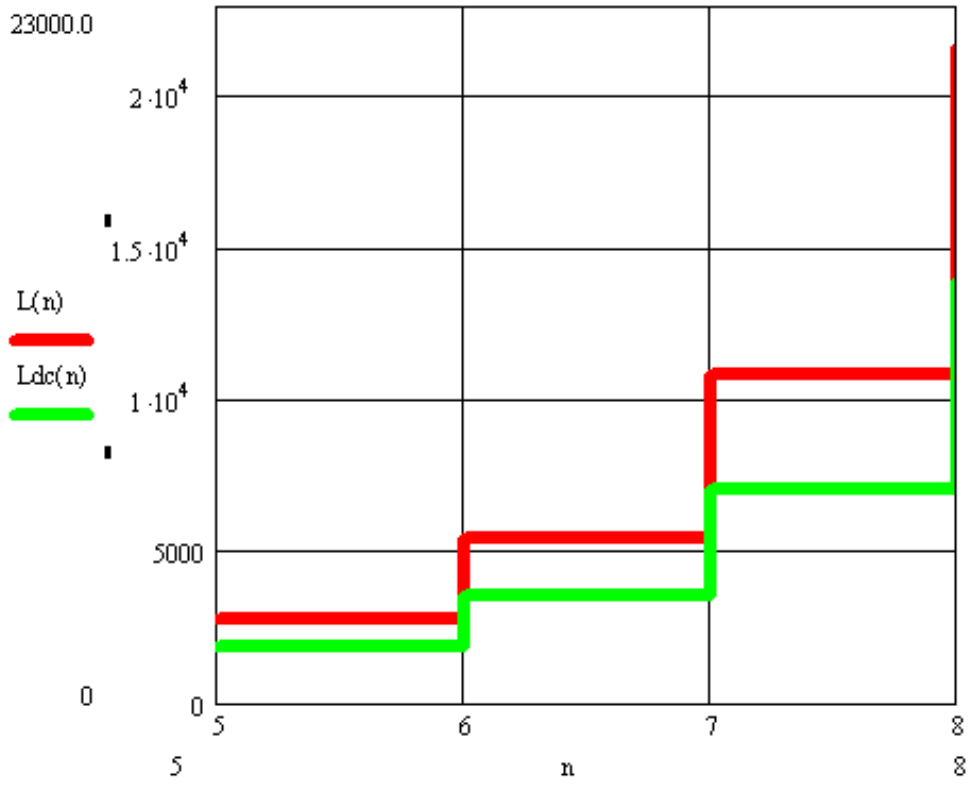
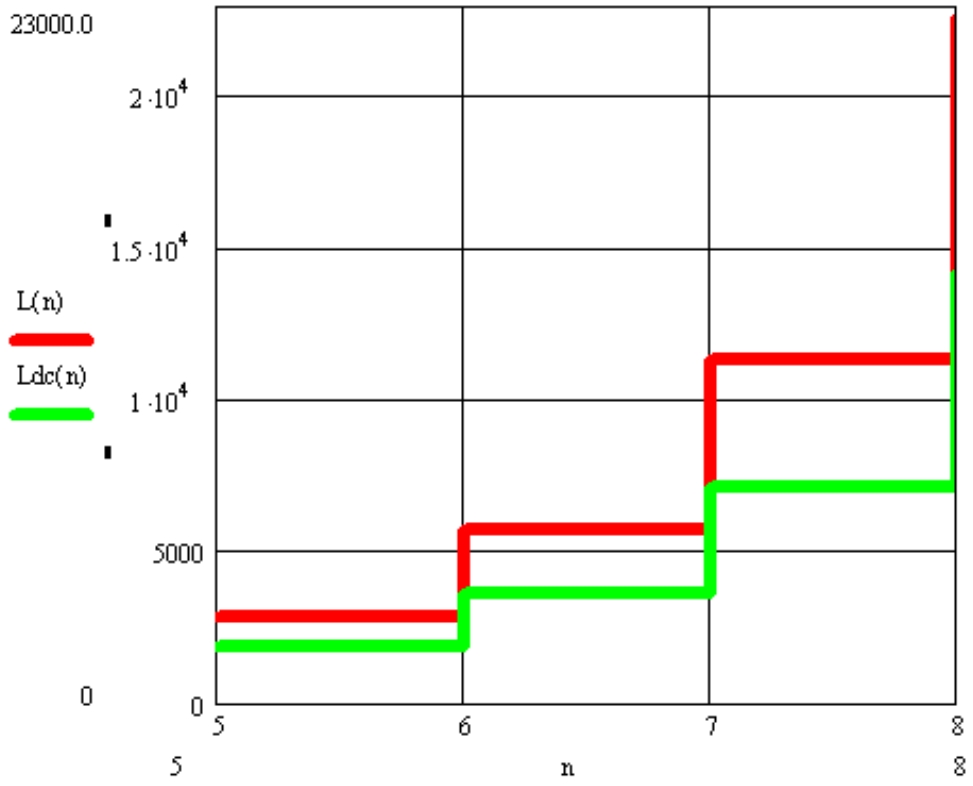


Рисунок 4.10. Сравнение сложности логических элементов n LUT - L(n)b DC n LUT - Ldc(n) при декомпозиции: k=4, m=8;

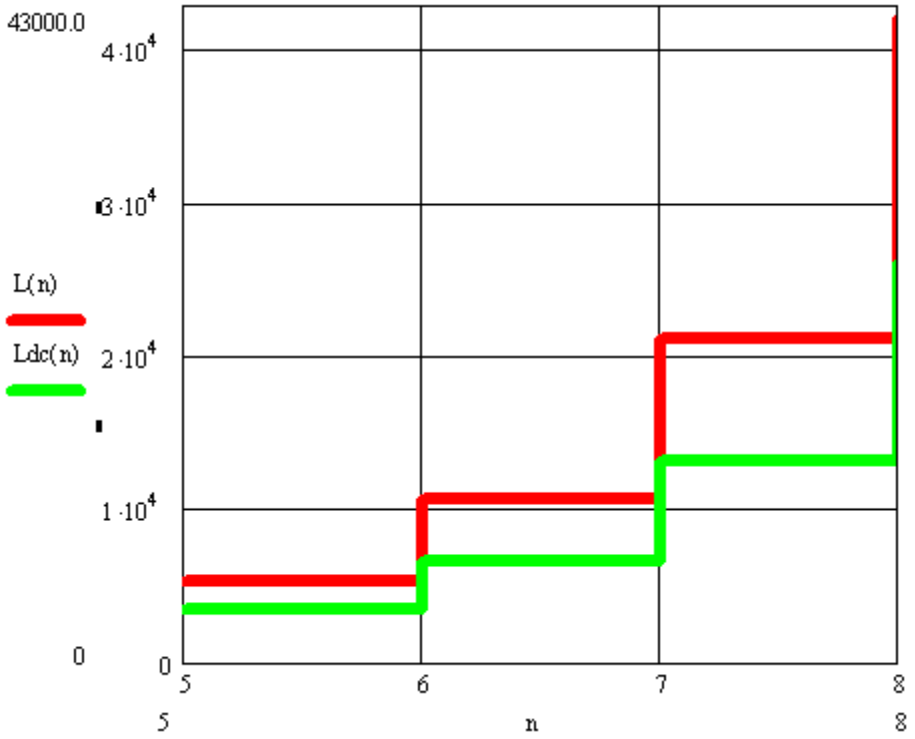


а)

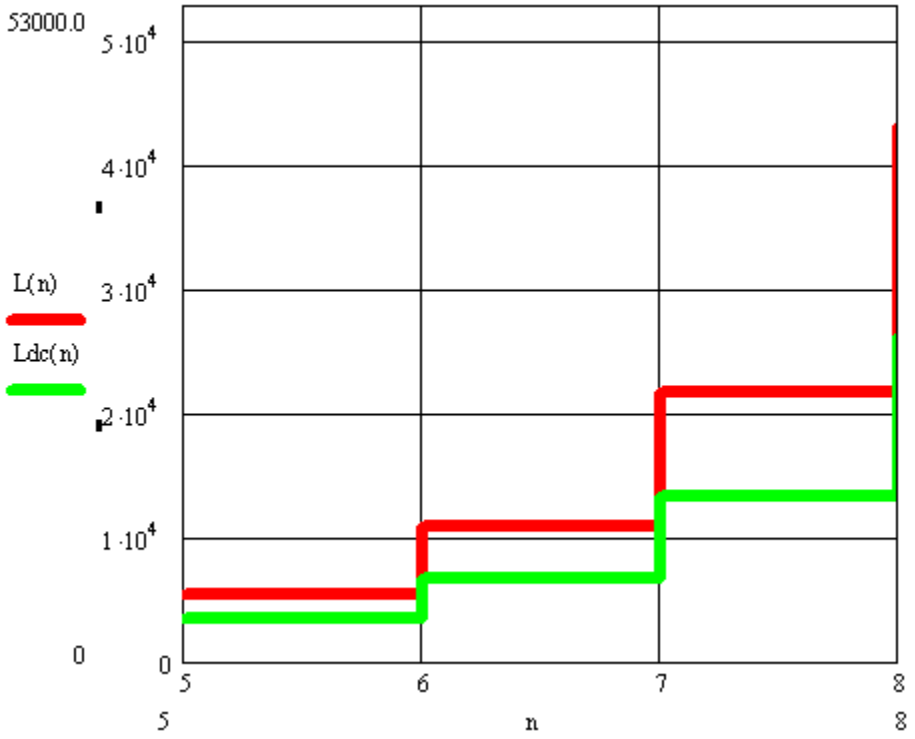


б)

Рисунок 4.11. Сравнение сложности логических элементов n LUT - L(n)b DC n LUT - Ldc(n) при декомпозиции: а) k=3, m=8; б) k=2, m=8;



а)



б)

Рисунок 4.12. Сравнение сложности логических элементов n LUT - L(n)b DC n LUT - Ldc(n) при декомпозиции: а) k=4, m=16; б) k=3, m=16;

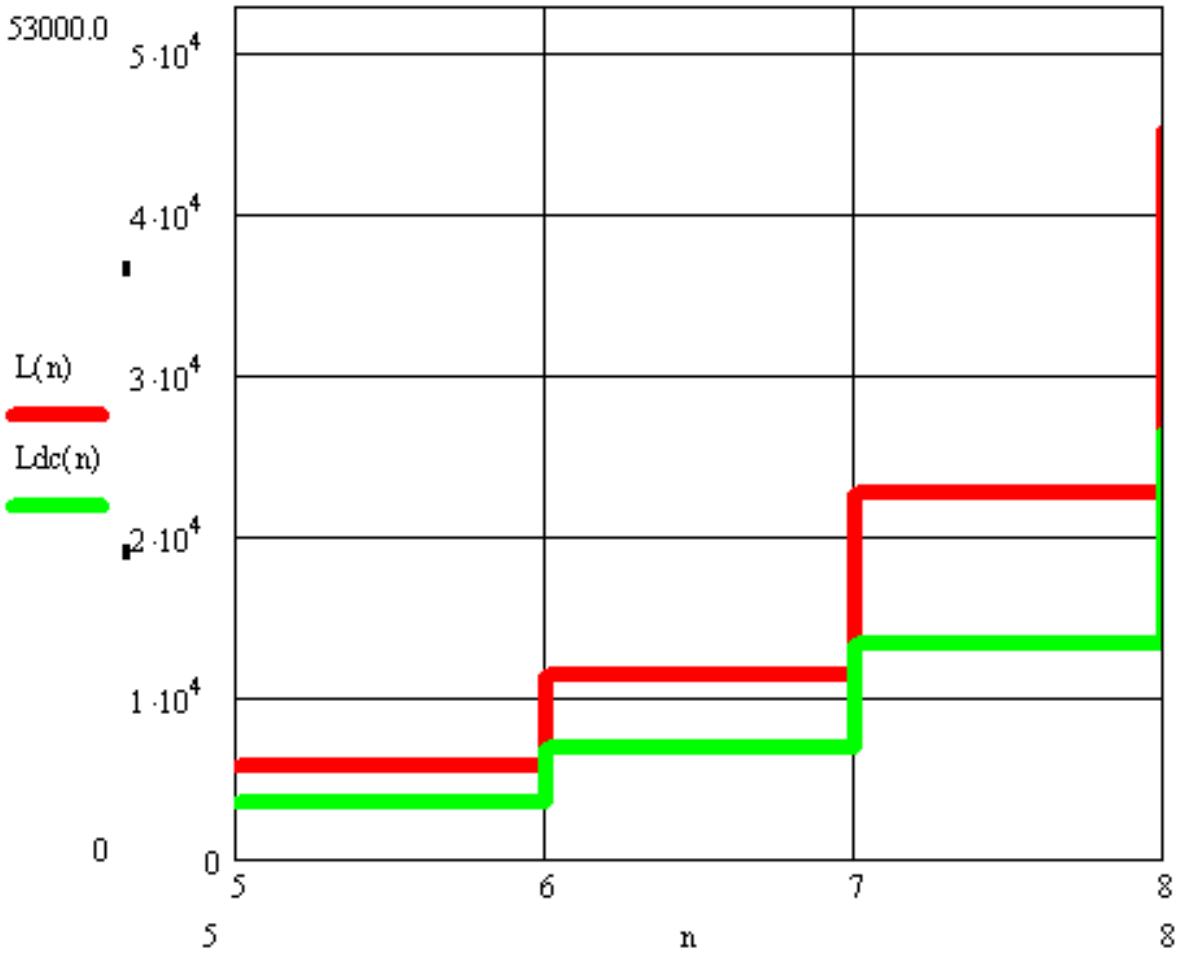


Рисунок 4.13. Сравнение сложности логических элементов n LUT - L(n)b DC n LUT - Ldc(n) при декомпозиции: k=2, m=16

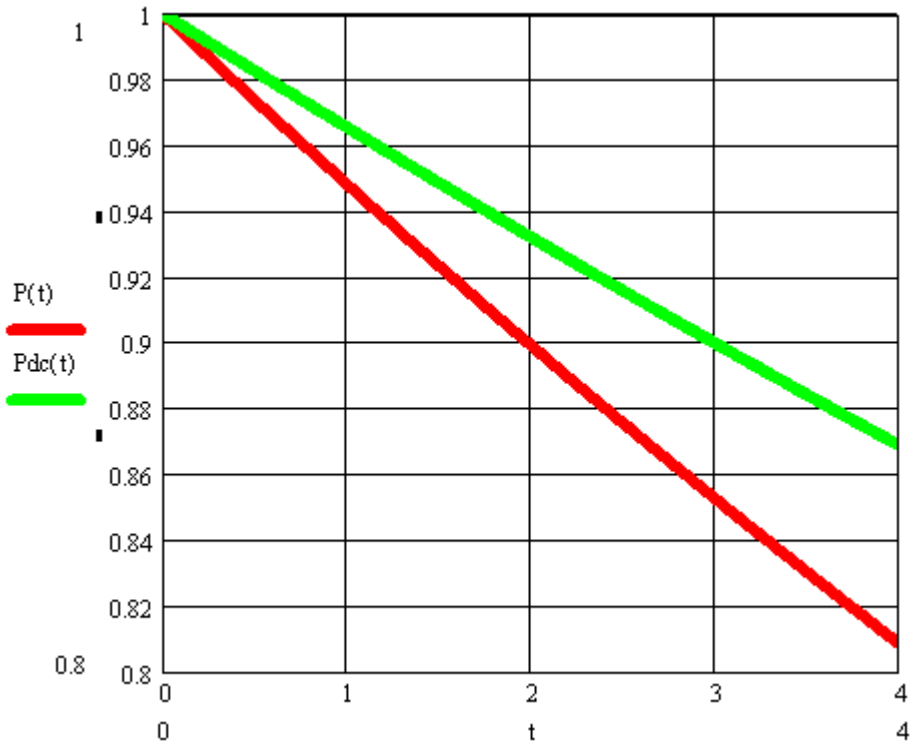
Таким образом, получаем ориентировочно на 40% меньше затрат при 6-7 переменных. Вероятность безотказной работы LUT исходя из сложности оценивается выражением:

$$P(t) = e^{-m(2^n \cdot 8 + 2 \cdot [2^{k+1} \cdot 2^{n-k} + 4 \cdot (2^{n-k} - 1)] + 2n) \cdot \lambda \cdot t}. \qquad (4.12)$$

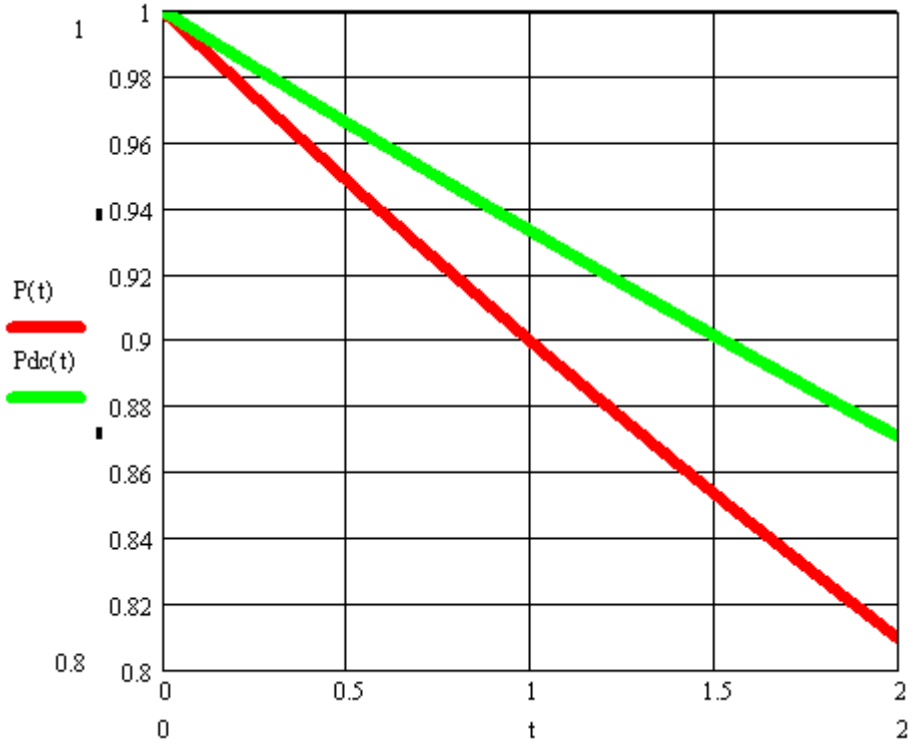
Вероятность безотказной работы DC LUT исходя из сложности оценивается выражением:

$$P_{dc}(t) = e^{-1 \cdot (2^{n+1} + [(2^{k+1} + 2k) \cdot 2^{n-k} + (2^{2^{n-k}+1} + 2^{n-k+1})] 2 + 2n + 6 \cdot m \cdot (2^n + 2)) \cdot \lambda \cdot t}. \qquad (4.13)$$

Сравнение выражений 4.12 и 4.13 представлено на рисунке 4.14:



а)



б)

Рисунок 4.14. Графики изменения вероятности безотказной работы - схемы LUT $P(t)$ и вероятности безотказной работы DC LUT $P_{dc}(t)$ при интенсивности отказов (сбоев) $\lambda = 10^{-5}$ 1/час, а) n=6, k=4, m=8; б) n=7, k=4, m=8.

4.3. Оценка сложности предлагаемого DNF-LUT

Для соблюдения в LUT ограничений Мида-Конвей [62] при реализации логических функций большого количества переменных необходима декомпозиция [76,77] сложного n-дерева по k-LUT, k Є {1,2,3,4}, n> = k, n:

$$L_{n.k} = 2^{\lfloor n \rfloor} \cdot 8 + (2^{\lfloor k \rfloor + 1} + 6\lfloor k \rfloor) \cdot \sum_{i=1}^{\left\lfloor \frac{\lfloor n \rfloor}{\lfloor k \rfloor} \right\rfloor} 2^{\lfloor n \rfloor - \lfloor i \rfloor \lfloor k \rfloor} + (2^{\lfloor n \rfloor - \left\lfloor \frac{\lfloor n \rfloor}{\lfloor k \rfloor} \right\rfloor \cdot \lfloor k \rfloor + 1} + 6 \cdot \left(\lfloor n \rfloor - \left\lfloor \frac{\lfloor n \rfloor}{\lfloor k \rfloor} \right\rfloor \cdot \lfloor k \rfloor \right)) + 6\lfloor n \rfloor, \quad (4.14)$$

где $2^{\lfloor k \rfloor + 1} + 6\lfloor k \rfloor$ - сложность одного k-дерева $\lfloor ... \rfloor$-округление в нижнюю строну (floor); таких деревьев (k-LUT) необходимо в первом слое $2^{\lfloor n \rfloor - \lfloor k \rfloor}$, затем нужно провести декомпозицию k-LUT этого первого слоя, получаем $2^{\lfloor n \rfloor - \lfloor k \rfloor - \lfloor k \rfloor}$ Всего необходимо i k-LUT, где i определяется из соотношения: $\lfloor i \rfloor = \left\lfloor \frac{\lfloor n \rfloor}{\lfloor k \rfloor} \right\rfloor$ всего $\sum_{i=1}^{\left\lfloor \frac{\lfloor n \rfloor}{\lfloor k \rfloor} \right\rfloor} 2^{\lfloor n \rfloor - \lfloor i \rfloor \lfloor k \rfloor}$ и последний LUT на $\lfloor n \rfloor - \left\lfloor \frac{\lfloor n \rfloor}{\lfloor k \rfloor} \right\rfloor \cdot \lfloor k \rfloor$ переменных. Реализация DC LUT оценивается выражением

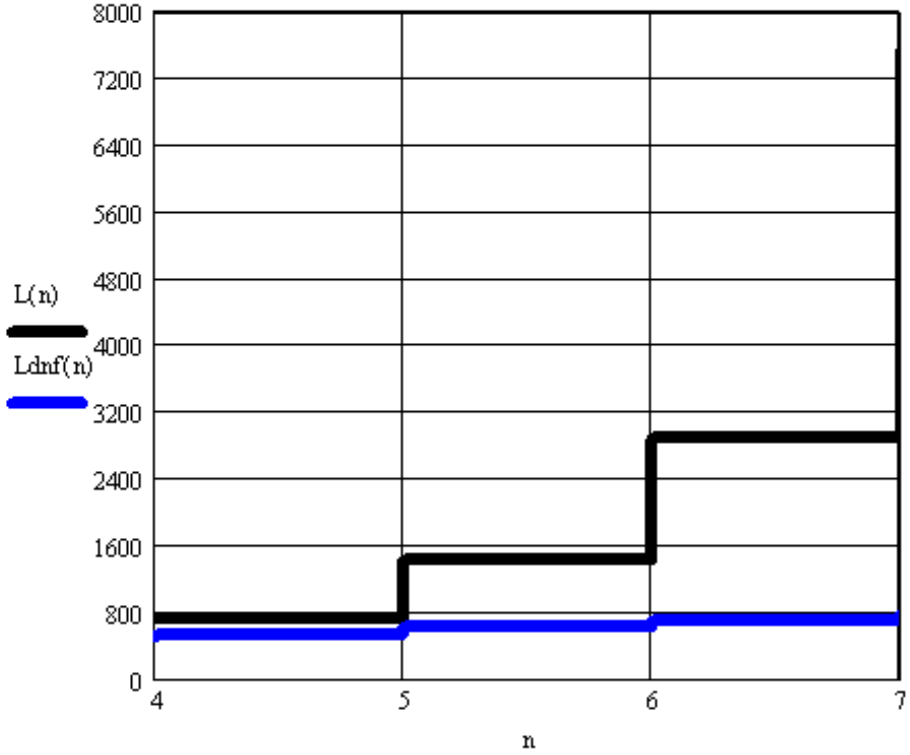
$$L_{dc-n.k} = 2^{\lfloor n \rfloor} \cdot 8 + (2^{\lfloor k \rfloor + 1} + 6\lfloor k \rfloor) \cdot \sum_{i=1}^{\left\lfloor \frac{\lfloor n \rfloor}{\lfloor k \rfloor} \right\rfloor} 2^{\lfloor n \rfloor - \lfloor i \rfloor \cdot \lfloor k \rfloor} + (2^{\lfloor n \rfloor - \left\lfloor \frac{\lfloor n \rfloor}{\lfloor k \rfloor} \right\rfloor \cdot \lfloor k \rfloor + 1} + 6 \cdot \left(\lfloor n \rfloor - \left\lfloor \frac{\lfloor n \rfloor}{\lfloor k \rfloor} \right\rfloor \cdot \lfloor k \rfloor \right)) + 6\lfloor n \rfloor + 2^{\lfloor n \rfloor} + 6m(2^n + 2) \quad (4.15)$$

Сравнение сложности ДНФ-LUT (Ldnf(n)) и LUT(L(n)) показано на рисунке 4.11:

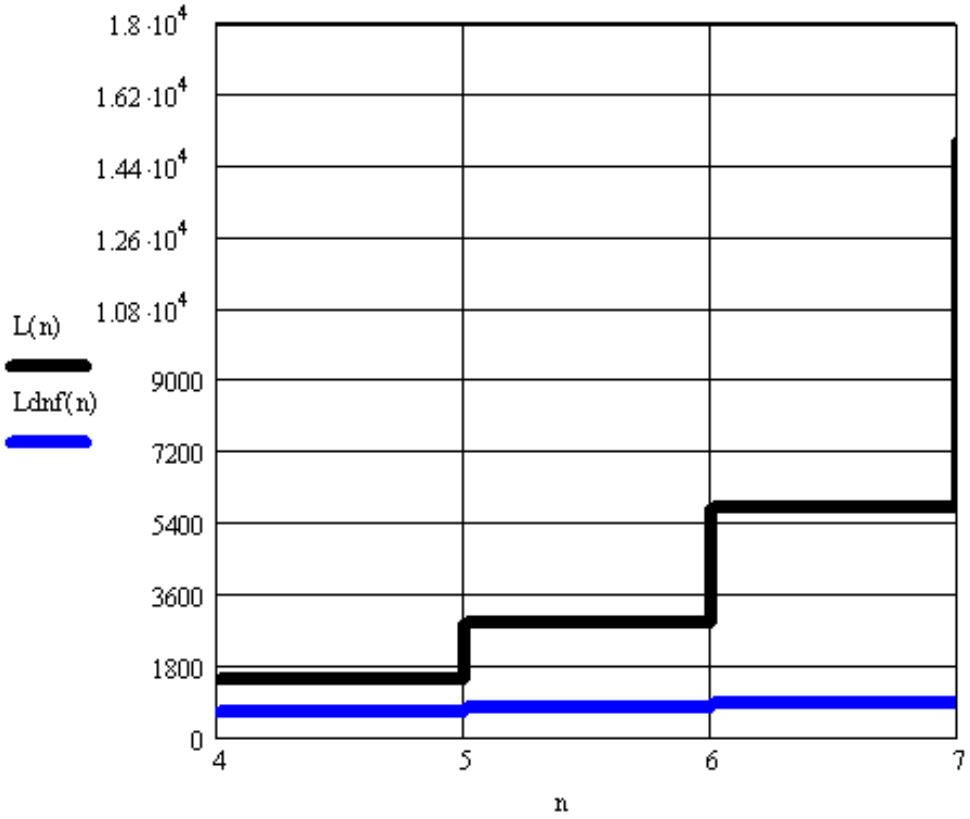
Сложность ADC LUT описывается выражением:

$$L_{adc-n.k} = 2^{\lfloor n \rfloor} \cdot 8 + (2^{\lfloor k \rfloor + 1} + 6\lfloor k \rfloor) \cdot \sum_{i=1}^{\left\lfloor \frac{\lfloor n \rfloor}{\lfloor k \rfloor} \right\rfloor} 2^{\lfloor n \rfloor - \lfloor i \rfloor \cdot \lfloor k \rfloor} + (2^{\lfloor n \rfloor - \left\lfloor \frac{\lfloor n \rfloor}{\lfloor k \rfloor} \right\rfloor \cdot \lfloor k \rfloor + 1} + 6 \cdot \left(\lfloor n \rfloor - \left\lfloor \frac{\lfloor n \rfloor}{\lfloor k \rfloor} \right\rfloor \cdot \lfloor k \rfloor \right)) + 6\lfloor n \rfloor + 2^{\lfloor n \rfloor}(17) + 6m(2^n + 2) + 14 \quad (4.16)$$

где $2^{\lfloor n \rfloor}(17)$ - сложность конфигурации выходов, 17 – сложность конфигурации корня дерева, $6m(2^n + 2)$ - сложности m блоков реализации функций в СДНФ.



а)



б)

Рисунок 4.15. Сравнение сложности ДНФ-LUT (Ldnf(n)) и LUT(L(n)) для n 4-7, а) m=4, k=4; б) m=8, k=4.

При использовании ДНФ-LUT получаем сложность в количестве транзисторов:

$$L_{dnf} = \lfloor v \rfloor (6\lfloor n \rfloor) + 6 \cdot \lfloor m \rfloor (\lfloor v \rfloor + 2) + 3 \left\lfloor \frac{n}{r} \right\rfloor \lfloor k \rfloor, \qquad (4.17)$$

где $\lfloor v \rfloor (6\lfloor n \rfloor)$ учитывает сложность реализации v настраиваемых конъюнкций; $6 \cdot \lfloor m \rfloor (\lfloor v \rfloor + 2)$ -сложность m блоков функций от v конъюнкций (реализация монтажного И), $3 \left\lfloor \frac{n}{r} \right\rfloor \lfloor k \rfloor$ -учитывает r ограничения Мида-Конвей и восстановитель по k конъюнкциям, сложностью 3 транзистора (инвертор и дополнительный p-транзистор). Сравнение сложности L0(n), L1(n), Ldc(n), Ldnf(n), Ladc(n) изображено на рисунке 4.12.

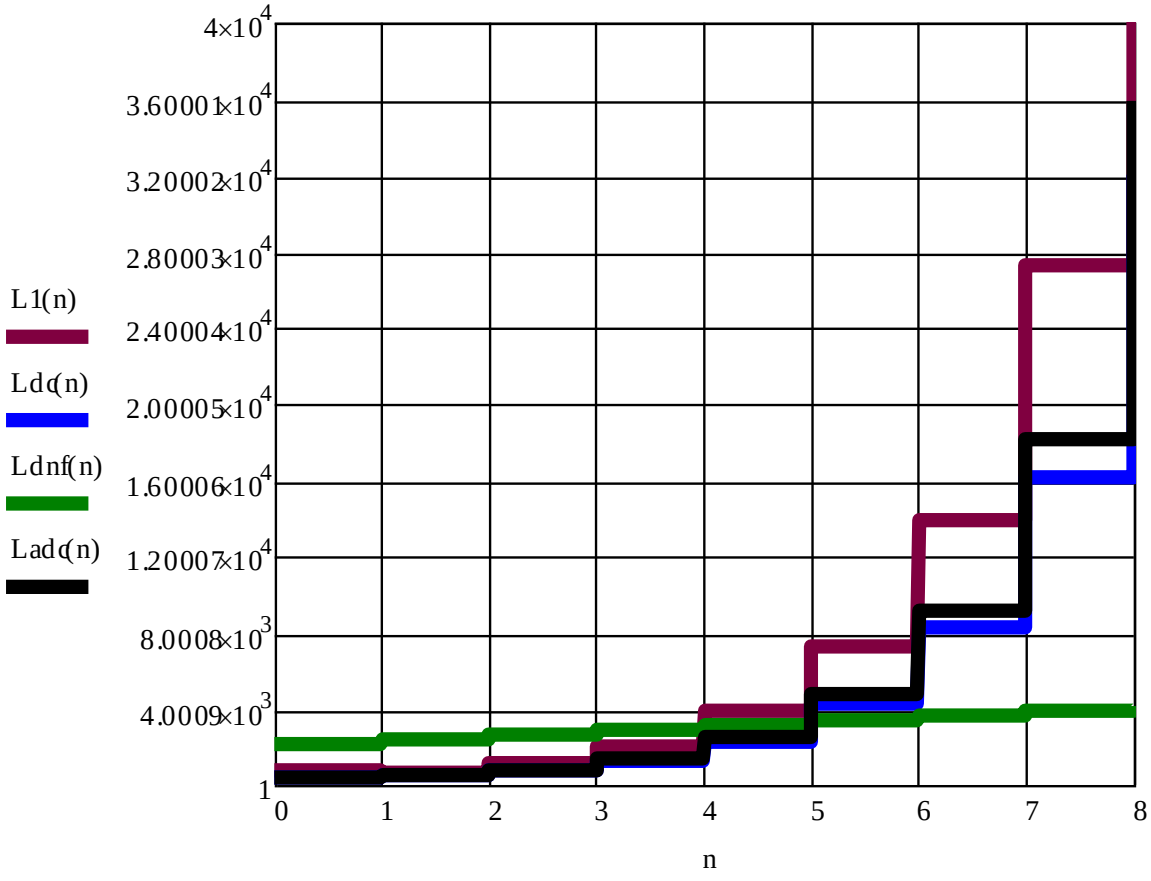
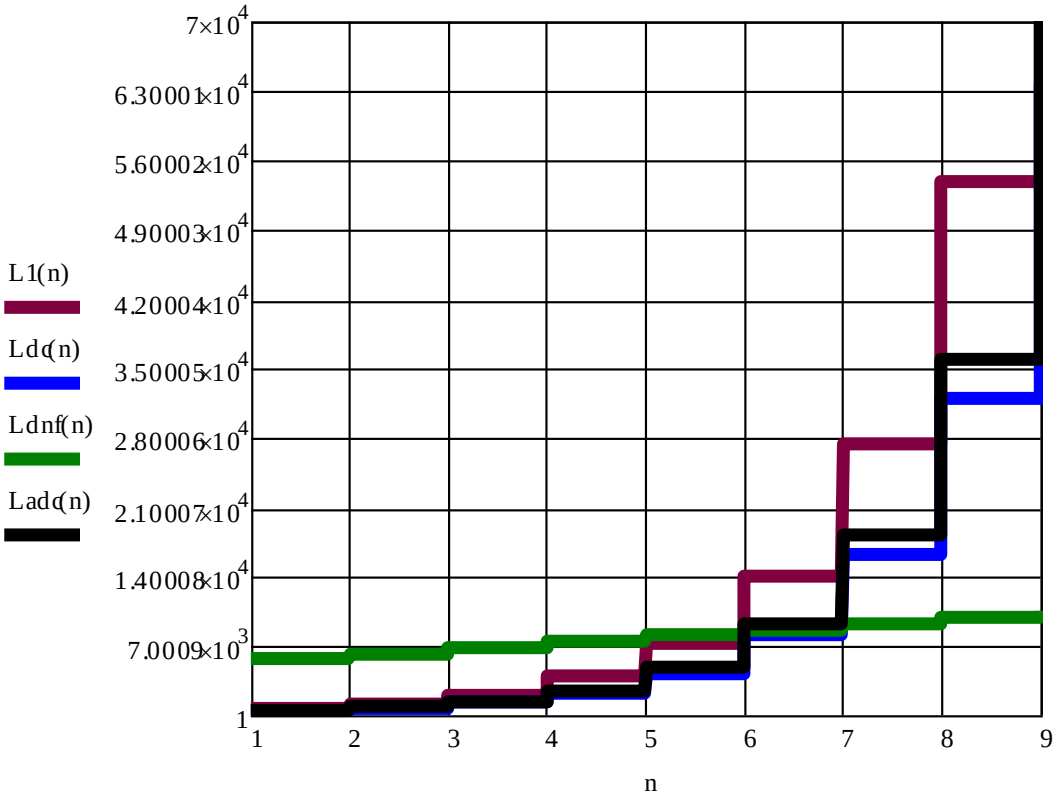
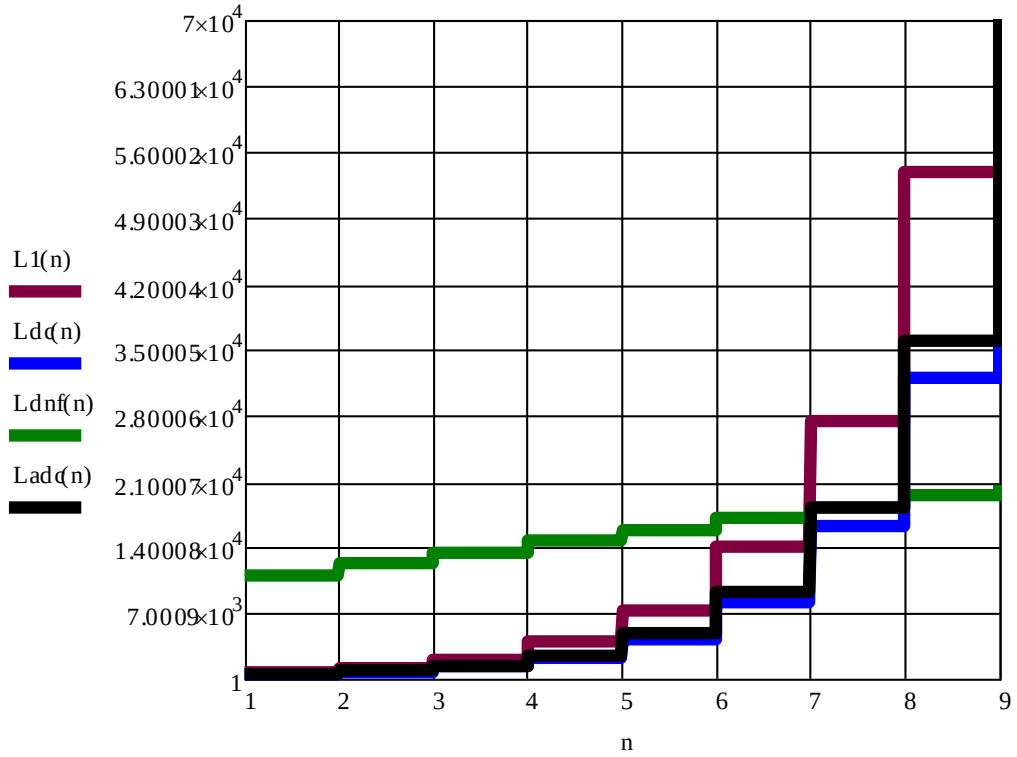


Рисунок 4.16. Сравнение сложности L1(n), Ldc(n), Ldnf(n), Ladc(n) при: a) m=8; k=3; v=20;

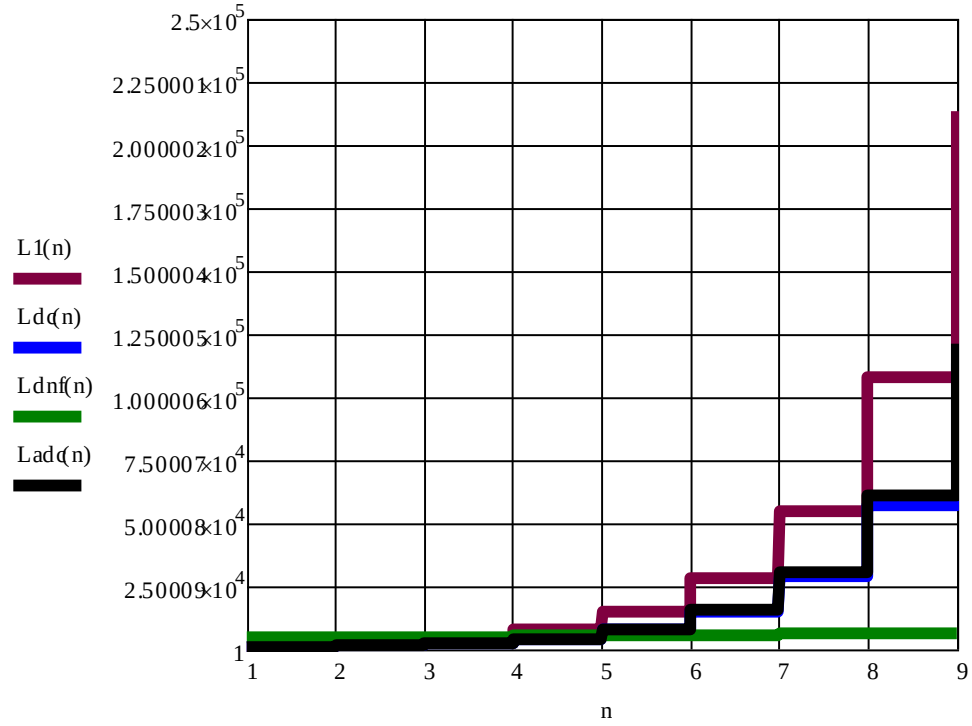


а)

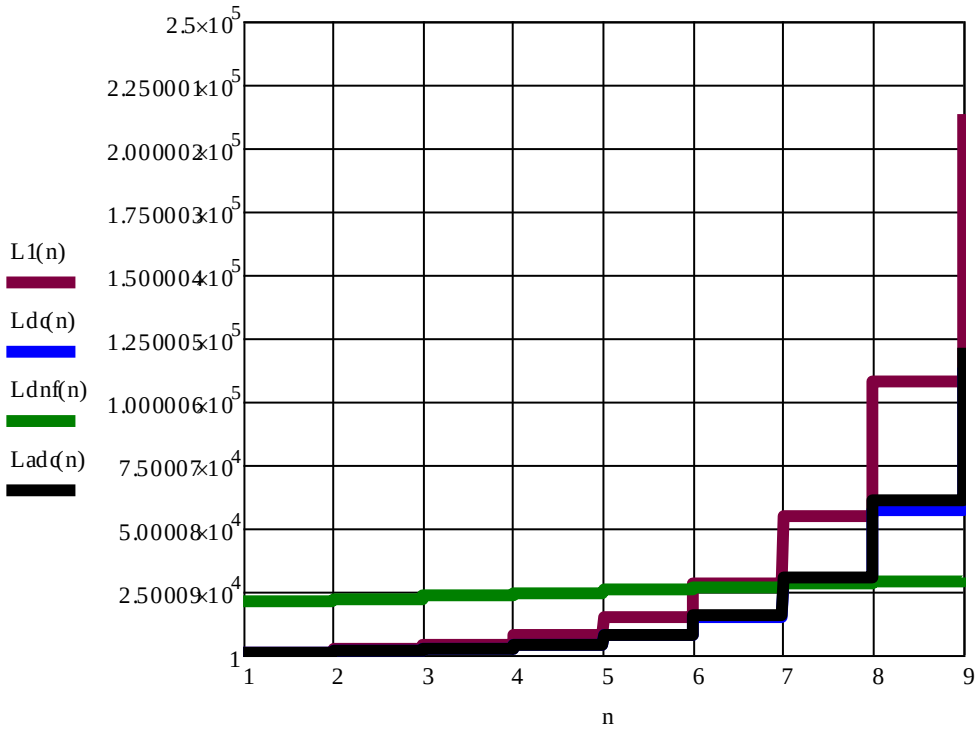


б)

Рисунок 4.17. Сравнение сложности L1(n), Ldc(n), Ldnf(n), Ladc(n) при: а) m=8; k=3; v=50; б) m=8; k=3; v=100;



а)



б)

Рисунок 4.18. Сравнение сложности L1(n), Ldc(n), Ldnf(n), Ladc(n) при: а) m=16, k=3, v=20; б) m=16, k=3, v=100

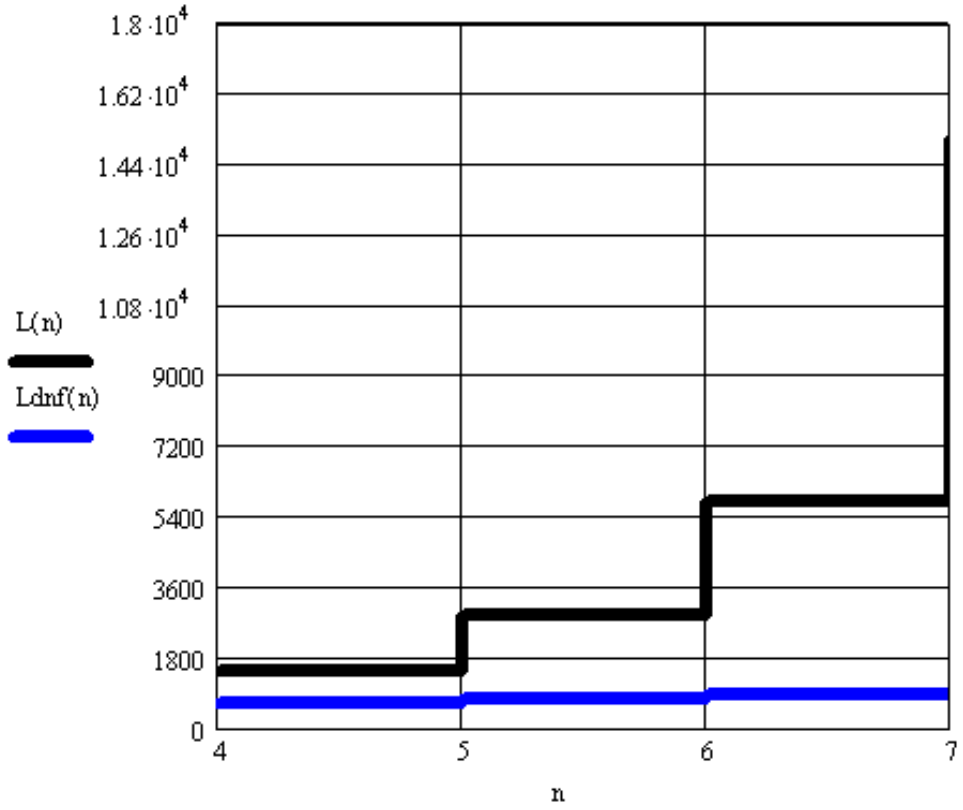


Рисунок 4.19. Сравнение сложности ДНФ-LUT (Ldnf(n)) и LUT(L(n)) для n 4-7: m=8, k=4; v=100

Таким образом, по сложности реализации систем логических функций в СДНФ адаптивный логический модуль ADC LUT выигрывает у LUT и не слишком проигрывает DC LUT, тем более, что ADC LUT является многофункциональным по сравнению с версиями LUT и DC LUT. Однако, при большом числе переменных предпочтительна ДНФ реализация программируемой логики, которая, в свою очередь проигрывает LUT и DC LUT при небольшом числе переменных (до 5-7), но резко вырывается вперед при большом числе переменных, хотя и очень сильно зависит от количества конъюнкций. Целесообразно иметь в составе адаптивных логических модулей как ADC LUT, так и программируемый блок ДНФ логики. В дальнейшем имеет смысл провести исследования по определению оптимального состава таких АЛМ.

4.4. Разработка алгоритма выбора оптимального набора логических элементов FPGA для реализации систем логических функций

В качестве вариантов реализации систем логических функций рассматриваются следующие. 1). LUT по числу требуемых функций в системе; 2). DC LUT на заданное максимальное число функций; 3) ДНФ-LUT на заданное максимальное число конъюнкций и функций; 4.) ADC LUT 5.) варианты комбинирования 1-4. Средние характеристики затрат w на реализацию различных систем логических функций получены путём анализа типовых проектов, загружаемых в ПЛИС. Сведём задачу выбора оптимальных наборов логических элементов для реализации заданных систем логических функций к задаче оптимизации назначений венгерским методом (Hungarian algorithm, Hungarian method) [78,79]. Соответствующее представление в терминах задачи о назначениях имеет вид Таблицы 4.1:

Таблица 4.1. Назначения вариантов наборов логических элементов для реализации различных систем логических функций в проектах

№ Варианты наборов логических элементов i	Средние характеристики различных систем логических функций j				
	1	2	3	…	s=n
1	W1.1	W1.2	W1.3	…	W1.s
2	W2.1	W2.2	W2.3	…	W2.s
3	W3.1	W3.2	W3.3	…	W3.s
.	.	.	.	.	.
v=n	Wv.1	Wv.2	Wv.3	…	Wv.s

Однако варианты реализации систем логических функций необходимо сравнивать не по одному, а по нескольким параметрам – по количеству транзисторов, по площади топологии, по энергопотреблению, по временной задержке. Предлагается решение венгерским методом по нескольким параметрам w с последующим построением из частных оптимальных решений множества Парето, в котором выбираются варианты при заданных

ограничениях [80,81]. Известный венгерский метод оптимизации (Hungarian algorithm, Hungarian method) [78,79] с матрицей W стоимостей w назначений n*n и унитарным назначением x (решение – матрица «не X») имеет модель:

$$\left\{\begin{array}{c} W=\left\|w_{ij}\right\|;colon(w_j);row(w_i);i=j=\overline{1,n}; \\ W\Rightarrow(X=\left\|x_{ij}\right\|)\Rightarrow(\overline{X}=\left\|\bar{x}_{ij}\right\|):if\,[x_{ij}=0]then(\bar{x}_{ij}=1),else(\bar{x}_{ij}=0); \\ [(\sum_{j=1}^{n}\bar{x}_{ij}=1)\&(\sum_{i=1}^{n}\bar{x}_{ij}=1)]\&[\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}\bar{x}_{ij}\to\min] \\ [\forall\mu\forall\xi[\mu\neq\xi]\&[colon(\bar{x}_{\mu})\cap colon(\bar{x}_{\xi})=\varnothing] \\ [row(\bar{x}_{\mu})\cap row(\bar{x}_{\xi})=\varnothing]\&[\xi=\overline{1,n}][\mu=\overline{1,n}]. \end{array}\right\} \quad (4.18)$$

Алгоритм «Hungarian method» [78,79] содержит следующие действия.

$$\begin{array}{l} Find[W\Rightarrow(X=\left\|x_{ij}\right\|)]: \\ Find\,(\min\ w_j\in[colon(w_j)]; \\ Find\,\{[colon(w_j)]-\min\ w_j\}; \\ 1.\quad Find\,(\min\ w_i\in[row(w_i)]; \\ Find\,\{[row(w_i)]-(\min\ w_i\}. \end{array} \quad (4.19)$$

$$If\{\left(\begin{array}{c} [(\sum_{j=1}^{n}\bar{x}_{ij}=1)(\sum_{i=1}^{n}\bar{x}_{ij}=1)]\& \\ [\forall\mu\forall\xi[\mu\neq\xi][colon(\bar{x}_{\mu})\cap colon(\bar{x}_{\xi})=\varnothing]\& \\ [row(\bar{x}_{\mu})\cap row(\bar{x}_{\xi})=\varnothing]\&[\xi=\overline{1,n}]\&[\mu=\overline{1,n}]. \end{array}\right)=True?\}then$$

$$2.\quad (\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}\bar{x}_{ij}\to\min)=\overline{X};Output(\overline{X});End; \quad (4.20)$$

$$else3.$$

$$\begin{array}{l} Find[(X^*=\left\|x^*{}_{ij}\right\|)=X\overset{delete}{-}M^0]:\forall i\forall j(x^*{}_{ij}\neq 0); \\ 3.\quad Find\ (M^0: \\ \{(\min\ \ [colon(w^0{}_j)])\&(\min\ \ [row(w^0{}_i)]\}). \end{array} \quad (4.21)$$

4. $Find(\theta = \min(X^* = \|x^*_{ij}\|))$.

5. $$Find\{X^\theta : colon(w^{X^\theta}{}_j) = \theta - colon(w^{X^*}{}_j)\}$$
$$\{X^\theta : row(w^{X^\theta}{}_i) = \theta + row(w^{M^0}{}_i)\}. \quad (4.22)$$

6. $Goto2$.

Схема известного алгоритма оптимизации венгерским методом [78,79] в соответствие с 4.18-4.22 представлена на рисунке 4.20.

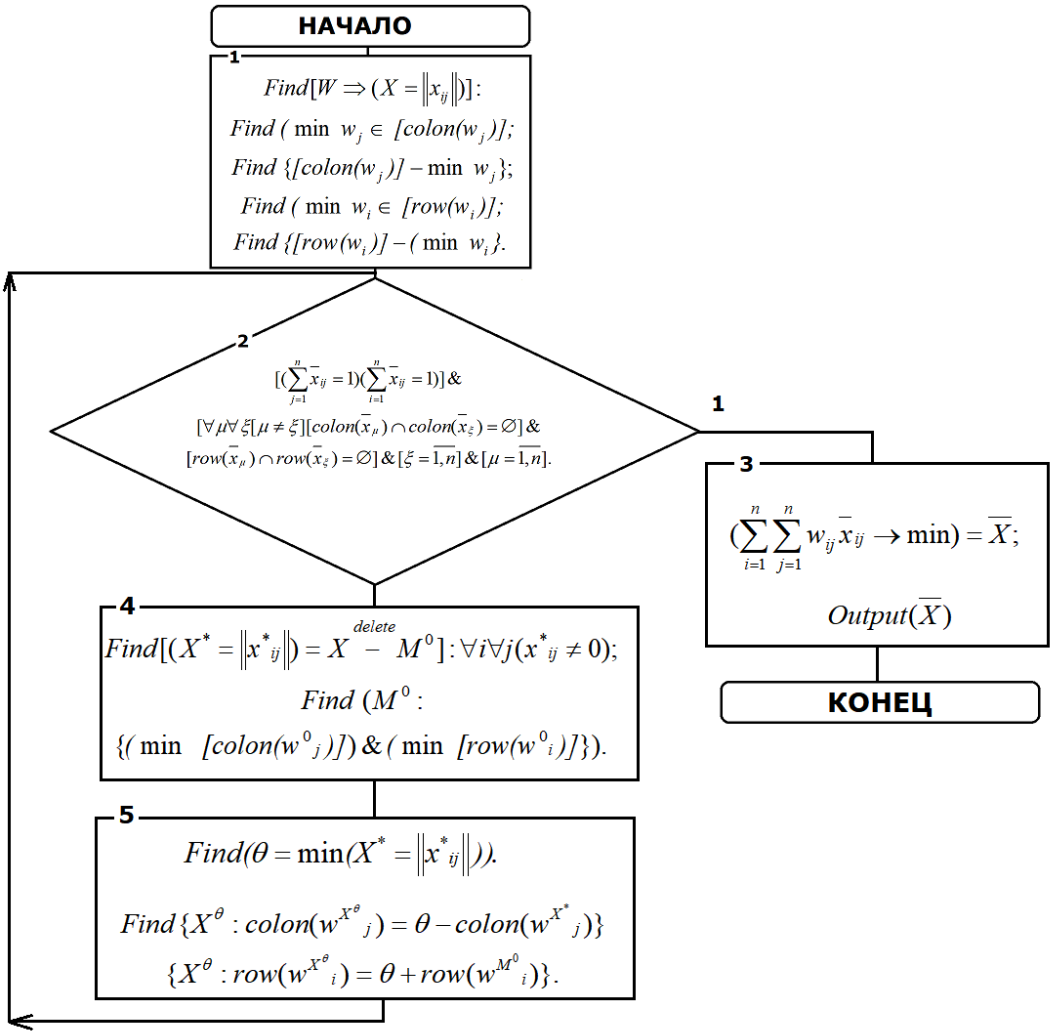


Рисунок 4.20. Схема известного алгоритма оптимизации венгерским методом

Предлагается использовать известный алгоритм оптимизации (Рисунок 4.13) итеративно - для получения нескольких назначений по параметрам

логических элементов и систем логических функций так, чтобы из частных решений в дальнейшем построить глобальную таблицу назначений и получить глобальные назначения, которые и описывают требуемый результат выбора [80,81]. Предлагаемый алгоритм оптимизации набора логических элементов ПЛИС FPGA для реализации типовых систем логических функций представлен на рисунке 4.21.

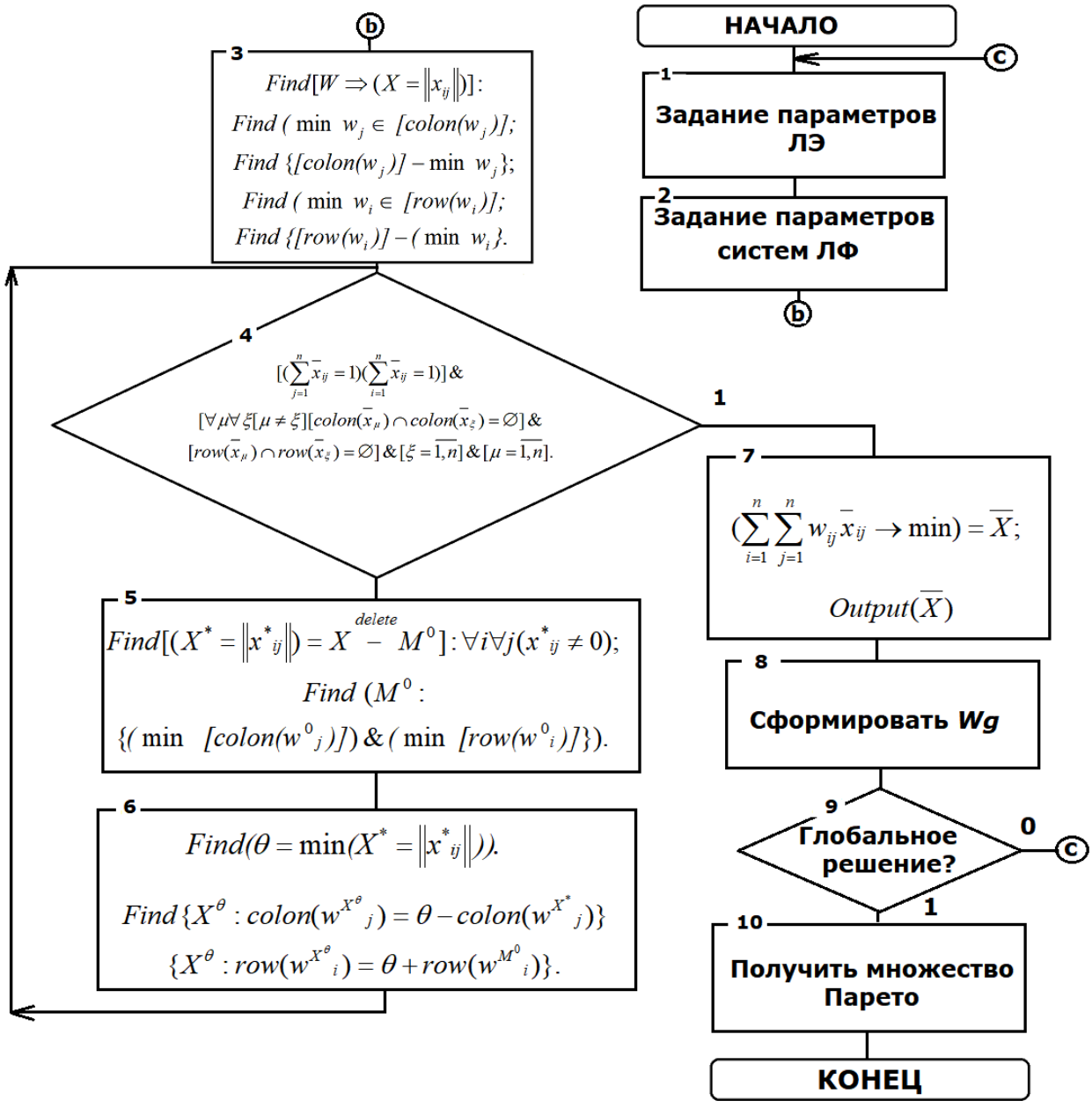


Рисунок 4.21. Алгоритм оптимизации набора логических элементов ПЛИС FPGA для реализации типовых систем логических функций

Вводятся процедуры задания параметров логических элементов (1) с использованием результатов расчета по полученным оценкам сложности и по результатам моделирования. Вводится процедура задания параметров систем логических функций (2) с использованием дополнительной информации, выходящей за рамки данного исследования. Вводится блок получения Парето-оптимальных решений (10). Разработанная программа [82] выбора оптимального набора логических элементов для реализации систем логических функций представлена в Приложении А.

4.5. Выбор оптимального набора логических элементов FPGA для реализации систем функций

Получим с помощью разработанной программы [82] оптимальные наборы ЛЭ для реализации систем функций с учетом показателей, предложенных в подразделах 4.1-4.4. Примем во внимание, что задержка с учётом декомпозиции DC-LUT в самом дешифраторе описывается выражением (4.23):

$$T_{n.k} = n + 2\left\lceil \frac{n}{\lfloor k \rfloor} \right\rceil, \tag{4.23}$$

где $2\left\lceil \frac{n}{\lfloor k \rfloor} \right\rceil$ учитывает задержку восстановления сигнал(восстановителя).

Задержка в блоках ИЛИ – фиксированная:

$$T_{n.k.OR} = 2. \tag{4.24}$$

В случае каскадирования необходимо учесть сложность декомпозиции аналогично (4.23). Итого для всего DC-LUT получаем выражение (4.24) без учета каскадирования:

$$T_{n.k} = n + 2\left\lceil \frac{n}{\lfloor k \rfloor} \right\rceil + 2. \tag{4.24}$$

. Для 1. последовательной цепочки с элементом ИЛИ (без каскадирования) получаем выражение (4.25):

$$T_{n.k.DNFposl} = 1 + 2n + 2\left\lceil \frac{n}{\lfloor k \rfloor} \right\rceil + 2. \tag{4.25}$$

Для параллельной цепочки получаем выражение (4.26):

$$T_{n.k.DNFpar} = 5 + 2\left\lceil \frac{n}{\lfloor k \rfloor} \right\rceil + 2. \qquad (4.26)$$

В случае использования варианта с подтягивающими транзисторами получаем выражение (4.27):

$$T_{n.k.R} = \tau_{NOT-R} + 2n + 2\left\lceil \frac{n}{\lfloor k \rfloor} \right\rceil + 2, \qquad (4.27)$$

где τ_{NOT-R}-задержка выходного инвертора конъюнкции с резистором.

результаты топологического моделирования, представленные в подразделе 3.6 при формировании множеств Парето по показателям сложности L (S), задержки T и энергопотребления E. с помощью предложенного алгоритма и разработанной программы множества предпочтительных вариантов. Результаты расчётов представлены на рисунке 4.22

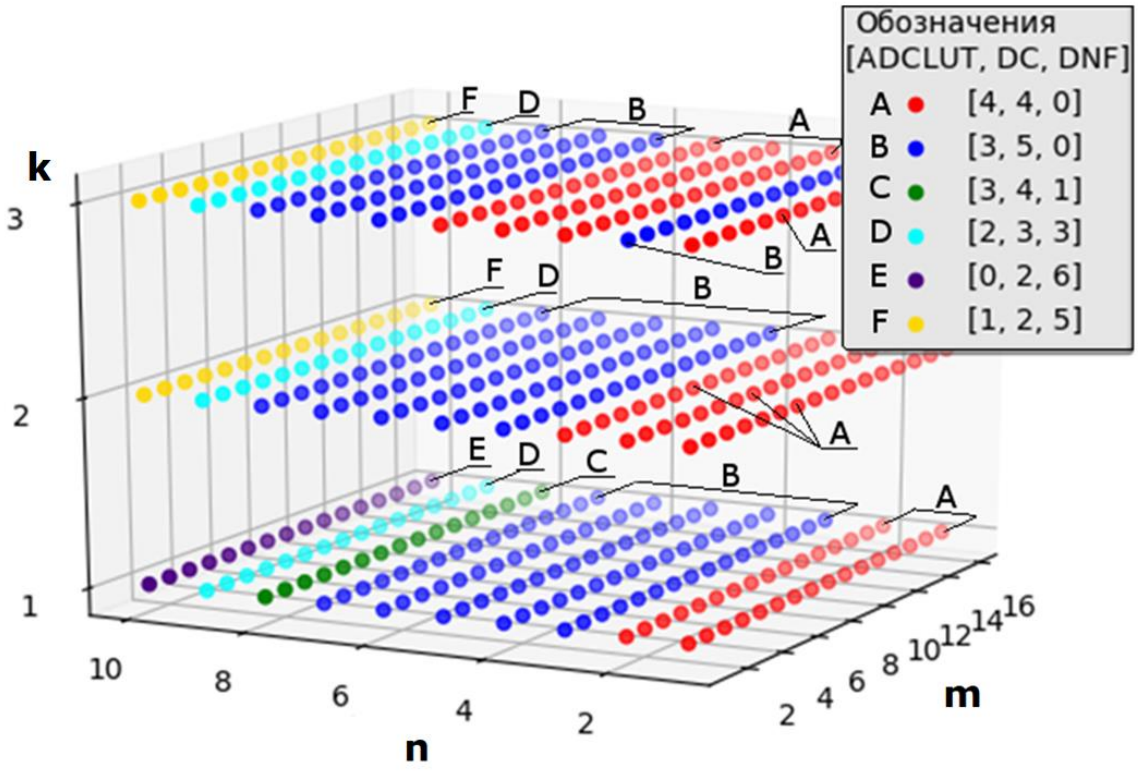


Рисунок 4.22. Результаты оптимизации – наборы A (4,4,0), B (3,5,0), C (3,4,1), D (2,3,3), E (0,2,6), F (1,2,5) для n=2,…10; m=2…16; k=1,2,3.

Пример полученного множества Парето с учетом максимальной задержки, энергопотребления, площади топологии по результатам рисунка 4.22 для m=4, n=4…7,k=1,v=20 представлен на рисунке 4.23.

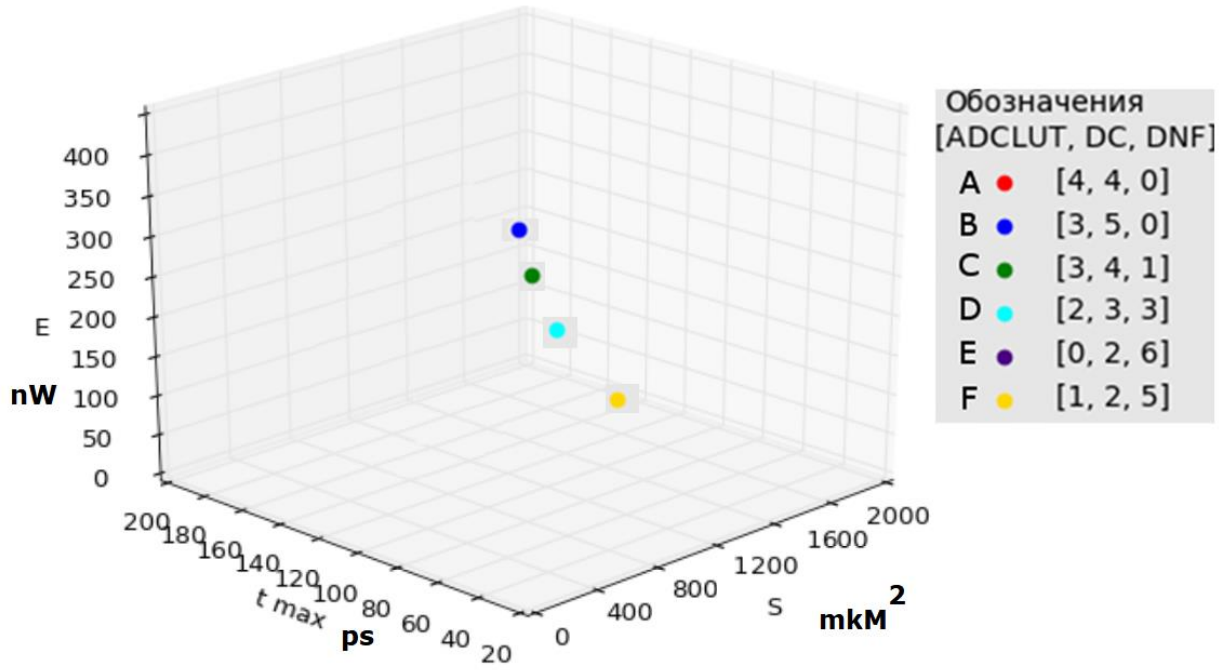


Рисунок 4.23. Множество Парето для параметров m=8, n=5…7, k=1, v=20…50

Полученные варианты и уточненные оценки (при доступном расширенном моделировании) могут быть использованы для формирования адаптивных логических элементов для ПЛИС трех ценовых категорий: малобюджетные ПЛИС с относительно невысоким быстродействием и небольшим количеством ЛЭ; ПЛИС «эконом класса» со средним быстродействием количеством ЛЭ и дорогостоящие ПЛИС с высоким быстродействием, сочетаемым с большим количеством элементов. Таким образом, например, варианты C, D, F могут быть использованы для компоновки соответствующих АЛМ различных категорий ПЛИС.

4.6. Выводы по главе 4

1. Предлагаемый DC-LUT FPGA предпочтительней по аппаратным затратам, чем известный LUT уже при количестве функций m=8 для числа переменных n=4. Наибольший выигрыш достигается при реализации дешифратора на n переменных, когда реализуются два в степени n функций, причем в каждой функции одна конъюнкция. В таких случаях целесообразно использовать ADC.

2. Предлагаемый логический элемент ПЛИС - DNF FPGA на основе ДНФ по сравнению с ЛЕ-СДНФ выигрывает при переходе к восьмиразрядным функциям (для n=k=m). При этом, существующий ЛЭ не может реализовать даже 10 разрядные функции, а предлагаемый имеет приемлемые затраты даже для 12-разрядных функций. Причём задержка предлагаемого варианта так же, как и известного, определяемого в основном длиной цепочки передающих транзисторов – n, определяется цепочкой транзисторов в блоках программируемых конъюнкций – это тоже n, а цепочки в блоках программируемых функций содержат всего один транзистор.

3. Оптимизация логики венгерским методом позволила установить, что обычный LUT не вошёл в оптимальный набор при рассматриваемых

п

а

р

а

м

е

т

р 4. Во множество Парето для параметров m=8, n=5…7, k=1, v=20…50 **в**ошли наборы B(3,5,0), C(3,4,1), D(2,3,3), F(1,2,5).

х

,

а

н

Заключение

Выполненная диссертационная работа посвящена решению актуальной научной задачи выбора оптимального набора логических элементов для адаптивно-гибридной реализации систем логических функций в ПЛИС FPGA. В диссертационной работе поставлены и решены следующие задачи исследования:

1. Разработан усовершенствованный метод реализации логических функций в СДНФ на основе предложенного адаптивного логического элемента ADC-LUT, который может работать как в режиме LUT, так и в режиме DC-LUT. Адаптивный ЛЭ в режиме DC-LUT реализует системы функций, что позволяет снизить аппаратные затраты ориентировочно на 15-20% при реализации 8 функций от 4 входных переменных.

2. Разработан усовершенствованный метод реализации логических функций в ДНФ, отличающийся тем, что использовано оригинальное кодирование конъюнкций, что способствует реализации систем функций большего числа переменных, практически нереализуемых в LUT.

3. Получены оценки сложности реализации систем логических функций на основе разработанных логических элементов, которые позволили оценить выигрыш в количестве транзисторов, в площади топологии и доказать предпочтительность DC-LUT в сравнении с LUT при реализации систем функций большого числа переменных.

4. Разработаны и исследованы модели предложенных логических элементов ПЛИС FPGA для реализации типовых систем логических функций, которые показывают, что DC-LUT, ADC-LUT обладают большими функциональными возможностями при незначительном снижении задержки и существенном выигрыше в энергопотреблении перед реализацией на нескольких существующих LUT систем, содержащих большое число функций от одинаковых переменных.

5. Разработан алгоритм оптимизации набора логических элементов FPGA для реализации систем логических функций, используемых в типовых

проектах на ПЛИС, который позволяет на основе венгерского метода и получения множества Парето установить наиболее предпочтительные наборы предлагаемых ЛЭ в АЛМ.

6. Направлениями дальнейшей работы может быть использование транзисторов Tri-Gate, а также разработка новых подходов к обеспечению надёжности предложенных ЛЭ, а также адаптация к области самосинхронной схемотехники.

Список сокращений

АЛМ – адаптивный логический модуль;

БИС – большая интегральная схема

БМК— базовые матричные кристаллы

БПИ – Блок преобразования информации

ДНФ – дизъюнктивная нормальная форма;

ИМС — интегральная микросхема;

ИПИ РАН - Институт проблем информатики Российской академии наук

ИПУ РАН - Институт проблем управления Российской академии наук

КЛБ - конфигурируемый логический блок;

КМОП - комплементарная структура металл-оксид-полупроводник

ЛЭ — логический элемент;

ЛС — логическая схема;

МОП - металл-оксид-полупроводник

ОЗУ— оперативное запоминающее устройство;

ПАО ПНППК - Публичное акционерное общество Пермская научно-производственная приборостроительная компания

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.

ТЦ МИЭТ – технологический центр Московского института электронной техники

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;

DC – Decoder

FPGA – Field-Programmable Gate Array

LAB - logic array block

LE - Logic Element (Basic Logic Element);

LUT – Look Up Table, генератор логических функций;

MLAB - Memory LAB

PAL - Programmable Array Logic

PLA - Programmable logic array

SB – Switch Box

SRAM - Static Random Access Memory

ULA - Uncommited Logic Array

Список литературы

1. Лазарев В.Г. Построение программируемых управляющих устройств / В.Г. Лазарев, Е.И. Пийль, Е.Н. Турута – М.: Энергоиздат, 1984.– 193с.
2. Баранов С.И. Цифровые устройства на программируемых БИС с матричной структурой / С.И. Баранов, В.А. Скляров – М.: Радио и связь,1986. –272с.
3. Programmable Array Logic. [Электронный ресурс]. – Режим доступа: https://www.electrical4u.com/programmable-array-logic/
4. Закревский А. Д. Алгоритмы синтеза дискретных автоматов. / А. Д. Закревский – М.: Наука, 1971. –511 с.
5. Лазарев В.Г. Построение программируемых управляющих устройств / В.Г. Лазарев, Е.И. Пийль, Е.Н. Турута – М.: Энергоиздат, 1984.– 193с.
6. Угрюмов Е. П. Цифровая схемотехника: учебное пособие / Е. П. Угрюмов – СПб: БХВ-Петербург, 2010. – 816 с.
7. Тюрин С.Ф. Вычислительная техника и информационные технологии. Цифровая схемотехника: учебное пособие. Пермь, издательство Перм. гос. техн. ун-та, 2008. –137 с.
8. Схемотехника: учеб. пособие / С.Ф. Тюрин, Пермь: Изд-во Перм. нац. иссл. политех. ун-та, 2017. –169с.
9. САПР "КОВЧЕГ" [Электронный ресурс]. – Режим доступа: http://www.asic.ru/index.php?option=com_content&view=article&id=61&Itemid=6.
10. Степченков Ю.А. Библиотека самосинхронных элементов для технологии БМК / Ю.А. Степченков //Проблемы разработки перспективных микроэлектронных систем – 2006. Сборник научных трудов /Под общ. ред. А.Л. Стемпковского. – М.: ИППМ РАН, 2006. – С. 259-264.
11. Каменских А.Н. Комбинированное резервирование самосинхронных схем. Автореферат дисс. … канд. техн. наук / Каменских Антон Николаевич Пермь, 2016 – 138 с.

12. Бернард К. Коул. Фирма Altera готовит производство стираемых ПЛИС на 5000 логических вентилей и 60 МГц / К. Коул. Бернард //Журнал "Электроника" том 61, No.10 (792), 1988г - пер. с англ. М.: Мир, 1988. С.22-26.
13. Бёрски Д. Быстродействующие матричные ИС, повышающие производительность вычислительных систем/ Д. Бёрски // Журнал "Электроника" том 61, No.25 (804), 1988г - пер. с англ. М.: Мир, 1988. С.17-26
14. Виды программируемой логики. [Электронный ресурс]. – Режим доступа: http://www.pvsm.ru/programmirovanie/87810.
15. Programmable Logic Devices. [Электронный ресурс]– Режим доступа: http://ee.sharif.edu/~logic_circuits_t/readings/PLD.pdf.
16. Brown S. Architecture of FPGAs and CPLDs: A Tutorial. [Электронный ресурс] / S. Brown, J. Rose – Режим доступа: http://www.eecg.toronto.edu/~jayar/pubs/brown/survey.pdf.
17. 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.
18. Строгонов А. Программируемая коммутация ПЛИС: взгляд изнутри [Электронный ресурс] / А. Строгонов, С. Цыбин – Режим доступа: http://www.kit-e.ru/articles/plis/2010_11_56.php.
19. Программируемые пользователем вентильные матрицы (FPGA) [Электронный ресурс]. – Режим доступа: http://digteh.ru/digital/FPGA.
20. Архитектура ПЛИС (FPGA) [Электронный ресурс]. – Режим доступа: http://marsohod.org/11-blog/265-fpga.
21. 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

22. Understanding How the New Intel®HyperFlex™ FPGA Architecture Enables Next Generation High-Performance Systems [Электронный ресурс]. – Режим доступа: https://www.altera.com/products/fpga/stratix-series/stratix10/features.html#hyperflexarchitecture
23. Обзор архитектуры ПЛИС семейства Virtex-5 [Электронный ресурс]. – Режим доступа: http://elektors.ru/radioelektronika/mikroshemy/2759-obzor-arhitektury-plis-semeystva-virtex-5.html
24. Капитанов В. Топологическая и временная оптимизация проектов на ПЛИС XILINX. [Электронный ресурс]/ В. Капитанов, П. Володин. – Режим доступа: http://www.compitech.ru/html.cgi/arhiv/00_01/stat_22.htm
25. Оптимизация конфигурации ПЛИС по площади (Уменьшение занимаемых ресурсов) [Электронный ресурс]. – Режим доступа: https://bovs.org/post/163/Optimizazia-konfigurazii-PLIS-po-plosadi-%28Umen_senie-zanimaemyh-resursov%29
26. FPGA Architecture White Paper. [Электронный ресурс]. – Режим доступа: https://www.altera.com/en_US/pdfs/literature/wp/wp-01003.pdf
27. Alireza Kaviani HYBRID FPGA ARCHITECTURE. [Электронный ресурс] / Alireza Kaviani Stephen Brown – Режим доступа: https://sydney.edu.au/engineering/electrical/people/philip.leong/UserFiles/File/theses/cwyu10.pdf
28. Chi Wai, Yu. Hybrid FPGA: Architecture and Interface. [Электронный ресурс] / Chi Wai, Yu. – Режим доступа: https://sydney.edu.au/engineering/electrical/people/philip.leong/UserFiles/File/theses/cwyu10.pdf
29. Manasi R. Mali PERFORMANCE OPTIMIZATION OF LUT IN FPGA USING CNFET. [Электронный ресурс] / Manasi R. Mali, Dr. S.D Pable, Prof R.S Khule. – Режим доступа: https://zenodo.org/record/60127#.WIMqglWLTcs

http: // www.ijesrt.com International Journal of Engineering Sciences & Research Technology

30. Cliff R. G. Look up table implementation of fast carry for adders and counters / R. G. Cliff, L. T. Cope, K. Veenstra, B. B. Pederson // European patent specification US 005274581A, publication 28.12.1993.
31. Yervant Z. Gest editors' introduction: Design for Yield and reliability / Z. Yervant, G. Dmytris // IEEE Design & Test of Computers. – May–June 2004. – Pp. 177-182.
32. 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.
33. Kharchenko V. Green IT Engineering: Concepts, Models,Complex Systems Architectures, Studies in Systems, Decision and Control, / V. Kharchenko, Y. Kondratenko, J. Kacprzyk // Heidelberg: Springer International Publishing, Vol. 74. Berlin, 2017, DOI: 10.1007/978-3-319-44162-7.
34. Nikil Mehta An ultra-low-energy, variation-tolerant FPGA architecture using component-specific mapping. Dissertation (Ph.D.), California Institute of Technology [Электронный ресурс] / Mehta, Nikil. – Режим доступа: http://thesis.library.caltech.edu/7226/1/Nikil-Mehta-2013.pdf
35. Drozd, A. The levels of target resources development in computer systems / Drozd, A. // Design & Test Symposium (EWDTS), 2014 East-West, pp. 1-5.
36. Tyurin, S.F. Retention of functional completeness of Boolean functions under "failures" of the arguments / S.F. Tyurin, // Automation and Remote Control 60 (9 PART 2) -1999- PP. 1360 – 1367.
37. Tyurin, S.F. Green Logic: Green LUT FPGA concepts, models and evaluations / S.F. Tyurin, // Studies in Systems, Decision and Control Volume 105, 2017 - P. 241-261 DOI: 10.1007/978-3-319-55595-9_12

38. Греков А.В. Повышение отказоустойчивости конфигурируемых блоков, программируемых логических интегральных схем на основе функционально полных толерантных элементов: диссертация на соискание ученой степени канд. техн. наук: 05.13.05 / Греков Артем Владимирович. - Пермь, 2011-267с.
39. Громов О.А. Повышение отказоустойчивости программируемых логических интегральных схем на основе КМОП элементов с избыточным базисом. Автореферат дис… канд. техн. наук: 05.13.05 / Громов Олег Александрович. - Пермь, 2013-16с.
40. Городилов А.Ю. Методы и алгоритмы диагностирования и реконфигурации логики высоконадёжных ПЛИС. Автореферат дисс. … канд. техн. наук / Городилов Алексей Юрьевич. - Пермь, 2016.-19с.
41. 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.
42. Аксенова Г.П. Контролепригодная архитектура для самотестирования в программируемых логических матричных структурах / Г.П. Аксенова // Автоматика и телемеханика. 2010. – №12. – С. 154-165.
43. Цыбин С.А. Проектирование высокоинтегрированных программируемых логических интегральных схем по субмикронным проектным нормам. Диссертация на соискание учёной степени канд. техн. наук: 05.27.01 [Электронный ресурс] / Цыбин Сергей Александрович – Воронеж, 2010 – 135с. – Режим доступа: http://www.dissercat.com/content/proektirovanie-vysokointegrirovannykh-programmiruemykh-logicheskikh-integralnykh-skhem-po-su. Научная библиотека диссертаций и авторефератов disserCat http://www.dissercat.com/content/proektirovanie-vysokointegrirovannykh-programmiruemykh-logicheskikh-integralnykh-skhem-po-su#ixzz3sl1sJ1o7
44. Давыдов С. И. Проектирование функциональных блоков программируемой логической интегральной схемы, конфигурируемых

с использованием метода сканирования пути. Диссертация на соискание учёной степени канд. техн. наук: 05.27.01 [Электронный ресурс] / Давыдов Сергей Игоревич Воронеж, 2013 – 111с. – Режим доступа: http://www.dslib.net/tverdoteln-elektronika/proektirovanie-funkcionalnyh-blokov-programmiruemoj-logicheskoj-integralnoj.html

45. Быстрицкий А.В. Проектирование структуры межсоединений программируемых логических интегральных схем. Диссертация на соискание учёной степени канд. техн. наук: 05.27.01 [Электронный ресурс] / Быстрицкий Алексей Викторович Воронеж, 2012 – 143с. – Режим доступа: http://www.dslib.net/tverdoteln-elektronika/proektirovanie-struktury-mezhsoedinenij-programmiruemyh-logicheskih-integralnyh.html
46. Строгонов А. Новая серия ПЛИС 5578, разработанная в рамках импортозамещения зарубежной электронной компонентной базы / А. Строгонов, С. Цыбин, П. Городков // Компоненты и технологии. 2017- Т. 2- № 187- С. 46-49.
47. Что такое критические технологии? [Электронный ресурс]. – Режим доступа: http://www.vest-news.ru/article.php?id=387.
48. Перечень критических технологий Российской Федерации 21.05. 2006 г. Пр-842. [Электронный ресурс]. – Режим доступа: http://mon.gov.ru/dok/ukaz/nti/4407/.
49. Указ президента Российской Федерации «Об утверждении приоритетных направлений развития науки, технологий и техники в Российской Федерации и перечня критических технологий Российской Федерации».12.05.2011. [Электронный ресурс]. – Режим доступа: http://mon.gov.ru/dok/npa/prez/8479/.
50. Проблемы создания отечественной элементной компонентной базы. [Электронный ресурс]. – Режим доступа: http://www.electronics.ru/journal/article/295.
51. Tyurin, S. Green Logic: Green LUT FPGA concepts, models and evaluations

(2017) Studies in Systems, Decision and Control, 105, pp. 241-261. www.springer.com/series/13304. doi: 10.1007/978-3-319-55595-9_12

52. Вихорев Р.В. Особенности моделирования и оптимизации комплекта новых логических элементов ПЛИС/Р.В. Вихорев //Вестник Пермского университета. Серия: Математика. Механика. Информатика-2018.-№ 3.-С.111-116.
53. Vikhorev R.V. Implementation of systems of logic functions based on LUT FPGA./ R.V. Vikhorev // Инновационные процессы в исследовательской и образовательной деятельности. - 2015. - Т.1. - С. 13-16.
54. Вихорев Р.В. Логическая ячейка для реализации систем функций / Р.В. Вихорев // 14-я Международная конференция «Авиация и космонавтика-2015» - Москва. -2015 - С. 170-171.
55. Тюрин С.Ф. Модифицированный логический элемент LUT FPGA. / С.Ф. Тюрин, А.Ю. Городилов, Р.В. Вихорев // Вестник Пермского университета. Серия: Математика. Механика. Информатика. - 2014. - № 1 (24). - С. 69-74.
56. Тюрин С.Ф. Программируемое логическое устройство /
57. С.Ф. Тюрин, А.Ю. Городилов, Р.В. Вихорев // Патент на изобретение №2547229; опубл. 10.04.2015, Бюл. №10. – 16 с.
58. Тюрин С.Ф. Программируемое логическое устройство /
С.Ф. Тюрин, Р.В. Вихорев, А.Ю. Плотникова // Патент на изобретение № 2602780; опубл. 20.11.2016, Бюл. №32. -19 с.
59. Тюрин С.Ф. Программируемое логическое устройство /
С.Ф. Тюрин, Р.В. Вихорев // Патент на изобретение №2573732; опубл. 27.01.2016, Бюл. №3. -15 с.
60. Тюрин, С.Ф. Усовершенствованный метод реализации в FPGA систем логических функций, заданных в СДНФ [Электронный ресурс] /
61. С.Ф. Тюрин, Р.В. Вихорев // Инженерный вестник Дона. - 2017. - № 1. - 10 с. – Режим доступа: http://www.ivdon.ru/ru/magazine/archive/n1y2017/4044

62. Вихорев Р.В. Программируемые логические элементы ПЛИС FPGA для реализации систем логических функций / Р.В. Вихорев, С.Ф. Тюрин // Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления.- 2017. - № 23. - С. 133-145.
63. Carver A. Mead Introduction to VLSI Systems [Электронный ресурс] / Carver A. Mead, Lynn Conway. – Режим доступа: http://ai.eecs.umich.edu/people/conway/VLSI/VLSIText/PP-V2/V2.pdf; https://ru.scribd.com/document/104510240/VLSI-Introduction-to-VLSI-Systems-Mead-amp-Conway.
64. Тюрин С.Ф. Адаптивный логический модуль ПЛИС с архитектурой FPGA. / С.Ф. Тюрин, Р.В. Вихорев // Вестник Рязанского государственного радиотехнического университета. - 2018. - №63. - С.69-76.
65. Тюрин С.Ф. Логический элемент на основе последовательной цепочки переменных для ДНФ конфигурирования FPGA. Проектирование и технология электронных средств. 2016. № 3. С. 50-55.
66. Вихорев Р.В. Усовершенствованные методы реализации программируемой логики. / Р.В. Вихорев, А.С. Прохоров, А.Ю. Скорнякова, С.Ф. Тюрин // Управление большими системами. УБС-2017 материалы XIV Всероссийской школы-конференции молодых ученых. - Пермь- 2017. - С. 306-315
67. Tyurin S. Advanced FPGA Look up tables. / S. Tyurin, A. Grekov, R. Vikhorev, A. Prokhorov // International Journal of Pure and Applied Mathematics. - 2017 - vol. 117 - no. 22. - PP. 143-147.
68. Сайт разработчика National Instruments [Электронный ресурс]. – Режим доступа: http://www.ni.com/multisim/
69. Вихорев Р.В. Моделирование и оптимизация инновационных логических элементов ПЛИС./ Р.В. Вихорев, А.С. Прохоров, С.Ф. Тюрин, А.С. Никитин // Вестник Пермского национального

исследовательского политехнического университета. Электротехника, информационные технологии, системы управления. - 2017. - № 24. - С. 192-208.

70. N. Paydavosi, BSIM4v4.8.0 MOSFET Model -User's Manual 2013 [Электронный ресурс] Режим доступа: http://bsim.berkeley.edu/BSIM4/BSIM480.zip
71. Сайт САПР MicroWind [Электронный ресурс] Режим доступа: https://www.microwind.net
72. Шунков В. Проектные нормы в микроэлектронике: где на самом деле 7 нанометров в технологии 7 нм [Электронный ресурс] / В. Шунков Режим доступа: https://habr.com/ru/post/423575/
73. Вихорев Р.В Топологическое моделирование логических элементов нейронных сетей / Р.В. Вихорев, М.С. Никитин, С.Ф. Тюрин // Нейрокомпьютеры: разработка, применение. - 2019. - № 73. - С. 45-55
74. Вихорев Р.В. Моделирование усовершенствованных устройств программируемой логики / Р.В.Вихорев, А.Ю.Скорнякова //Вестник пермского университета. Серия: Математика. Механика. Информатика,-2017,-Вып.3-С.77-81.
75. Тюрин С.Ф., И.И. Безукладников. Особенности логики FPGA Stratix III. Информационно-измерительные и управляющие системы. 2016. № 9. С. 12-16.
76. Vikhorev R. Universal logic cells to implement systems functions. / R. Vikhorev // Proceedings of the 2016 IEEE North West Russia Section Young Researchers in Electrical and Electronic Engineering Conference, EIConRusNW-2016, 2016 - PP. 404-406. (Scopus)
77. Vikhorev R. Improved FPGA logic elements and their simulation. / R. Vikhorev //Proceedings of the 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, EIConRus - 2018 - PP. 275-280. (Scopus)

78. Harold W. Kuhn, The Hungarian Method for the assignment problem / W. Kuhn Harold // Naval Research Logistics Quarterly, 2- 1955- P.83–97 doi:10.1002/nav.3800020109
79. Hungarian algorithm [Электронный ресурс]. – Режим доступа: http://www.hungarianalgorithm.com/solve.php.
80. Тюрин С.Ф Выбор набора конфигурируемых логических элементов с использованием венгерского метода / С.Ф. Тюрин, А.С. Никитин, Р.В. Вихорев, А.Ю. Скорнякова // Вестник пермского университета. Серия: Математика. Механика. Информатика – 2017 - Вып. 2(37) - С. 65-68.
81. Никитин А.С. Оптимизация LUT FPGA на основе модифицированного венгерского метода /А.С. Никитин, Р.В. Вихорев, А.Ю. Скорнякова //Управление большими системами УБС-2017 Материалы 14 Всероссийской школы-конференции молодых ученых - Пермь, - 2017.- С. 563-572.
82. Свидетельство о государственной регистрации программы для ЭВМ №2017663289 «Программа оптимизации набора логических элементов модифицированным венгерским методом «ВЕННИТ»» / Тюрин С.Ф., Никитин А.С., Вихорев Р.В., Скорнякова А.Ю., Прохоров А.С. Дата регистрации 28.11.2017

Приложение А

Программа оптимизации набора логических элементов модифицированным венгерским методом «ВЕННИТ»

1.Описание программы «ВЕННИТ»

1.1. Общие сведения.

Для работы программы требуется наличие установленного интерпретатора языка Python, и наличие библиотеки tkinter, предназначенной для построения графического интерфейса (GUI), которая входит в стандартный дистрибутив языка Python.

1.2. Функциональное назначение.

Программа по заданной размерности строит таблицу сложностей так, что каждой строке таблицы соответствует некоторая формула, содержащая аргументы m, n, r – а каждому столбцу – набор из этих трёх аргументов. Каждой ячейке таблицы приходит в соответствие некоторое число, вычисленное по формуле и аргументам соответствующей строки и столбца.

При этом программа выбирает некоторые элементы таблицы таким образом, что в каждой строке и каждом столбце есть строго по одному выбранному элементу, и сумма выбранных элементов является минимально возможной для данного условия.

1.3. Описание логической структуры.

Программа состоит из интерпретируемого программного кода. Текст программы состоит из алгоритма, рассчитывающего результат по заданному условию и графический интерфейс, являющийся некоторой обёрткой над алгоритмом, позволяющий задавать условия задачи, а также рассчитывать матрицу по заданным условиям, и выводить результат работы в удобном виде.

1.4. Используемые технические средства.

Для исполнения программы достаточно ЭВМ с интерпретатором языка Python. Программа может работать на любой ЭВМ под управлением

операционных систем семейств Windows, macOS, GNU/Linux, BSD и любых других, на которые поставляется Python.

1.5. Вызов и загрузка.

Вызов программы осуществляется путём запуска интерпретируемых файлов используя любую программу-интерпретатор языка Python.

1.6. Входные данные.

Входными данными программы является заданная размерность квадратной матрицы, а также заданные для каждой строки соответствующей формулы, а каждому столбцу - соответствующих аргументов формул.

1.7. Выходные данные.

Выходными данными является: список паросочетаний вида “[строка, столбец]” размерности, соответствующей размерности задачи, являющихся результатом работы алгоритма решения задачи. Помимо этого, программа выводит таблицу с вычисленными числами по заданным входным данным, с выделенными элементами матрицы, соответствующих решению задачи.

2. Текст программы.

```
#!/usr/bin/python
# -*- encoding: UTF-8 -*-
from tkinter import *
from math import inf as INF
def hungarian(matrix): ## реализация венгерского алгоритма на матрице matrix
    height = int(len(matrix))
    width = int(len(matrix[0]))
    u=[0]*height
    v=[0]*width
    markIndices=[]
    for i in range(0, width):
        markIndices.append(-1)
    for i in range(0, height):
```

```
        links=[-1]*width
        mins=[INF]*width

    visited=[0]*width
        markedI = i
        markedJ = -1
        while markedI != -1:
            j=-1
            for j1 in range(0, width):
                if not visited[j1]:
                    if matrix[markedI][j1] - u[markedI] - v[j1] < mins[j1]:
                        mins[j1] = matrix[markedI][j1] - u[markedI] - v[j1]
                        links[j1]=markedJ
                    if j==-1 or mins[j1] < mins[j]:
                        j=j1
            delta = mins[j]
            for j1 in range(0, width):
                if visited[j1]:
                    u[markIndices[j1]]+=delta
                    v[j1]-=delta
                else:
                    mins[j1]-=delta
            u[i]+=delta
            visited[j]=1
            markedJ=j
            markedI = markIndices[j]
        while links[j]!=-1:
            markIndices[j]=markIndices[links[j]]
            j=links[j]
        markIndices[j]=i
```

```
            result=[]
            for j in range(0, width):

                if markIndices[j]!=-1:
               result.append([markIndices[j],j])
            ##вывод вида [row, col]
            return result
        vals_row = []
        vals_col = []
        vals_row_text=[]
        vals_col_text=[]
        def create_table(config_root): ## формирование таблицы и вывод результатов
            size = scale_size.get()
            global vals_row_text
            global vals_col_text
            vals_row_text=[]
            vals_col_text=[]
            for i in range(0, size):
                row_info=vals_row[i].get()
                col_info=vals_col[i].get()
                vals_row_text.append(row_info)
                vals_col_text.append(col_info)
            frames_table=[]
            table = [[0] * size] * size
            matrix = configure_table(table) ## формирование таблицы значений
            if matrix: ## в случае успешного формирования таблицы значений
                result= hungarian(matrix) ## поиск совершенного паросочетания с минимальной стоимостью
                table_root = Tk() ## формирование таблицы для вывода на экран
```

```
                for i in range(0, size):
                    frames_table.append(Frame(table_root))

                    frames_table[i].pack(side='top')
                    for j in range(0, size):
                        table[j][i]=Entry(frames_table[i])
                        table[j][i].pack(side='left')
                        table_field = str(matrix[i][j])
                        for char_index in range(0, len(table_field)):
                            table[j][i].insert(char_index,table_field[char_index])
                        if [i,j] in result: ## выделение ячеек, соответствующих оптимальному решению
                            table[j][i].configure( readonlybackground='red',bd='1',highlightthickness='1',highlightbackground='black',font = "Helvetica 12 bold",justify="center",state='readonly')
                        else:
                            table[j][i].configure( readonlybackground='white',bd='1',highlightthickness='1',highlightbackground='black',font = "Helvetica 12 bold",justify="center",state='readonly')
                table_root.title('Матрица назначений')
                table_root.resizable(False,False)
                table_root.mainloop()
            else: ## в случае неуспешного формирования таблицы значений выводим сообщение об ошибке
                throw_error()
        def throw_error():
            error_root = Tk()
            error_root.title("Ошибка")
            l=Label(error_root, text='Ошибка! Проверьте правильность введенных формул и параметров!')
```

```
        error_root.resizable(False, False)
        error_root.mainloop()

    def configure_table(table): ## формирование таблицы значений на основе введенных данных
        ## в случае успешного выполнения возвращает таблицу значений, иначе - False
        size = scale_size.get()
        table=[]
        print(vals_row_text)
        print(vals_col_text)
        for row in range(0,size):
            table.append([])
            for col in range(0,size):
                try:
                    formula = 'table[row].append('+vals_row_text[row]+')'
                    m,n,k = vals_col_text[col].split()
                    m=int(m)
                    n=int(n)
                    k=int(k)
                    exec(formula)
                except:
                    return False
            return table
    def config_table(root): ## вывод на экран таблицы для ввода данных
        global vals_row_text
        global vals_col_text
        global vals_row
        global vals_col
        size = scale_size.get()
```

```
        vals_row = []
        labels_row = []

        frames_row = []
        vals_col = []
        labels_col = []
      frames_col = []
        config_root = Tk()
        config_root.resizable(False, False)
        config_root.title("Параметры матрицы")
        config_root.geometry('520x'+str(size*20+50))
        while len(vals_row_text)<size:
            vals_row_text.append('')
        while len(vals_col_text)<size:
            vals_col_text.append('')
        config_frame_top = Frame(config_root)
        config_frame_top.pack()
        left_frame = Frame(config_frame_top)
        left_frame.pack(side='left')
        right_frame = Frame(config_frame_top)
        right_frame.pack(side='right')
        config_rows_label = Label(left_frame, text="Формулы для строк")
        config_rows_label.pack()
        config_cols_label = Label(right_frame, text="Параметры для
столбцов[m,n,k]")
        config_cols_label.pack()
        for i in range(0, size):
            frames_row.append(Frame(left_frame))
            vals_row.append(Entry(frames_row[i]))
```

```
                labels_row.append(Label(frames_row[i],
text='Формула '+str(i+1)))
                frames_row[i].pack()

                vals_row[i].pack(side='right')
                labels_row[i].pack(side='left')
                frames_col.append(Frame(right_frame))
                vals_col.append(Entry(frames_col[i]))

             labels_col.append(Label(frames_col[i], text='Столбец '+str(i+1)))
                frames_col[i].pack()
                vals_col[i].pack(side='right')
                labels_col[i].pack(side='left')
                vals_row[i].insert(0, vals_row_text[i])
                vals_col[i].insert(0, vals_col_text[i])
        config_button = Button(config_root, text="Расчитать")
        config_button.pack()
        config_button.bind("<Button-1>",create_table)
        config_root.mainloop()
    if __name__ == '__main__':
        root = Tk()
        root.title("Решение задачи о назначениях")
        scale_size_label=Label(root,text='Размерность    матрицы',font='arial
12')
        scale_size_label.pack()
        scale_size                                                    =
Scale(root,orient=HORIZONTAL,length=300,from_=2,to=20,tickinterval=18,reso
lution=1)
        scale_size.pack()
```

```
main_button = Button(root, text='Сформировать таблицу', font='arial
14')
main_button.pack()
main_button.bind("<Button-1>",config_table)
root.resizable(False, False)
root.mainloop()
```

Приложение Б

Дополнительные результаты топологического моделирования в системе автоматизированного проектирования специализированных (заказных) интегральных схем MicroWind

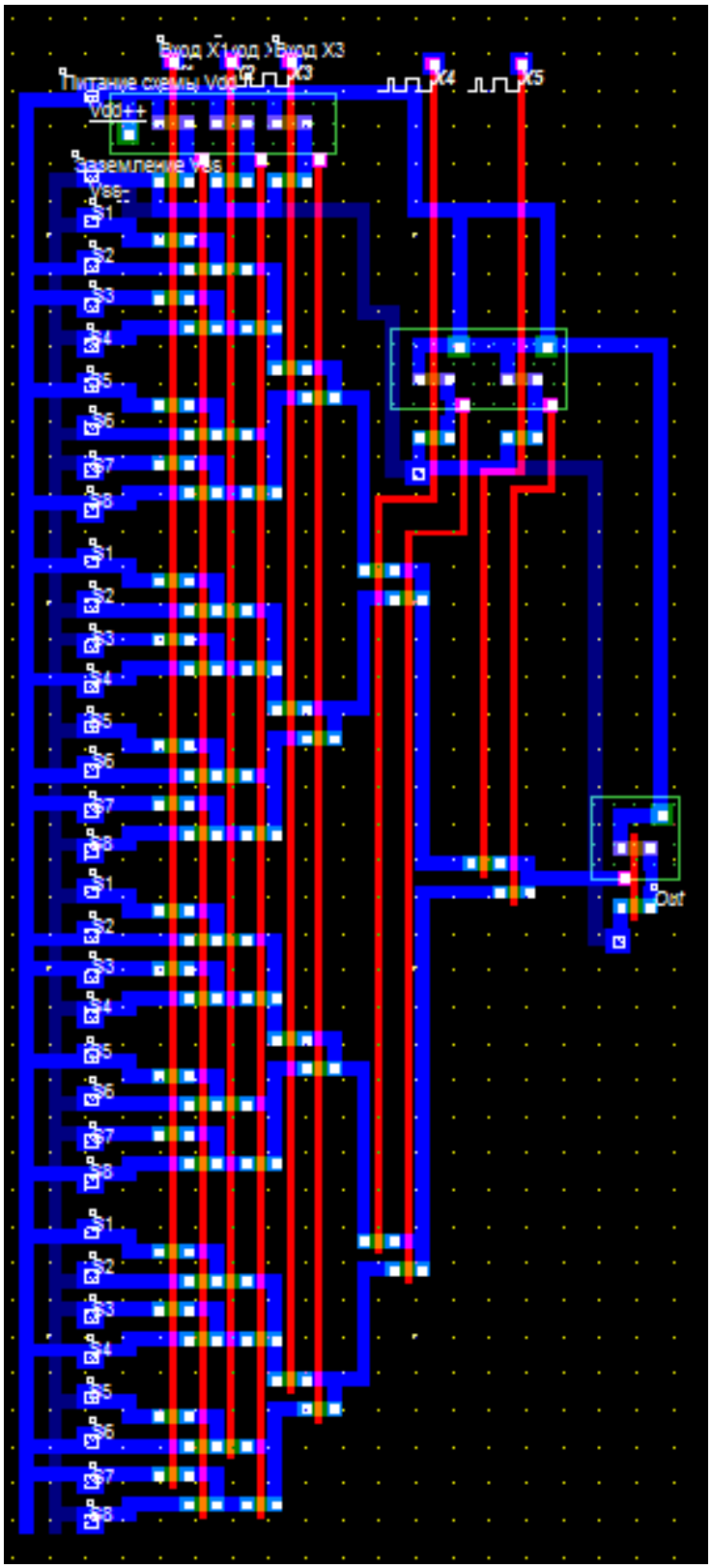


Рисунок ПБ.1. Топология ЛЭ 5 –LUT, 32 нм

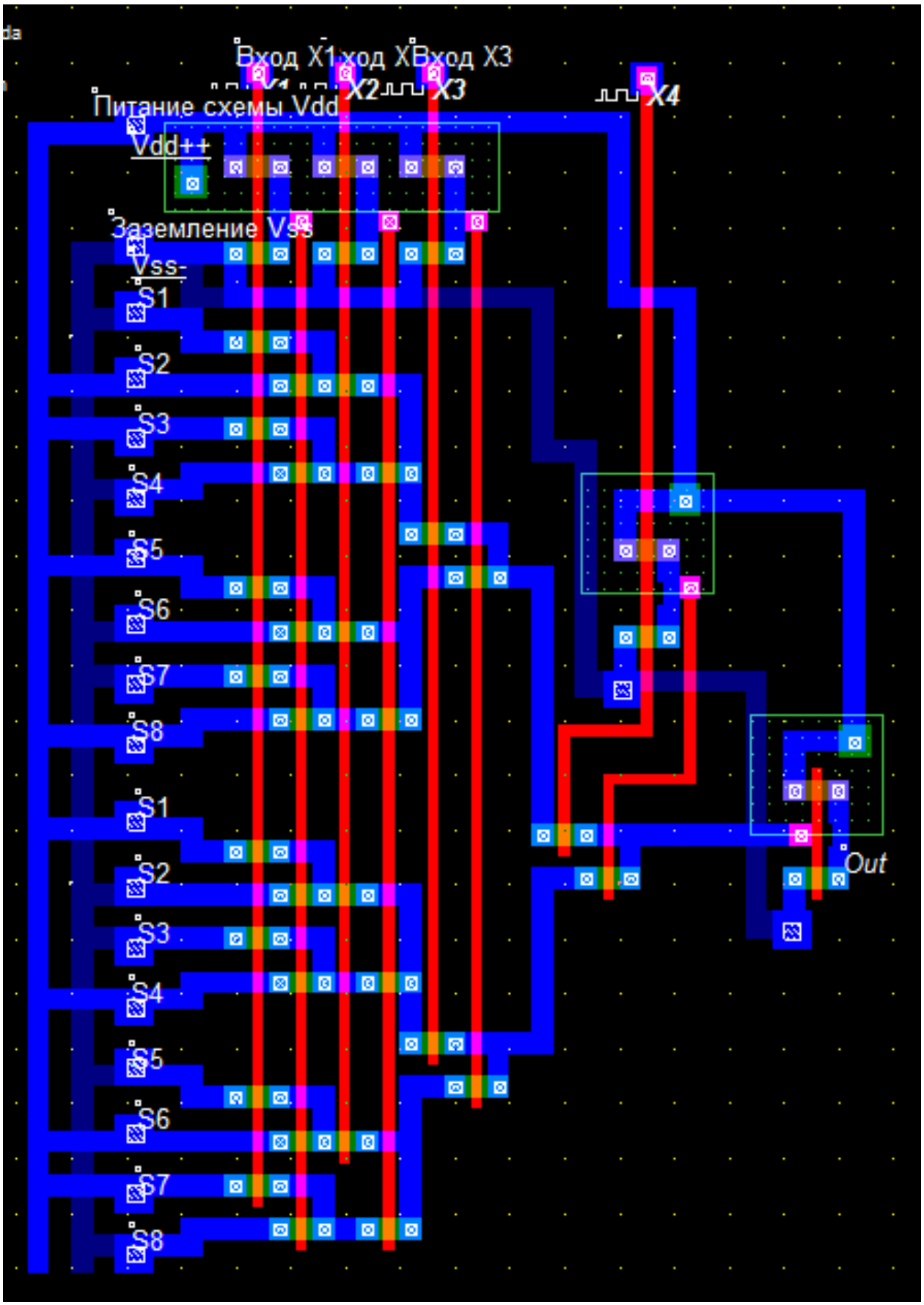


Рисунок ПБ.2. Топология 4-LUT, 32 нм

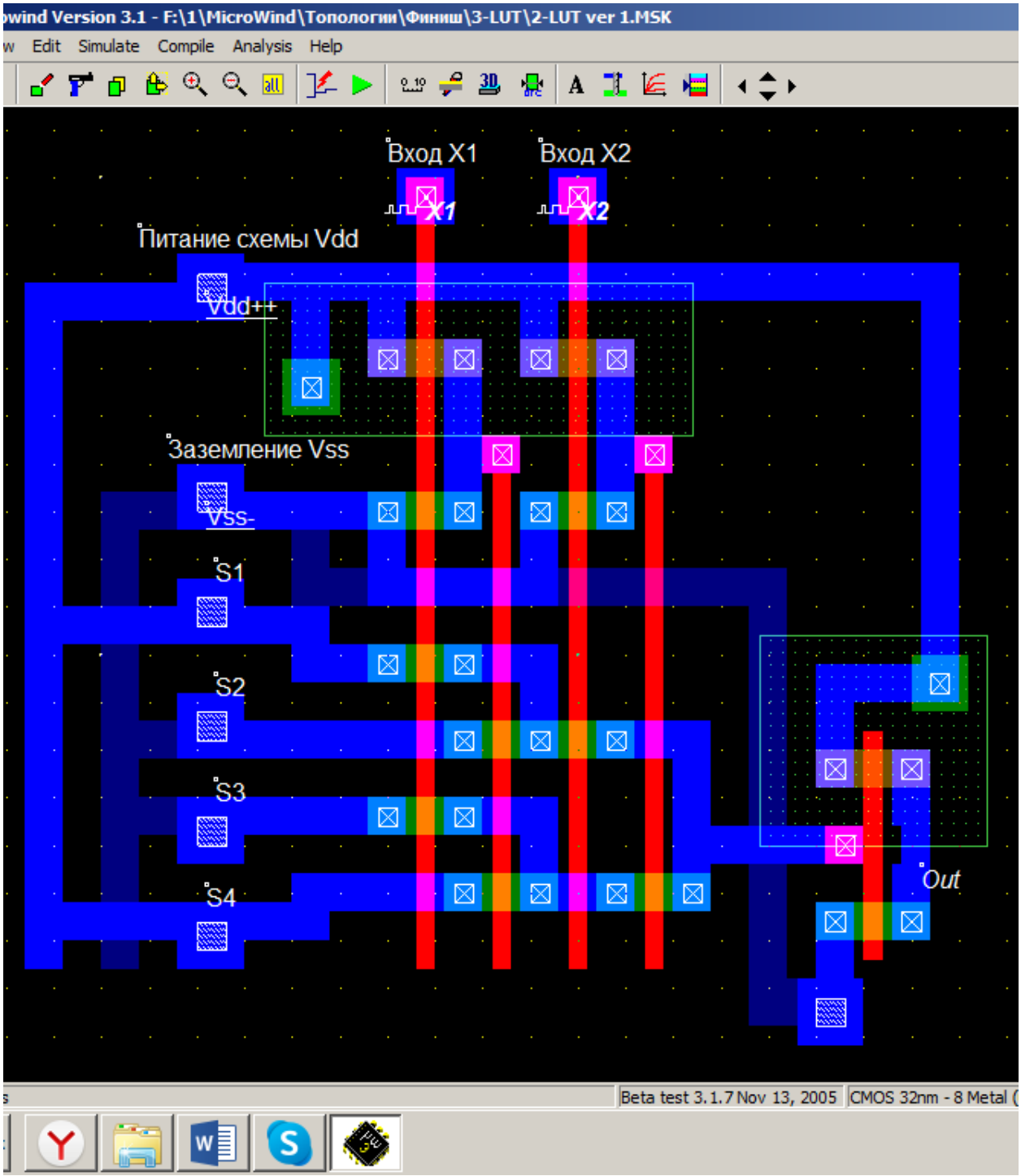


Рисунок ПБ.3. Топология 2-LUT, 32 нм

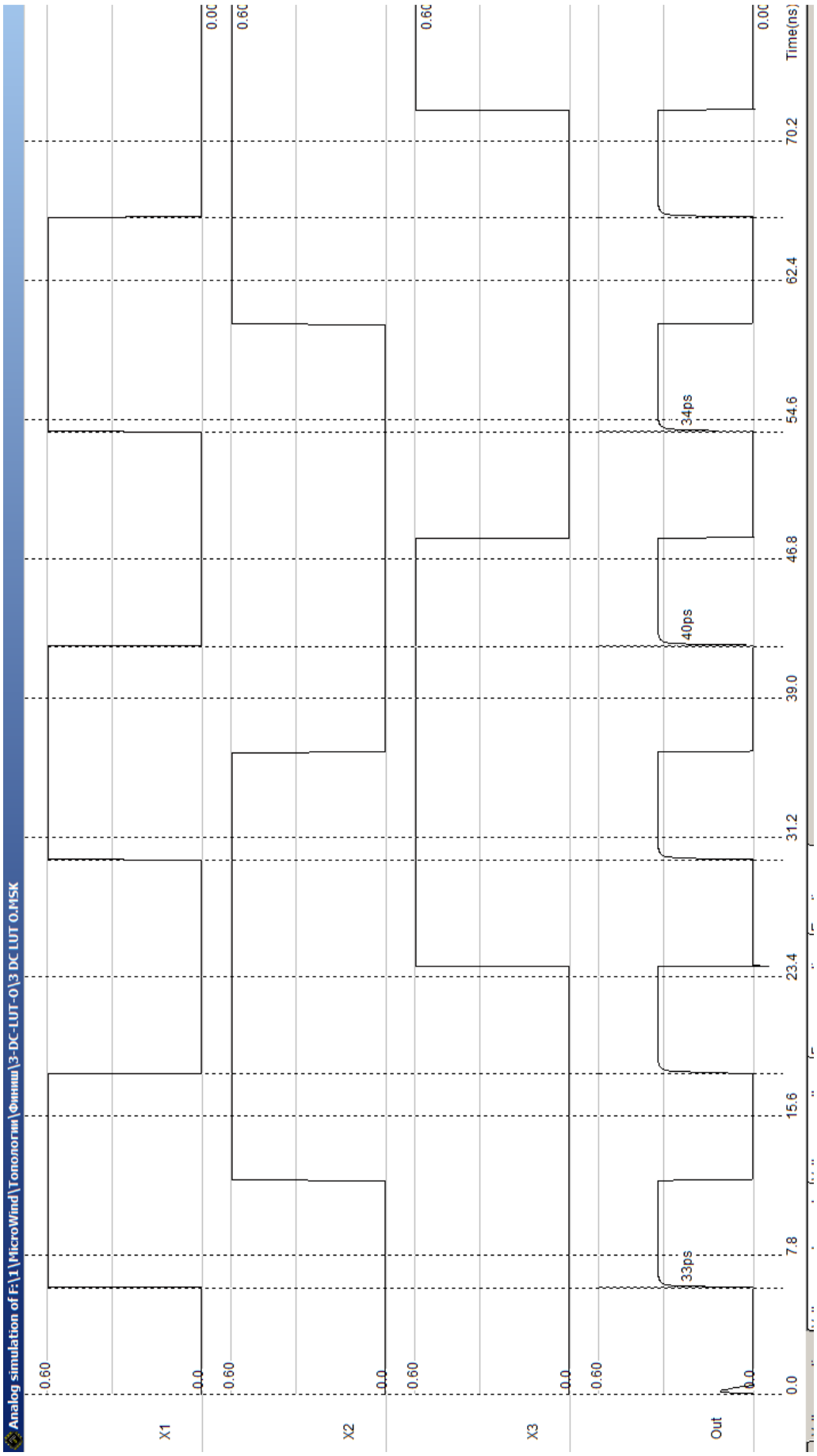


Рисунок ПБ.4 Осциллограмма работы 3-DC-LUT-O

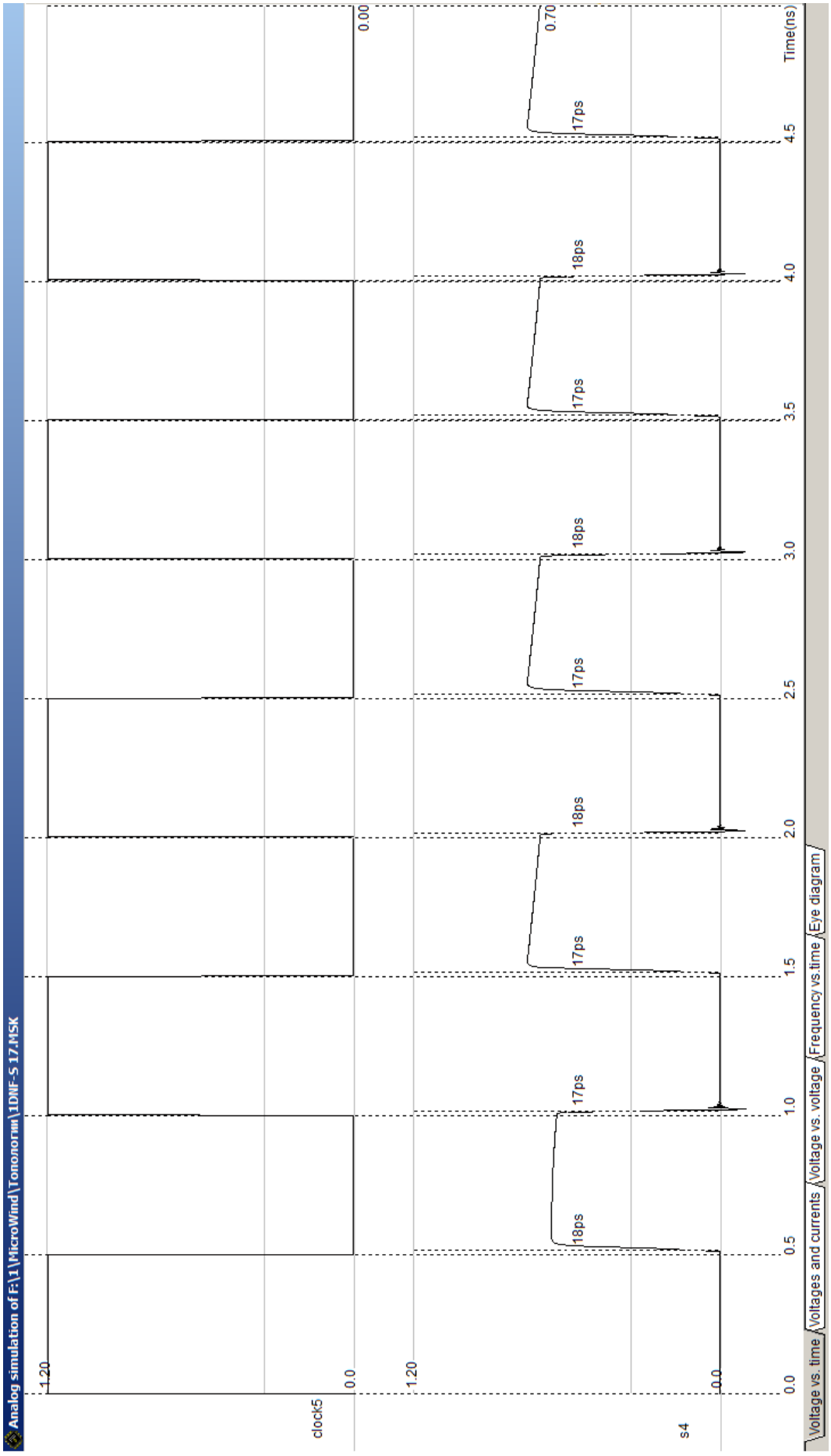


Рисунок ПБ.5 Осциллограмма работы 1-DNF-S

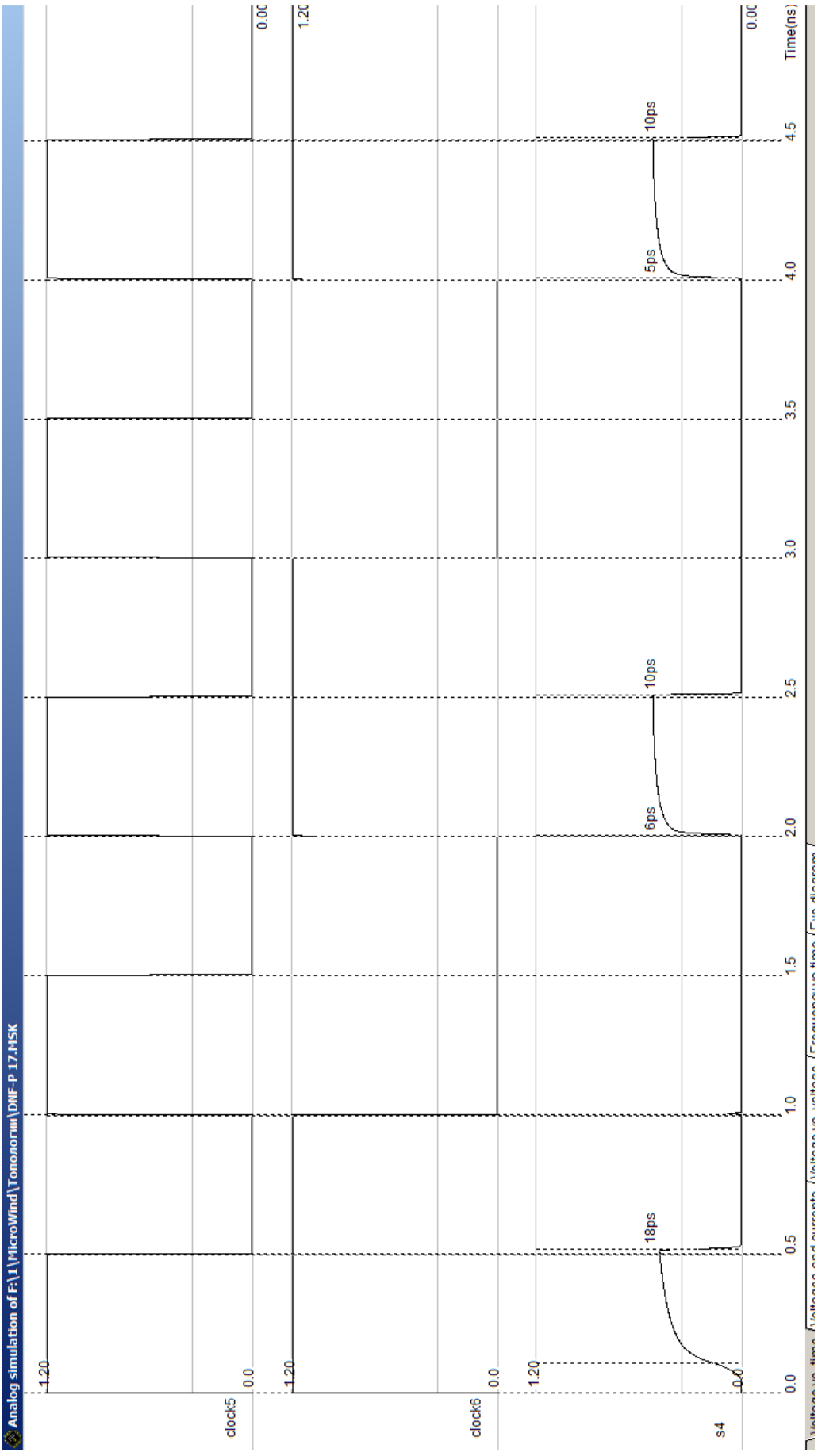


Рисунок ПБ.6 Осциллограмма работы 2-DNF-P

Приложение В

Акты внедрения результатов диссертационного исследования

УТВЕРЖДАЮ
Директор ФИЦ ИУ РАН
_____ И.А. Соколов
«21» 12 2018 г.

Акт о внедрении

результатов диссертационных исследований

Вихорева Руслана Владимировича

Настоящим актом подтверждается, что в научно-исследовательской работе Института проблем информатики «Федерального исследовательского центра» «Информатика и управление» Российской академии наук: «Информационные, управляющие и телекоммуникационные системы» использовались следующие научные результаты, полученные в кандидатской диссертации аспиранта кафедры “Автоматика и Телемеханика” Пермского национального исследовательского политехнического университета Вихорева Руслана Владимировича: раздел 5 части №3 отчета (направления): "Концептуальные и методологические основы создания семейства потоковых самосинхронных процессоров и средств поддержки их проектирования" (№ госрегистрации 117030650038):

1. Усовершенствованный метод реализации систем логических функций в СДНФ на основе предложенного элемента ПЛИС.
2. Усовершенствованный метод реализации систем логических функций в ДНФ на основе предложенного элемента ПЛИС.
3. Оценки сложности реализации систем логических функций на основе разработанных методов и логических элементов.
4. Алгоритм оптимизации набора логических элементов ПЛИС.

Разработанные методы используются при проектировании синхронных и самосинхронных программируемых логических устройств на основе отечественной элементной базы, при этом обеспечивается снижение аппаратных затрат на реализацию систем логических функций от 15-20%.

Зав. отд., научный руководитель направления НИР,
канд. техн. наук Ю.А. Степченков

ПУБЛИЧНОЕ АКЦИОНЕРНОЕ ОБЩЕСТВО «ПЕРМСКАЯ НАУЧНО-ПРОИЗВОДСТВЕННАЯ ПРИБОРОСТРОИТЕЛЬНАЯ КОМПАНИЯ»
Россия, 614990, г. Пермь, ул. 25 Октября, 106
Тел.: +7 (342) 240 05 02, факс: +7 (342) 280 97 19
E-mail: root@ppk.perm.ru, www.ppk.perm.ru

PUBLIC JOINT STOCK "PERM SCIENTIFIC INDUSTRIAL INSTRUMENT-MAKING COMPANY"
Russia, 614990, Perm, 25th October St., 106
Phone: +7 (342) 240 05 02, Fax: +7 (342) 280 97 19
E-mail: root@ppk.perm.ru, www.ppk.perm.ru

УТВЕРЖДАЮ
Генеральный директор ПАО «ПНППК»
_____ А.Г. Андреев
«26» 12 2017 г.

АКТ

внедрения результатов диссертационной работы
Вихорева Руслана Владимировича.

Комиссия в составе:

председатель: исполнительный директор – первый заместитель ГД – главный конструктор В.С. Ермаков

члены комиссии: заместитель ГД по науке – директор НТЦ А.В. Субботин

начальник конструкторского отдела С.А. Пичугин

составили настоящий акт о том, что результаты исследований, изложенные в диссертационной работе Р.В. Вихорева «Логические элементы ПЛИС FPGA для реализации систем функций», представленной на соискание ученой степени кандидата технических наук, используются в ПАО «ПНППК». Диссертационная работа Р.В. Вихорева выполнена в рамках проекта ПАО «ПНППК» по созданию перспективной навигационной системы. Р.В. Вихоревым получены следующие новые научные результаты:

1. Усовершенствованный метод реализации логических функций в совершенной дизъюнктивной нормальной форме на основе соответствующего адаптивного логического элемента
2. Усовершенствованный метод реализации систем логических функций в дизъюнктивной нормальной форме.
3. Получены оценки сложности реализации систем логических функций на основе разработанных логических элементов
4. Алгоритм оптимизации набора логических элементов программируемых логических интегральных схем field-programmable gate array для реализации типовых систем логических функций.

Полученные научные и практические результаты позволяют снизить аппаратные затраты на реализацию систем логических функций в блоке БПИ-002.

Председатель комиссии _____ В.С. Ермаков

Члены комиссии _____ А.В. Субботин

_____ С.А. Пичугин

УТВЕРЖДАЮ
Проректор по учебной работе ПНИПУ
_____ /Лобов Н.В./
«30» 11 2018 г.

АКТ

внедрения в учебный процесс кафедры «Автоматика и телемеханика» ФГБОУ ВО ПНИПУ результатов диссертационной работы Вихорева Руслана Владимировича на тему «Логические элементы ПЛИС FPGA для реализации систем функций»

Комиссия в составе:

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

Члены комиссии: Заневский Э.С.– к.т.н., проф. кафедры «Автоматика и телемеханика»

Фрейман В.И. - к.т.н., проф. кафедры «Автоматика и телемеханика»

составили настоящий акт о том, что результаты диссертационного исследования Вихорева Руслана Владимировича внедрены в учебный процесс кафедры «Автоматика и телемеханика» ФГБОУ ВО Пермский национальный исследовательский политехнический университет в рамках практических занятий профильных дисциплин «Электроника», «Проектирование дискретных устройств», «Схемотехника» для бакалавриата направления 27.03.04 «Управление в технических системах».

Результаты диссертационной работы были использованы в разработанных и внедренных в учебный процесс практических занятиях типа:

1. Разработка усовершенствованного метода реализации логических функций в совершенной дизъюнктивной нормальной форме на основе соответствующего адаптивного логического элемента.
2. Разработка усовершенствованного метода реализации систем логических функций в дизъюнктивной нормальной форме.
3. Оценка сложности реализации систем логических функций на основе разработанных логических элементов.
4. Разработка алгоритма оптимизации набора логических элементов программируемых логических интегральных схем field-programmable gate array для реализации типовых систем логических функций.

Эффект от внедрения результатов диссертационной работы заключается в повышении уровня знаний, умений и владений в соответствии со стандартом ФГОС ВО по направлению 27.03.04 «Управление в технических системах».

Председатель:

д.т.н., проф. зав. кафедрой АТ _____ / Южаков А.А. /

Члены комиссии:

к.т.н., проф. кафедры АТ _____ / Заневский Э.С. /

к.т.н., проф. кафедры АТ _____ / Фрейман В.И. /