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

На правах рукописи

Скорнякова Александра Юрьевна

# КОНФИГУРИРУЕМЫЕ ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ ДЛЯ САМОСИНХРОННЫХ СХЕМ

05.13.05 – Элементы и устройствавычислительной техники и систем управления

ДИССЕРТАЦИЯ
на соискание ученой степени
кандидата технических наук

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

Пермь – 2020

# ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

**Актуальность работы.** Сегодня, несмотря на определенные трудности, продолжает активно развиваться самосинхронная (СС) схемотехника, в том числе и для проектирования энергоэффективной, «зеленой» логики (Green Computing), а также для систем высокой надёжности [1,2], работающих в широком диапазоне температур [3,4]. Однако существующие методы ориентированы в основном на заказную (ASIC, например, конвейерный сопроцессор для микропроцессоров 1890ВМ8Я и 1890ВМ9Я [4]) и полузаказную реализацию базовых матричных кристаллов (БМК или ULA), например, библиотека самосинхронных элементов для БМК серии 5503/5507 [5]. В то же время, для проектирования аппаратуры активно используются программируемые логические интегральные схемы (ПЛИС) типы FPGA, которые позволяют существенно сократить время на разработку устройства и уменьшить стоимость проектирования сложных систем [6]. Базой для проектирования ПЛИС типа FPGA выступают Look Up Table (LUT) [7-13]. Имеется тенденция совмещения СС схемотехники и программируемой логики [14-20], но строго самосинхронные программируемые логические интегральные схемы (ПЛИС) до настоящего времени не созданы. Это усложняет проектирование и сдерживает масштабы использования самосинхронных схем (ССС) по сравнению с синхронной программируемой логикой. В наноэлектронике, где начинают влиять квантовые эффекты и возникают сложности с глобальной синхронизацией блоков, ССС могли бы стать одной из альтернатив для развития перспективных квантовых вычислений [11]. Таким образом, актуальным является проведение исследований в области совмещения самосинхронного подхода и ПЛИС путём создания конфигурируемых самосинхронных логических элементов. **Поэтому объектом исследования** являются самосинхронные схемы и ПЛИС.

Впервые о самосинхронном подходе заговорили в 1959 г. после публикации научных трудов Маллера Д.Е [21]. Большой вклад в развитие ССС внесла группа

советских ученых под руководством Варшавского В.И. Сегодня самосинхронный подход рассмотрен в работах с различных сторон, например, в работах Маллера Д.Е [21], Варшавского В.И [22], Плеханова Л.П [3] и ряда других авторов были рассмотрены основные свойства ССС.

Также были рассмотрены проблемы проектирования ССС в работах таких авторов как: Варшавский В.И [22], Махаровский В.Б [23,24], Яковлев А. [25], Плеханов Л.П [3], Степченков Ю.А и др. [26], Цирлин Б. С. [27], Денисов А.Н. [28,29] Бобков С.Г. [30,31], Сурков А.В. [4], J. Brady [32], A. M. Francis [33], J. Holmes, J. Di, H. A. Mantooth [34], K. M. Fant, S. A. Brantd [35], S. C. Smith [36] и др.

Проблемы надежности ССС рассмотрены в работах Варшавского В.И [37], Мараховского В.Б [38,39], Степченкова Ю.А [40], Дьяченко Ю.Г [41]. Позднее была затронута проблема создания отказоустойчивых ССС, которая рассмотрена в работах Каменских А.Н, Степченкова Ю.А и Тюрина С.Ф. [1,2]. Однако эти исследования в основном ориентированы на проектирование ССС в специализированных базисах, таких как: БМК или функционально-полные толерантные элементы (ФПТЭ). В работе Тюрина С.Ф [42] поднимался вопрос о недостаточности исследования направления синтеза ССС на основе универсальных настраиваемых блоков, но дальнейшего решения вопроса выполнено не было.

Таким образом, необходима детальная проработка и оценка универсальных настраиваемых моделей. **Поэтому предметом исследования** являются методы синтеза конфигурируемых логических элементов в ССС.

**Целью исследования** является решение научно-технической задачи разработки методов синтеза конфигурируемых логических элементов для ССС.

Для достижения поставленной цели в рамках диссертационного исследования следует решить следующие частные **задачи:**

1. Разработать метод реализации конфигурируемого самосинхронного генератора логических функций на основе стандартных логических КМОП элементов (ЛЭ);

2. Разработать метод реализации конфигурируемого самосинхронного генератора логических функций на основе Look Up Table (LUT), используемого в ПЛИС;

3. Разработать метод реализации конфигурируемого самосинхронного генератора систем логических функций, заданных в СДНФ, на основе дешифратора DC LUT и блоков дизъюнкций;

4. Разработать метод реализации конфигурируемого самосинхронного генератора систем логических функций, заданных в ДНФ, на основе блоков конъюнкций и дизъюнкций;

5. Получить оценки сложности в количестве транзисторов, площади, задержки, потребляемой мощности для реализации систем логических функций на основе разработанных логических элементов;

6. Разработать алгоритм выбора оптимального набора конфигурируемых логических элементов для реализации типовых систем логических функций.

Под конфигурированием в данной работе подразумевается настройка элемента константами или записью значений в ячейки памяти SRAM.

В диссертационном исследование используются **методы и средства** схемотехнического моделирования, топологического моделирования, анализа и синтеза схем, описание и анализ моделей Маллера. Применяемые методы и средства основаны на положениях дискретной математики, теории булевых функций и автоматов, комбинаторики, теории надежности, принципах КМОП схемотехники, программирования.

**Научная новизна** диссертационного исследования заключается следующем:

- Разработан новый метод реализации конфигурируемого самосинхронного генератора функций (ГФ) на основе библиотечного базиса 2И-

2ИЛИ-НЕ, отличающийся тем, что он адаптирован к условиям работы в ССС. Для этого применяется парафазная дисциплина кодирования сигнала и используется фаза спейсера, причем для согласованности работы блоков схемы при количестве переменных n>1 спейсер каждого слоя блоков изменяется;

- Разработан новый метод реализации конфигурируемого самосинхронного генератора логических функций в ССС по принципу LUT (Look Up Table), используемому в программируемых логических интегральных схемах (ПЛИС типа FPGA), отличающийся тем, что используется дополнительная ветвь дерева передающих транзисторов, активируемая в фазе спейсера, а двойственный канал универсального логического элемента настраивается инверсными константами;
- Разработан новый метод реализации конфигурируемого самосинхронного генератора систем логических функций в СДНФ на основе DC LUT FPGA и блока дизъюнкций, отличающийся тем, что он адаптирован к работе в ССС;
- Разработан новый метод реализации конфигурируемого самосинхронного генератора систем логических функций в ДНФ на основе блоков конъюнкций и дизъюнкций, отличающийся тем, что он адаптирован к работе в ССС;
- Разработан алгоритм выбора оптимального набора конфигурируемых логических элементов для реализации типовых систем логических функций, отличающийся тем, что он реализует многокритериальную оптимизацию путем нахождения Парето-оптимальных вариантов.

**Основные положения, выносимые на защиту:**

- Существующие методы синтеза ССС ориентированы в основном на заказную (ASIC) и полузаказную реализацию (БМК или ULA). Известные примеры использования асинхронного подхода в ПЛИС (FPGA) не являются строго самосинхронными. Все это сдерживает развитие цифровых технологий в

области наноэлектроники и квантовых вычислений (п.2 паспорта специальности).

- Предлагаемые новые методы синтеза конфигурируемых логических элементов для самосинхронных ПЛИС, разработанные элементы для реализации логических функций в СДНФ и в ДНФ, проведенное схемотехническое, топологическое моделирование и проверка полумодулярности с использованием модели Маллера подтверждают возможность совмещения строго самосинхронной обработки информации и ПЛИС на уровне логического элемента (п.3 паспорта специальности).
- Полученные оценки сложности, площади кристалла, потребляемой мощности и задержки позволяют с использованием предложенного алгоритма получить оптимальные наборы конфигурируемых логических элементов для ССС и заданных систем логических функций и рекомендуются при разработке самосинхронных ПЛИС (п.2 паспорта специальности).
- Для реализации систем логических функций, зависящих от одних и тех же аргументов при среднем числе входных переменных, целесообразно применение предлагаемого элемента DC LUT ST. Наибольший эффект достигается при реализации системы функций унитарного декодирования набора переменных. При большом числе переменных (больше 8) в связи с экспоненциальным ростом сложности элемент LUT ST нереализуем, поэтому рекомендуется элемент ДНФ LUT ST (п.4 паспорта специальности).

**Достоверность исследования** подтверждаются соответствием аналитических выводов и результатов моделирования, полученных в системах схемотехнического и топологического проектирования, проверке в подсистеме TRANAL, а также использованием апробированного математического аппарата булевой алгебры, теории автоматов, теории надежности, схемотехники и программирования. Полученные результаты не противоречат теоретическим и практическим положениям, известным из научных публикаций отечественных и зарубежных исследователей в рассматриваемой предметной области.

**Практическая значимость диссертационной работы** диссертационной работы заключается в том, что предложенные методы реализации конфигурируемых логических элементов для самосинхронных схем используются в перспективных разработках элементов Института проблем информатики Федерального исследовательского центра "Информатика и управление" Российской академии наук.

Полученные научные и практические результаты используются в учебном процессе кафедры «Автоматика и телемеханика» Пермского национального исследовательского политехнического университета.

**Апробация работы.** Основные теоретические и практические результаты работы докладывались на научно-технических конференциях: Микроэлектроника и информатика-2015, г. Зеленоград; ЛЭТИ - IEEE North West Russia Section Young Researchers in Electrical and Electronic Engineering Conference, (EIConRusNW-2016, EIConRusNW-2018, EIConRusNW-2020) г. Санкт-Петербург; Элементная база отечественной радиоэлектроники: импортозамещение и применение-2015, г. Нижний Новгород; Авиация и космонавтика – 2015, г. Москва; всероссийская школа-конференция молодых ученых «Управление большими системами УБС» (УБС-2017) г. Пермь; всероссийская научно-техническая конференция «Автоматизированные системы управления и информационные технологии» (АСУИТ-2020) г. Пермь; международная выставка SEMIEXPO RUSSIA 2017, г. Москва и в других международных и региональных конференциях.

**Публикации.** Основные результаты диссертационной работы опубликованы в 18 научных работах, из них пять статей в изданиях, включенных в перечень ВАК, три в изданиях, индексируемых в Scopus, три патента на изобретение и одно свидетельство о регистрации программы для ЭВМ.

**Структура и объем диссертационной работы.** Данная работа состоит из введения, четырех глав, заключения, списка терминов и сокращений, списка использованной литературы, который включает в себя 104 источника и 6

приложений. Основная часть работы изложена на 150 страницах, содержит 102 рисунка и 21 таблицу.

Автор выражает благодарность коллективу кафедры «Автоматика и телемеханика» ПНИПУ и лично Тюрину Сергею Феофентовичу, Степченкову Юрию Афанасьевичу, Дъяченко Юрию Георгиевичу.

# ГЛАВА 1. АНАЛИЗ СУЩЕСТВУЮЩИХ МЕТОДОВ ПОСТРОЕНИЯ САМОСИНХРОННЫХ СХЕМ И СРЕДСТВ РЕАЛИЗАЦИИ ЛОГИЧЕСКИХ ФУНКЦИЙ. ПОСТАНОВКА ЗАДАЧИ ИССЛЕДОВАНИЯ

## 1.1 Принцип самосинхронности

Самосинхронный подход для проектирования схем был основан американским профессором прикладной математики Дэвидом Маллером, но в его работах он назывался асинхронным. Опубликованная в 1959 году статья "A Theory of Asynchronous Circuits" [21], стала одной из основ для развития асинхронной схемотехники. В работах Маллера Д.Е. явно доказаны и выявлены [43] существующие особенности самосинхронных схем, такие как независимость работы от задержек элементов. Принято, что задержки существенны только на выходах элементов, а задержками межсоединений после разветвлений можно пренебречь. Это свойство самосинхронных схем нашло практическое применение, поскольку реализация данных схем влечет за собой отсутствие состязаний, а также широкий диапазон внешних факторов безошибочной работы.

Следующим важнейшим этапом развития самосинхронных схем являются работы Варшавского В.И. и его сотрудников. В результате их работы были созданы конкретные схемы, названные самосинхронными [22]. Также в работах Варшавского В.И. впервые было обращено внимание на еще одно важное свойство самосинхронных схем - самопроверяемость. Свойство означает, что при возникновении константных неисправностей типа залипания (КНЗ), одиночных или кратных, схема останавливается, что гарантирует достоверность работы схемы. При наличии такого свойства есть возможность самодиагностики и саморемонта, а это значит, что есть возможность построения резервированных, надежных схем.

В начале XXI века группа перестала существовать как целое, один из известных исследователей группы - Алекс Яковлев, сегодня работает в университете Нью-Касла. Он развивает направление «Energy-modulated

computing» (Энергетически модулированные вычисления) [44,45], основанное на ССС.

В настоящее время единственной группой специалистов в России, которая занимается развитием самосинхронных схем [46], является группа в составе ИПИ РАН, ученые из двух пермских институтов с 1995г. являются их научными партнерами. Основная их цель – проектирование конкретных ССС. Сегодня ими созданы самосинхронный микропроцессор [40], самосинхронное устройство умножения-сложения [47], самосинхронное арифметико-логическое устройство [48], которые превосходят свои синхронные аналоги по ряду характеристик.

В [2] было дано определение самосинхронных схем. Самосинхронной схемой называется такая схема, которую можно описать диаграммой переходов, и эта диаграмма должна быть полумодулярной.

Диаграмма переходов полумодулярна, если ее возбужденные переменные не могут стать устойчивыми без изменения своего значения. [2, 3, 21]. На рисунке 1.1 показана полумодулярная диаграмма переходов.

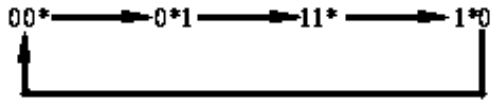



Рисунок 1.1 – Полумодулярная диаграмма переходов.

На рисунке 1.1 видно, что каждое из возбужденных состояний «правильно» становится устойчивыми. Если состояние, помеченное *, после перехода имеет такое же значение, но без *, то такой переход называется конфликтным [3] и означает, что такая диаграмма переходов не полумодулярна.

Таким образом, чтобы определить является ли схема самосинхронной нужно выполнить оценку диаграммы переходов данной схемы на полумодулярность, если она полумодулярна, то схема самосинхронная.

В [2, 3, 21] была предложена классификация ССС по распространению сигнала в схеме, которая включает три группы:

1. Speed-independent (SI) – схемы, независимые от задержек элементов или скорости распространения сигнала в схеме. В таких схемах берутся какие-то

средние значения задержек элементов и межсоединений, задержка межсоединений не превышает минимальной задержки элемента.

2. Delay-insensitive (DI) – схемы, нечувствительные к задержкам. Такие схемы работают по реальным задержкам элементов и межсоединений;

3. Quasi-delay-insensitive – схемы, квази-нечувствительные к задержкам. Такие схемы не зависят от задержек элементов и межсоединений.

Если классифицировать ССС по принципу проектирования, то можно выделить два типа:

1. Квазисамосинхронные - самым распространенным методом построения таких схем служит метод модели максимальной задержки. Для таких схем характерно использование модели комбинационной схемы. Формирование сигнала окончания переходного процесса позволяет устранить технологический разброс и ошибки оценки времени распространения сигнала.

2. Строго самосинхронные схемы - суть работы таких схем заключается в том, что определяется момент окончания переходного процесса в каждом блоке схемы.

В данной исследовательской работе речь идет о строго самосинхронных схемах.

Для проектирования ССС наиболее оптимальным является применение метода парафазного кодирования со спейсером. Парафазное представление сигналов означает, что каждый сигнал передается по двум каналам (проводникам). Таким образом, этот сигнал может принимать три значения "DATA0" (канал 1в логической 1, канал 2 в логическом 0), "DATA1" (канал 1 в логическом 0, канал 2 в логической 1) и NULL (канал 1=0, канал 2=0).

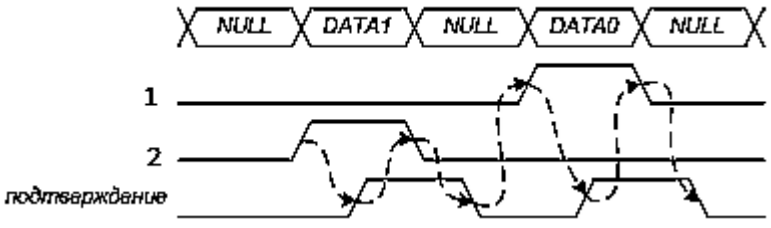



Рисунок 1.2 - Парафазный протокол передачи данных.

Как видно из рисунка 1.2, состояние NULL используется для разделения DATA1 и DATA0 во времени.

В работе ССС выделяют две фазы – рабочую и спейсерную фазы. В рабочей фазе происходят функциональные преобразования, для которых схема предназначена, а в спейсерной фазе происходит подготовка к следующим преобразованиям. Обычно в качестве спейсера используют легко различимые наборы сигналов: либо все 0, либо все 1. Для определения того, в какой фазе находится устройство в данный момент времени, используется индикатор окончания переходного процесса.

К основным преимуществам ССС относятся [2,3,40,43,49] следующие свойства:

- Отсутствие состязаний (гонок) сигналов;
- Устойчивое функционирование в широком диапазоне внешних условий, которое ограничено только физической работоспособностью транзисторов;
- Сниженное энергопотребление за счет способности работать при пониженном напряжении питания;
- Увеличение срока работы схемы за счет нечувствительности к разбросу и изменению параметров, вызванных старением элементов;
- Полностью самопроверяемые при возникновении константных неисправностей (КНЗ);
- Отказоустойчивые за счет относительно простого построения надежных схем.

Совокупностью перечисленных свойства обладают только ССС, поэтому актуально развитие методов и средств унифицированной реализации систем логических функций.

## 1.2 Анализ методов и средств синтеза самосинхронных схем

Развитие самосинхронных схем активно проявляется в мировой науке [50-54]. Но анализ самосинхронных схем до сих пор остается сложной задачей. Но

несмотря на это, количество подходов для синтеза ССС увеличивается, так, например, в работах Маллера и группы во главе с В. И. Варшавским применялась только одна группа подходов, в работе [3,43] они названы событийными.

Суть событийного подхода заключается в представлении схемы в графическом виде, это могут быть, например, диаграммы переходов, сети Петри или сигнальные графы. Кроме того, схемы должны иметь замкнутое описание [3,43], что на практике порождает трудность построения таких схем. Используя событийный подход для анализа схемы можно выявить выполнение двух основных свойств ССС. Это независимость работы схемы от задержек элементов и самопроверяемость [3,43]. Они выполняются автоматически после проверки схемы на полумодулярность. Следовательно, полумодулярность является косвенным признаком наличия этих свойств. Отмечается, что такой анализ схемы дает мало информации об анализируемой схеме. Также существует сложность использования событийного подхода на практике и невозможность использования иерархического анализа схемы. Поэтому в [3,43] предложен функциональный подход, который позволяет решить ряд недостатков событийного подхода.

Суть функционального подхода заключается в представлении схемы в виде логических функций, используя самосинхронное кодирование, преимущественно парафазное [3,43]. Схемы, в отличии от событийного подхода, имеют разомкнутый вид. Кроме этого схемы должны обладать двухфазной дисциплиной работы, т.е. чередованием рабочей фазы и фазы гашения (спейсера). Но, несмотря на разомкнутость схемы, работа должна проходить с замкнутой общей обратной связью для обеспечения правильного чередования фаз [3,43]. Также при данном подходе необходимо учитывать начальные состояния схемы, которые могут повлиять на свойства самосинхронной схемы. При использовании функционального подхода тоже проверяются два основных свойства, но в отличии от событийного подхода каждое свойство проверяется в отдельности. При анализе таким методом получают больше сведений о схеме,

например, где произошел конфликтный переход в случае нарушения свойства, или какое значение фазы гашения и так далее. Однако в [3,55] отмечается проблема анализа больших самосинхронных схем на нижнем уровне, где описание должно учитывать не только работу элементов, но и все возможные реальные состояния, и переходы между ними [3,55]. Как результат, получается описание состояний большой размерности, которое требует несоразмерного использования вычислительных ресурсов, поэтому в [3,55,56] предлагается использование структурных методов в иерархическом подходе.

Суть структурных методов заключается в отказе от вычисления логических функций, переходу к анализу взаимосвязей элементов между собой и подключенных к ним сигналов. Затем выделяют комбинационную часть и все бистабильные ячейки. Одна бистабильная ячейка состоит из 2х элементов с перекрестными связями [3,55]. После комбинационную часть и бистабильные ячейки оформляют как фрагмент, который в последующем проверяют на выполнение двух основных свойств ССС. Использование структурных методов при анализе больших ССС позволяет существенно сократить использование вычислительных ресурсов, что доказано в работе [3,55].

Одним из средств анализа самосинхронных схем является САПР Forcage, в которой используется функциональный подход. В данной САПР имеется подсистема анализа Tranal, которая позволяет определить наличие у схемы свойства полумодулярности.

Подсистема Tranal предназначена для проверки корректности самосинхронной схемы в независимости от задержек элементов. За основу описания схемы в подсистеме принята модель Маллера. Схема работоспособна, если происходит переход из одного состояния в другое. Наглядным представлением переходов состояний является представление их в виде графов переходов.

В подсистеме Tranal происходит преобразование описания схемы в виде системы уравнений во внутреннее представление графа переходов.

Результатом анализа является констатация факта полумодулярности схемы. По факту окончания процесса моделирования на экран выводятся наиболее существенные характеристики схемы, такие как: число проанализированных состояний, число слоев, их ширина и пр. Пользователю предоставляется возможность управления процедурой моделирования. Он может задать нужные параметры для схемы, которые влияют на эффективность оценки схемы. На рисунке 1.3 представлена блок-схема процедуры управления ходом анализа схемы пользователем.

На рисунке 1.3 цифрами описаны следующие блоки:

1. Задание верхней границы числа состояний;
2. Априорные параметры схемы задаются? (условие выполняется автоматически, без участия пользователя);
3. Анализ с сообщением о результатах (выполняется автоматически, без участия пользователя);
4. Есть ли нарушение корректности схемы? (условие выполняется автоматически, без участия пользователя);
5. Число сгенерированных слоев превышает заданное длиной рабочего цикла? (условие выполняется автоматически, без участия пользователя);
6. Нужен ли анализ с новыми начальными условиями? (условие выполняется автоматически, без участия пользователя);
7. Задание признаков цикличной и живой схемы;
8. Задается ли длина рабочего участка?
9. Задание границ рабочего цикла;
10. Ввод длины начального участка;
11. Нужен ли запуск с новыми управляющими параметрами? (условие выполняется автоматически, без участия пользователя);
12. Нужна ли смена контрольного слоя текущим? (условие выполняется автоматически, без участия пользователя);

13. Увеличение максимальной длины рабочего цикла (выполняется автоматически, без участия пользователя);

14. Устранение ошибок схемы;

15. Ввод начального состояния.

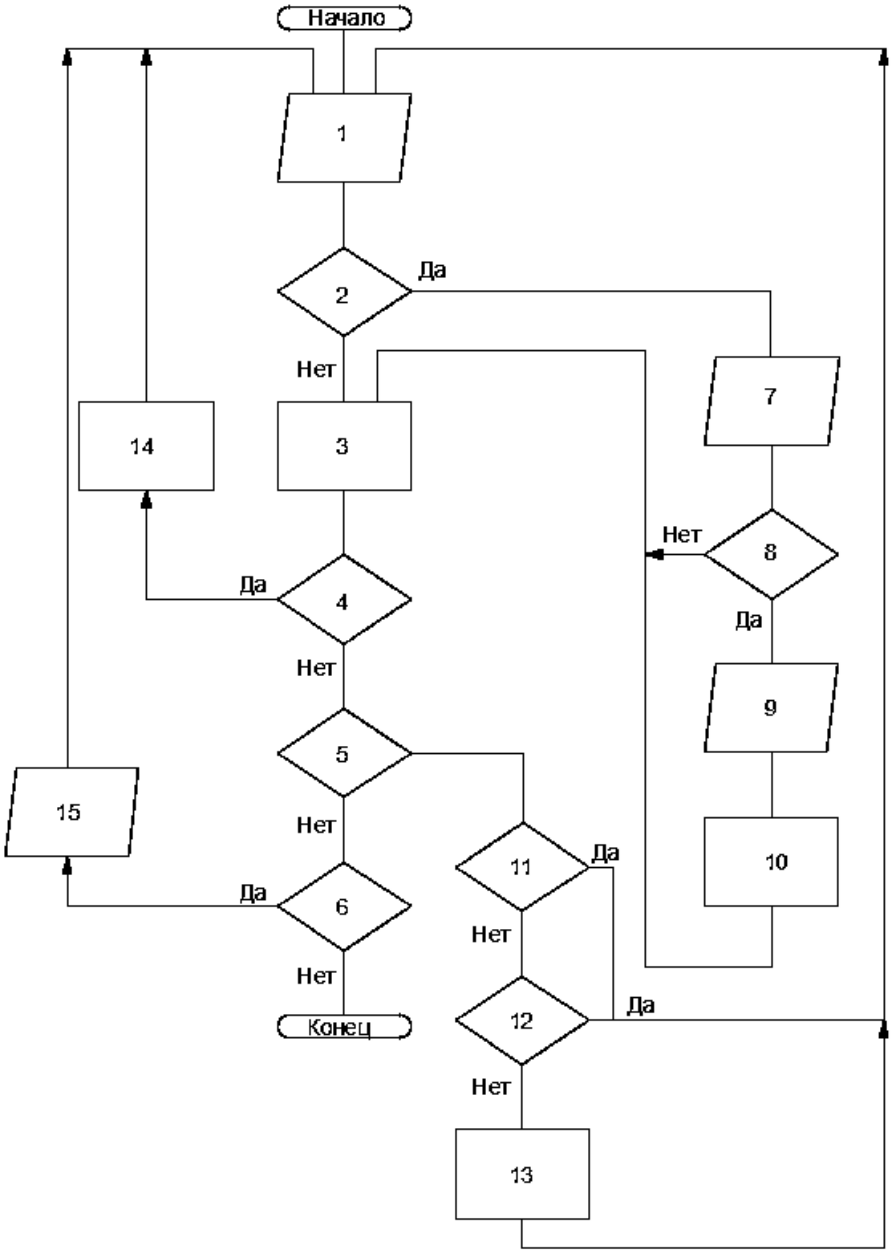



Рисунок 1.3 Процедура управлением ходом анализа.

Таким образом, видно, что в процессе моделирования пользователь сам выбирает параметры, на основании которых будет выполняться моделирование.

В САПР «Forcage» языками для программирования являются: TurboPascal и используются специальные символы, например:

- ; Разделяет уравнения
- ^ Инверсии (после названия сигнала или перед всем правая часть уравнения)
- * Конъюнкция
- | Дизъюнкции
- $ Отделяет уравнений из начального состояния.

На основании данной САПР будет проводиться анализ разработанных моделей элементов для подтверждения их принадлежности к классу ССС.

Кроме определения свойства полумодулярности, требуется проверить функциональность разработанных элементов, для этого можно использовать различные САПР. Например, САПР Ковчег – она отличается от других тем, что имеет встроенную библиотеку элементов для проектирования самосинхронных полузаказных БМК микросхем серии 5503/5507 [5]. Данная библиотека содержит более 200 элементов. Все элементы разделены на два больших класса:

1. Базовые элементы (логические элементы, мультиплексоры, гистерезисные триггеры, счетчики и др.)
2. Макроэлементы.

Логические элементы библиотеки реализуют четыре простые функции: конъюнкцию (И), дизъюнкцию (ИЛИ), штрих Шеффера (И-НЕ) и стрелку Пирса (ИЛИ-НЕ), а также сложные функции, которые состоят из нескольких простых: конъюнкцию, дизъюнкцию и инверсию. Мультиплексоры реализуют функцию передачи входных сигналов, в зависимости от управляющего кода, на выход. Преобразователи из бифазного сигнала преобразуют сигнал в парафазный. Так же присутствуют элементы, выполняющие логическую функцию сравнения двух двоичных чисел – компараторы. Гистерезисные триггеры выполняют роль индикаторов окончания перехода из рабочей фазы в фазу спейсера. Счетчики позволяют построить самосинхронные счетчики с единичным или нулевым спейсером [5].

Макроэлементы – это функционально сложные элементы, представляющие собой комбинацию базовых элементов, реализующий единую сложную функцию. Каждый макроэлемент содержит индикатор окончания переходного процесса. Библиотека САПР Ковчег включается в себя две группы макроэлементов. В первую группу входят триггеры, во вторую сдвиговые регистры [5].

Кроме самосинхронной библиотеки в САПР Ковчег имеются и синхронные. Данная САПР предназначена для проектирования КМОП БИС на основе базовых матричных кристаллов серий 5503, 5507, 5521, 5528 и 5529 [57]. САПР Ковчег включает в себя следующие подсистемы:

- Функционально-логическое моделирование;
- Размещение ячеек на поле БМК;
- Синтез топологии;
- Специализированный топологический редактор;
- Верификация;
- Расчет параметров топологии;
- Аттестация проекта.

Каждая из подсистем позволяет получить различные характеристики схем, например, временные диаграммы, оценить устойчивость схемы к воздействиям внешней среды, выполнить анализ влияния топологических параметров схемы, получить расчет задержек и другое. На сайте разработчиков [57] можно скачать версию, в которой полный цикл проектирования может быть осуществлен для специализированной библиотеки серии 5503, которая не включает в свой состав самосинхронные элементы. Возможность проектировать новые элементы на транзисторном уровне недоступна без специального разрешения от разработчиков, поэтому для проектирования новых элементов использованы доступные программные продукты.

Для проектирования новых элементов на транзисторном уровне предлагается использовать САПР NI Multisim [58]. Данная САПР применяется

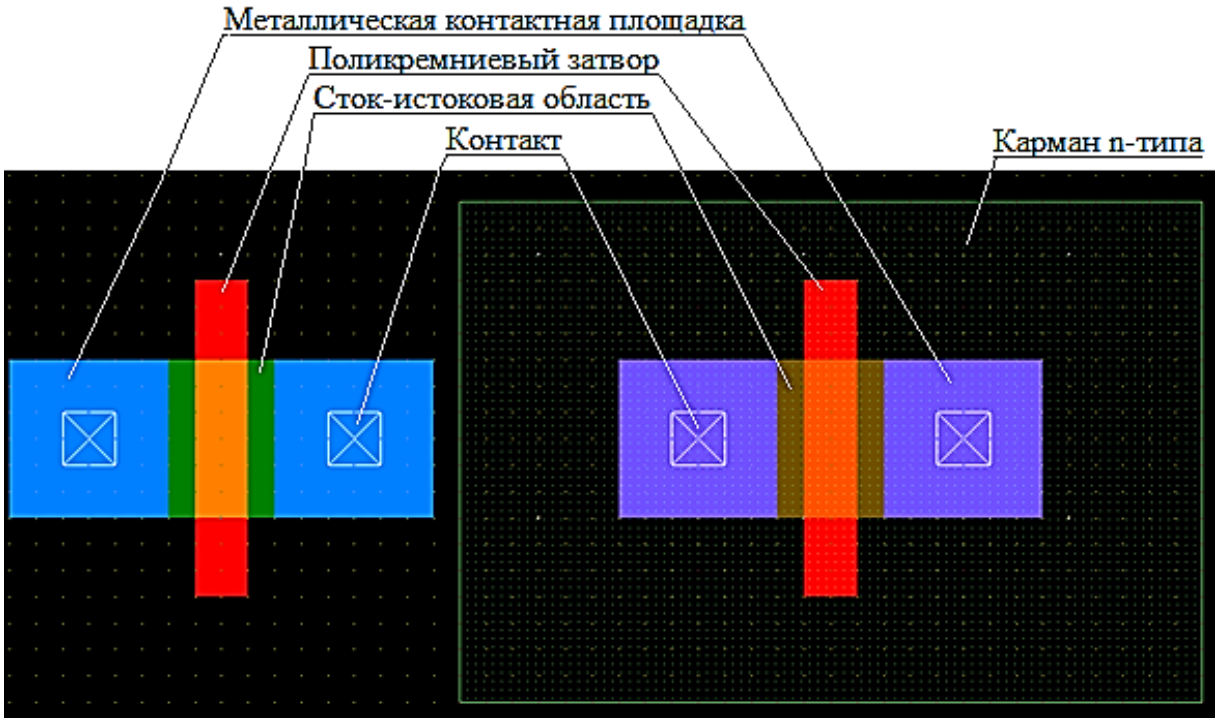
для моделирования аналоговой, цифровой и силовой электроники. Позволяет проектировать схемы на готовых компонентах реальных или виртуальных, а также на транзисторах КМОП или ТТЛ, причем база компонентов включает в себя более 1200 SPICE-моделей от ведущих производителей. Кроме компонентов в САПР имеются виртуальные приборы, которые позволяют произвести измерения, например, тока, напряжения и др. С помощью NI Multisim можно выполнить различный анализ схем: анализ цепей по току, анализ переходных процессов и др. В NI Multisim нет возможности оценить влияние топологических параметров схемы, поэтому для этих целей используется САПР MicroWind.

САПР MicroWind позволяет проектировать схемы на топологическом уровне. Выбор компонентов можно осуществлять двумя способами: генерировать виртуальные модели элементов или выполнить построение элементов вручную с учетом норм проектирования. Например, для построения транзисторов необходимо выполнить четыре или пять этапов [59] в зависимости от типа:

1. Формируем поликремневый затвор;
2. Формируем сток-истоковые области;
3. Формируем металлические контактные площадки;
4. Формируем контакты;
5. Формируем карман проводимостью n-типа для размещения транзистора p-типа.

После выполнения данных этапов необходимо проверить правильность построения, т.к. в программе используется специальная единица λ, которая зависит от выбранной технологии (нанометрового диапазона).

На рисунке 1.4 показаны топологии транзисторов для норм проектирования 90нм.




а) б)

Рисунок 1.4 Внешний вид транзисторов: а) N-тип, б) P-тип.

В результате проектирования можно оценить площадь, занимаемую на кристалле, получить временные диаграммы и другое.

### 1.3 Анализ существующих асинхронных ПЛИС

В [14-16] описываются возможности создания специальных асинхронных ПЛИС. В работах [17,18] описываются подходы, позволяющие использовать стандартные ПЛИС. Трудности заключаются в реализации так называемых арбитров, позволяющих устранить негативные последствия состязаний (гонок). Схема такого арбитра описывается в [19], где описывается стандартный реконфигурируемый логический блок ПЛИС FPGA (Field-Programmable Gate Array) – рисунок 1.3.1, на три переменных, состоящий из трех мультиплексоров (так написано, а фактически это коммутаторы связей [7]) для трёх входов (слева), функциональный блок FU, 3 коммутатора для выходов и тристабильные драйверы (справа).

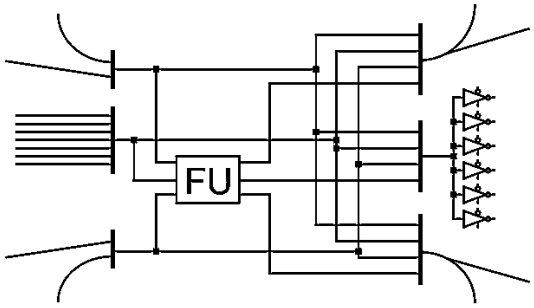



Рисунок 1.5 - Реконфигурируемый логический блок.

Сам функциональный блок FU, содержащий генератор функций - LUT (Look Up Table)] предлагается доработать с целью адаптации к реализации асинхронных схем – рисунок 1.6.

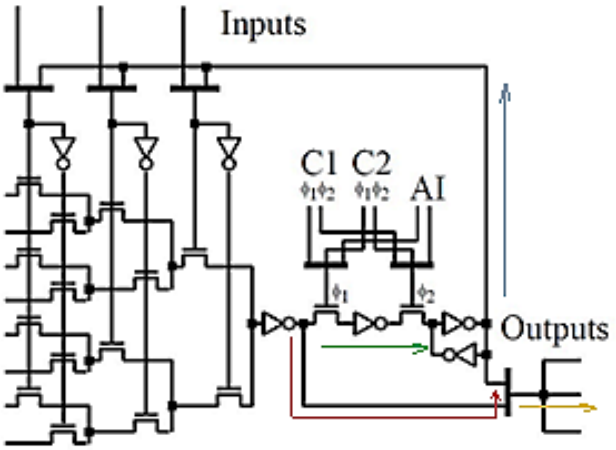



Рисунок 1.6 – Логический элемент LUT, адаптированный для реализации асинхронных схем.

Если в стандартном LUT выход дерева транзисторов может передаваться в матрицу локальных связей непосредственно, либо через D-триггер, то здесь с помощью дополнительного коммутатора возможно двухфазное управление по двум цепям синхронизации C1 и C2, кроме того, возможна реализация асинхронного режима A. Выход триггера через коммутаторы переменных может быть использован как обратная связь-на один из трёх входов переменных. Предлагаемый встроенный (special, built-in) арбитр изображен на рисунке 1.7.

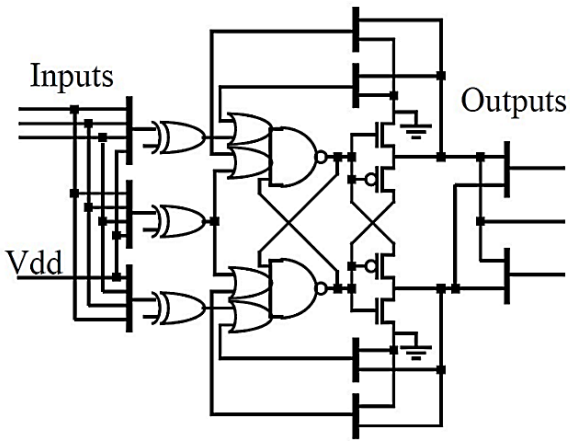



Рисунок 1.7 – Арбитр для FPGA с инвертируемыми входами.

Входы могут инвертироваться. Очевидно, это достигается программированием неподключенных входов трех элементов сложения по модулю два (исключающее ИЛИ). Подчёркивается необходимость иметь соотношение 15:1 между количеством логических блоков и блоками арбитров. Асинхронный арбитр формирует сигналы очерёдности запросов, схема для двух запросов приведена, например, в [20] – рисунок 1.8.

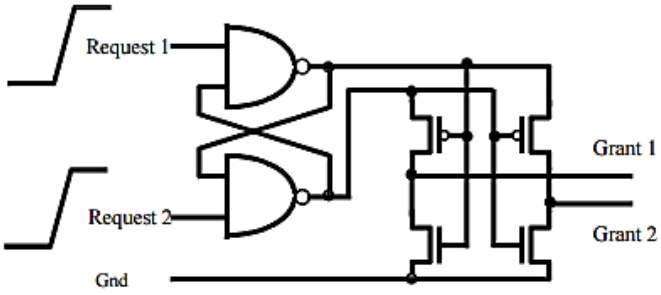



Рисунок 1.8 – Асинхронный арбитр на два запроса.

Таким образом, арбитр представляет собой входной SR триггер с инверсным управлением и два выходных инвертора, входы которых подключены к одному их выходов SR триггера, но питание (+) каждого инвертора подключено к другому выходу SR триггера. Выполним моделирование арбитра в системе NI Multisim [58]. При нулевых входах запросов S (Requst 1) =0, R (Requst 2) =0 на выходах очерёдности Grant1, Grant 2 – нули – рисунок 1.9.

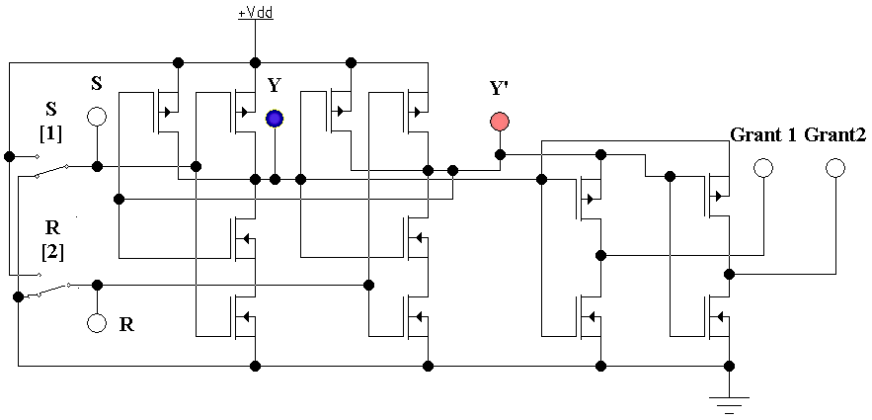



Рисунок 1.9 – Моделирование асинхронного арбитра: Grant1=0, Grant 2=0.

При получении первого запроса S (Requst 1) =1, R (Requst 2) =0 схема формирует Grant1=1, Grant 2=0 – рисунок 1.10.

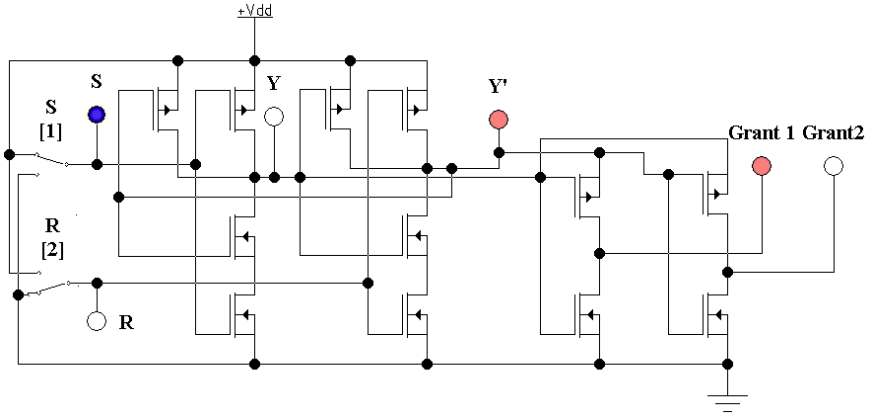



Рисунок 1.10 – Моделирование асинхронного арбитра: Grant1=1, Grant 2=0.

Если первым поступает второй запрос - формируется Grant1=0, Grant 2=1 – рисунок 1.11.

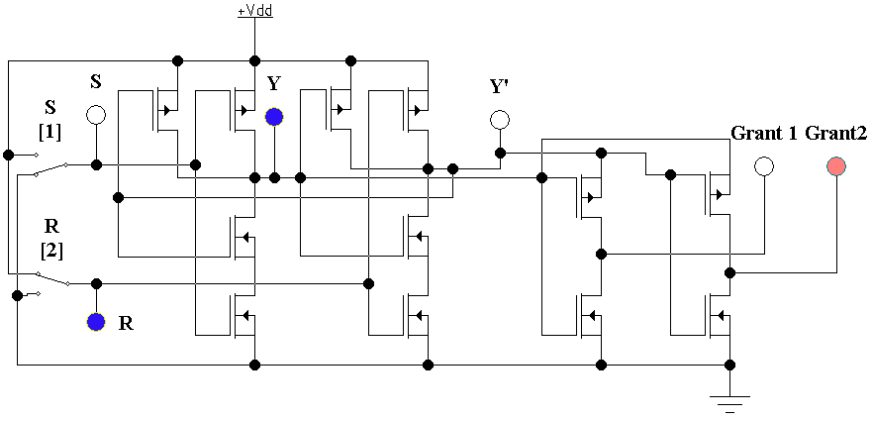



Рисунок 1.11 – Моделирование асинхронного арбитра: Grant1=0, Grant 2=1.

При задании S (Requst 1) =1, R (Requst 2) =1 из нулевого состояния предпочтение отдаётся первому, формируется Grant1=1, Grant 2=0 – рисунок 1.12.

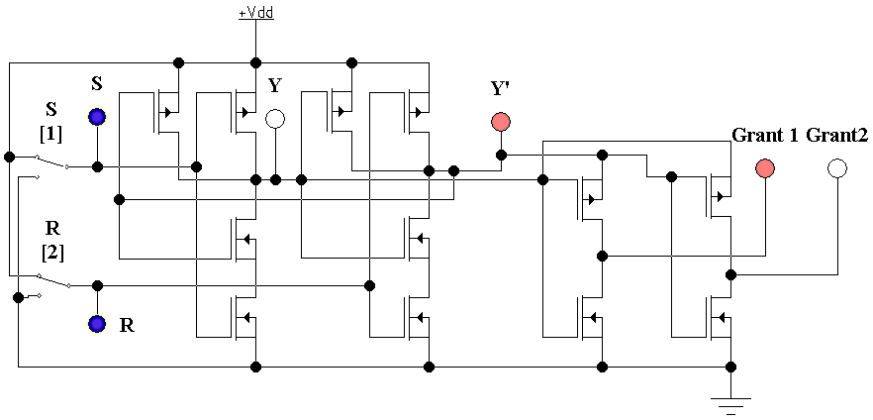



Рисунок 1.12 – Моделирование асинхронного арбитра: Grant1=1, Grant 2=0.

Таким образом, получаемые в такой ПЛИС асинхронные схемы будут относиться к классу независимых от задержки, а точнее квази-независимых от задержки.

В работе [60] предложена архитектура STACC (Self-timed Array of Configurable Cells), рисунок 1.13.

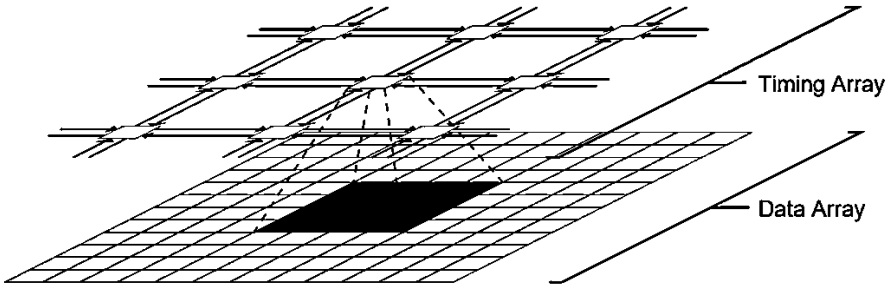



Рисунок 1.13 – Архитектура STACC.

Архитектура применима к мелкозернистым FPGA, то есть для логических элементов, представляющих собой простые вентили, и использует модель Bundled-delay, то есть «со связанной задержкой», хорошо подходит для конвейеризации. Получаемые схемы также относятся к классу квази-независимых от задержки. Такой подход развит в [61] в сторону реконфигурируемых конвейерных структур. В предлагаемой архитектуре [61] используются гистерезисные триггеры (С-элементы Маллера) – рисунок 1.14.

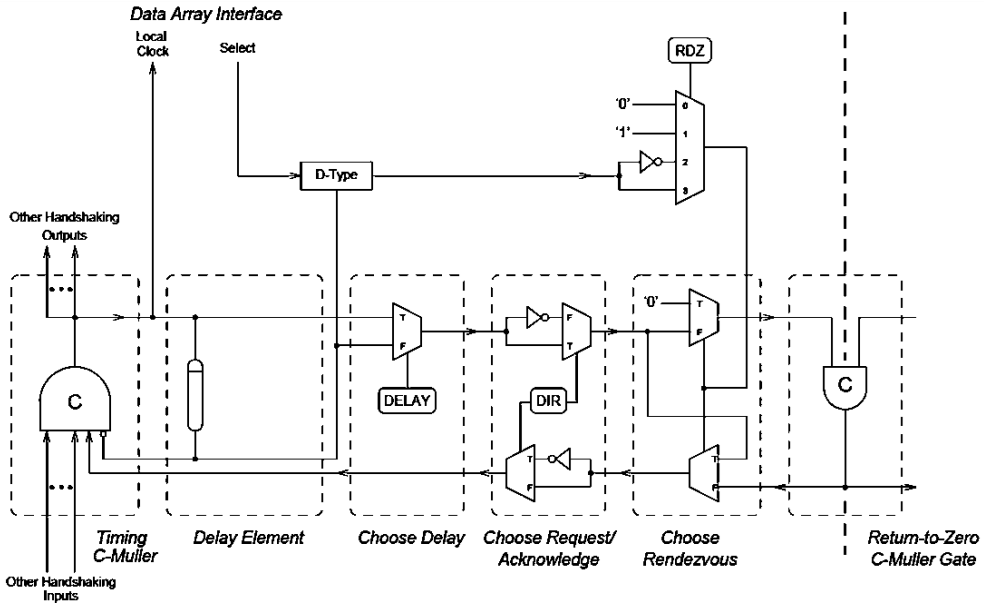



Рисунок 1.14. Использование С-элементов Маллера в архитектуре STACC.

В [62] рассматривается реализация самосинхронного подхода (self-timed cell) в ПЛИС Xilinix (1994 г.). Однако подробности устройства ячейки не раскрываются.

Среди работ нулевых годов XXI можно отметить [9,63], в которых берётся за основу ранее предложенный реконфигурируемый логический элемент для реализации NCL (Null Convention Logic) схем – рисунок 1.15 модифицируется для использования LUT - рисунок 1.16 а), б).

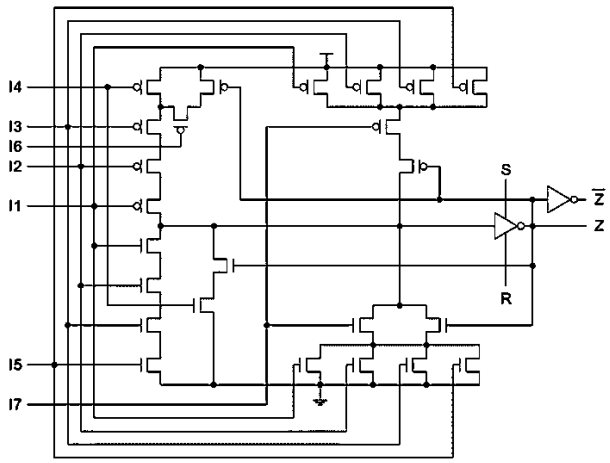
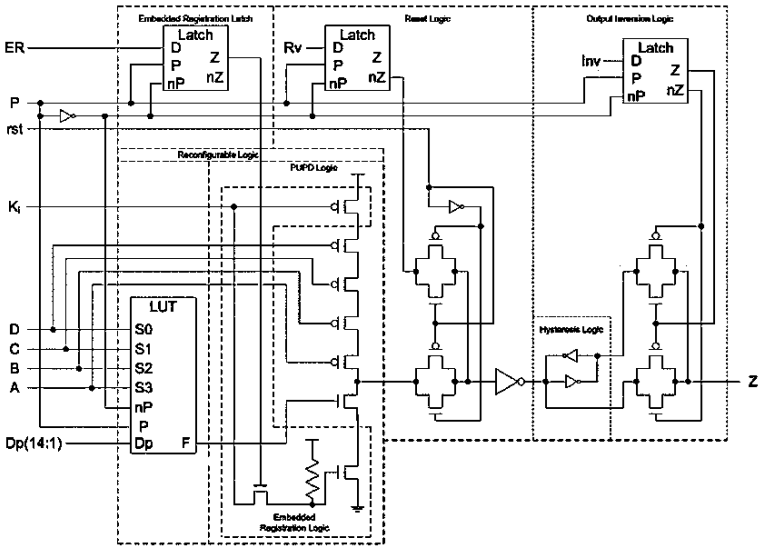


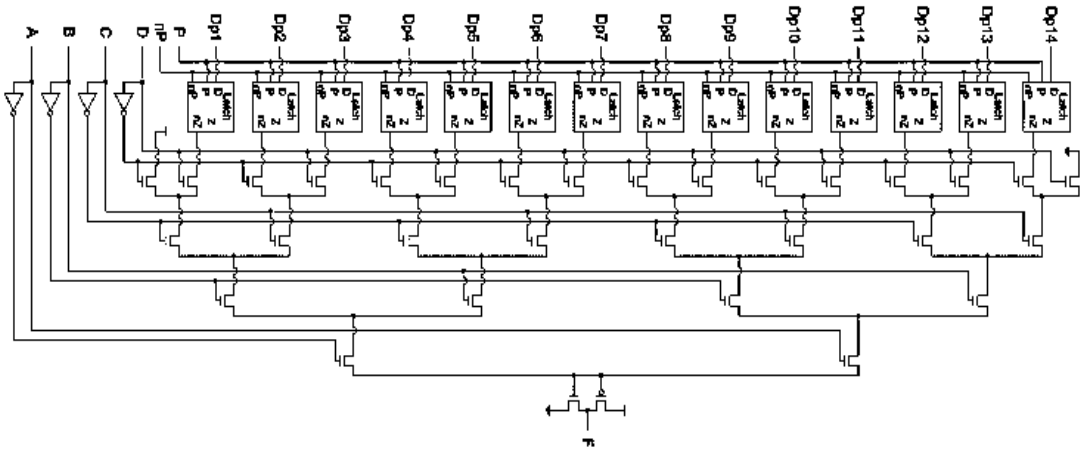

Рисунок 1.15 – Реализация NCL логики ПЛИС - реконфигурируемый сигналами I1–I7 NCL элемент.

Исходный реконфигурируемый NCL-элемент на рисунке 1.15 состоит из 28 транзисторов и может быть запрограммирован (по входам I1-I7) на 8 из 27

функций порогового элемента. Имеется обратная связь (z), реализованы, но не показаны на рисунке сброс S и установка R.




а)




б)

Рисунок 1.16 – Реализация NCL схемы логики ПЛИС:

а) LUT, как NCL элемент; б) модифицированный LUT.

На рисунке 1.16 а) логика сброса и установки раскрыты, причём подпись (Histeresis Logic) не оставляет сомнения, что речь идёт о гистерезисном триггере. Также указан блок PUPD Logic (PullUp, Pull Down). Таким образом, речь идёт о ПЛИС, использующей NCL. На рисунке 1.16 б) детализируется LUT - использованы 14 бит настройки, вместо полных 16 на 4 переменные, так как на наборах «все единицы» формируется ноль (подключено к «Нулю вольт» - все нули, все единицы, если подключено к шине питания).

В [13] описывается представляемая как асинхронная ПЛИС Speedster22i фирмы Achronix (якобы спроектирована по заказу Интел). Но из описания становится ясно, что эта ПЛИС имеет две иерархических сети тактирования – глобальную Global Clock Generator (GCG) и прямую. Имеются четыре глобальных источника тактирования GCG по четырём углам микросхемы – рисунок 1.17.

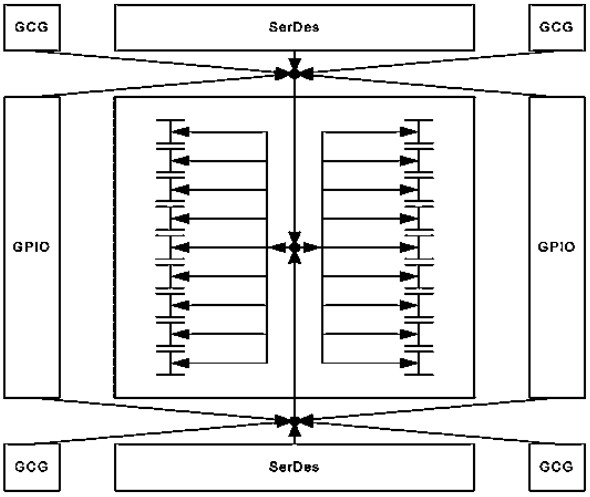



Рисунок 1.17 – Глобальная сеть тактирования ПЛИС Speedster22i.

Локальная тактировка обеспечивается «региональными» устройствами Regional Clock Manager (RCM). Прямая сеть тактирования основана на внутренних генераторах и обеспечивает значительно более низкую задержку. Таким образом, речь идёт о множественной распределённой синхронизации – рисунок 1.18.

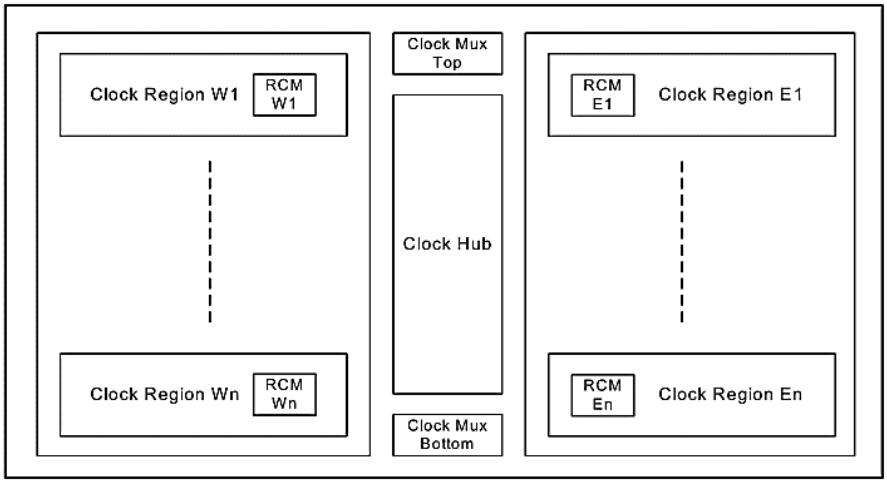



Рисунок 1.18 – Восточный E и западный W «регионы» тактирования ПЛИС Speedster22i.

Анализ показывает, что направление асинхронных ПЛИС всё ещё находится на этапе исследований, большинство ПЛИС синхронные. По запросу «Self Timed FPGA» выводится более 40 источников.

Так, описывается самосинхронная память, имеющая широкий температурный диапазон, высокую радиационную стойкость (проект НАСА) [64].

Описания строго самосинхронных (ССС) ПЛИС в доступных источниках не обнаружено [65-70].

В настоящее время происходит переход на технологии ПЛИС с транзисторами Tri-Gate (FPGAs with Tri-Gate Technology), например, Intel Tri-Gate, Stratix 10 FPGA [71] с проектными нормами 14 нм и ниже. В таких ПЛИС используется конвейеризация. Направление ПЛИС в свое время привело к созданию целых систем на кристалле - SoC, но это планарные устройства. Дальнейшее развитие технологий позволило создать SiP (System-in-Package), что можно перевести как систему в объёмном пакете - 3D системы [72]. В то же время технологии тактирования остаются прежними. Причём при распространении тактового сигнала по длинным связям в ПЛИС, имеющей миллионы логических элементов (адаптивных логических модулей -АЛМ), возникающие нежелательные явления – фазовые и/или частотные отклонения – джиттер (англ. jitter — дрожание) – ещё более усугубляются. Поэтому в ПЛИС используют модули цифровой автоподстройки тактового сигнала (DLL - Delay Locked Loop), блоки DCM (Digital Clock Manager) с функциями умножения и деления частоты, сдвига фазы, а также блоки фазовой подстройки частоты (PLL - Phase Locked Loop) [73].

Возможно, одним из направлений тактирования в ПЛИС может стать комбинирование самосихронного и синхронного подхода для использования преимуществ того и другого и сглаживания недостатков.

Кроме этого в работе [8] были предложены усовершенствованные методы реализации систем логических функций в FPGA и CPLD. Была доказана их

эффективность по сравнению с известными методами реализации логических функций, но они не могут использоваться в исходном виде для создания ССС ПЛИС.

### 1.4. Постановка задачи разработки конфигурируемых логических элементов для самосинхронных схем

Анализ объекта исследования показал, что проектирование самосинхронных схем в основном реализуется на специализированных базисах или функционально-полных толерантных элементах и нет каких-то универсальных, настраиваемых базисов, которые можно использовать в типовых проектах.

Кроме развития самосинхронного подхода, для проектирования аппаратуры активно используются ПЛИС, которые существенно позволяют сократить время проектирования и стоимость проектирования сложных систем. Выполненный анализ существующих асинхронных ПЛИС показал, что данное направление все еще находится на этапе исследований и большинство представленных схем не являются строго самосинхронными.

Таким образом, анализ показал необходимость решения научно-технической задачи разработки методов синтеза конфигурируемых логических элементов для ССС. Поэтому предлагается исследовать возможность совмещения самосинхронного подхода и ПЛИС для создания конфигурируемых самосинхронных логических элементов.

Формальная постановка общей задачи исследования формулируется следующим образом:

**Дано:** множество неунифицированных (специализированных) комбинационных логических преобразователей, каждое преобразование $\psi_i$ характеризуется числом входных переменных $n$, количеством логических функций $m$ в системе, количеством систем $y$, числом конъюнкций $w$ и индексом $s$ использования этого логического преобразователя.

**Получить:** множество $E = \{\varepsilon_1, \varepsilon_2, \dots \varepsilon_u\}$ методов унифицированной самосинхронной реализации систем логических функций. Конфигурирование производить либо константами, либо записью значений в ячейки памяти SRAM. Конфигурация связей в данной работе не рассматривается. Каждый из методов характеризуется:

a) Оценкой сложности **L** (в количестве необходимых транзисторов):

$$L = \{L_{\tau_1}(n, m, w), L_{\tau_2}(n, m, w), \dots . L_{\tau_e}(n, m, w)\} \quad (1.1)$$

b) Оценкой быстродействия **T** (задержки, измеряемой количеством транзисторов в самом длинном пути со входа схемы на выход):

$$T = \{T_{\tau_1}(n, m, w), T_{\tau_2}(n, m, w), \dots . T_{\tau_e}(n, m, w)\} \quad (1.2)$$

c) Оценкой площади, занимаемой на кристалле **S:**

$$S = \{S_{\tau_1}(n, m, w), S_{\tau_2}(n, m, w), \dots . S_{\tau_e}(n, m, w)\} \quad (1.3)$$

d) Оценкой потребляемой мощности **P:**

$$P = \{P_{\tau_1}(n, m, w), P_{\tau_2}(n, m, w), \dots . P_{\tau_e}(n, m, w)\} \quad (1.4)$$

При получении оценок необходимо учесть существующие ограничения схемной реализации методов в современных заказных и полузаказных микросхемах. Для подтверждения работоспособности методов выполнить функциональное и топологическое моделирование, а также проверку полумодулярности схемотехнических решений (элементов) $\Omega = \{\omega_1, \omega_2, \dots \omega_u\}$.

Получить оптимальные наборы H элементов:

$$H = \langle \langle \omega_1(n, m, w), s \rangle, \langle \omega_2(n, m, w), s \rangle, \dots \langle \omega_u(n, m, w), s \rangle \rangle,$$

такой, что $[L(H) \to min] \vee [T(H) \to min] \vee [S(H) \to min] \vee [P(H) \to min]$

### 1.5. Выводы по главе

1. Анализ существующих методов синтеза ССС показал, что в настоящее время он остается труднорешаемой задачей, но, несмотря на это, появляются новые подходы. Была исследована работа Каменских А.Н, в которой была решена научно-техническая задача усовершенствования существующего

аппарата синтеза ССС для комбинированного резервирования, но конфигурируемая реализация функций рассмотрена не была.

2. Анализ существующих асинхронных ПЛИС показал, что появляется множество статей на исследуемую тему, но данное направление все еще находится на этапе исследований, и большинство представленных в статьях схем не являются строго самосинхронными. Некоторые работы посвящены использованию известных ПЛИС с доработкой, например, использование арбитров, которые позволяют устранить негативные последствия состязаний сигналов. Предлагается архитектура STACC с использованием в чистом виде С-элементов Маллера. Кроме этих двух вариантов предлагается реконфигурируемый логический элемент для реализации NCL (Null Convention Logic). Однако подробности устройства ячейки не раскрываются. В открытых источниках описаний по самосинхронным ПЛИС обнаружено не было. Была исследована работа Вихорева Р.В, в которой была решена научно-техническая задача выбора оптимального набора логических элементов для реализации логических функций в ПЛИС, в результате которой были разработаны новые эффективные логические элементы.

3. Выполненный анализ показывает актуальность проведения исследований в области создания методов реализации конфигурируемых самосинхронных генераторов логических функций, заданных в СДНФ и ДНФ.

4. Одним из перспективных направлений является модификация существующих элементов, используемых для построения ПЛИС, к условиям работы в ССС.

# ГЛАВА 2. РАЗРАБОТКА МЕТОДОВ РЕАЛИЗАЦИИ СИСТЕМ ЛОГИЧЕСКИХ ФУНКЦИЙ НА ОСНОВЕ КОНФИГУРИРУЕМЫХ САМОСИНХРОННЫХ ЭЛЕМЕНТАХ

## 2.1. Разработка метода реализации конфигурируемого самосинхронного генератора логических функций на основе стандартных логических элементов

Метод ориентирован на использование элементной базы БМК, поэтому предполагается использовать универсальный логический модуль, в качестве которого необходимо применить библиотечные элементы.

Для разработки самосинхронного элемента требуется адаптировать библиотечные элементы к условиям работы в самосинхронных схем. Ввиду этого необходимо реализовать два канала передачи данных: прямой и двойственный, индикацию каждого блока схемы и фиксирование окончания переходных процессов в каждом компоненте схемы.

Для реализации логической функции от $n$ переменных $X_j$ предлагается использовать библиотечный элемент 2И-2ИЛИ-НЕ (A22OI) [5], описываемый выражением:

$$F = \overline{I0 \cdot I1 \vee I2 \cdot I3} \qquad (2.1)$$

где F – выход элемента 2И-2ИЛИ-НЕ, I0, I1, I2, I3 – входы элемента.

Суть метода заключается в том, чтобы использовать два входа из четырех в качестве информационных входов, а два входа для настройки (конфигурирования). Для реализации фазы спейсера предлагается использовать дополнительные внешние блоки, через которые на входы переменных $X_j$, $X_j$’ будут подаваться значения переменных и сигнала разрешения - спейсера (Sp). Таким образом, метод заключается в реализации каналов, индикаторов и элементов, фиксирующих окончания переходных процессов.

**Теорема 2.1.** О реализации самосинхронного генератора функций на библиотечном элементе БМК 2И-2ИЛИ-НЕ [5,74].

Докажем, что функция

$$f(x_1x_2x_3x_4) = (\overline{x_1} \vee \overline{x_2})(\overline{x_3} \vee \overline{x_4}) = \overline{x_1x_2 \vee x_3x_4} \quad (2.2)$$

2И-2ИЛИ- НЕ или ФПТ2-элемент [74] позволяет синтезировать конфигурируемый СС элемент, реализующий любую логическую функцию *n* переменных.

Построим универсальный элемент для реализации любой функции одной переменной, заменяя:

$$x_1 = d_0;\ x_2 = \bar{x};\ x_3 = d_1;\ x_4 = x \quad (2.3)$$

Используя (2.2) и (2.3) получим:

$$f(x\bar{x}d_0d_1) = \overline{d_0\bar{x} \vee d_1x} \quad (2.4)$$

При этом $\bar{x}, x$ – парафазная переменная. Двойственный канал в этом случае трактуется как:

$$\overline{f(x\bar{x}d_0d_1)} = \overline{\overline{d_0}\bar{x} \vee \overline{d_1}x} \quad (2.5)$$

Действительно,

$$\overline{d_0\bar{x} \vee d_1x} = \left(\overline{d_0} \vee x\right)\left(\overline{d_1} \vee \bar{x}\right) = \overline{d_0}\,\overline{d_1} \vee \overline{d_1}x \vee \overline{d_0}\bar{x} \vee x\bar{x} = \overline{d_0}\bar{x} \vee \overline{d_1}x \quad (2.6)$$

Причем $x\bar{x} = 0$ по закону противоречия, $\overline{d_0}\,\overline{d_1} \vee \overline{d_1}x \vee \overline{d_0}\bar{x} = \overline{d_0}\bar{x} \vee \overline{d_1}x$ по закону обобщенного склеивания.

Следовательно, получаем для формирования основной *F* и двойственной *F'* логических функций одной переменной *x* с основной и инверсной конфигурационными настройками *d*:

$$\begin{cases} F(x\bar{x}d_0d_1) = \overline{d_0\bar{x} \vee d_1x} \\ F'(x\bar{x}d_0d_1) = \overline{\overline{d_0}\bar{x} \vee \overline{d_1}x} \end{cases} \quad (2.7)$$

Функциональная полнота системы (2.7) для n=1 очевидна, поскольку настройка *d* позволяет получить как константы 0,1, так и функции повторения и инверсии (отрицания).

Докажем реализацию спейсера. При $x = \bar{x} = 0$ получаем:

$$\begin{cases} F(00d_0d_1) = \overline{d_0 0 \vee d_1 0} = 1 \\ F'(00d_0d_1) = \overline{\overline{d_0}0 \vee \overline{d_1}0} = 1 \end{cases} \quad (2.8)$$

Таким образом, выполняются условия для реализации любой логической функции одной переменной путем конфигурирования настройкой *d*.

Для реализации функций большего числа переменных итеративно выполняется каскадирование (2.7). Примем в качестве базиса индукции. Пусть реализована система на *i* переменных:

$$\begin{cases} F_i\left(X_i\overline{X}_\iota d_0 \dots d_{2^i-1}\right) \\ F_i{}'\left(X_i\overline{X}_\iota\overline{d_0} \dots \overline{d_{2^\iota-1}}\right) \end{cases} \quad (2.9)$$

где $X_i\overline{X}_\iota$ - парафазный вектор *i* переменных.

Покажем, что *i*+1 получим реализацию любой функции *i*+1 переменной. Действительно, принимая (2.9) в качестве конфигурационных данных, получим:

$$\begin{cases} F_{i+1}(F_iX_{i+1}\overline{X_{\iota+1}}) \\ F_{i+1}{}'(F_i{}'X_{i+1}\overline{X_{\iota+1}}) \end{cases} \quad (2.10)$$

Выражение (2.10) реализует условия функциональной полноты и спейсера относительно $X_{i+1}\overline{X_{\iota+1}}$ с учетом (2.9). При *i*+1=*n* получаем требуемую реализацию. Теорема доказана.

Использование индикации и Г-триггеров осуществляется известным образом.

Модель элемента генератора функций для n=1 показана на рисунке 2.1.

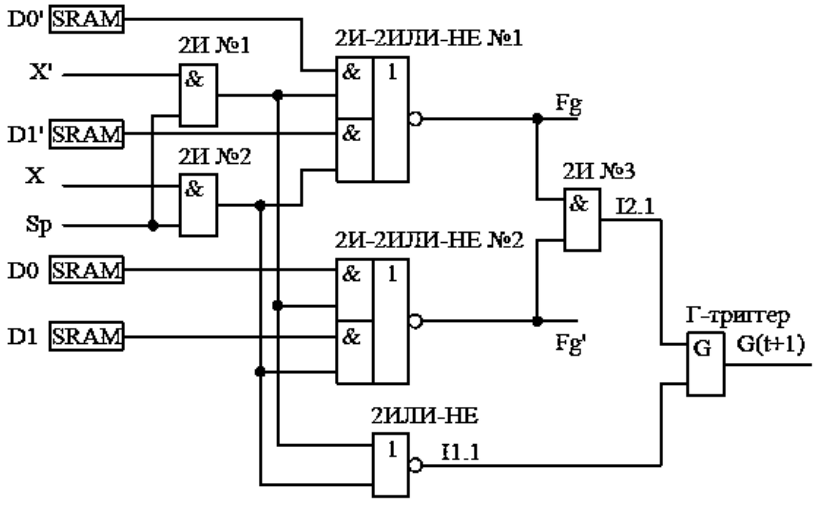



Рисунок 2.1 – Модель генератора функций (ГФ) одной переменной.

Элементы 2И №1 и 2И №2 – реализуют блоки входного набора, элемент 2И-2ИЛИ-НЕ №1 реализует прямой канал, а элемент 2И-2ИЛИ-НЕ №2 реализует двойственный канал. Элементы 2ИЛИ-НЕ и 2И №3 используются в качестве индикаторов, а Г-триггер используется для фиксирования окончания

переходных процессов в схеме. Для организации самосинхронизации используется обратная связь, не показанная на рисунке 2.1 - с выхода G(t+1) Г-триггера на вход Sp блоков 2И №1 и 2И №2.

Для реализации элемента на две и более переменных предлагается использовать каскадирование элемента на одну переменную, при этом необходимо учитывать настройку основного канала таким образом, что при четном числе каскадов инверсия настройки не требуется, при нечетном требуется. Тогда выражение, описывающее данного условие:

$$D^{\sigma};\ \begin{matrix}\sigma = 0\ if\ [(2i-1) = n] = \ true \\ \sigma = 1\ if\ [2i = n] = \ true\end{matrix}\ , i = 1, 2^n \tag{2.11}$$

где σ=1 для прямой настройки, σ=0 для инверсной.

Таким образом принцип каскадирования может быть пояснен таблицей:

Таблица 2.1 Каскадирование элементов ГФ:

| **i+1 каскад** | **i каскад** |
|---|---|
| Fgi+1.1 | D0.i |
| Xi+1’ | Xi’ |
| Fgi+1.2 | D1.i |
| Xi+1 | Xi |
| Fg’i+1.1 | D0’.i |
| Xi+1’ | Xi’ |
| Fg’i+1.2 | D1’.i |
| Xi+1 | Xi |

Получим выражения, описывающие модель генератора функций для n=2. Выражения, описывающие работу основных и двойственных каналов первого слоя (2.12):

$$\begin{cases} Fg11 = (D0 \cdot X1' \cdot Sp \vee D1 \cdot X1 \cdot Sp)' \\ Fg12 = (D0' \cdot X1' \cdot Sp \vee D1' \cdot X1 \cdot Sp)' \\ Fg13 = (D2 \cdot X1' \cdot Sp \vee D3 \cdot X1 \cdot Sp)' \\ Fg14 = (D2' \cdot X1' \cdot Sp \vee D3' \cdot X1 \cdot Sp)' \end{cases} \tag{2.12}$$

где *Fg11, Fg13*- выходы основных каналов, *Fg12, Fg14*- выходы инверсных каналов, *Sp* – нулевой спейсер, *D0, D1, D2, D3* – настройка ячейки SRAM.

Выражение, описывающее работу основного канала второго слоя:

$$Fg = [Fg11 \cdot (Sp' \vee X2') \vee Fg13 \cdot (Sp' \vee X2)]' \tag{2.13}$$

Выражение, описывающе работу двойственного канала второго слоя:

$$Fg' = [Fg12 \cdot (Sp' \vee X2') \vee Fg14 \cdot (Sp' \vee X2)]' \quad (2.14)$$

где *Sp'* – единичный спейсер.

Индикация первого слоя описывается выражениями:

$$\begin{cases} I1.1 = (Fg11 \cdot Fg12)' \\ I1.2 = (Sp \cdot X1' \vee Sp \cdot X1)' \\ I1.3 = (Fg13 \cdot Fg14)' \\ G1(t+1) = [I1.1' \cdot I1.2 \cdot I1.3' \vee G1(t) \cdot (I1.1' \vee I1.2 \vee I1.3')]' \end{cases} \quad (2.15)$$

где $I1.1$, $I1.3$ – индикаторы выходов, $I1.2$ – индикатор входов, $G1(t+1)$ – индикатор окончания переходных процессов первого слоя.

Выполним описание индикации второго слоя:

$$\begin{cases} I2.1 = (Fg \cdot Fg')' \\ I2.2 = (Sp' \cdot X2' \cdot Sp' \cdot X2)' \\ G2(t+1) = [I2.1 \cdot I2.2 \cdot G1(t) \vee G2(t)(I2.1 \vee I2.2 \vee G1(t))]' \end{cases} \quad (2.16)$$

где $I2.1$ – индикатор выхода, $I2.2$ – индикатор входа, $G2(t+1)$ –общий индикатор схемы.

Модель элемента генератора функций для n=2:

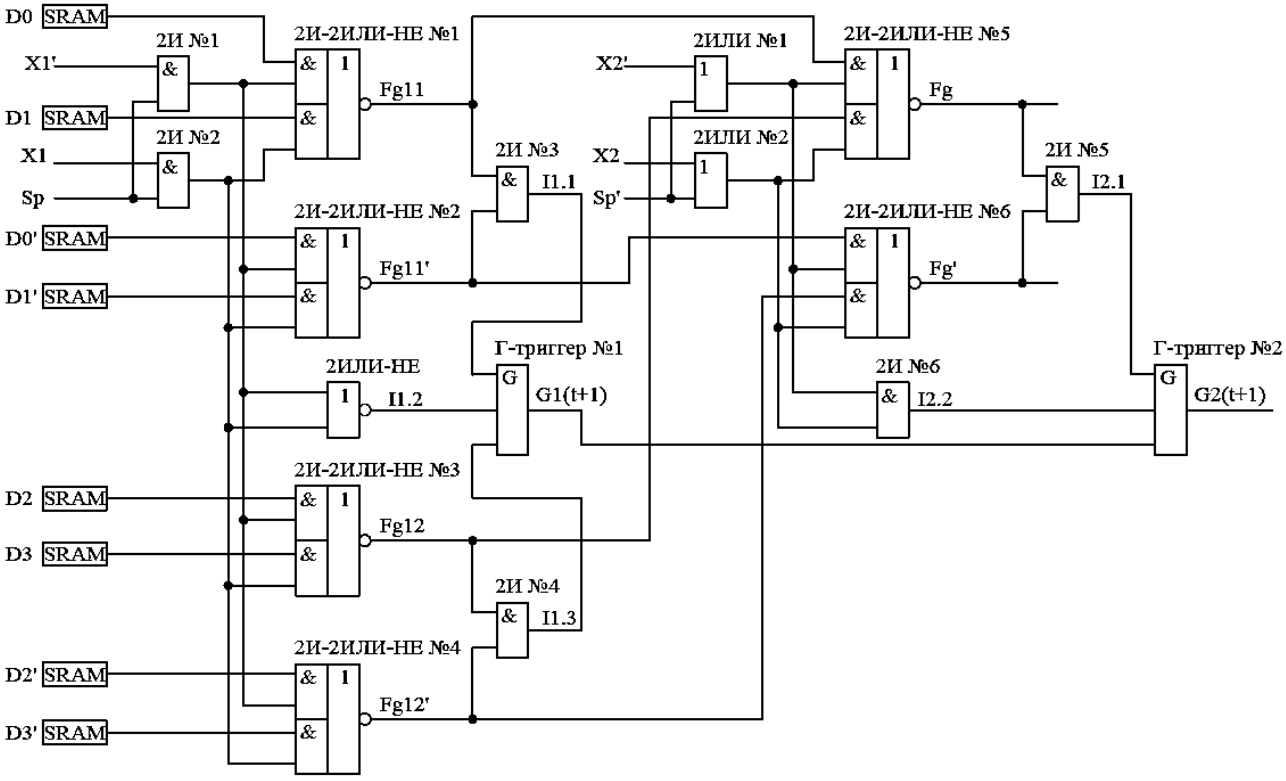



Рисунок 2.2 – Модель генератора функций двух переменных.

Элементы 2И №1, 2И №2 и 2ИЛИ №1, 2ИЛИ №2 – реализуют блоки входного набора для первого и второго каскада. Элементы 2И-2ИЛИ-НЕ №1, 2И-2ИЛИ-НЕ №3 и 2И-2ИЛИ-НЕ №5 реализуют прямой канал. Для реализации двойственного канала используются элементы 2И-2ИЛИ-НЕ №2, 2И-2ИЛИ-НЕ №4 и 2И-2ИЛИ-НЕ №6. Элементы 2ИЛИ-НЕ, 2И №3, 2И №4, 2И №5 и 2И №6 используются в качестве индикаторов. Г-триггеры используется для фиксирования окончания переходных процессов в схеме. Для организации самосинхронизации используется обратная связь, не показанная на рисунке 2.2 - с выхода G2(t+1) Г-триггера на вход Sp блоков 2И №1 и 2И №2, а также на вход Sp’ блоков 2ИЛИ №1 и 2ИЛИ №2, причем с инверсией.

Для подтверждения принадлежности разработанного метода к классу ССС, выполним проверку схемы на полумодулярность. Для этого получим модель Маллера для каждого узла схемы на рисунке 2.1.

D0=1; D1=0; neD0=0; neD1=1; X=1; neX=0; / Описываем постоянные схемы

V1=neX*Sp; V2=X*Sp; / Описываем входы на основной и двойственный каналы

F1=^D0*V1|D1*V2; / Описываем основной канал

F2=^neD0*V1|neD1*V2; / Описываем двойственный канал

I11=^V1|V2; Описываем индикаторы входов

I21=^F1*F2; I2=^I21; / Описываем индикатор выхода

I2i=^I2*I11|I2k(I2|I11); I2k=^I2i; / Г-триггер

E=I2k; / Управляющий сигнал

$F1*F2*I2i ^ / Описываем начальные состояния схемы (В начальном состоянии схема находится в фазе спейсера.)

Полученный результат (рисунок 2.3) подтверждает принадлежность предложенного в разработанном методе элемента к классу самосинхронных схем [3].

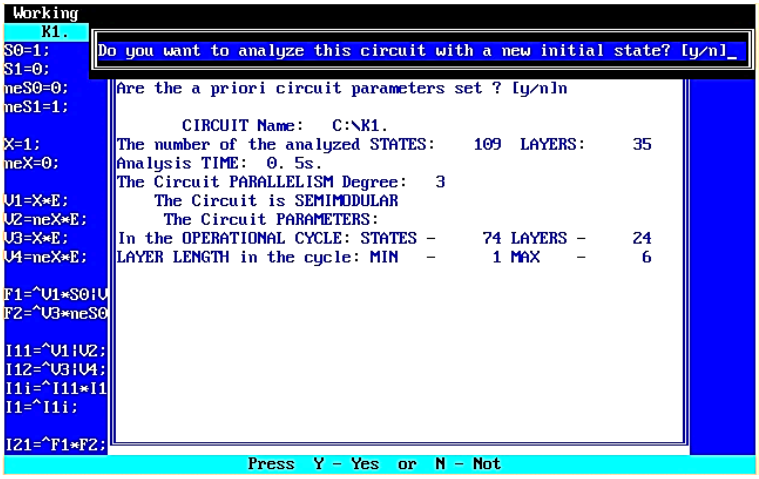



Рисунок 2.3 – Результат анализа полумодулярности генератора функций одной переменной с нулевым спейсером в САПР Forcage (подсистема TRANAL).

Таким же образом выполним проверку модели генератора функций двух переменных (рисунок 2.2)

D0=0; D1=1; D2=1; D3=0; /Задаем константы

X1=1; X2=0; / Задаем переменные

neX1=^X1; neX2=^X2;

neD0=^D0; neD1=^D1; neD2=^D2; neD3=^D3;

V1=Sp*neX1; V2=Sp*X1; / Описываем входные элементы первого слоя

V3= Sp ^|neX2; V4= Sp ^|X2; / Описываем входные элементы второго слоя (Sp ^ - единичный спейсер)

F11=^D0*V1|D1*V2; F12=^neD0*V1|neD1*V2; F13=^D2*V1|D3*V2; F14=^neD2*V1|neD3*V2; / Описываем элементы 2И-2ИЛИ-НЕ первого слоя

F1=^F11*V3|F13*V4; F2=^F12*V3|F14*V4; / Описываем элементы 2И-2ИЛИ-НЕ второго слоя

I11=^V1|V2; I1v=^F11*F12; I2v=^F13*F14; / Описываем индикаторы входов/выходов первого слоя

G1i=^I11*I1v^*I2v^|G1ii(I11|I1v^|I2v^); G1ii=^G1i; / Описываем индикатор окончания переходных процессов первого слоя (С-элемент)

I21=V3*V4; Iv=^F1|F2; / Описываем индикаторы входов/выходов второго слоя

G2i=^I21*Iv*G1ii|G2ii(I21|Iv|G1ii); G2ii=^G2i; / Описываем индикатор окончания переходных процессов схемы (С-элемент)

E=G2ii; / Задаем нулевой спейсер

$G2i^ / Задаем начальное состояние схемы.

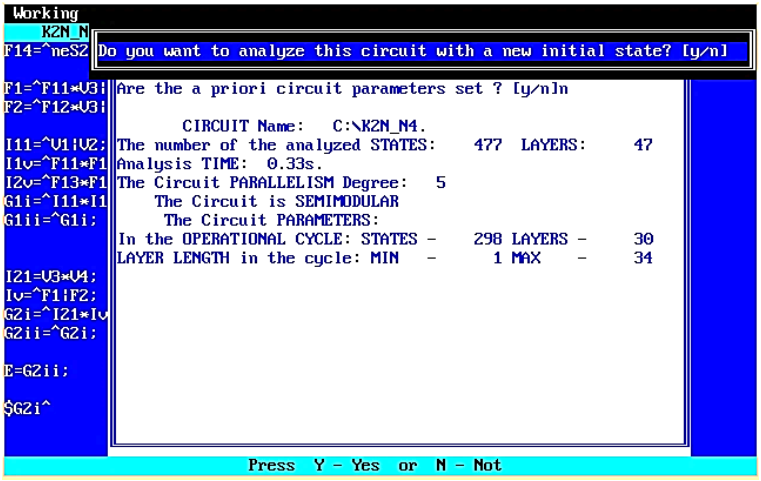



Рисунок 2.4 – Результат анализа полумодулярности генератора функций двух переменных.

Как видно на рисунке 2.4, модель генератора функций двух переменных является полумодулярной, как и модель генератора функций одной переменной. Таким образом, можно получить модели генераторов функций на любое число переменных. Проверка моделей на реализацию других функций представлена в приложении №3.

## 2.2. Разработка метода реализации конфигурируемого самосинхронного генератора логических функций по принципу LUT

Метод ориентирован на использование универсального элемента программируемых логических интегральных схем, так называемого Look Up Table «просмотровой таблицы». Как и в предыдущем методе, требуется адаптировать его к условиям работы ССС.

Существо метода заключается в том, чтобы модифицировать дерево передающих транзисторов, применяемое в существующих синхронных LUT (Look Up Table) FPGA (Field-Programmable Gate Array) [7,75,76]. Выражение, описывающее работу исходного элемента:

$$F_{out.l} = \bigvee_{i=0}^{2^n-1} \left( \&_{j=1}^{2n} X_j^{\sigma(i,j)} \cdot D_i \right), l = 1, m \quad (2.17)$$

где $F_{out.l}$ – корень дерева, $l$-ая логическая функция, $\sigma(i,j)$ – показатель инверсирования переменной $X_j$, двоичный эквивалент которого указывает на $j$-ую из $2^n$ входов (проекция $\Psi_j(B_i)$), $\sigma(i,j)$=1 для $X_j$; $\sigma(i,j)$=0 для $X_j'$; $D_i$ – настройка логической функции по одному из $2^n$ входов, осуществляемая путём конфигурирования ПЛИС записью конфигурационного файла в оперативную память SRAM.

Для адаптации к ССС предлагается введение ветви спейсера, активируемого при $X_j = X_j' = 0$:

Таким образом выражение (2.17) с учётом реализации спейсера примет вид:

$$F_{out.l.st} = \bigvee_{i=0}^{2^n-1} \left( \&_{j=1}^{2n} X_j^{\sigma(i,j)} \cdot D_i \right) \vee \left( \&_{j=1}^{n} X_j X_j' \right)', l = 1, m \quad (2.18)$$

где $\&_{j=1}^{n} X_j X_j'$ - цепочка реализации спейсера.

Для реализации двойственного канала конфигурируемого элемента необходима инверсия выражения (2.17).

**Теорема 2.2.** Докажем, что

$$F'_{out.l.st} = \bigvee_{i=0}^{2^n-1} \left( \&_{j=1}^{2n} X_j^{\sigma(i,j)} \cdot D'_i \right) \vee \left( \&_{j=1}^{n} X_j X_j' \right)', l = 1, m \quad (2.19)$$

где $F'_{out.l.s}$ – значение двойственной логической функции.

Для этого рассмотрим множество векторов:

$$\vec{X_i} \in \left( \bigvee_{i=0}^{2^n-1} \left( \&_{j=1}^{2n} X_j^{\sigma(i,j)} \right) \right), \quad (2.20)$$

тогда

$$\overline{\bigvee_{i=0}^{2^n-1} \left( \vec{X_i} \cdot D_i \right)} = \&_{i=0}^{2^n-1} \left( \overline{\vec{X_i}} \vee D'_i \right), \quad (2.21)$$

где операция отрицания (инверсии) и соответствующий закон Де Моргана применены к векторам, а не к отдельным переменным, $D'_i$- отрицание $i$-го бита настройки.

Отрицание одного вектора может рассматриваться как дизъюнкция остальных векторов. То есть:

$$\overset{2^n-1}{\underset{i=0}{\&}}\left(\overline{\vec{X}_i} \vee D'_i\right) = \overset{2^n-1}{\underset{i=0}{\&}}\left[\overset{2^n}{\underset{\forall\xi\neq 1}{\vee}}\left(\vec{X}_\xi\right) \vee (D'_i)\right]. \quad (2.22)$$

Пусть $n$=1, тогда вектора состоят из одного бита, получим:

$$\overset{1}{\underset{i=0}{\&}}\left(\overline{\vec{X}_i} \vee D'_i\right) = \overset{1}{\underset{i=0}{\&}}\left[\left(\vec{X}_\xi\right) \vee (D'_i)\right]. \quad (2.23)$$

Согласно закона обобщенного склеивания получаем:

$$\left(D'_0 \vee \vec{X}_1\right)\left(D'_1 \vee \vec{X}_0\right) = D'_0\vec{X}_0 \vee D'_1\vec{X}_1 \vee D'_0D'_1 = D'_0\vec{X}_0 \vee D'_1\vec{X}_1. \quad (2.24)$$

Пусть (2.24) справедливо для $n$:

$$\overset{2^n-1}{\underset{i=0}{\&}}\left[\overset{2^n}{\underset{\forall\xi\neq 1}{\vee}}\left(\vec{X}_\xi\right) \vee (D'_i)\right] = \overset{2^n-1}{\underset{i=0}{\vee}}\left[\left(\vec{X}_i\right) \cdot (D'_i)\right]. \quad (2.25)$$

Покажем, что (2.25) будет справедливо для $n+1$:

$$\overset{2^{n+1}-1}{\underset{i=0}{\&}}\left[\overset{2^{n+1}}{\underset{\forall\xi\neq 1}{\vee}}\left(\vec{X}_\xi\right) \vee (D'_i)\right] = \overset{2^{n+1}-1}{\underset{i=0}{\vee}}\left[\left(\vec{X}_i\right) \cdot (D'_i)\right]. \quad (2.26)$$

Увеличение $n+1$ эквивалентно удвоению членов левой части (2.26):

$$\overset{2^{n+1}-1}{\underset{i=0}{\&}}\left[\overset{2^{n+1}}{\underset{\forall\xi\neq 1}{\vee}}\left(\vec{X}_\xi\right) \vee (D'_i)\right] = \left(\overset{2^n-1}{\underset{i=0}{\&}}\left[\overset{2^n-1}{\underset{\forall\xi\neq 1}{\vee}}\left(\vec{X}_\xi\right) \vee (D'_i)\right]\right) \wedge$$

$$\wedge\left(\overset{2^{n+1}-1}{\underset{i=0}{\&}}\left[\overset{2^{n+1}-1}{\underset{\forall\xi\neq 1}{\vee}}\left(\vec{X}_\xi\right) \vee (D'_i)\right]\right) \quad (2.27)$$

Поскольку

$$\forall(\xi \neq \pi)[(\vec{X}_\xi) \cdot (\vec{X}_\pi)] = 0, \quad (2.28)$$

Задача сводится к случаю (2.23), то есть

$$\left(\overset{2^n-1}{\underset{i=0}{\&}}\left[\overset{2^n-1}{\underset{\forall\xi\neq 1}{\vee}}\left(\vec{X}_\xi\right) \vee (D'_i)\right]\right) \wedge \left(\overset{2^{n+1}-1}{\underset{i=2^n}{\&}}\left[\overset{2^{n+1}-1}{\underset{\forall\xi\neq 1}{\vee}}\left(\vec{X}_\xi\right) \vee (D'_i)\right]\right) = \left(\overset{2^n-1}{\underset{i=0}{\vee}}\left[\left(\vec{X}_i\right) \cdot (D'_i)\right]\right) \wedge$$

$$\left(\overset{2^{n+1}-1}{\underset{i=2^n}{\vee}}\left[\left(\vec{X}_i\right) \cdot (D'_i)\right]\right) = \overset{2^{n+1}-1}{\underset{i=0}{\vee}}\left[\left(\vec{X}_i\right) \cdot (D'_i)\right]. \quad (2.29)$$

Поэтому в предлагаемой структуре возможен только нулевой спейсер.

Очевидно, что при $X_j = X_j^{'} = 0$ все члены

$$\left(\overset{2n}{\underset{j=1}{\&}} X_j^{\sigma(i,j)} \cdot D'_i\right) \quad (2.30)$$

попарно ортогональны и в основном и в двойственном каналах, поэтому получаем:

$$\begin{cases} F_{out.l.st} = \bigvee_{i=0}^{2^n-1} \left( \&_{j=1}^{2n} X_j^{\sigma(i-1,j)} \cdot D_i \right) \vee \left( \&_{j=1}^{n} X_j X_j' \right)', \\ F'_{out.l.st} = \bigvee_{i=0}^{2^n-1} \left( \&_{j=1}^{2n} X_j^{\sigma(i-1,j)} \cdot D'_i \right) \vee \left( \&_{j=1}^{n} X_j X_j' \right)' , l = 1, m \end{cases} \quad (2.31)$$

Таким образом, теорема доказана.

Рассмотрим выражение (2.18) для *n*=1, это элементарный мультиплексор 2-1 вида [77]:

$$Fout.l.st(X) = 0(X') \vee 1(X), \quad (2.32)$$

где $D_0 = 0; D_1 = 1$.

Вводим спейсер:

$$Fout.l.st(X) = 0(X') \vee 1(X) \vee (X'X)' \quad (2.33)$$

При введении двойственного канала реализуется универсальный элемент для ССС – LUT-ST, с учетом индикатора окончания переходных процессов в каналах получаем выражением (2.34):

$$\begin{cases} Fout.l.st(X) = 0(X') \vee 1(X) \vee (X'X)'; \\ F'out.l.st(X) = 0'(X') \vee 1'(X) \vee (X'X)'; \\ I = (Fout.l.st \cdot F'out.l.st)' \end{cases} \quad (2.34)$$

Где *Fout.l.st (X)* – основной канал, *F'out.l.st (X)* – двойственный канал, *I* – индикатор завершения переходного процесса в каналах.

Элемент 1-LUT-ST с двойственным каналом и индикатором [78-80] изображён на рисунке.2.5.

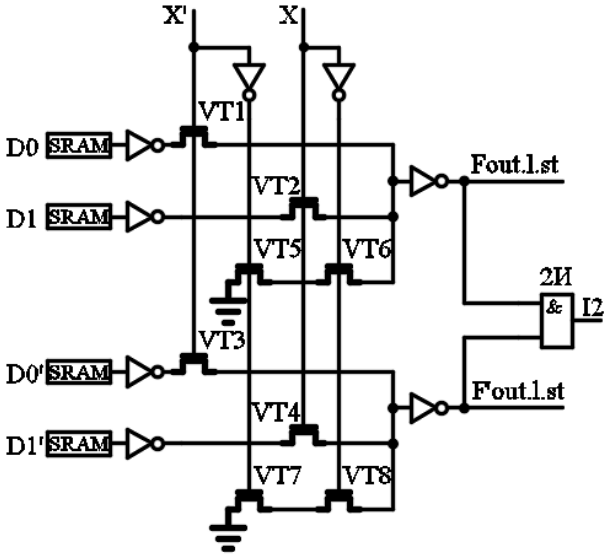



Рисунок 2.5 –Элемент 1-LUT-ST с двойственным каналом и индикатором.

Дерево передающих транзисторов VT1-VT2 и VT5-VT6 реализует прямой канал, а дерево транзисторов VT3-VT4 и VT7-VT8 реализует двойственный канал. Кроме этого транзисторы VT5-VT8 реализуют фазу спейсера. Инверторы на входах и выхода каналов предназначены для восстановления уровня сигналов. Элемент 2И реализует индикатор выходов каналов и фиксирует окончание переходных процессов.

Рассмотрим выражение (2.18) для $n$=2, это мультиплексор 4-1 вида:

$$F(X1, X2) = 0[(X2)'(X1)'] \vee 1[(X2)'(X1)] \vee 2[(X2)(X1)'] \vee 3[(X2)(X1)] \quad (2.35)$$

В таком случае имеем множество векторов X:

$$\vec{X} = \left\{\vec{X_0}, \vec{X_1}, \vec{X_2}, \vec{X_3}\right\} : a\vec{X_0} \vee b\vec{X_1} \vee c\vec{X_2} \vee d\vec{X_3}. \quad (2.36)$$

Продемонстрируем справедливость (2.19):

$$\overline{a\vec{X}_0 \vee b\vec{X}_{1,} \vee c\vec{X_2} \vee d\vec{X_3}} = (\overline{a} \vee \overline{\vec{X_0}})(\overline{b} \vee \overline{\vec{X_1}})(\overline{c} \vee \overline{\vec{X_2}})(\overline{d} \vee \overline{\vec{X_3}}). \quad (2.37)$$

Действительно:

$$(\overline{a} \vee \vec{X_1} \vee \vec{X_2} \vee \vec{X_3})(\overline{b} \vee \vec{X_0} \vee \vec{X_2} \vee \vec{X_3})(\overline{c} \vee \vec{X_0} \vee \vec{X_1} \vee \vec{X_3})(\overline{d} \vee \vec{X_0} \vee \vec{X_1} \vee \vec{X_2}). \quad (2.38)$$

Применяя закон дистрибутивности:

$$[(\overline{a} \vee \vec{X_1})(\overline{b} \vee \vec{X_0}) \vee (\vec{X_2} \vee \vec{X_3})][(\overline{c} \vee \vec{X_3})(\overline{d} \vee \vec{X_2}) \vee (\vec{X_0} \vee \vec{X_1})]. \quad (2.39)$$

Следовательно,

$$[(\overline{a}\overline{b} \vee \overline{a}\vec{X_0} \vee \overline{b}\vec{X_1} \vee \vec{X_0}\vec{X_1}) \vee (\vec{X_2} \vee \vec{X_3})][(\overline{c}\overline{d} \vee \overline{c}\vec{X_2} \vee \overline{d}\vec{X_3} \vee \vec{X_2}\vec{X_3}) \vee (\vec{X_{0,}} \vee \vec{X_1})]. \quad (2.40)$$

Далее, применяя закон обобщённого склеивания, в связи с ортогональностью векторов с разными номерами получим:

$$[(\overline{a}\vec{X_0} \vee \overline{b}\vec{X_1}) \vee (\vec{X_2} \vee \vec{X_3})][(\overline{c}\vec{X_2} \vee \overline{d}\vec{X_3}) \vee (\vec{X_0} \vee \vec{X_1})]. \quad (2.41)$$

Следовательно,

$$\overline{a}\vec{X_0}\vec{X_0} \vee \overline{b}\vec{X_1}\vec{X_1} \vee \overline{c}\vec{X_2}\vec{X_2} \vee \overline{d}\vec{X_3}\vec{X_3}. \quad (2.42)$$

Окончательно получим:

$$\overline{a}\vec{X_0} \vee \overline{b}\vec{X_1} \vee \overline{c}\vec{X_2} \vee \overline{d}\vec{X_3}. \quad (2.43)$$

Таким образом, элемент 2-LUT-ST описывается выражением с учетом индикатора окончания переходных процессов в каналах схемы:

$$\begin{cases} F(X1, X2) = 0[(X2)'(X1)'] \vee 1[(X2)'(X1)] \vee 2[(X2)(X1)'] \vee \\ \qquad \vee 3[(X2)(X1)] \vee [(X2)(X2)'(X1)(X1)']'; \\ F'(X1', X2') = 0'[(X2)'(X1)'] \vee 1'[(X2)'(X1)] \vee 2'[(X2)(X1)'] \vee \\ \qquad \vee 3'[(X2)(X1)] \vee [(X2)(X2)'(X1)(X1)']'; \\ I = (F \cdot F')' \end{cases} \quad (2.44)$$

Предлагаемая схема 2-LUT-ST для *n*=2, изображена на рисунке.2.6.

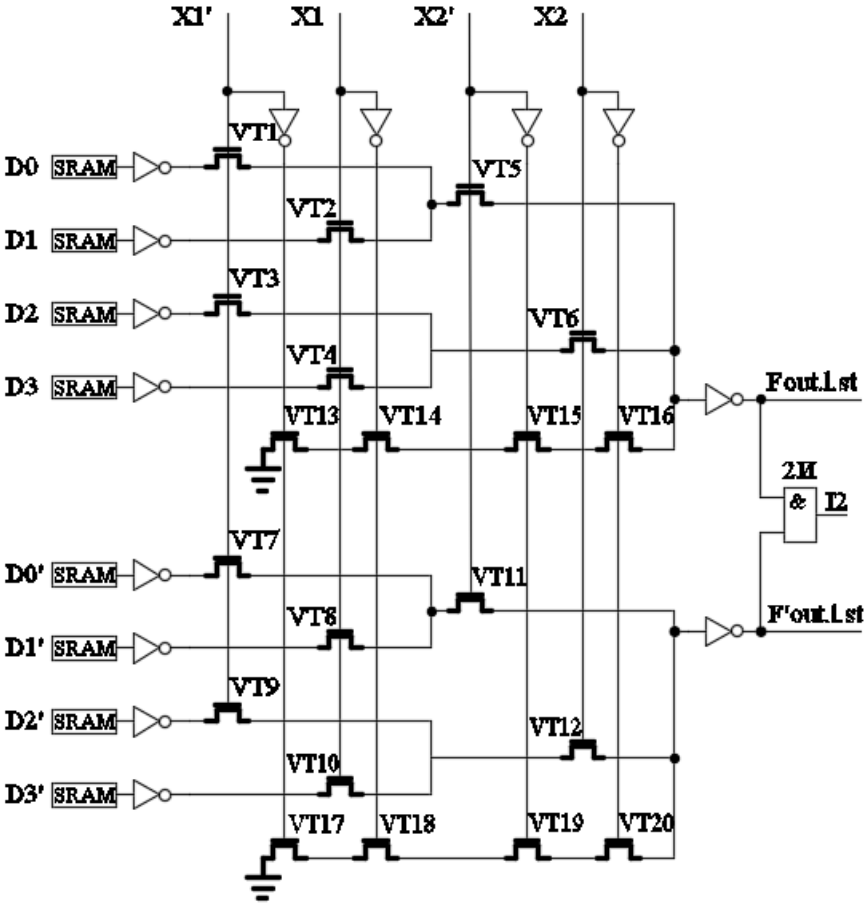



Рисунок 2.6 – Элемент 2-LUT-ST с двойственным каналом и индикатором.

Дерево передающих транзисторов VT1-VT6 и VT13-VT16 реализует прямой канал, а дерево транзисторов VT7-VT12 и VT17-VT20 реализует двойственный канал. Кроме этого, транзисторы VT13-VT20 реализуют фазу спейсера. Инверторы на входах и выходах каналов предназначены для восстановления уровня сигналов. Элемент 2И реализует индикатор выходов каналов и фиксирует окончание переходных процессов. На устройство получен патент РФ [81].

В элементе 2-LUT-ST (рисунок 2.6) цепочки спейсера состоят из четырех последовательно соединенных транзисторов n-типа, что является граничным значением в ограничениях Мида-Конвей [82], поэтому для числа переменных n>2 использовать такой подход в построении каналов не рекомендуется. Также на рисунках 2.5 и 2.6 показан только индикатор на выходе схемы, но в ССС необходимо индицировать не только выходы, но и входы [3], для этого предлагается использовать элементы 2ИЛИ-НЕ. Кроме этого индикатор служит для фиксирования окончания переходных процессов в схеме, для этой цели предлагается использовать Г-триггер (гистерезисный триггер, С-элемент Маллера). Выражение (2.7) описывает работу Г-триггера. Таким образом, выражения, описывающие работу элемента одной переменной с индикаторами и Г-триггерами:

$$\begin{cases} Fout.l.st = \left(\overline{D0}\left[\overline{X' \cdot Sp}\right] \vee \overline{D1}\left[\overline{X \cdot Sp}\right] \vee \left[\overline{X' \cdot Sp}\right] \cdot \left[\overline{X \cdot Sp}\right]\right)' \\ F'out.l.st = (\overline{D0'}\left[\overline{X' \cdot Sp}\right] \vee \overline{D1'}\left[\overline{X \cdot Sp}\right] \vee \left[\overline{X' \cdot Sp}\right] \cdot \left[\overline{X \cdot Sp}\right])' \\ I1 = \left(\left[\overline{X' \cdot Sp}\right] \vee \left[\overline{X \cdot Sp}\right]\right)' \\ I2 = Fout.l.st \cdot F'out.l.st \\ G(t+1) = (I1 \cdot I2 \vee G(t)(I1 \vee I2))' \end{cases} \quad (2.45)$$

Модель элемента LUT-ST одной переменной:

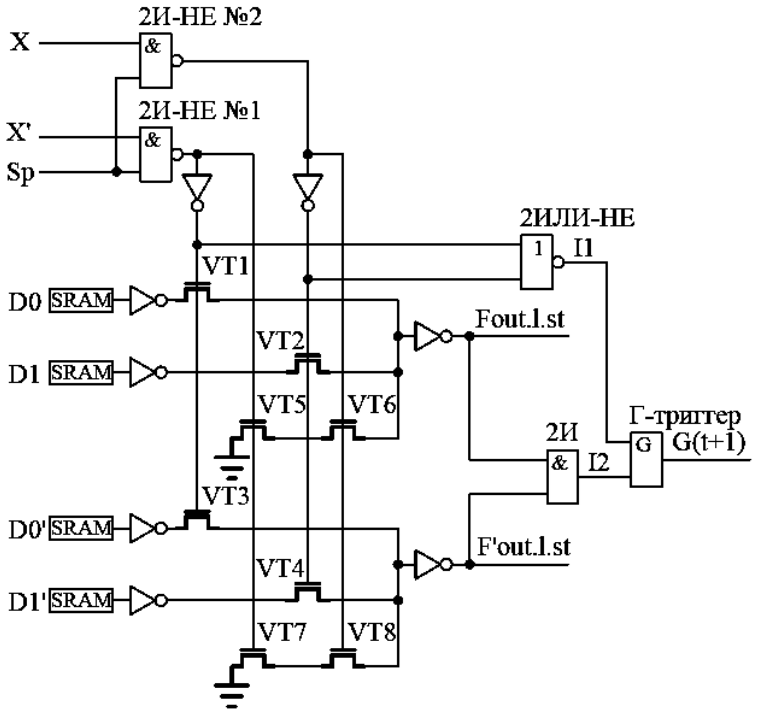



Рисунок 2.7 –Элемент 1-LUT-ST с разомкнутой обратной связью.

Блоки 2И №1 и 2И №2 – реализуют блоки входного набора, дерево передающих транзисторов VT1-VT2 и VT5-VT6 реализует прямой канал, а дерево транзисторов VT3-VT4 и VT7-VT8 реализует двойственный канал. Кроме этого, транзисторы VT5-VT8 реализуют фазу спейсера. Инверторы на входах и выходах каналов предназначены для восстановления уровня сигналов. Элементы 2ИЛИ-НЕ и 2И используются в качестве индикаторов. Для организации самосинхронизации используется обратная связь, не показанная на рисунке 2.7, с выхода G(t+1) Г-триггера на вход Sp блоков 2И-НЕ №1 и 2И-НЕ №2.

Ввиду ограничения Мида-Конвей для реализации элемента на n>2 предлагается использовать каскадирование элемента на одну переменную, при этом необходимо учитывать согласованность каналов. Это достигается путем инвертирования значений выходов каналов с предыдущего каскада.

Таким образом принцип каскадирования может быть пояснен таблицей:

Таблица 2.2 Каскадирование элементов LUT ST

Начало таблицы 2.2

| **i+1 каскад** | **i каскад** |
|---|---|
| (Fi+1.1)’ | (D0.i)’ |
| Xi+1’ | Xi’ |

Окончание таблицы 2.2

| **i+1 каскад** | **i каскад** |
|---|---|
| (Fi+1.2)’ | (D1.i)’ |
| Xi+1 | Xi |
| (F’i+1.1)’ | (D0’.i)’ |
| Xi+1’ | Xi’ |
| (F’i+1.2)’ | (D1’.i)’ |
| Xi+1 | Xi |

Тогда получим выражения, описывающие работу элемента LUT-ST двух переменных:

$$\begin{cases} F11 = \left(\overline{D0}\left[\overline{X1' \cdot Sp}\right] \vee \overline{D1}\left[\overline{X1 \cdot Sp}\right] \vee [\overline{X1' \cdot Sp}] \cdot [\overline{X1 \cdot Sp}]\right)' \\ F'11 = \left(\overline{D0'}\left[\overline{X1' \cdot Sp}\right] \vee \overline{D1'}\left[\overline{X1 \cdot Sp}\right] \vee [\overline{X1' \cdot Sp}] \cdot [\overline{X1 \cdot Sp}]\right)' \\ F12 = \left(\overline{D2}\left[\overline{X2' \cdot Sp}\right] \vee \overline{D3}\left[\overline{X2 \cdot Sp}\right] \vee [\overline{X2' \cdot Sp}] \cdot [\overline{X2 \cdot Sp}]\right)' \\ F'12 = \left(\overline{D2'}\left[\overline{X2' \cdot Sp}\right] \vee \overline{D3'}\left[\overline{X2 \cdot Sp}\right] \vee [\overline{X2' \cdot Sp}] \cdot [\overline{X2 \cdot Sp}]\right)' \\ I11 = \left(\left[\overline{X1' \cdot Sp}\right] \vee \left[\overline{X1 \cdot Sp}\right]\right)' \\ I12 = F11 \cdot F'11 \\ I13 = F12 \cdot F'12 \\ Fout.l.st = \left(\overline{F11}\left[\overline{X2' \cdot Sp}\right] \vee \overline{F12}\left[\overline{X2 \cdot Sp}\right] \vee [\overline{X2' \cdot Sp}] \cdot [\overline{X2 \cdot Sp}]\right)' \\ F'out.l.st = \left(\overline{F'11}\left[\overline{X2' \cdot Sp}\right] \vee \overline{F'12}\left[\overline{X2 \cdot Sp}\right] \vee [\overline{X2' \cdot Sp}] \cdot [\overline{X2 \cdot Sp}]\right)' \\ I21 = \left(\left[\overline{X2' \cdot Sp}\right] \vee \left[\overline{X2 \cdot Sp}\right]\right)' \\ I22 = Fout.l.st \cdot F'out.l.st \\ G1(t+1) = (I11 \cdot I12 \cdot I13 \vee G1(t)(I11 \vee I12 \vee I13))' \\ G2(t+1) = (I21 \cdot I22 \cdot G1(t) \vee G2(t)\left(I11 \vee I12 \vee G1(t)\right))' \end{cases} \quad (2.46)$$

где $F11$, $F12$, $F_{out.l.st}$ – выходы прямого канала первого и второго слоев, $F'11$, $F'12$, $F'_{out.l.st}$ – выходы двойственных каналов первого и второго слоев, $I11$, $I21$ – выходы индикаторов входов, $I12$, $I13$, $I22$ – выходы индикаторов выходов каналов первого и второго слоев, $G1(t+1)$, $G2(t+1)$ – выходы Г-триггеров, фиксирующих окончания переходных процессов в блоках схемы.

Модель элемента LUT-ST двух переменных показана на рисунке 2.8. Блоки 2И №1, 2И №2, 2И №3 и 2И №3 реализуют блоки входных наборов первого и второго каскадов. Деревья передающих транзисторов VT1-VT2, VT5-VT6 и VT9-VT10, VT13-VT14 реализуют прямой канал первого каскада, а деревья транзисторов VT3-VT4, VT7-VT8 и VT11-VT12, VT15-VT16 реализуют

двойственный канал первого каскада. Транзисторы VT17-VT18 и VT21-VT22 реализуют прямой канал второго каскада, транзисторы VT19-VT20 и VT23-VT24 реализуют двойственный канал второго каскада. Кроме этого, транзисторы VT5-VT8, VT13-VT16, VT21-VT24 реализуют фазу спейсера. Инверторы на выходах каналов предназначены для восстановления уровня сигналов. Инверторы, подключенные к истокам транзисторов VT17, VT18, VT19, VT20, предназначены для согласованности работы каскадов. Элементы 2ИЛИ-НЕ №1, 2ИЛИ-НЕ №2, 2И №1, 2И №2 и 2И №3 используются в качестве индикаторов. Для фиксирования окончания переходных процессов в каждом блоке схемы используются гистерезисные триггеры (Г-триггеры). Для организации самосинхронизации используется обратная связь, не показанная на рисунке 2.8, с выхода G2(t+1) Г-триггера №2 на входы Sp блоков 2И-НЕ №1, 2И-НЕ №2, 2И-НЕ №3, 2И-НЕ №4.

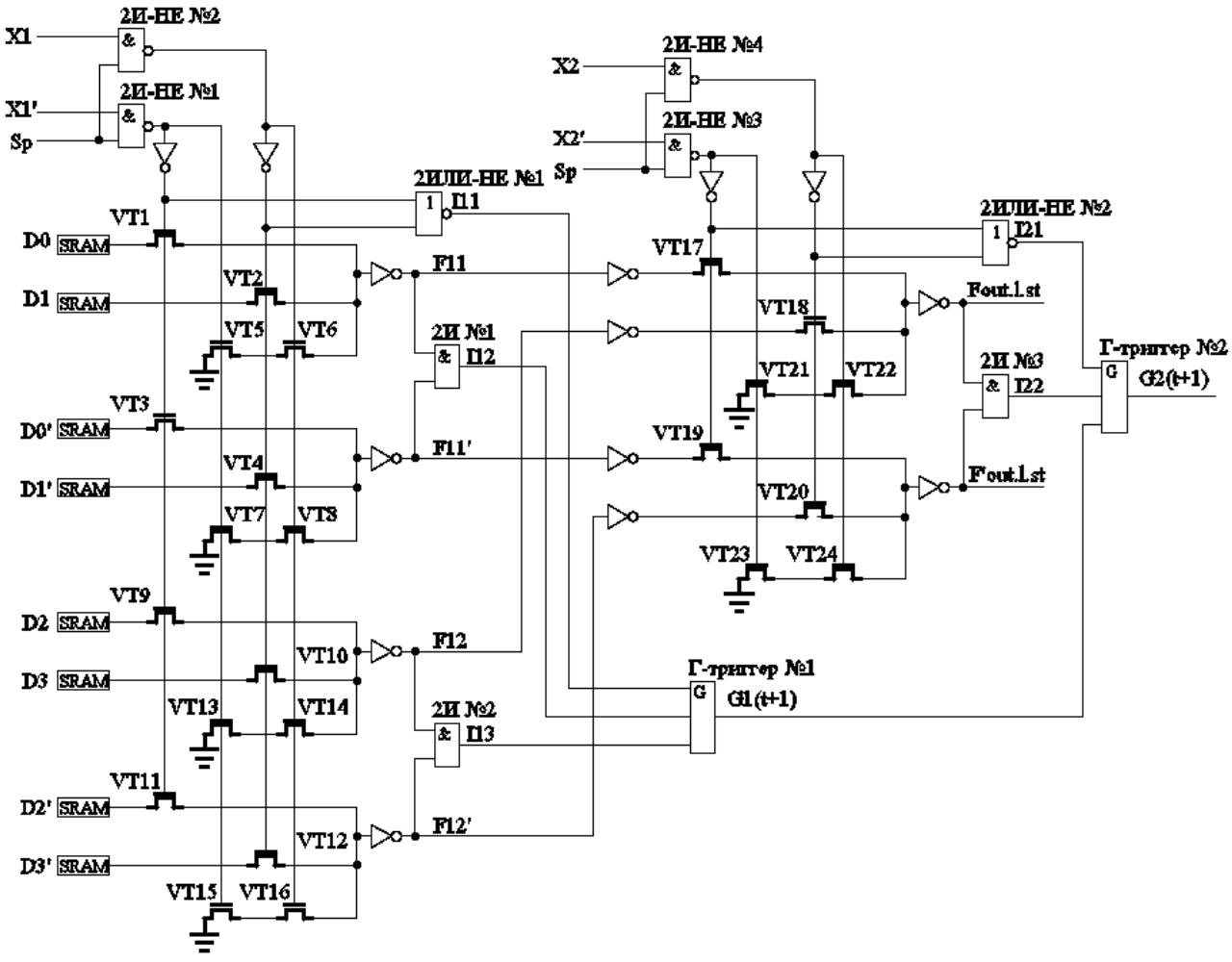



Рисунок 2.8 – Элемент 2-LUT-ST с разомкнутой обратной связью.

Для подтверждения, что исходный элемент адаптирован к условиям работы в СССС, выполним анализ полученной модели LUT-ST двух переменных на полумодулярность. Для этого опишем модель Маллера схемы на рисунке 2.8.

D0=0; D1=1; D2=1; D3=0; / Описываем конфигурирование элемента

X1=0; X2=0; / Задаем значения переменных

neD0=^D0; neD1=^D1; neD2=^D2; neD3=^D3;

neX1=^X1; neX2=^X2; Gr=0;

V1=^neX1*E; V2=^X1*E; V3=^neX2*E; V4=^X2*E;

SP=Gr(V1*V2); SP1=Vc(V3*V4);

/Описание работы каналов/

F11=^D0*V1^|D1*V2^|SP; neF11=^neD0*V1^|neD1*V2^|SP;

F12=^D2*V1^|D3*V2^|SP; neF12=^neD2*V1^|neD3*V2^|SP;

F=^F11*V3^|F12*V4^|SP1; neF=^neF11*V3^|neF12*V4^|SP1;

/Описание индикаторов схемы/

I11=^V1^|V2^; I12=^F11*neF11; I13=^F12*neF12;

G1t=^I11*I12^*I13^|G1ti(I11|I12^|I13^); G1ti=^G1t; /G-триггер

I21=V3^|V4^; I22=^F*neF;

G2t=^I21*I22*G1ti|G2ti(I21|I22|G1ti); G2ti=^G2t; E=G2ti; / Спейсер

$G2t / Начальные состояния схемы

Результат анализа схемы LUT-ST двух переменных показан на рисунке 2.9.

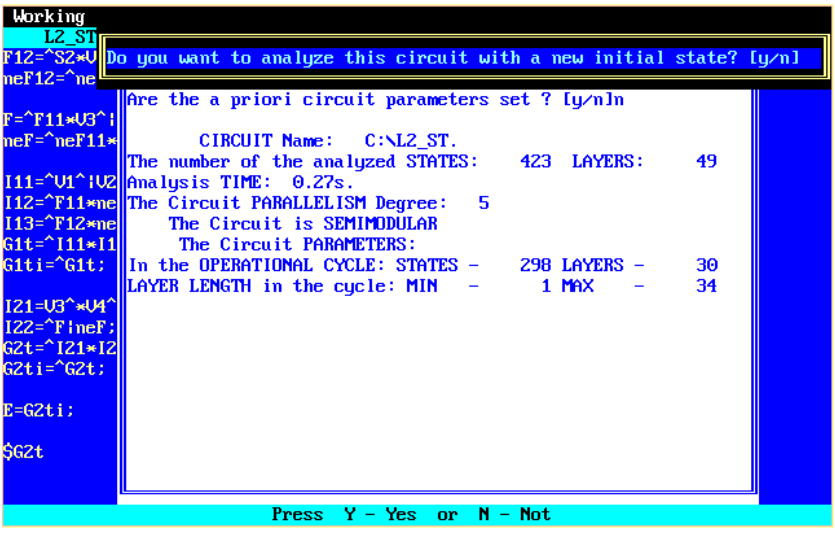



Рисунок 2.9. Результат анализа 2LUT-ST на полумодулярность в САПР Forcage.

На рисунке 2.9 видно, что схема является полумодулярной (semimodular), а значит, обладает двумя основными свойствами самосинхронных схем. Проверка моделей на реализацию других функций представлена в приложении №3.

## 2.3. Разработка метода реализации конфигурируемого самосинхронного генератора систем логических функций, заданных в СДНФ

Метод ориентирован на использование предложенного в работе [8] дешифратора DC LUT для ПЛИС типа FPGA.

Для реализации систем логических функций в СДНФ предлагается использовать реверс дерева передающих транзисторов [83-90]. Дешифрация входного набора описывается выражением:

$$F_i' = \underset{j=1}{\overset{2n}{\&}} X_j^{\sigma(i,j)} \cdot Z_{out}', i = 0, 2^n - 1 \quad (2.47)$$

Введём входной сигнал in вместо $Z_{out}$, получим:

$$F_{out.i}' = \underset{j=1}{\overset{n}{\&}} X_j^{\sigma(i,j)} \cdot in_i', i = 0, 2^n - 1 \quad (2.48)$$

При реализации (2.49) в виде дерева передающих транзисторов нарушается условие ортогональности сигналов, так как в случае не активации одного из передающих транзисторов вход одного из выходных инверторов получается «оборванным». Ортогональность обеспечивается в случае:

$$F_{out.i} = \underset{j=1}{\overset{n}{\&}} (X_j^{\sigma(i,j)} \cdot in_i' \vee X_j^{\sigma'(i,j)} \cdot in_i), i = 0, 2^n - 1. \quad (2.49)$$

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

$$F_{out.i} = \underset{j=1}{\overset{n}{\&}} (X_j^{\sigma(i,j)}) (in_i') \vee \underset{j=1}{\overset{n}{\&}} (X_j^{\sigma'(i,j)})(in_i), i = 0, 2^n - 1, \quad (2.50)$$

выражение (2.50) описывает обеспечение ортогональности по каждой переменной в каждой ветви дерева [8].

Существо метода заключается в модификации дерева передающих транзисторов путем введения дополнительной цепочки спейсера аналогично (2.17):

$$F_{out.d.st} = \underset{j=1}{\overset{n}{\&}} (X_j^{\sigma(i,j)} \cdot in_i' \vee X_j^{\sigma'(i,j)} \cdot in_i ) \vee ( \underset{\mu=1}{\overset{n}{\&}} X_\mu X_\mu')', i = 0, 2^n - 1. \quad (2.51)$$

Для реализации двойственного канала предполагается инверсия выражения

$$\underset{j=1}{\overset{n}{\&}} (X_j^{\sigma(i,j)} \cdot in_i' \vee X_j^{\sigma'(i,j)} \cdot in_i ). \quad (2.52)$$

Используя результат доказательства теоремы 2.2 с учётом двойственности логических операций, получим:

$$\left[ \underset{j=1}{\overset{n}{\&}} (X_j^{\sigma(i,j)} \cdot in_i' \vee X_j^{\sigma'(i,j)} \cdot in_i ) \right]' = \underset{j=1}{\overset{n}{\&}} (X_j^{\sigma(i,j)} \cdot in_i \vee X_j^{\sigma'(i,j)} \cdot in_i'). \quad (2.53)$$

Таким образом, получаем для предлагаемого DC LUT-ST с учетом индикатора выходов:

$$\begin{cases} F_{out.d.st} = \underset{j=1}{\overset{n}{\&}} \left(X_j^{\sigma(i,j)} \cdot in_i' \vee X_j^{\sigma'(i,j)} \cdot in_i\right) \vee \left( \underset{\mu=1}{\overset{n}{\&}} X_\mu X_\mu' \right)' \\ F'_{out.d.st} = \underset{j=1}{\overset{n}{\&}} (X_j^{\sigma(i,j)} \cdot in_i \vee X_j^{\sigma'(i,j)} \cdot in_i') \vee ( \underset{\mu=1}{\overset{n}{\&}} X_\mu X_\mu')', i = 0, 2^n - 1 \\ I = [(F_{out.i.st})(F'_{out.i.st})]' \end{cases} \quad (2.54)$$

Для одноразрядного 1DC LUT-ST, получим выражения, описывающие работу элемента с учетом индикаторов и Г-триггеров:

$$\begin{cases} F0_{out.d.st} = [\overline{\overline{X' \cdot Sp}} \cdot Ground \vee \overline{\overline{X \cdot Sp}} \cdot Vcc \vee \overline{X' \cdot Sp} \cdot \overline{X \cdot Sp}]' \\ F1_{out.d.st} = [\overline{\overline{X \cdot Sp}} \cdot Ground \vee \overline{\overline{X' \cdot Sp}} \cdot Vcc \vee \overline{X' \cdot Sp} \cdot \overline{X \cdot Sp}]' \\ F0'_{out.d.st} = [\overline{\overline{X' \cdot Sp}} \cdot Vcc \vee \overline{\overline{X \cdot Sp}} \cdot Ground \vee \overline{X' \cdot Sp} \cdot \overline{X \cdot Sp}]' \\ F1'_{out.d.st} = [\overline{\overline{X \cdot Sp}} \cdot Vcc \vee \overline{\overline{X' \cdot Sp}} \cdot Ground \vee \overline{X' \cdot Sp} \cdot \overline{X \cdot Sp}]' \\ I1 = \overline{\overline{X \cdot Sp}} \vee \overline{\overline{X' \cdot Sp}} \\ I2 = F0_{out.d.st} \cdot F0'_{out.d.st} \\ I3 = F1_{out.d.st} \cdot F1'_{out.d.st} \\ G(t+1) = [I1 \cdot I2 \cdot I3 \vee G(t)(I1 \vee I2 \vee I3)]' \end{cases} \quad (2.55)$$

Предлагаемый 1DC LUT-ST с G-триггером и индикатором входов изображён на рисунке 2.10.

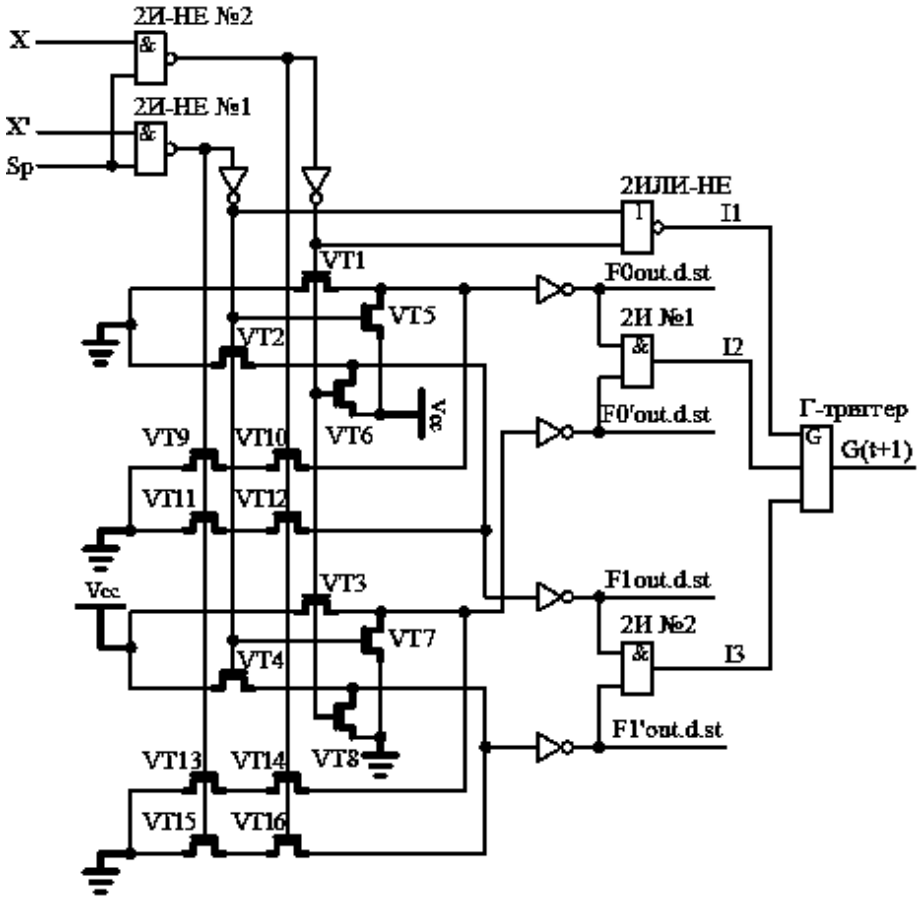



Рисунок 2.10 – Элемент 1DC-LUT-ST с разомкнутой обратной связью.

Блоки 2И-НЕ №1 и 2И-НЕ №2 – реализуют блоки входного набора, дерево передающих транзисторов VT1-VT2, VT5-VT6 и VT9-VT12 реализует прямой канал, а дерево транзисторов VT3-VT4, VT5-VT6 и VT13-VT16 реализует двойственный канал. Кроме этого, транзисторы VT9-VT16 реализуют фазу спейсера по каждой из ветви дерева. Инверторы на выходах предназначены для восстановления уровня сигналов. Элементы 2ИЛИ-НЕ, 2И №1 и 2И №2 используются в качестве индикаторов. Для организации самосинхронизации используется обратная связь, не показанная на рисунке 2.10, с выхода G(t+1) Г-триггера на вход Sp блоков 2И-НЕ №1 и 2И-НЕ №2.

Так же, как в предыдущем методе, требуется учитывать ограничения в количестве последовательно соединенных транзисторов для построения элементов на n>2, поэтому для реализации элемента на n>2 предлагается использовать каскадирование элемента на одну переменную. В отличии от предыдущего метода, использования только инвертирования выходов каналов

предыдущего каскада недостаточно для сохранения дешифрации выходных сигналов. Поэтому требуется учитывать подключение транзисторов ортогональности так, чтобы в каскаде со старшей переменной подключения были попарно инверсны подключениям каскада с младшими переменными, причем во всех каскадах с младшими переменными эти подключения должны быть одинаковыми.

Таким образом, принцип каскадирования может быть пояснен таблицей 2.3:

Таблица 2.3 Каскадирование элементов DC LUT ST:

| **i+2 каскад** | **i+1 каскад** | **i каскад** |
|---|---|---|
| (Fi+1.1)’ | (Fi.1)’ | Ground |
| Xi-1’ | Xi’ | Xi+1’ |
| Ground | Ground | Vcc |
| Xi-1 | Xi | Xi+1 |
| (F’i+1.1)’ | (F’i.1)’ | Vcc |
| Xi-1’ | Xi’ | Xi+1’ |
| Vcc | Vcc | Ground |
| Xi-1 | Xi | Xi+1 |

На устройство получен патент РФ [91].

Построим таким образом элемент DC-LUT-ST для n=2. Тогда выражения, описывающие работу элемента:

$$\begin{cases} F01_{out.d.st} = [\overline{\overline{X2'\cdot Sp}\cdot Ground} \vee \overline{\overline{X2\cdot Sp}\cdot Vcc} \vee \overline{X2'\cdot Sp}\cdot\overline{X2\cdot Sp}]' \\ F11_{out.d.st} = [\overline{\overline{X2\cdot Sp}\cdot Ground} \vee \overline{\overline{X2'\cdot Sp}\cdot Vcc} \vee \overline{X2'\cdot Sp}\cdot\overline{X2\cdot Sp}]' \\ F01'_{out.d.st} = [\overline{\overline{X2'\cdot Sp}\cdot Vcc} \vee \overline{\overline{X2\cdot Sp}\cdot Ground} \vee \overline{X2'\cdot Sp}\cdot\overline{X2\cdot Sp}]' \\ F11'_{out.d.st} = [\overline{\overline{X2\cdot Sp}\cdot Vcc} \vee \overline{\overline{X2'\cdot Sp}\cdot Ground} \vee \overline{X2'\cdot Sp}\cdot\overline{X2\cdot Sp}]' \\ F0_{out.d.st} = [\overline{\overline{X1'\cdot Sp}\cdot\overline{F01_{out.d.st}}} \vee \overline{\overline{X1\cdot Sp}\cdot Ground} \vee \overline{X1'\cdot Sp}\cdot\overline{X1\cdot Sp}]' \\ F1_{out.d.st} = [\overline{\overline{X1\cdot Sp}\cdot\overline{F01_{out.d.st}}} \vee \overline{\overline{X1'\cdot Sp}\cdot Ground} \vee \overline{X1'\cdot Sp}\cdot\overline{X1\cdot Sp}]' \\ F2_{out.d.st} = [\overline{\overline{X1'\cdot Sp}\cdot\overline{F11_{out.d.st}}} \vee \overline{\overline{X1\cdot Sp}\cdot Ground} \vee \overline{X1'\cdot Sp}\cdot\overline{X1\cdot Sp}]' \\ F3_{out.d.st} = [\overline{\overline{X1\cdot Sp}\cdot\overline{F11_{out.d.st}}} \vee \overline{\overline{X1'\cdot Sp}\cdot Ground} \vee \overline{X1'\cdot Sp}\cdot\overline{X1\cdot Sp}]' \\ F0'_{out.d.st} = [\overline{\overline{X1'\cdot Sp}\cdot\overline{F01'_{out.d.st}}} \vee \overline{\overline{X1\cdot Sp}\cdot Vcc} \vee \overline{X1'\cdot Sp}\cdot\overline{X1\cdot Sp}]' \\ F1'_{out.d.st} = [\overline{\overline{X1\cdot Sp}\cdot\overline{F01'_{out.d.st}}} \vee \overline{\overline{X1'\cdot Sp}\cdot Vcc} \vee \overline{X1'\cdot Sp}\cdot\overline{X1\cdot Sp}]' \\ F2'_{out.d.st} = [\overline{\overline{X1'\cdot Sp}\cdot\overline{F11'_{out.d.st}}} \vee \overline{\overline{X1\cdot Sp}\cdot Vcc} \vee \overline{X1'\cdot Sp}\cdot\overline{X1\cdot Sp}]' \\ F3'_{out.d.st} = [\overline{\overline{X1\cdot Sp}\cdot\overline{F11'_{out.d.st}}} \vee \overline{\overline{X1'\cdot Sp}\cdot Vcc} \vee \overline{X1'\cdot Sp}\cdot\overline{X1\cdot Sp}]' \\ I11 = \overline{\overline{X2\cdot Sp}} \vee \overline{\overline{X2'\cdot Sp}} \\ I12 = F0_{out.d.st}\cdot F0'_{out.d.st} \end{cases} \quad (2.56)$$

$$\begin{cases} I13 = F1_{out.d.st} \cdot F1'_{out.d.st} \\ I21 = \overline{\overline{X1 \cdot Sp}} \vee \overline{\overline{X1' \cdot Sp}} \\ I22 = F0_{out.d.st} \cdot F0'_{out.d.st} \\ I23 = F1_{out.d.st} \cdot F1'_{out.d.st} \\ I24 = F2_{out.d.st} \cdot F2'_{out.d.st} \\ I25 = F3_{out.d.st} \cdot F3'_{out.d.st} \\ G1(t+1) = [I11 \cdot I12 \cdot I13 \vee G1(t)(I11 \vee I12 \vee I13)]' \\ G2(t+1) = [I21 \cdot I22 \cdot I23 \vee G2(t)(I21 \vee I22 \vee I23)]' \\ G3(t+1) = [I24 \cdot I25 \cdot G2(t) \vee G3(t)(I24 \vee I25 \vee G2(t))]' \\ G4(t+1) = [G1(t) \cdot G3(t) \vee G4(t)(G1(t) \vee G3(t))]' \end{cases} \quad (2.57)$$

где $F01$, $F11$, $F0$, $F1$, $F2$, $F3$ –выходы прямого канала первого и второго слоев, $F01'$, $F11'$, $F0'$, $F1'$, $F2'$, $F3'$ – выходы двойственных каналов первого и второго слоев, $I11$, $I21$ – выходы индикаторов входов, $I12$, $I13$, $I22$, $I23$, $I24$, $I25$ – выходы индикаторов выходов каналов первого и второго слоев, $G1(t+1)$, $G2(t+1)$, $G3(t+1)$, $G4(t+1)$ – выходы Г-триггеров, фиксирующих окончания переходных процессов в блоках схемы.

Модель элемента 2DC-LUT-ST показана на рисунке 2.11.

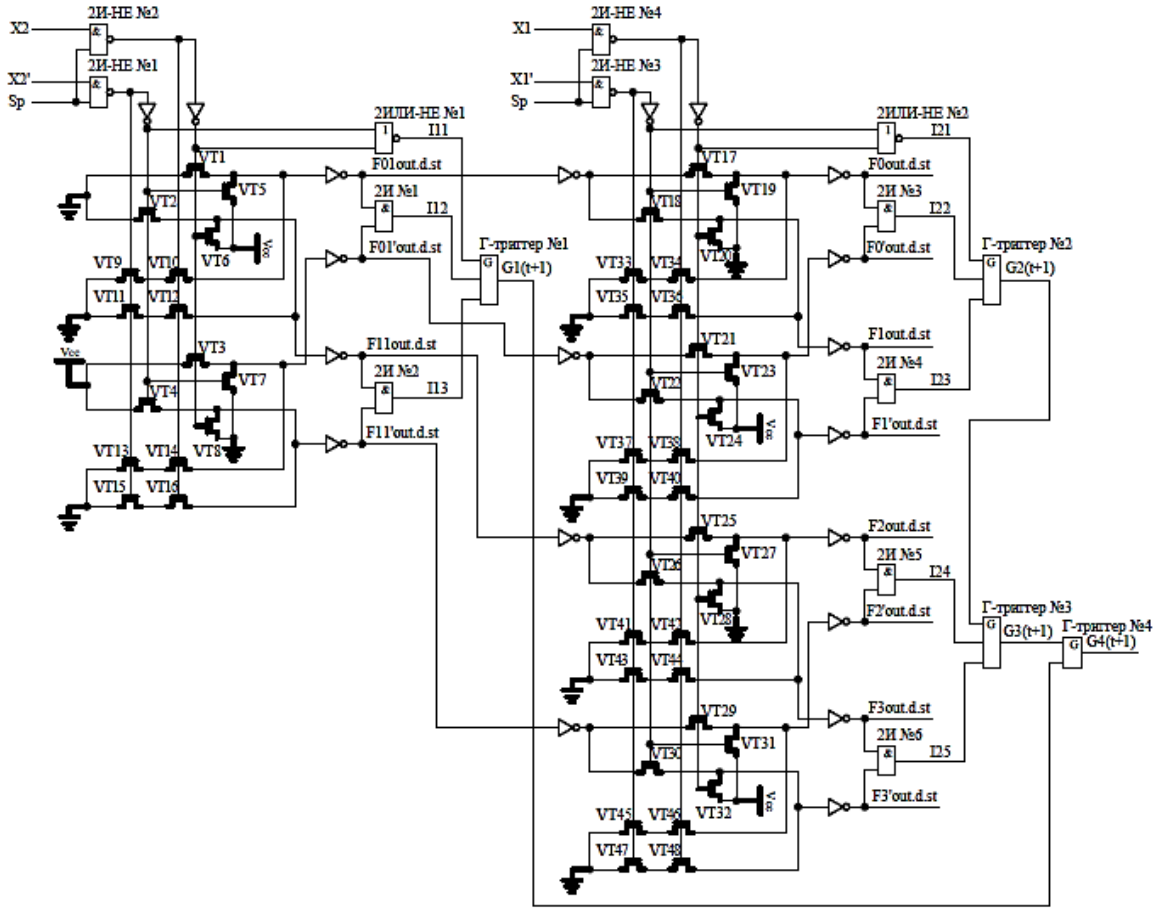



Рисунок 2.11 Элемент 2DC-LUT-ST с разомкнутой обратной связью.

Блоки 2И-НЕ №1, 2И-НЕ №2, 2И-НЕ №3 и 2И-НЕ №4 реализуют блоки входного набора первого и второго каскадов. Дерево передающих транзисторов VT1-VT2, VT5-VT6 и VT9-VT12 реализует прямой канал первого каскада, а дерево транзисторов VT3-VT4, VT5-VT6 и VT13-VT16 реализует двойственный канал первого каскада. Деревья передающих транзисторов VT17-VT20 и VT33-VT36, VT25-VT28 и VT41-VT44 реализуют прямые каналы второго каскада. Деревья передающих транзисторов VT21-VT24 и VT37-VT40, VT29-VT32 и VT45-VT48 реализуют двойственные каналы второго каскада. Кроме этого, транзисторы VT9-VT16 и VT33-VT48 реализуют фазу спейсера по каждой из ветви деревьев. Транзисторы VT5-VT8, VT19-VT20, VT23-VT24, VT27-VT28 и VT31-VT32 используются в качестве транзисторов ортогональности, причем подключение стоков выполнено с учетом согласованности работы каналов при каскадировании. Инверторы на выходах каналов предназначены для восстановления уровня сигналов. Инверторы, подключенные к истокам транзисторов VT17-VT18, VT21-VT22, VT25-VT26 и VT29-VT30 второго каскада, предназначены для согласованности работы каскадов. Элементы 2ИЛИ-НЕ №1, 2ИЛИ-НЕ №2, 2И №1, 2И №2, 2И №3, 2И №4, 2И №5 и 2И №6 используются в качестве индикаторов. Гистерезисные триггеры Г-триггер №1, Г-триггер №2, Г-триггер №3 и Г-триггер №4 предназначены для фиксирования окончания переходных процессов в каждом блоке схемы. Для организации самосинхронизации используется обратная связь, не показанная на рисунке 2.11, с выхода G4(t+1) Г-триггера на входы Sp блоков 2И-НЕ №1, 2И-НЕ №2, 2И-НЕ №3 и 2И-НЕ №4.

На основании полученных моделей можно получить элемент на любое число переменных с соблюдением согласованности слоев и подключением транзисторов ортогональности к шинам питания или земли.

Выполним проверку разработанной модели, показанной на рисунке 2.11, на полумодулярность. Для этого опишем все блоки схемы:

Gr=0; Vc=1; X1=1; X2=0; neX1=^X1;neX2=^X2; /Описываем все постоянные схемы

V1=^neX1*E; V2=^X1*E; V3=^neX2*E; V4=^X2*E; / Описываем блоки входного набора

SP1=Gr(V3*V4); SP=Gr(V1*V2); / Цепочки спейсера

/Работа прямых и двойственных каналов первого каскада

F0=Gr*V3^|Vc*V4^|SP1; neF0=Vc*V3^|Gr*V4^|SP1;

F1=Gr*V4^|Vc*V3^|SP1; neF1=Vc*V4^|Gr*V3^|SP1;

/Работа прямых и двойственных каналов второго каскада

F01=^F0*V1^|Gr*V2^|SP; neF01=^neF0*V1^|Vc*V2^|SP;

F11=^F0*V2^|Gr*V1^|SP; neF11=^neF0*V2^|Vc*V1^|SP;

F02=^F1*V1^|Gr*V2^|SP; neF02=^neF1*V1^|Vc*V2^|SP;

F12=^F1*V2^|Gr*V1^|SP; neF12=^neF1*V2^|Vc*V1^|SP;

/Работа индикаторов схемы

I1=^V3^|V4^; I2=F0^*neF0^; I3=F1^*neF1^; G1=^I1*I2*I3|G1i(I1|I2|I3);

G1i=^G1; I11=^V1^|V2^; I12=F01*neF01; I13=F11*neF11; I14=F02*neF02;

I15=F12*neF12; G11=^I11*I12*I13|G11i(I11|I12|I13); G11i=^G11;

G12=^I14*I15*G11i|G12i(I14|I15|G11i); G12i=^G12;

G2=^G12i*G1i|G2i(G12i|G1i); G2i=^G2;

E=G2i; /Обратная связь

$G2 / Начальное состояние схемы

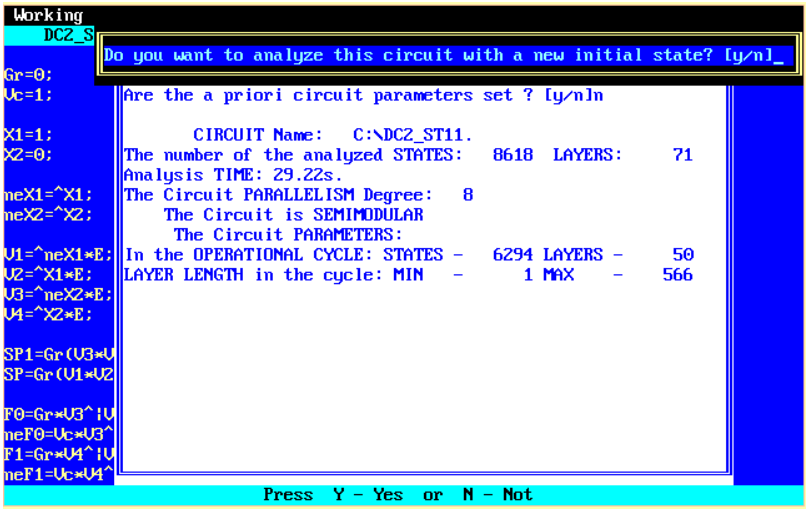



Рисунок 2.12 Результат проверки элемента 2DC-LUT-ST на полумоделярность.

На рисунке 2.12 показан положительный результат анализа схемы на полумодулярность (The Circuit is SEMIMODULAR).

Для реализации m функций от одинаковых переменных предлагается использовать специальные блоки настройки. Блоки m логических функций также должны иметь двойственный канал и индикацию каждого элемента. Таким образом, для программирования значений *m* логических функций предлагается:

$$\begin{cases} Z = \bigvee_{i=1}^{2^n-1} (F_{out.d.st} \vee D_i \vee \overline{Sp}) \\ Z' = \bigvee_{i=1}^{2^n-1} (F'_{out.d.st} \cdot D'_i \vee \overline{Sp}) \end{cases} \quad (2.58)$$

где $D$ – настройка вхождения конституент $i$ в данную функцию из $m$ функций системы.

Модель блока настройки с двойственным каналом, индикаторами и Г-триггером для двух выходов каналов показана на рисунке 2.12. Для реализации прямого канала используются элементы 3ИЛИ №1, 3ИЛИ №2 и элемент 2И-НЕ, для реализации двойственного канала используются элементы 2И-2ИЛИ №1, 2И-2ИЛИ №2 и элемент 2ИЛИ-НЕ №1. Остальные элементы используются для индицирования блоков схемы. Для организации самосинхронизации используется обратная связь, не показанная на рисунке 2.12, с выхода G(t+1) Г-триггера на вход Sp инвертора.

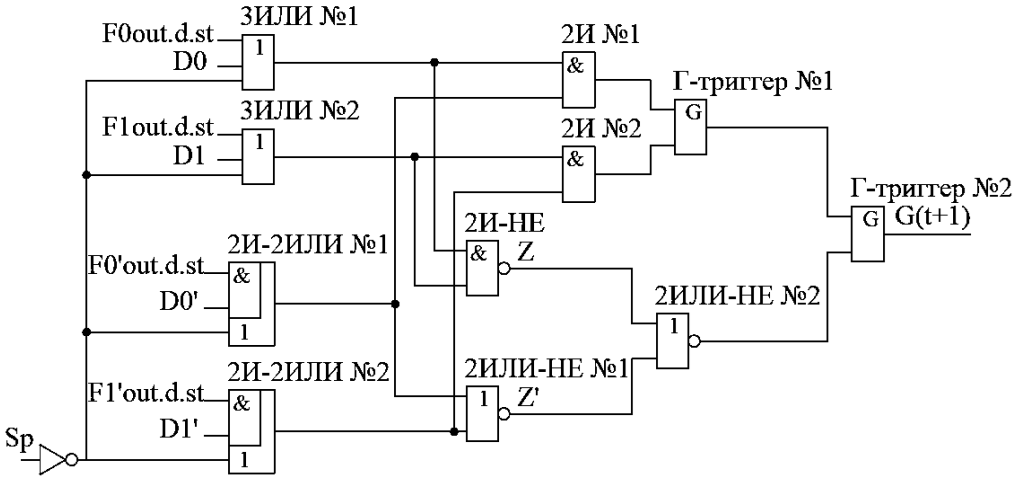



Рисунок 2.13 Блок дизъюнкции настройки функции.

Для реализации прямого канала используются элементы 3ИЛИ №1, 3ИЛИ №2 и элемент 2И-НЕ, для реализации двойственного канала используются

элементы 2И-2ИЛИ №1, 2И-2ИЛИ №2 и элемент 2ИЛИ-НЕ №1. Остальные элементы используются для индицирования блоков схемы. Для организации самосинхронизации используется обратная связь, не показанная на рисунке 2.13, с выхода G(t+1) Г-триггера на вход Sp инвертора.

Так как число выходов элементов DC LUT ST имеет зависимость $2^n$, то разработку блока настройки на число выход $F_{out.d.st}$>2 предлагается реализовывать каскадированием блока настройки на два выхода, причем входов у блока настройки должно быть четное количество. Таким образом, работу блока на четыре выхода каналов, без учета индикаторов и Г-триггеров, можно описать:

$$\begin{cases} Z = \overline{(\overline{F0_{out.d.st} \vee D_0 \vee \overline{Sp}}) \cdot (\overline{F1_{out.d.st} \vee D_1 \vee \overline{Sp}})} \vee \overline{F2_{out.d.st} \vee D_2 \vee \overline{Sp}} \wedge \\ \quad \overline{\wedge F3_{out.d.st} \vee D_3 \vee \overline{Sp}} \\ Z' = \overline{(\overline{F0'_{out.d.st} \cdot D_0' \vee \overline{Sp}}) \vee (\overline{F1'_{out.d.st} \cdot D_1' \vee \overline{Sp}})} \cdot \overline{F2'_{out.d.st} \cdot D_2' \vee \overline{Sp} \vee} \\ \quad \overline{\vee F3'_{out.d.st} \cdot D_3' \vee \overline{Sp}} \end{cases} \quad (2.59)$$

Модель блока настройки с двойственным каналом, индикаторами и Г-триггером для четырех выходов каналов показана на рисунке 2.14:

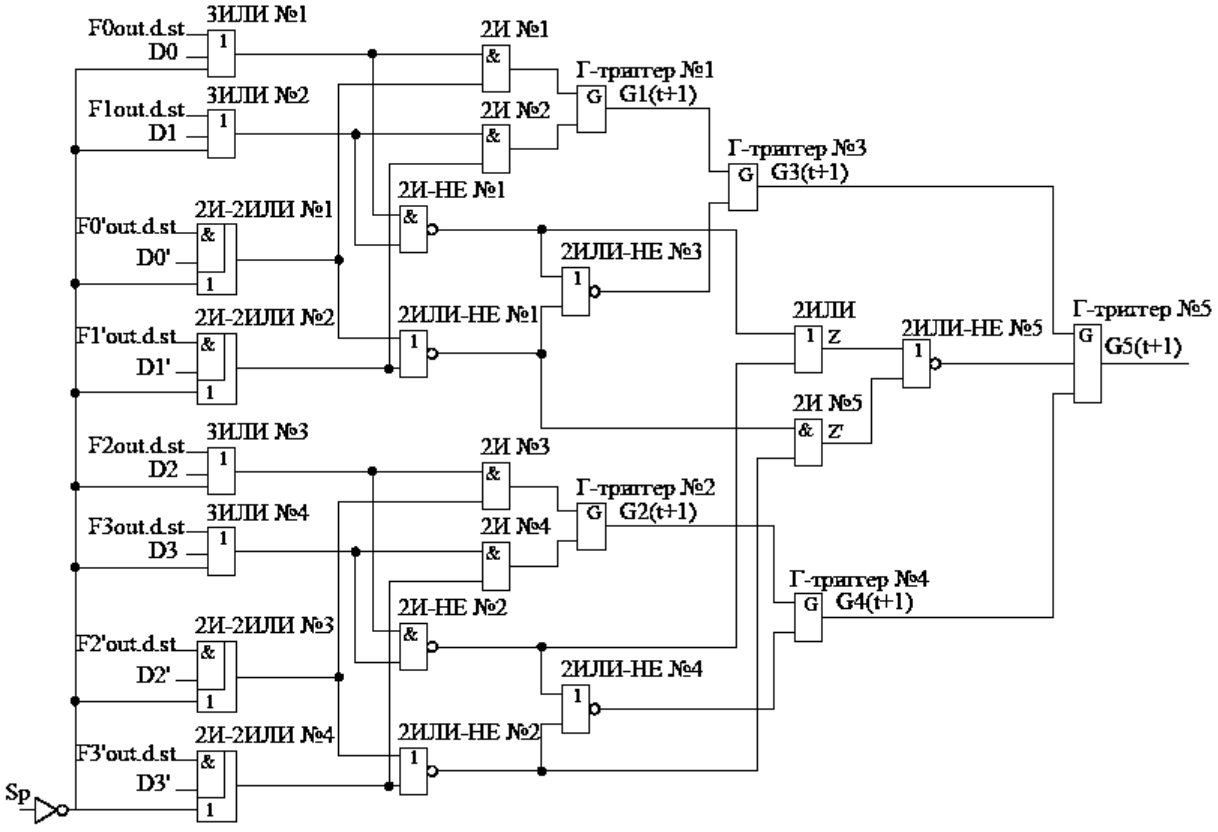



Рисунок 2.14 Блок дизъюнкции настройки функции
для четырех выходов каналов.

Для реализации прямого канала используются элементы 3ИЛИ №1, 3ИЛИ №2, 3ИЛИ №3, 3ИЛИ №4, 2И-НЕ №1, 2И-НЕ №2 и 2ИЛИ, для реализации двойственного канала используются элементы 2И-2ИЛИ №1, 2И-2ИЛИ №2, 2И-2ИЛИ №3, 2И-2ИЛИ №4, 2ИЛИ-НЕ №1, 2ИЛИ-НЕ №2 и 2И №5. Остальные элементы используются для индицирования блоков схемы. Для организации самосинхронизации используется обратная связь, не показанная на рисунке 2.14, с выхода G5(t+1) Г-триггера на вход Sp инвертора.

Аналогичным образом можно получить блок настройки для большого количества выходов каналов.

Для подтверждения, что исходный элемента адаптирован к условиям работы в ССС, выполним проверку блока дизъюнкции на полумодулярность. Для этого получим модель Маллера для схемы на рисунке 2.14.

D0=0; D1=1; D2=1; D3=0; neD0=1; neD1=0; neD2=1; neD3=0; /Описание констант

F0=0; nF0=1; F1=1; nF1=0; F2=1; nF2=0; F3=1; nF3=0; / описываем значения с выходов канала

/Описываем работу прямого канала/

A=D0|F0|Sp^; B=D1|F1|Sp^; A2=D2|F2|Sp^; B2=D3|F3|Sp^; / элементы 3ИЛИ

Z1=^A*B; Z2=^A2*B2; / элементы 2И-НЕ

Z=Z1|Z2; / элемент 2ИЛИ

/Описываем работу двойственного канала/

C=neD0*nF0|Sp^; D=neD1*nF1|Sp^; C2=neD2*nF2|Sp^; k2=neD3*nF3|Sp^; / Элементы 2И-2ИЛИ

nZ1=^C|D; nZ2=^C2|k2; /Элементы 2ИЛИ-НЕ

nZ=nZ1*nZ2; / Элемент 2И

/Описываем работу индикаторов схемы/

I=^Z|nZ; I0=A*C; I1=B*D; I2=A2*C2; I3=B2*k2; I4=^z1|nZ1; I5=^z2|nZ2;

/Описываем работу гистерезисных триггеров/

g1=^I0*I1|g1i(I0|I1); g1i=^g1; g2=^I2*I3|g2i(I2|I3); g2i=^g2;

g3=^I4*g1i|g3i(I4|g1i); g3i=^g3; g4=^I5*g2i|g4i(I5|g2i); g4i=^g4;

g5=^I*g3i*g4i|g5i(I|g3i|g4i); g5i=^g5;

Sp=g5i; / Описываем замкнутость схемы

$g5 /Начальное состояние схемы (спейсерная фаза)

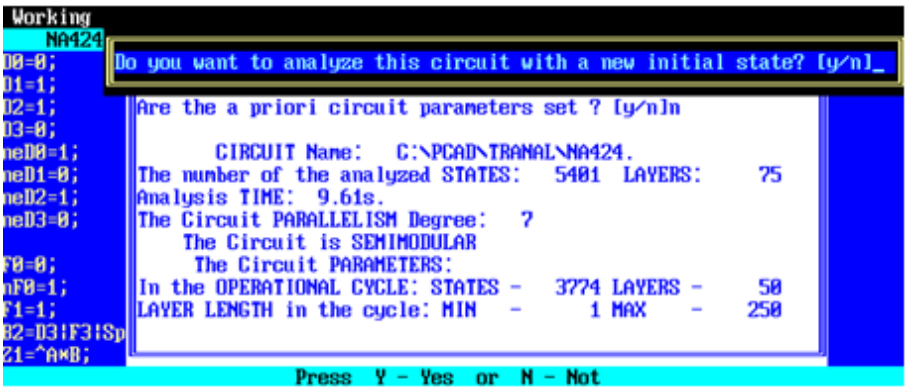



Рисунок 2.15 – Результат анализа блока дизъюнкции настройки функции для четырех выходов каналов в САПР Forcage (подсистема TRANAL).

На рисунке 2.15 видно, что схема является полумодулярной (semimodular), а значит обладает двумя основными свойствами самосинхронных схем. Остальные результаты проверки на полумодулярность показаны в приложении №3.

## 2.4. Разработка методов реализации конфигурируемых самосинхронных генераторов систем логических функций, заданных в ДНФ

### 2.4.1 Разработка метода конфигурируемых элементов с использованием библиотеки элементов БМК

Метод ориентирован на использование элементной базы БМК, поэтому предполагается использовать универсальный логический модуль, в качестве которого необходимо применить библиотечные элементы.

В качестве элементов для построения одного самосинхронного блока конъюнкций, реализующего один разряд, предлагается использовать библиотечные элементы 2И-2ИЛИ (A21O) и 3ИЛИ (OR3). Выражения, описывающие работу выбранных элементов:

$$F_{out.2and2or} = I0 \cdot I1 \vee I2 \tag{2.60}$$

$$F_{out.3or} = I0 \vee I1 \vee I2 \tag{2.61}$$

где $F_{out.2and2or}$ – выход элемента 2И-2ИЛИ, $F_{out.3or}$ – выход элемента 3ИЛИ, $I0$, $I1$, $I2$– входы элементов.

Существо метода состоит в том, чтобы использовать каждый вход по определенному назначению: один вход использовать в качестве настройки (конфигурирования), второй вход использовать как информационный и третий вход использовать для реализации фазы спейсера. Таким образом метод заключается в реализации каналов, индикаторов и элементов, фиксирующих окончания переходных процессов. Для реализации прямого канала предлагается использовать два элемента 2И-2ИЛИ и элемент 2ИЛИ, а для реализации двойственного канала два элемента 3ИЛИ с инверсной настройкой и один элемент 2И. Выражения, описывающие работу каналов для n=1:

$$\begin{cases} F = \overline{(X} \cdot D0 \vee Sp') \vee \overline{(X'} \cdot D1 \vee Sp') \\ F' = (X \vee D0' \vee Sp') \cdot (X' \vee D1' \vee Sp') \end{cases} \tag{2.62}$$

где $X$, $X'$ – входные переменные, $D0$, $D1$ – прямая настройка, $D0'$, $D1'$ – инверсная настройка, $Sp$– спейсер.

В качестве индикаторов входов и выходов предлагается использовать элемент 2И, на выходе которого устанавливается логическая 1 только при условии, что на оба входа приходят 1, в других случаях на выходе 0.

Для фиксирования времени переходного процесса используются Г-триггеры, которые меняют свое состояние только при всех 0 или 1 на входах, во всех других состояние выхода сохраняется. Тогда выражения, описывающие работу всего блока:

$$\begin{cases} F = \overline{(X} \cdot D0 \vee \overline{Sp(t)}) \vee \overline{(X'} \cdot D1 \vee \overline{Sp(t)}) \\ F' = \left(X \vee D0' \vee \overline{Sp(t)}\right) \cdot \left(X' \vee D1' \vee \overline{Sp(t)}\right) \\ I = F \cdot F' \\ I1.1 = \overline{(X} \cdot D0 \vee \overline{Sp(t)}) \cdot \left(X \vee D0' \vee \overline{Sp(t)}\right) \\ I1.2 = \overline{(X'} \cdot D1 \vee \overline{Sp(t)}) \cdot \left(X' \vee D1' \vee \overline{Sp(t)}\right) \\ G1(t+1) = [I1.1 \cdot I1.2 \vee G1(t)(I1.1 \vee I1.2)]' \\ Sp(t+1) = \left[I \cdot G1(t) \vee E(t)\left(I \vee G1(t)\right)\right]' \end{cases} \tag{2.63}$$

где $I$ – выход индикатора выходов каналов, $I1.1$, $I1.2$ – выходы индикаторов входов, $G1(t+1)$ – выход Г-триггера, фиксирующего окончание переходного процесса в блоках 2И-2ИЛИ и 3ИЛИ, $Sp(t+1)$ – выход Г-триггера, фиксирующего окончание переходного процесса во всем блоке.

Модель блока одного разряда представлена на рисунке 2.16:

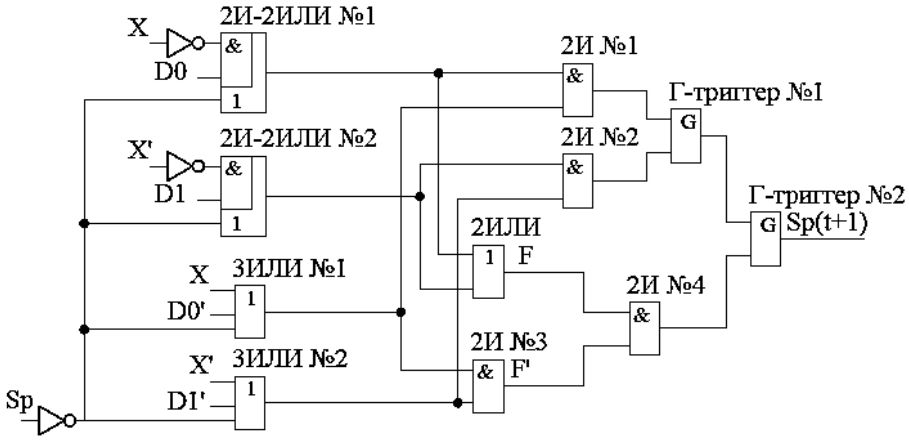



Рисунок 2.16 - Модель блока конъюнкций одного разряда.

Элементы 2И-2ИЛИ №1, 2И-2ИЛИ №2 и 2ИЛИ реализуют прямой канал. Для реализации двойственного канала предлагается использовать элементы 3ИЛИ №1, 3ИЛИ №2 и 2И №3. Остальные элементы используются в качестве индикаторов.

Для реализации блока конъюнкций одного разрядов на n>2 предлагается использовать каскадирование блока конъюнкций одного разряда. Тогда выражения, описывающие работу модели:

$$\begin{cases} F = \overline{(X1 \cdot D0 \vee \overline{E(t)})} \vee \overline{(X1' \cdot D1 \vee \overline{E(t)})} \cdot \overline{(X2 \cdot D2 \vee \overline{E(t)})} \vee \overline{(X2' \cdot D3 \vee \overline{E(t)})} \\ F' = \left(X1 \vee D0' \vee \overline{E(t)}\right) \cdot \left(X1' \vee D1' \vee \overline{E(t)}\right) \vee \left(X2 \vee D2' \vee \overline{E(t)}\right) \cdot \left(X2' \vee D3' \vee \overline{E(t)}\right) \\ I = F \cdot F' \\ I1.1 = \overline{(X1 \cdot D0 \vee \overline{E(t)})} \cdot \left(X1 \vee D0' \vee \overline{E(t)}\right) \\ I1.2 = \overline{(X1' \cdot D1 \vee \overline{E(t)})} \cdot \left(X1' \vee D1' \vee \overline{E(t)}\right) \\ I1.3 = \overline{(X2 \cdot D2 \vee \overline{E(t)})} \cdot \left(X2 \vee D2' \vee \overline{E(t)}\right) \\ I1.4 = \overline{(X2' \cdot D3 \vee \overline{E(t)})} \cdot \left(X2' \vee D3' \vee \overline{E(t)}\right) \\ I2.1 = \overline{(X1 \cdot D0 \vee \overline{E(t)})} \vee \overline{(X1' \cdot D1 \vee \overline{E(t)})} \cdot \left(X1 \vee D0' \vee \overline{E(t)}\right) \cdot \left(X1' \vee D1' \vee \overline{E(t)}\right) \\ I2.2 = \overline{(X2 \cdot D2 \vee \overline{E(t)})} \vee \overline{(X2' \cdot D3 \vee \overline{E(t)})} \cdot \left(X2 \vee D2' \vee \overline{E(t)}\right) \cdot \left(X2' \vee D3' \vee \overline{E(t)}\right) \\ G1.1(t+1) = [I1.1 \cdot I1.2 \vee G1.1(t)(I1.1 \vee I1.2)]' \\ G1.2(t+1) = [I1.3 \cdot I1.4 \vee G1.2(t)(I1.3 \vee I1.4)]' \\ G2.1(t+1) = [G1.1(t) \cdot I2.1 \vee G2.1(t)(G1.1(t) \vee I2.1]' \end{cases} \quad (2.64)$$

$$\begin{cases} G1(t+1) = [G2.1(t) \cdot G2.2(t) \vee G1(t)(G2.1(t) \vee G2.2(t)]' \\ E(t+1) = \left[I \cdot G1(t) \vee E(t)\left(I \vee G1(t)\right)\right]' \end{cases} \quad (2.65)$$

где $I$ – выход индикатора выходов каналов, $I1.1$, $I1.2$, $I1.3$, $I1.4$ – выходы индикаторов входов первого слоя, $I2.1$, $I2.2$ – выходы индикаторов входов второго слоя, $G1.1(t+1)$, $G1.2(t+1)$ – выходы Г-триггеров, фиксирующих окончание переходного процессов в блоках первого слоя, $G2.1(t+1)$, $G2.2(t+1)$ - выходы Г-триггеров, фиксирующих окончание переходного процессов в блоках второго слоя, $G1(t+1)$ – выход Г-триггера, фиксирующего окончание переходных процессов в первом и во втором слоях, $E(t+1)$ – выход Г-триггера, фиксирующего окончание переходного процесса во всем блоке.

Модель блока конъюнкций для двух переменных показана на рисунке 2.17. Элементы 2И-2ИЛИ №1, 2И-2ИЛИ №2, 2И-2ИЛИ №3, 2И-2ИЛИ №4, 2ИЛИ №1, 2ИЛИ №2 и 2И №9 реализуют прямой канал. Для реализации двойственного канала предлагается использовать элементы 3ИЛИ №1, 3ИЛИ №2, 3ИЛИ №3, 3ИЛИ №4, 2И №3, 2И №7 и 2ИЛИ №3. Остальные элементы используются в качестве индикаторов.

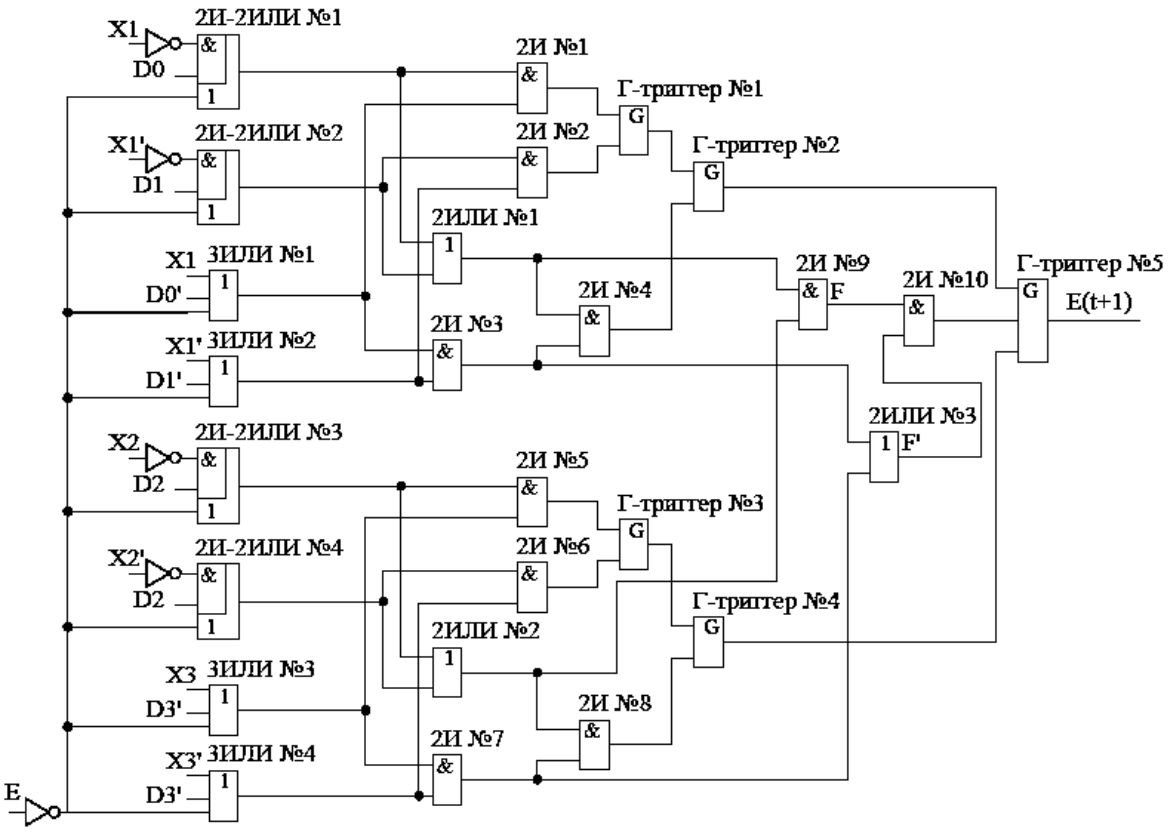



Рисунок 2.17 - Модель блока конъюнкций двух разрядов.

Для реализации *m* функций от *n* переменных предлагается использовать блоки настройки, предложенные в разработке метода реализации конфигурируемого самосинхронного генератора систем логических функций, заданных в СДНФ, показанные на рисунках 2.13, 2.14, но подключение выполнять инверсно, т.е выход прямого канала блока конъюнкций к входам двойственных каналов блока дизъюнкций и выход двойственного канала блока конъюнкций к входам прямых каналов блока дизъюнкций.

Для подтверждения принадлежности разработанной модели к классу ССС выполним анализ модели для n=2 на полумодулярность, используя САПР Forcage подсистему Tranal. Выполним описание модели Маллера:

S0=0; S1=1; S2=1; S3=0; neS0=1; neS1=0; neS2=1; neS3=0; /Задаем константы

X1=0; nX1=1; X2=1; nX2=0; /Задаем значения переменных

A=S0|X1|E^; B=S1|nX1^|E^; A2=S2|X2|E^; B2=S3|nX2^|E^; / Описываем каждый элемент прямого канала

Z1=A*B; Z2=A2*B2; Z=z1|z2; /Описываем работу прямого канала

C=neS0*nX1|E^; / Описываем каждый элемент двойственного канала

D=neS1*X1^|E^; C2=neS2*nX2|E^; D2=neS3*X2^|E^; nZ1=C|D; nZ2=C2|D2;

nZ=nZ1*nZ2; /Описываем работу двойственного канала

I=Z*nZ; / Описываем работу индикатора выходов

I0=A*C; / Описываем работу индикатора входов первого слоя

I1=B*D; I2=A2*C2; I3=B2*D2; I4=z1*nZ1; I5=z2*nZ2; / Описываем работу индикатора входов второго слоя

g1=^I0*I1|g1i(I0|I1); g1i=^g1; / Описываем работу каждого Г-триггера

g2=^I2*I3|g2i(I2|I3); g2i=^g2; g3=^I4*g1i|g3i(I4|g1i); g3i=^g3;

g4=^I5*g2i|g4i(I5|g2i); g4i=^g4; g5=^g3i*g4i|g5i(g3i|g4i); g5i=^g5;

G=^I*g5i|Gi(I|g5i); Gi=^G;

E=Gi; / Описываем замкнутость элемента

$G / Задаем начальное состояние схемы

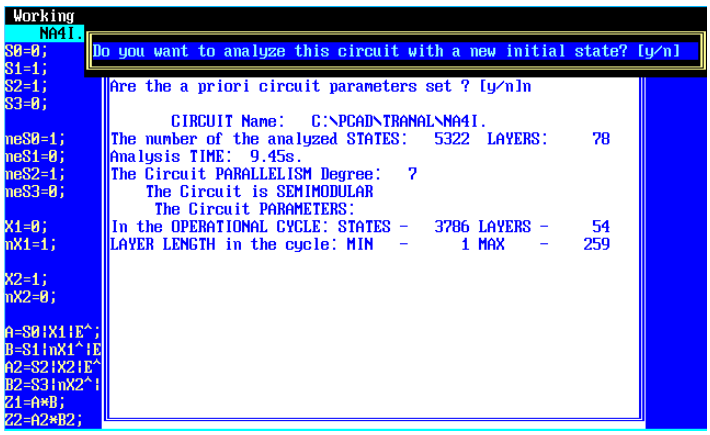



Рисунок 2.18 – Результат анализа блока конъюнкций двух разрядов в САПР Forcage (подсистема TRANAL).

На рисунке 2.18 видно, что схема является полумодулярной (semimodular), а значит обладает двумя основными свойствами самосинхронных схем. Кроме принадлежности схем к ССС, на рисунке 2.18 можно посмотреть сколько времени длился анализ, сколько было проанализировано состояний и слоев и др. Остальные проверки на полумодулярность показаны в приложении №3.

### 2.4.2 Разработка метода конфигурируемых элементов с использованием подтягивающих резисторов

Метод ориентирован на использование универсального элемента в программируемых логических матрицах (ПЛМ), называемого блоком конъюнкций.

Суть метода заключается в том, чтобы модифицировать существующий принцип реализации функций, заданных в ДНФ в программируемых логических матрицах и в ПЛИС типа CPLD [8]. Одна ячейка имеет вид:

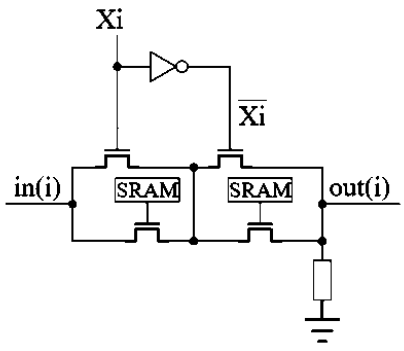



Рисунок 2.19 – Модель существующей ячейки i-го блока конъюнкции с подтягивающим резистором.

Представленная модель описывается выражением:

$$out(i) = in(i) \cdot [Xi \vee Sram] \cdot [\overline{Xi} \vee Sram] \quad (2.66)$$

Таким образом, для реализации j-ой конъюнкции от n переменных требуется:

$$out(i) = \underset{i=1}{\overset{n}{\&}} (in(i) \cdot [Xi \vee Sram] \cdot [\overline{Xi} \vee Sram]), j = 1, w \quad (2.67)$$

Данная ячейка позволяет вычислить существенность переменной или несущественность, но существуют варианты настройки, при которых цепочки транзисторов не активируются и на выходе схемы возникает так называемый обрыв, в таком случае на выходе ячейки устанавливают подтягивающие резисторы.

Для адаптации модели (рисунок 2.18) к условиям работы в ССС предлагается ввести в ячейку цепочку спейсера и блок отсечки, тогда выражение, описывающее работы схемы:

$$out(i) = \underset{i=1}{\overset{n}{\&}} (in(i) \cdot [Xi \vee Sram] \cdot [\overline{Xi} \vee Sram] \cdot [Xi \vee \overline{Xi}] \vee [Xi\overline{Xi}]), j = 1, w \quad (2.68)$$

Для реализации двойственного канала конфигурируемого элемента предлагается использовать инверсный вход и подключение подтягивающего резистора на шину противоположного знака. Тогда выражение, описывающее работу двойственного канала:

$$out'(i) = \underset{i=1}{\overset{n}{\&}} (in'(i) \cdot [Xi \vee Sram] \cdot [\overline{Xi} \vee Sram] \cdot [Xi \vee \overline{Xi}] \vee [Xi\overline{Xi}]), j = 1, w \quad (2.69)$$

Рассмотрим выражения 2.67 и 2.68 для n=1, out(i) обозначим F, в качестве входов in(i) используем шину питания и шину земли:

$$\begin{cases} F = [Vcc \cdot (X \vee D0) \cdot (X' \vee D1) \cdot (X \vee X') \vee (XX')]' \\ F' = [Ground \cdot (X \vee D0) \cdot (X' \vee D1) \cdot (X \vee X') \vee (XX')]' \end{cases} \quad (2.70)$$

Кроме каналов в элементе также используются блоки входного набора, индикаторы входов и выходов и Г-триггеры, фиксирующие окончания переходных процессов в схеме. Тогда выражения, описывающие работу блока:

$$\begin{cases} F = \left[ \begin{array}{l} Vcc \cdot \left(\overline{\overline{X' \cdot Sp(t)} \vee D0}\right) \cdot \left(\overline{\overline{X \cdot Sp(t)} \vee D1}\right) \cdot \left(\overline{\overline{X' \cdot Sp(t)} \vee \overline{X \cdot Sp(t)}}\right) \vee \\ \vee \left(\overline{X' \cdot Sp(t)} \cdot \overline{X \cdot Sp(t)}\right) \end{array} \right]' \\ F' = \left[ \begin{array}{l} Ground \cdot \left(\overline{\overline{X' \cdot Sp(t)} \vee D0}\right) \cdot \left(\overline{\overline{X \cdot Sp(t)} \vee D1}\right) \cdot \left(\overline{\overline{X' \cdot Sp(t)} \vee \overline{X \cdot Sp(t)}}\right) \vee \\ \vee \left(\overline{X' \cdot Sp(t)} \cdot \overline{X \cdot Sp(t)}\right) \end{array} \right]' \\ I = F \cdot F' \\ I1 = \left[\overline{\overline{X' \cdot Sp(t)} \vee \overline{X \cdot Sp(t)}}\right]' \\ E(t+1) = [I \cdot I1 \vee Sp(t)(I \vee I1)]' \end{cases} \quad (2.71)$$

где $I$ – выход индикатора выходов каналов, $I1$– выход индикатора входов, $E(t+1)$ – выход Г-триггера, фиксирующего окончание переходного процесса во всем блоке.

Модель блока на один разряд показана рисунке 2.20.

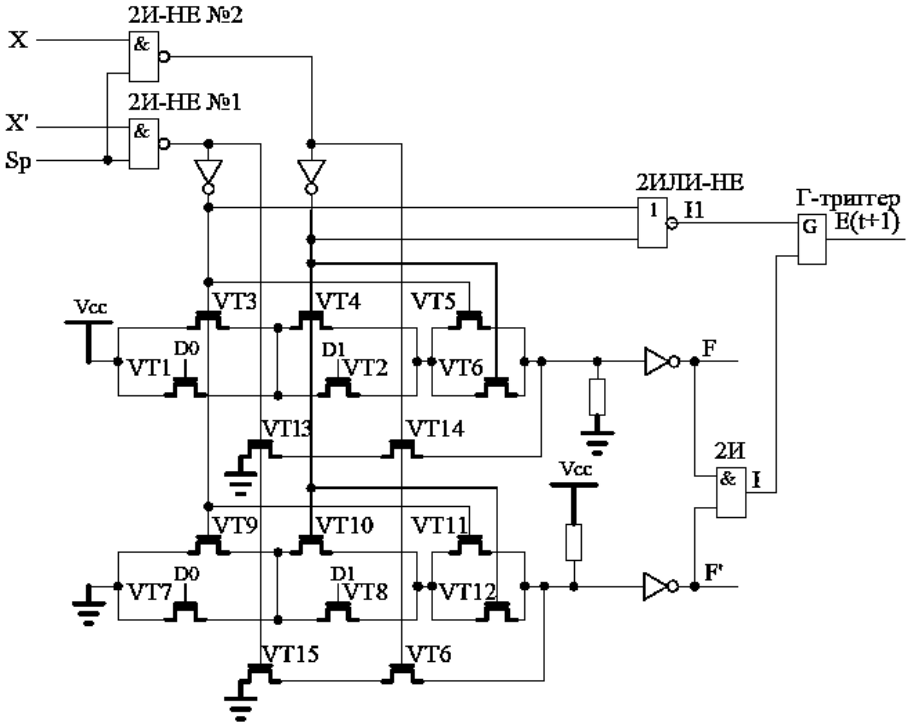



Рисунок 2.20 – Модель блока одной конъюнкции с подтягивающим резистором для n=1.

Для реализации прямого канала используются транзисторы VT1-VT6 и транзисторы VT13-VT14. Двойственный канал реализован на транзисторах VT7-VT12 и VT15-VT16. Кроме этого, транзисторы VT5-VT6 и VT11-VT12 – реализуют отсечку. Для соблюдения логики работы ССС используются подтягивающие резисторы, которые подтягивают уровень 0 или 1 при возникновении ситуации, когда какой-нибудь транзистор не активируется и

канале образуется «обрыв». Транзисторы VT13-VT16 также реализуют фазу спейсера. Элементы 2И-НЕ №1 и 2И-НЕ №2 реализуют блоки входного набора, на которые подается сигнал с выхода последнего Г-триггера для реализации самосинхронизации схемы. На рисунке 2.20 обратная связь не показана. Элементы 2ИЛИ-НЕ и 2И реализуют индикаторы. Г-триггер требуется для фиксирования времени окончания переходных процессов в схеме.

Для реализации блока конъюнкций с подтягивающим резистором двух и более разрядов предлагается использовать каскадирование блока конъюнкций одного разряда. Тогда работа модели для n=2 описывается выражениями:

$$
\begin{cases}
F = \begin{bmatrix} [Vcc \cdot \left(\overline{\overline{X1' \cdot Sp(t)} \vee D0}\right) \cdot \left(\overline{\overline{X1 \cdot Sp(t)} \vee D1}\right) \cdot \left(\overline{\overline{X1' \cdot Sp(t)} \vee \overline{X1 \cdot Sp(t)}}\right) \vee \\ \vee \left(\overline{X1' \cdot Sp(t)} \cdot \overline{X1 \cdot Sp(t)}\right)]' \vee [Vcc \cdot \left(\overline{\overline{X2' \cdot Sp(t)} \vee S2}\right) \cdot \left(\overline{\overline{X2 \cdot Sp(t)} \vee D3}\right) \wedge \\ \wedge \left(\overline{\overline{X2' \cdot Sp(t)} \vee \overline{X2 \cdot Sp(t)}}\right) \vee \left(\overline{X2' \cdot Sp(t)} \cdot \overline{X2 \cdot Sp(t)}\right)]' \end{bmatrix} \\
F' = \begin{bmatrix} [Ground \cdot \left(\overline{\overline{X1' \cdot Sp(t)} \vee D0}\right) \cdot \left(\overline{\overline{X1 \cdot Sp(t)} \vee D1}\right) \cdot \left(\overline{\overline{X1' \cdot Sp(t)} \vee \overline{X1 \cdot Sp(t)}}\right) \vee \\ \vee \left(\overline{X1' \cdot Sp(t)} \cdot \overline{X1 \cdot Sp(t)}\right)]' \cdot [Ground \cdot \left(\overline{\overline{X2' \cdot Sp(t)} \vee D2}\right) \wedge \\ \wedge \left(\overline{\overline{X2 \cdot Sp(t)} \vee D3}\right) \cdot \left(\overline{\overline{X2' \cdot Sp(t)} \vee \overline{X2 \cdot Sp(t)}}\right) \vee \left(\overline{X2' \cdot Sp(t)} \cdot \overline{X2 \cdot Sp(t)}\right)]' \end{bmatrix} \\
I1.1 = \left[\overline{\overline{X1' \cdot Sp(t)} \vee \overline{X1 \cdot Sp(t)}}\right]' \\
I1.2 = \left[\overline{\overline{X2' \cdot Sp(t)} \vee \overline{X2 \cdot Sp(t)}}\right]' \\
I1.3 = \begin{bmatrix} [Vcc \cdot \left(\overline{\overline{X1' \cdot Sp(t)} \vee D0}\right) \cdot \left(\overline{\overline{X1 \cdot Sp(t)} \vee D1}\right) \cdot \left(\overline{\overline{X1' \cdot Sp(t)} \vee \overline{X1 \cdot Sp(t)}}\right) \vee \\ \vee \left(\overline{X1' \cdot Sp(t)} \cdot \overline{X1 \cdot Sp(t)}\right)]' \vee [Ground \cdot \left(\overline{\overline{X1' \cdot Sp(t)} \vee D0}\right) \wedge \\ \wedge \left(\overline{\overline{X1 \cdot Sp(t)} \vee D1}\right) \left(\overline{\overline{X1' \cdot Sp(t)} \vee \overline{X1 \cdot Sp(t)}}\right) \vee \left(\overline{X1' \cdot Sp(t)} \cdot \overline{X1 \cdot Sp(t)}\right)]' \end{bmatrix} \\
I1.4 = \begin{bmatrix} [Vcc \cdot \left(\overline{\overline{X2' \cdot Sp(t)} \vee D2}\right) \cdot \left(\overline{\overline{X2 \cdot Sp(t)} \vee D3}\right) \cdot \left(\overline{\overline{X2' \cdot Sp(t)} \vee \overline{X2 \cdot Sp(t)}}\right) \vee \\ \vee \left(\overline{X2' \cdot Sp(t)} \cdot \overline{X2 \cdot Sp(t)}\right)]' \vee [Ground \cdot \left(\overline{\overline{X2' \cdot Sp(t)} \vee D2}\right) \wedge \\ \wedge \left(\overline{\overline{X2 \cdot Sp(t)} \vee D3}\right) \left(\overline{\overline{X2' \cdot Sp(t)} \vee \overline{X2 \cdot Sp(t)}}\right) \vee \left(\overline{X2' \cdot Sp(t)} \cdot \overline{X2 \cdot E(t)}\right)]' \end{bmatrix} \\
I = F \cdot F' \\
G1.1(t+1) = [I1.1 \cdot I1.3 \vee G1.1(t)(I1.1 \vee I1.3)]' \\
G1.2(t+1) = [I1.2 \cdot I1.4 \vee G1.2(t)(I1.2 \vee I1.4)]' \\
G1(t+1) = \left[G1.1(t) \cdot G1.2(t) \vee G1(t)\left(G1.1(t) \vee G1.2(t)\right)\right]' \\
E(t+1) = \left[I \cdot G1(t) \vee E(t)\left(I \vee G1(t)\right)\right]'
\end{cases}
\tag{2.72}
$$

где $I$ – выход индикатора выходов каналов, $I1.1$, $I1.2$, $I1.3$, $I1.4$ – выходы индикаторов входов, $G1.1(t+1)$, $G1.2(t+1)$ – выходы Г-триггеров, фиксирующих окончание переходных процессов в блоках, $G1(t+1)$ – выход Г-триггера, фиксирующего окончание переходных процессов во всех индикаторах и Г-триггерах, $E(t+1)$ – выход Г-триггера, фиксирующего окончание переходного процесса во всем блоке.

Модель блока показана на рисунке 2.21.

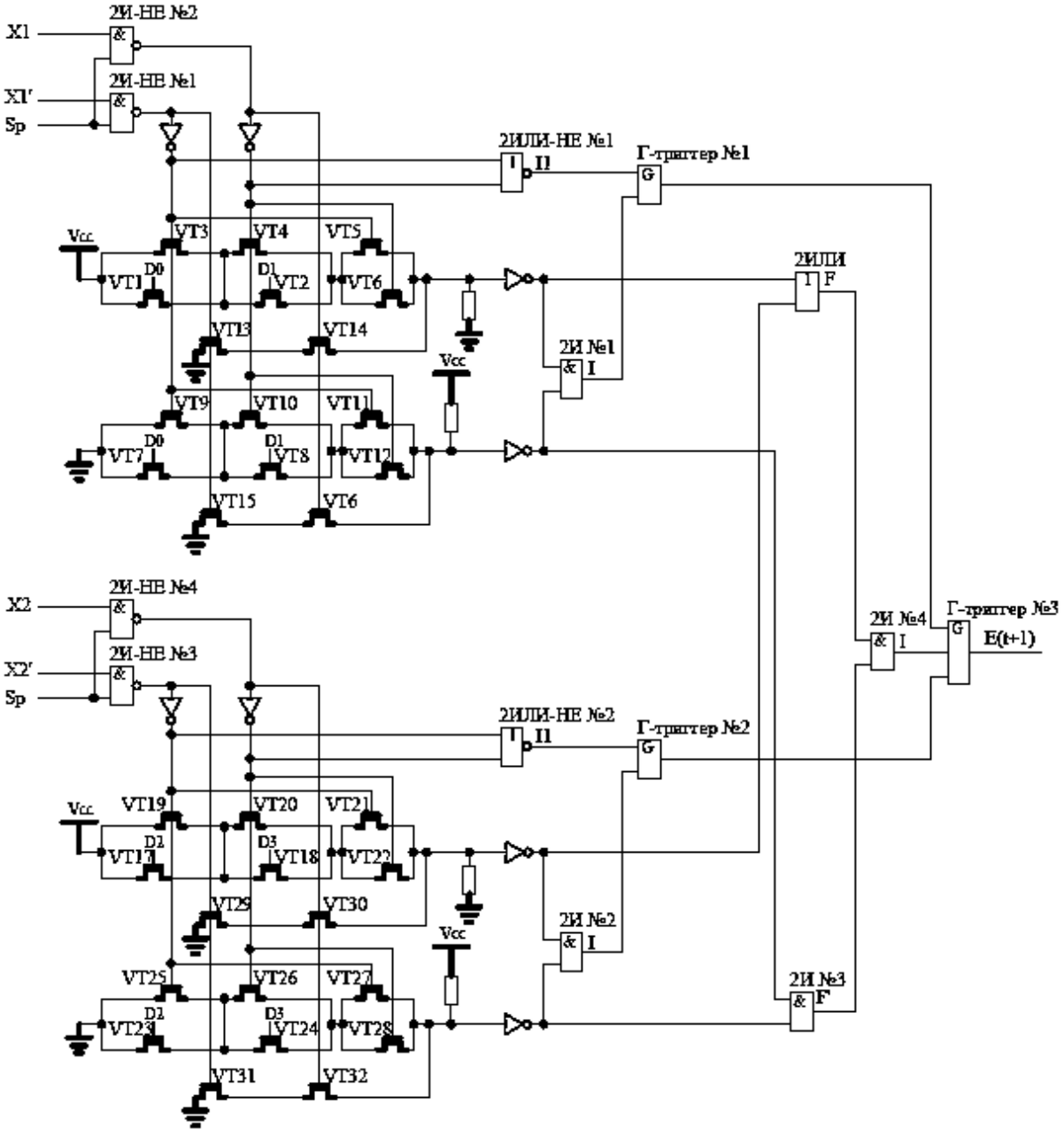



Рисунок 2.21 – Модель блока конъюнкций с подтягивающим резистором для n=2.

Для реализации прямого канала используются транзисторы VT1-VT6 и VT13-VT14 для одной конъюнкции. Для реализации прямого канала второй конъюнкции используются транзисторы VT17-VT22 и VT29-VT30. Двойственные каналы реализованы на транзисторах VT7-VT12 и VT15-VT16, VT25-VT28 и VT31-VT32 для первой и второй конъюнкций соответственно. Кроме этого, транзисторы VT5-VT6, VT11-VT12, VT21-VT22 и VT27-VT28 – реализуют отсечку. Транзисторы VT13-VT16 и VT29-VT32 реализуют фазу спейсера. Элементы 2ИЛИ и 2И №3 реализуют прямой и двойственный каналы двух конъюнкций соответственно. Элементы 2И-НЕ №1, 2И-НЕ №2, 2И-НЕ №3, 2И-НЕ №4 реализуют блоки входного набора, на которые подается сигнал с выхода последнего Г-триггера для реализации самосинхронизации схемы. На рисунке 2.21 обратная связь не показана. Остальные элементы используются в качестве индикаторов.

Таким образом можно строить блоки на большее число разрядов.

Для реализации m функций от n переменных предлагается использовать блоки настройки, предложенные в разработке метода реализации конфигурируемого самосинхронного генератора систем логических функций, заданных в СДНФ, показанные на рисунках 2.13, 2.14, но подключение должно быть выполнено инверсно, т.е выход прямого канала блока конъюнкций к входам двойственных каналов блока дизъюнкций и выход двойственного канала блока конъюнкций к входам прямых каналов блока дизъюнкций.

Для подтверждения принадлежности разработанной модели к классу ССС выполним анализ модели для n=2 на полумодулярность, используя САПР Forcage подсистему Tranal. Выполним описание модели Маллера:

```
Gr=0; Vc=1; / Задаем постоянные в схеме
X1=1; nX1=^X1; X2=1; nX2=^X2; / Описываем выходные переменные
S0=0; S1=1; S2=0; S3=1; / Задаем постоянные конфигурирования схемы
V1=^nX1*E; / Описываем каждый элемент схемы
V2=^X1*E;
```

Sp=Gr(V1*V2); / Описываем цепочку спейсера

V3=^nX2*E; V4=^X2*E; Sp1=Gr(V3*V4);

F1=^Vc(V1^|S0*V2^|S1*V1^|V2^)|Sp; / Описываем работу прямого канала первого слоя

nF1=^Gr(V1^|S0*V2^|S1*V1^|V2^)|Sp; / Описываем работу двойственного канала первого слоя

F2=^Vc(V3^|S2*V4^|S3*V3^|V4^)|Sp1; / Описываем работу прямого канала первого слоя

nF2=^Gr(V3^|S2*V4^|S3*V3^|V4^)|Sp1; / Описываем работу двойственного канала первого слоя

I=^V1^|V2^; I1=F1*nF1; / Описываем работу индикаторов схемы

G1=^I*I1|G1i(I|I1); G1i=^G1; / Описываем работу Г-триггера

I0=^V3^|V4^; I2=F2*nF2;

G2=^I0*I2|G2i(I0|I2); G2i=^G2;

F=F1|F2; / Описываем работу прямого канала второго слоя

nF=nF1|nF2; / Описываем работу двойственного канала второго слоя

If=F*nF; / Описываем работу индикатора выходов каналов

G3=^G1i*G2i|G3i(G1i|G2i); G3i=^G3; / Описываем работу Г-триггеров

G=^If*G3i|Gi(If|G3i); Gi=^G;

E=Gi; / Описываем замкнутость схемы

$G / Задаем начальные условия схемы

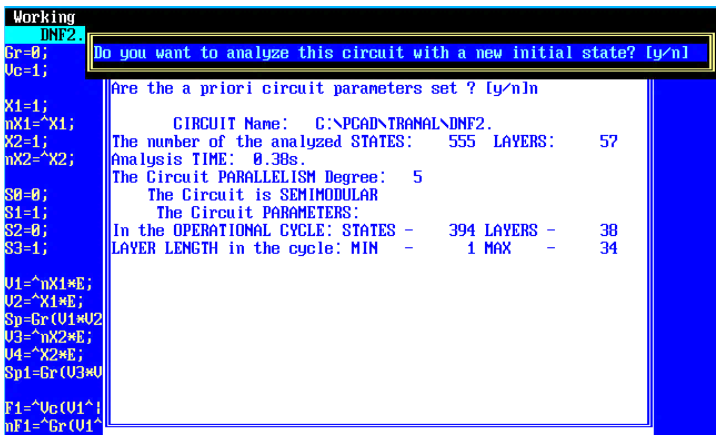



Рисунок 2.22 – Результат анализа блока конъюнкций двух разрядов с подтягивающим резистором в САПР Forcage (подсистема TRANAL).

На рисунке 2.22 видно, что схема является полумодулярной, а значит ССС.

Из-за использования подтягивающих резисторов в схеме снижается быстродействие, увеличивается площадь, занимаемая на кристалле и увеличивается потребляемая мощность. Остальные результаты проверки на полумодулярность показаны в приложении №3.

### 2.4.3 Разработка метода конфигурируемых элементов с использованием транзисторов ортогональности

Метод ориентирован на использование предложенного в работе [8] усовершенствованного элемента для ПЛМ с целью исключения недостатков схемы с подтягивающими резисторами.

Предлагается один разряд блока конъюнкций ДНФ-LUT [8] строить так, как изображено на рисунке 2.23

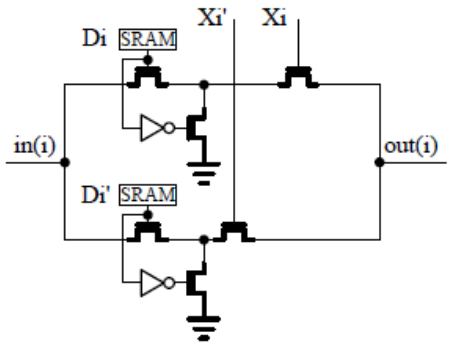



Рисунок 2.23. Один разряд блока конъюнкций ДНФ-LUT.

В предложенной схеме нет случаев, когда цепочка остается неактивированной, при любой из настроек SRAM хотя бы одна ветвь активирована. Это достигается за счет транзисторов, названных транзисторами ортогональности – затворы подключены к настройке SRAM, истоки подключены к шине нуля, а стоки к сток-истоковой области транзисторов, управляемыми настройками SRAM и значением переменных. Тогда выражение, описывающее работу блока:

$$out(i) = \underset{i=1}{\overset{n}{\&}} in(i) \cdot [(Di \vee Ground \cdot \overline{Di}) \cdot Xi \vee (Di' \vee Ground \cdot \overline{Di'}) \cdot Xi'] \qquad (2.73)$$

Сущность метода заключается в адаптации исходного блока к условиям работы в ССС. Для этого предлагается ввести цепочку спейсера. Реализовать

двойственный канал предлагается путем подключения входа in(i) к шине нуля, а истоки транзисторов ортогональности, обеспечивающие активацию одной из ветвей, к шине питания. Выражения, описывающие работу каналов:

$$\begin{cases} out(i) = \underset{i=1}{\overset{n}{\&}}[in(i) \cdot [(Di \vee Ground \cdot \overline{Di}) \cdot Xi \vee (Di' \vee Ground \cdot \overline{Di'}) \cdot Xi'] \vee (Xi \cdot Xi')]' \\ out'(i) = \underset{i=1}{\overset{n}{\&}}\left[in'^{(i)} \cdot [(Di \vee Vcc \cdot \overline{Di}) \cdot Xi \vee (Di' \vee Vcc \cdot \overline{Di'}) \cdot Xi'] \vee (Xi \cdot Xi')\right]' \end{cases} \quad (2.74)$$

где *out(i)* – выход прямого канала *i*-го блока, *out'(i)* – выход двойственного канала *i*-го блока, *Di*, *Di'* – настройка SRAM.

Как уже было сказано ранее, для реализации ССС элемента требуются дополнительные элементы. С учетом вышесказанного получим выражения, описывающие блок конъюнкций одной переменной:

$$\begin{cases} F = [Vcc \cdot \left[(D0 \vee Ground \cdot \overline{D0}) \cdot \left(\overline{\overline{Sp(t) \cdot X'}}\right) \vee (D1 \vee Ground \cdot \overline{D1}) \cdot \left(\overline{\overline{Sp(t) \cdot X}}\right)\right] \vee \\ \quad \vee \left(\overline{Sp(t) \cdot X'} \cdot \overline{Sp(t) \cdot X}\right)]' \\ F' = [Ground \cdot \left[(D0 \vee Vcc \cdot \overline{D0}) \cdot \left(\overline{\overline{Sp(t) \cdot X'}}\right) \vee (D1 \vee Vcc \cdot \overline{D1}) \cdot \left(\overline{\overline{Sp(t) \cdot X}}\right)\right] \vee \\ \quad \vee \left(\overline{Sp(t) \cdot X'} \cdot \overline{Sp(t) \cdot X}\right)]' \\ I1 = \left[\overline{\overline{Sp(t) \cdot X'}} \vee \overline{\overline{Sp(t) \cdot X}}\right]' \\ I = F \cdot F' \\ E(t+1) = [I1 \cdot I \vee E(t)(I1 \vee I)]' \end{cases} \quad (2.75)$$

где *F* – выход прямого канала блока, *F'* – выход двойственного канала блока, *D*0, *D*1 – настройка SRAM, *I*1 – выход индикатора входов, *I* – выход индикатора выходов каналов, *E*(*t*+1) – выход Г-триггера, фиксирующего окончания переходных процессов в схеме.

Модель ССС блока конъюнкций одного разряда показана на рисунке 2.24. Элементы 2И-НЕ №1 и 2И-НЕ №2 реализуют блоки входного набора, дерево передающих транзисторов VT1-VT4, VT9-VT10 и VT13-VT14 реализует прямой канал. Для реализации двойственного канала используются транзисторы VT5-VT8, VT11-VT12 и VT15-VT16. Кроме этого, транзисторы VT13-VT16 реализуют цепочки спейсеров. Инверторы на выходах предназначены для восстановления уровня сигналов. Остальные элементы используются в качестве

индикаторов. Для организации самосинхронизации используется обратная связь с выхода G(t+1) Г-триггера на вход Sp блоков 2И-НЕ №1 и 2И-НЕ №2.

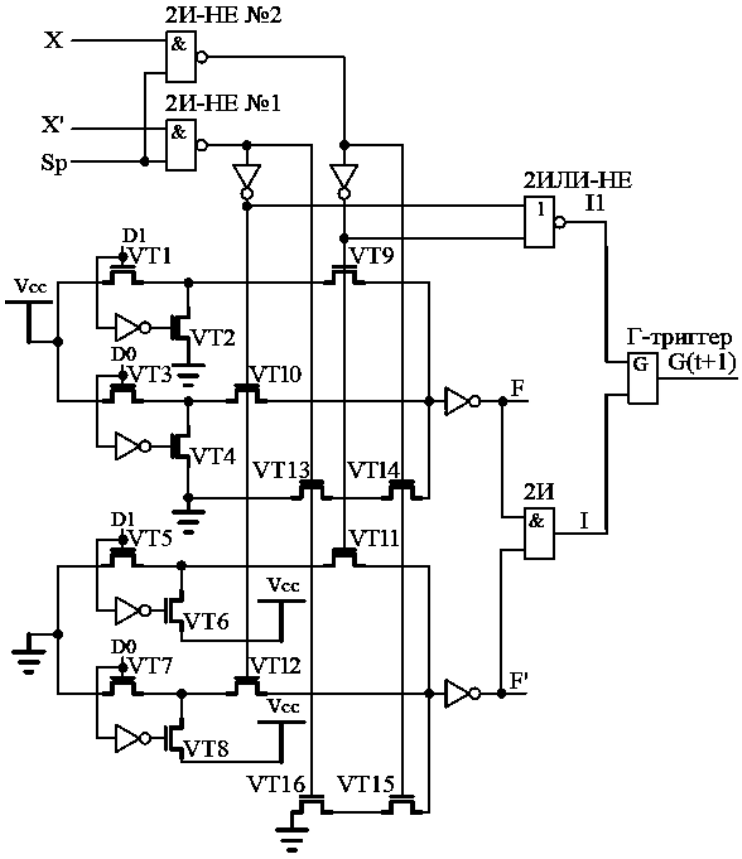



Рисунок 2.24. – Модель ССС блока конъюнкций одного разряда для n=1.

Для реализации блока конъюнкций с транзисторами ортогональности двух и более разрядов предлагается использовать каскадирование блока конъюнкций одного разряда. Тогда работа модели для n=2 описывается выражениями:

$$\begin{cases} F = \{[Vcc \cdot [(D0 \vee Ground \cdot \overline{D0}) \cdot \left(\overline{\overline{Sp(t) \cdot X1'}}\right) \vee (D1 \vee Ground \cdot \overline{D1}) \cdot \left(\overline{\overline{Sp(t) \cdot X1}}\right)] \vee \\ \quad \vee \left(\overline{Sp(t) \cdot X1'} \cdot \overline{Sp(t) \cdot X1}\right)]' \vee [Vcc \cdot [(D2 \vee Ground \cdot \overline{D2}) \cdot \left(\overline{\overline{Sp(t) \cdot X2'}}\right) \vee \\ \quad \vee \ (D3 \vee Ground \cdot \overline{D3}) \cdot \left(\overline{\overline{Sp(t) \cdot X2}}\right)] \vee \left(\overline{Sp(t) \cdot X2'} \cdot \overline{Sp(t) \cdot X2}\right)]'\} \\ F' = \{[Ground \cdot [(D0 \vee Vcc \cdot \overline{D0}) \cdot \left(\overline{\overline{Sp(t) \cdot X1'}}\right) \vee (D1 \vee Vcc \cdot \overline{D1}) \cdot \left(\overline{\overline{Sp(t) \cdot X1}}\right)] \vee \\ \quad \vee \left(\overline{Sp(t) \cdot X1'} \cdot \overline{Sp(t) \cdot X1}\right)]' \cdot [Ground \cdot [(D2 \vee Vcc \cdot \overline{D2}) \cdot \left(\overline{\overline{Sp(t) \cdot X2'}}\right) \vee \\ \quad \vee \ (D3 \vee Vcc \cdot \overline{D3}) \cdot \left(\overline{\overline{Sp(t) \cdot X2}}\right)] \vee \left(\overline{Sp(t) \cdot X2'} \cdot \overline{Sp(t) \cdot X2}\right)]' \\ I1.1 = \left[\overline{\overline{Sp(t) \cdot X1'} \vee \overline{Sp(t) \cdot X1}}\right]' \end{cases} \quad (2.76)$$

$$\begin{cases} I1.2 = \left[\overline{\overline{Sp(t)\cdot X2'}} \vee \overline{\overline{Sp(t)\cdot X2}}\right]' \\ I1.3 = [Vcc\cdot[(D0 \vee Ground\cdot\overline{D0})\cdot\left(\overline{\overline{Sp(t)\cdot X1'}}\right) \vee (D1 \vee Ground\cdot\overline{D1}) \wedge \\ \quad \wedge\left(\overline{\overline{Sp(t)\cdot X1}}\right)] \vee \left(\overline{Sp(t)\cdot X1'}\cdot\overline{Sp(t)\cdot X1}\right)]'\cdot[Ground\cdot[(D0 \vee Vcc\cdot\overline{D0}) \wedge \\ \quad \wedge\left(\overline{\overline{Sp(t)\cdot X1'}}\right) \vee (D1 \vee Vcc\cdot\overline{D1})\cdot\left(\overline{\overline{Sp(t)\cdot X1}}\right)] \vee \left(\overline{Sp(t)\cdot X1'}\cdot\overline{Sp(t)\cdot X1}\right)]' \\ I1.4 = [Vcc\cdot[(D2 \vee Ground\cdot\overline{D2})\cdot\left(\overline{\overline{Sp(t)\cdot X2'}}\right) \vee (D3 \vee Ground\cdot\overline{D3}) \wedge \\ \quad \wedge\left(\overline{\overline{Sp(t)\cdot X2}}\right)] \vee \left(\overline{Sp(t)\cdot X2'}\cdot\overline{Sp(t)\cdot X2}\right)]'\cdot[Ground\cdot[(D2 \vee Vcc\cdot\overline{D2}) \wedge \\ \quad \wedge\left(\overline{\overline{Sp(t)\cdot X2'}}\right) \vee (D3 \vee Vcc\cdot\overline{D3})\cdot\left(\overline{\overline{Sp(t)\cdot X2}}\right)] \vee \left(\overline{Sp(t)\cdot X2'}\cdot\overline{Sp(t)\cdot X2}\right)]' \\ G1.1(t+1) = [I1.1\cdot I1.3 \ \vee G1.1(t)(\ I1.1 \vee I1.3)]' \\ G1.2(t+1) = [I1.2\cdot I1.4 \ \vee G1.2(t)(\ I1.2 \vee I1.4)]' \\ G1(t+1) = \left[G1.1(t)\cdot G1.2(t) \ \vee G1(t)\left(G1.1(t) \vee G1.2(t)\right)\right]' \\ I = F\cdot F' \\ E(t+1) = [G1(t)\cdot I \vee E(t)(G1(t) \vee I\ )]' \end{cases} \quad (2.77)$$

где $I$ – выход индикатора выходов каналов, $I1.1$, $I1.2$, $I1.3$, $I1.4$ – выходы индикаторов входов, $G1.1(t+1)$, $G1.2(t+1)$ – выходы Г-триггеров, фиксирующих окончание переходных процессов в блоках, $G1(t+1)$ – выход Г-триггера, фиксирующего окончание переходных процессов во всех индикаторах и Г-триггерах, $E(t+1)$ – выход Г-триггера, фиксирующего окончание переходного процесса во всем блоке.

Модель блока конъюнкций одного разряда для n=2 показана на рисунке 2.25. Элементы 2И-НЕ №1, 2И-НЕ №2, 2И-НЕ №3 и 2И-НЕ №4 реализуют блоки входного набора. Транзисторы VT1-VT4, VT9-VT10, VT13-VT14, VT17-VT20, VT11-VT12, VT29-VT30 и элемент 2ИЛИ реализуют прямой канал. Для реализации двойственного канала используются транзисторы VT5-VT8, VT11-VT12, VT15-VT16, VT21-VT24, VT27-VT28, VT131-VT32 и элемент 2И №3. Кроме этого, транзисторы VT13-VT16 и VT129-VT32 реализуют цепочки спейсеров. Инверторы на выходах предназначены для восстановления уровня сигналов. Остальные элементы используются в качестве индикаторов. Для организации самосинхронизации используется обратная связь с выхода E(t+1) Г-триггера на вход Sp блоков 2И-НЕ №1, 2И-НЕ №2, 2И-НЕ №3 и 2И-НЕ №4.

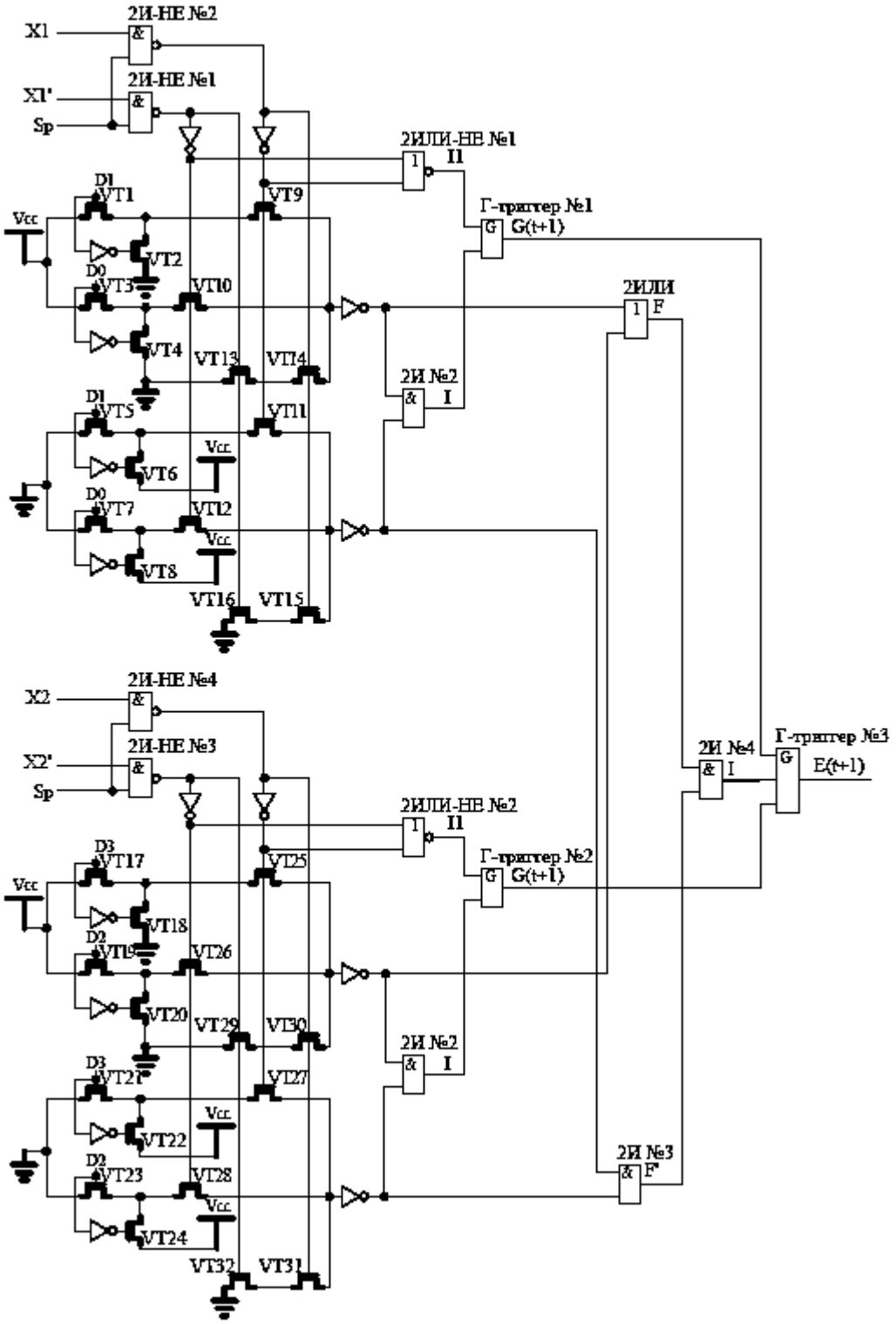



Рисунок 2.25. – Модель ССС блока конъюнкций двух разрядов.

Таким образом можно строить блоки на большее число разрядов. На устройство получен патент РФ [92].

Для реализации *m* функций от *n* переменных предлагается использовать блоки настройки, предложенные в разработке метода реализации конфигурируемого самосинхронного генератора систем логических функций,

заданных в СДНФ, показанные на рисунках 2.13, 2.14, но подключение должно быть выполнено инверсно, т.е выход прямого канала блока конъюнкций к входам двойственных каналов блока дизъюнкций и выход двойственного канала блока конъюнкций к входам прямых каналов блока дизъюнкций.

Для подтверждения принадлежности разработанной модели к классу ССС выполним анализ модели для n=2 на полумодулярность, используя САПР Forcage подсистему Tranal. Выполним описание модели Маллера:

Gr=0; Vc=1; / Задаем постоянные схемы

X1=1; nX1=^X1; X2=1; nX2=^X2; / Описываем значения переменных

S0=0; S1=1; S2=1; S3=0; / Задаем конфигурацию

V1=^nX1*E; V2=^X1*E; / Описываем входные блоки

Sp=Gr(V1*V2); / Описываем работу спейсера

V3=^nX2*E; V4=^X2*E; / Описываем входные блоки

Sp1=Gr(V3*V4); / Описываем работу спейсера

A=V2^(S1*Vc|S1^*Gr); B=V1^(S0*Vc|S0^*Gr); / Описываем работу цепочек прямого канала первого слоя

nA=V2^(S1*Gr|S1^*Vc); nB=V1^(S0*Gr|S0^*Vc); / Описываем работу цепочек двойственного канала первого слоя

F1=^A|B|Sp; nF1=^nA|nB|Sp; / Описываем прямой и двойственный каналы первого слоя

A1=V4^(S3*Vc|S3^*Gr); B1=V3^(S2*Vc|S2^*Gr); / Описываем работу цепочек прямого канала первого слоя

nA1=V4^(S3*Gr|S3^*Vc); nB1=V3^(S2*Gr|S2^*Vc); / Описываем работу цепочек двойственного канала первого слоя

F2=^A1|B1|Sp1; nF2=^nA1|nB1|Sp1; / Описываем прямой и двойственный каналы первого слоя

I=^V1^|V2^; I1=F1*nF1; G1=^I*I1|G1i(I|I1); G1i=^G1; / Описываем работу блоков индикаторов первого слоя

I0=^V3^|V4^; I2=F2*nF2; G2=^I0*I2|G2i(I0|I2);G2i=^G2;

F=F1|F2; / Описываем работу прямого канала второго слоя

nF=nF1*nF2; / Описываем работу двойственного канала второго слоя

If=F*nF; / Описание работы индикатора выходов

G3=^G1i*G2i|G3i(G1i|G2i);  G3i=^G3; / Описание работы Г-триггеров

G=^If*G3i|Gi(If|G3i); Gi=^G;

E=Gi; / Описание замкнутости схемы

$G / Задание начальных условий

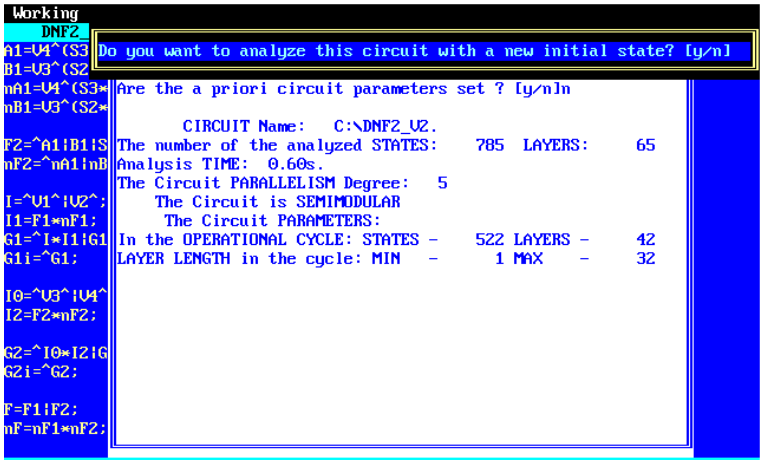



Рисунок 2.26 – Результат анализа блока конъюнкций двух разрядов с транзисторами ортогональности в САПР Forcage (подсистема TRANAL).

На рисунке 2.26 видно, что схема является полумодулярной, а значит ССС. Остальные результаты проверки элементов на полумодулярность показаны в приложении №3.

## 2.5. Выводы по главе

В главе отражены следующие научные и практические результаты:

1. Предложен метод самосинхронной реализации генератора функций (ГФ) на основе библиотечного базиса 2И-2ИЛИ-НЕ, отличающийся тем, что библиотечный элемент адаптирован к условиям работы в ССС. Для этого применяется парафазная дисциплина кодирования сигнала и используется фаза спейсера, причем для согласованности работы блоков схемы при количестве переменных n>1 спейсер каждого слоя блоков должен изменяется. Кроме этого, двойственность функции инверсных каналов достигается не отрицанием переменных и самой функции, а инвертированием настройки (конфигурации).

2. Предложен метод самосинхронной реализации логической функции в ССС по принципу LUT (Look Up Table), используемому в программируемых логических интегральных схемах (ПЛИС типа FPGA), отличающийся тем, что используется дополнительная ветвь дерева передающих транзисторов, активируемая в фазе спейсера, а двойственный канал универсального логического элемента настраивается инверсными константами.

3. Предложен метод унифицированной самосинхронной реализации систем логических функций в СДНФ, отличающийся тем, что логическая функция в ССС реализуется по принципу дешифратора DC LUT, адаптированного к работе в ССС. Кроме этого, двойственность функции инверсных каналов достигается путем инверсного подключения каналов и транзисторов ортогональности относительного прямого канала.

4. Предложен метод унифицированной самосинхронной реализации систем логических функций, заданных в ДНФ, отличающийся тем, что логическая функция реализуется по известному ДНФ в ПЛМ, адаптированному к условиям работы в ССС. Кроме этого, двойственность функции инверсных каналов достигается не отрицанием переменных и самой функции, а инвертированием настройки (конфигурации).

5. Анализ разработанных средств реализации логических функций в ССС на полумодулярность показал, что разработанные модели на библиотечном элементе 2И-2ИЛИ-НЕ - ГФ, LUT-ST, DC-LUT-ST и ДНФ-LUT-ST являются самосинхронными.

# ГЛАВА 3. МОДЕЛИРОВАНИЕ РАЗРАБОТАННЫХ КОНФИГУРИРУЕМЫХ САМОСИНХРОННЫХ ЭЛЕМЕНТОВ

## 3.1. Моделирование разработанного генератора логических функций на основе стандартных логических элементов

### 3.1.1 Моделирование ГФ на логическом уровне

Выполним моделирование разработанной модели генератора функций двух переменных на основе библиотечного элемента 2И-2ИЛИ-НЕ [77,93-95], изображенной на рисунке 2.1.2, в системе автоматизированного проектирования «КОВЧЕГ», разработанной НПК «Технологический центр» МИЭТ. САПР «КОВЧЕГ» предназначен для проектирования микросхем на базовых матричных кристаллах (БМК) серий 5503, 5507, 5521 и 5529 [57].

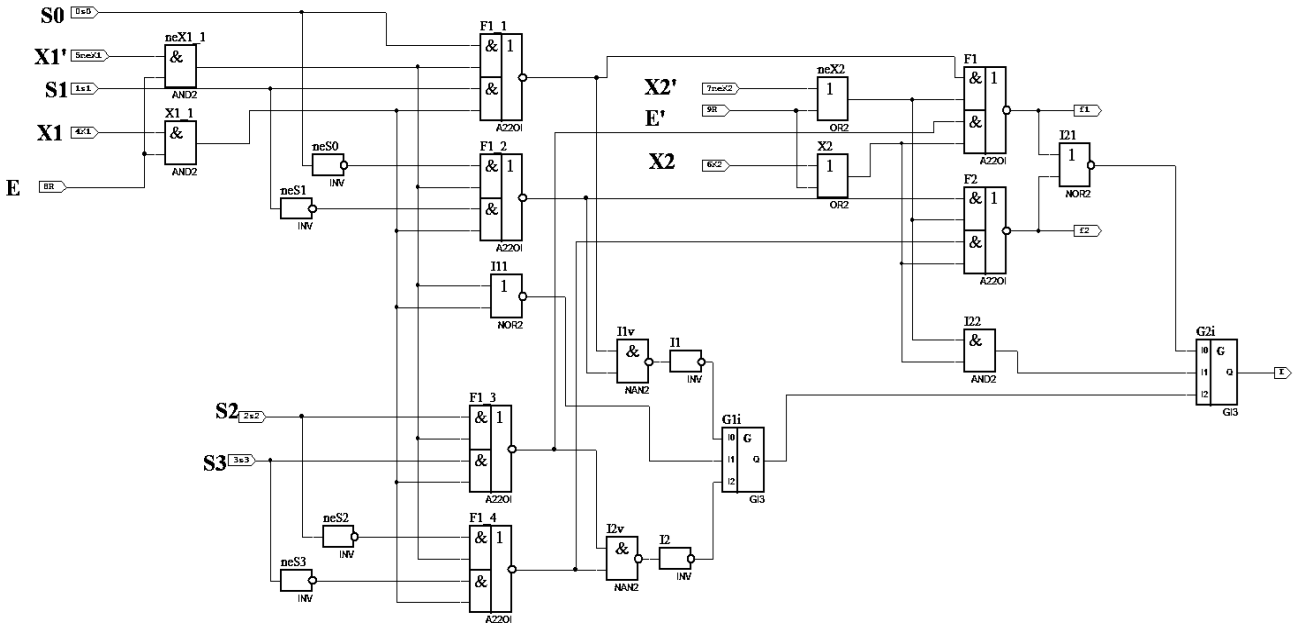



Рисунок 3.1 – Модель генератора функций двух переменных в САПР «Ковчег».

На рисунке 3.1 элементы F1_1, F2_1, F1 – библиотечные элементы 2И-2ИЛИ-НЕ, реализующие основной канал, элементы F1_2, F2_2, F2 реализуют двойственный канал. Элементы, на основе элемента 2И: X1_1, neX1_1, X2, neX2 реализуют блоки входного набора. Индикаторы входов I11, I22 реализованы на элементах 2ИЛИ-НЕ и 2И соответственно. Элементы I1v, I2v, I21 реализуют

индикаторы выходов. Кроме этого в схеме используются 3-х входовые G-триггеры, которые фиксируют окончание переходных процессов в первом слое и во всей схеме.

Для оценки работоспособности схемы необходимо написать тесты, в которых описываются начальные состояния схемы и задаются входные переменные. Для схемы на рисунке 3.1 можно написать 16 тестов, в которых будут учтены все возможные входные наборы. Полное описание тестов приведено в приложении №4.

Test7:

E = 1,0,1,0,1,0,1,0; / Задаем спейсер для первого слоя, на который приходит сигнал с выходного G-триггера

E’ = 0,1,0,1,0,1,0,1; / Задаем спейсер для второго слоя, на который приходит сигнал с выходного G-триггера

D0 = 0; D1 = 1; D2 = 1; D3 = 0; / Задание констант (SRAM)

X1 = 0,0,1,0,1,0,0,0; neX1 = 1,0,0,0,0,0,1,0; X2 = 0,0,0,0,1,0,1,0; neX2 = 1,0,1,0,0,0,0,0; / Задание все возможные состояния переменных с чередованием фаз

Результат моделирования показан на рисунке 3.2:

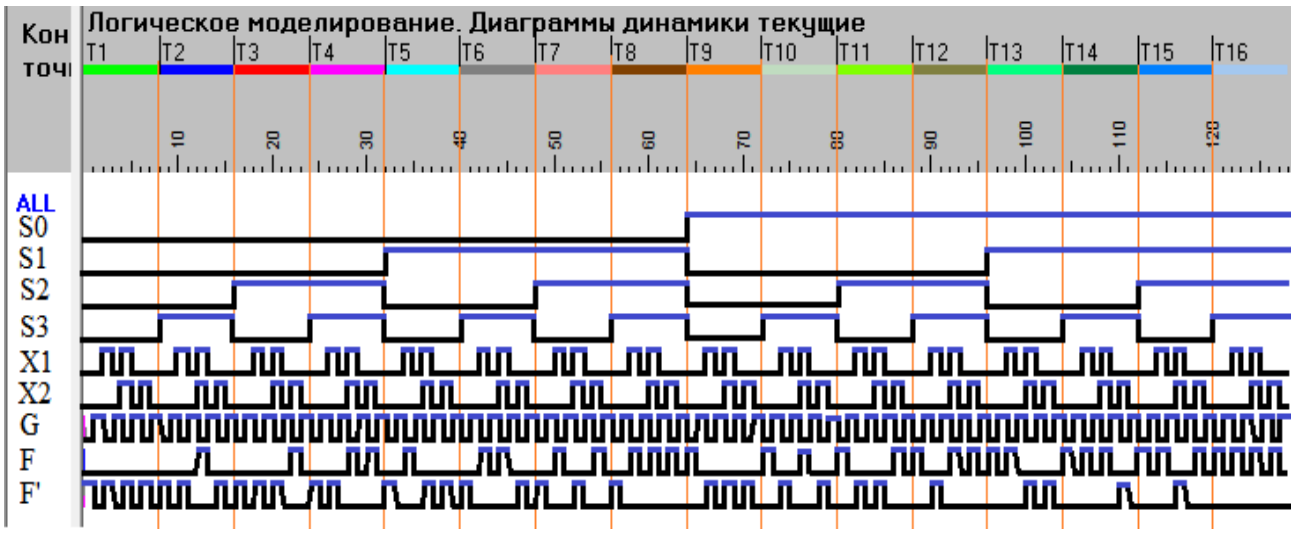



Рисунок 3.2 – Результат моделирования для генератора функций двух переменных в САПР «Ковчег».

Каждый из тестов реализует одну из возможных функций, например, тест Т7 реализует функцию «исключающее ИЛИ», а тест Т15 реализует функцию «Штрих Шеффера».

Выполним анализ результатов на примере теста №7 – функция «исключающее ИЛИ» («Сумма по модулю 2»). Выходы каналов F и F’ в рабочей фазе должны быть инверсны друг другу, в фазе спейсера F=F’, выход G Г-триггера G2i в рабочей фазе должен быть в 0, в фазе спейсера – 1.


Рисунок 3.3 – Анализ теста №7 для генератора функций двух переменных в САПР «Ковчег».

На рисунке 3.3 рабочие фазы каналов F и F’ {10} и {01} чередуются фазой спейсера {00}. Выход G в 0 в рабочей фазе, в 1 в фазе спейсера. Функция «исключающее ИЛИ» реализуется правильно, поэтому полученные диаграммы подтверждают работоспособность предложенного элемента.

Сведем полученный результат для реализации функции, например, «исключающее ИЛИ» в таблицу 3.1, «Эквивалентность» в таблицу 3.2, «Обратная импликация» в таблицу 3.3, «Штрих Шеффера» в таблицу 3.4. Таблица истинности для всех настроек представлена в приложении №4.

Таблица 3.1. – Таблица истинности генератора функций двух переменных. Функция «исключающее ИЛИ».

| **№ теста** | **S0S1S2S3** | **X1X2** | **F** | **F’** |
|---|---|---|---|---|
| 7 | 0110 | 00 | 0 | 1 |
| | | 01 | 1 | 0 |
| | | 11 | 0 | 0 |
| | | 10 | 1 | 1 |

Таблица 3.2. – Таблица истинности генератора функций двух переменных. Функция «Эквивалентность».

| № теста | S0S1S2S3 | X1X2 | F | F’ |
|---|---|---|---|---|
| 10 | 1001 | 00 | 1 | 0 |
| | | 01 | 0 | 1 |
| | | 11 | 1 | 0 |
| | | 10 | 0 | 1 |

Таблица 3.3. – Таблица истинности генератора функций двух переменных. Функция «Импликация».

| № теста | S0S1S2S3 | X1X2 | F | F’ |
|---|---|---|---|---|
| 10 | 1011 | 00 | 1 | 0 |
| | | 01 | 1 | 0 |
| | | 11 | 1 | 0 |
| | | 10 | 0 | 1 |

Таблица 3.4. – Таблица истинности генератора функций двух переменных. Функция «Штрих Шеффера».

| № теста | S0S1S2S3 | X1X2 | F | F’ |
|---|---|---|---|---|
| 15 | 1110 | 00 | 1 | 0 |
| | | 01 | 1 | 0 |
| | | 11 | 0 | 1 |
| | | 10 | 1 | 0 |

Таблица 3.5. – Таблица истинности генератора функций двух переменных. Функция «Дизъюнкция».

| № теста | S0S1S2S3 | X1X2 | F | F’ |
|---|---|---|---|---|
| 8 | 0111 | 00 | 0 | 1 |
| | | 01 | 1 | 0 |
| | | 11 | 1 | 0 |
| | | 10 | 1 | 0 |

Таким образом, выполняя конфигурирование генератора функций на основе стандартных КМОП элементов, можно получить любую функцию, при этом самосинхронность разработанного элемента сохраняется.

### 3.1.2. Моделирование ГФ на топологическом уровне

Выполним разработку топологии элемента ГФ двух переменных в САПР MicroWind. Топология показана на рисунке 3.4

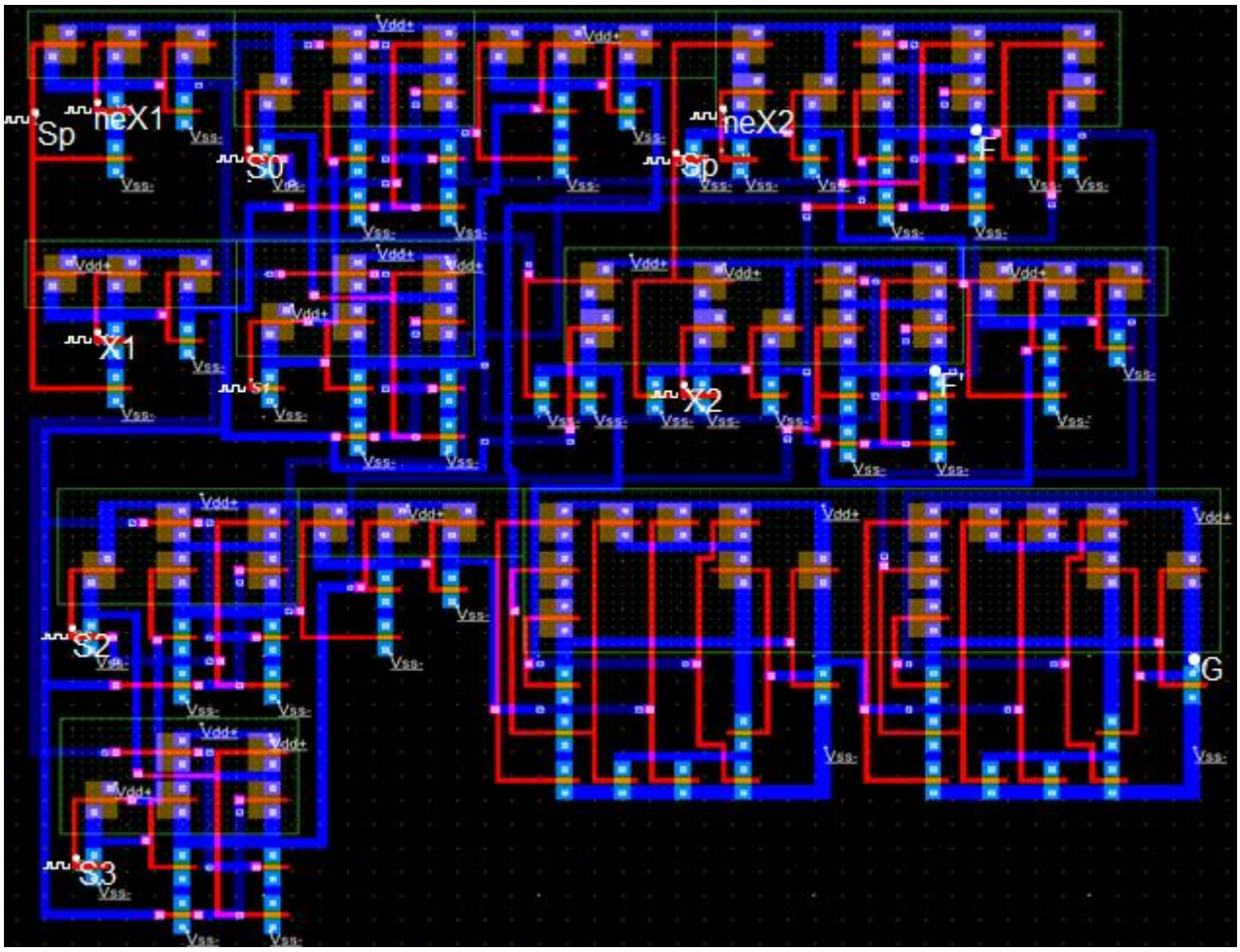



Рисунок 3.4 – Топология генератора функций двух переменных с разомкнутой обратной связью.

Вход Sp – реализует фазу спейсера и служит для подключения обратной связи с выхода последнего Г-триггера. X1, neX1, X2, neX2 – входы переменных первого каскада и второго соответственно. S0-S3 – входы настройки, которые позволяют выполнить конфигурацию элемента константами или выполнить подключение к ячейками SRAM с загруженными таблицами истинности логической функции. F – выход прямого канала, F’ – выход двойственного канала. G – выход последнего гистерезисного триггера схемы.

Выполним проверку реализации функций «исключающее ИЛИ» - таблица истинности 3.1 и «Обратная импликация» - таблица истинности 3.3.

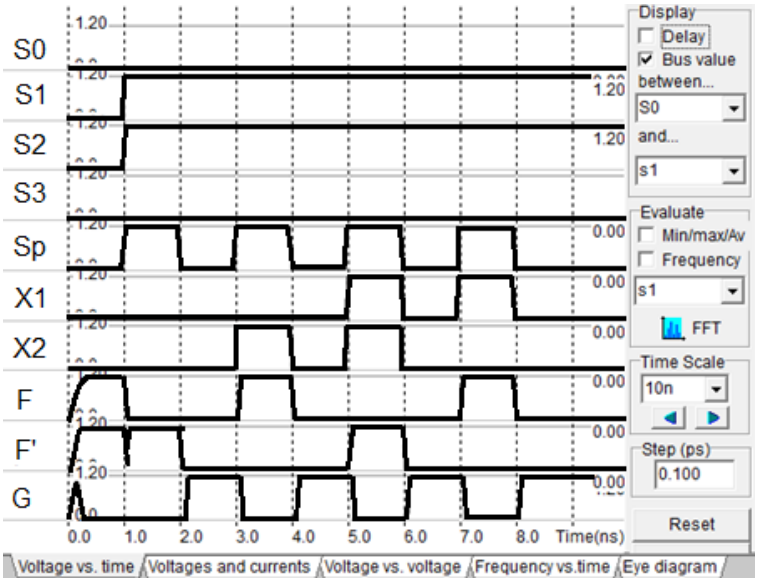



Рисунок 3.5- Результат моделирования генератора функций двух переменных, реализация функции «исключающее ИЛИ».

На рисунке 3.5 показан результат реализации функции «исключающее ИЛИ», который соответствует таблице истинности 3.1. Нулевой такт длительностью 1 нс – установочный, в анализе не учитывается. В рабочей фазе выходы F, F’ инверсны друг другу, в фазе спейсера F=F’=0. Выход Г-триггера в рабочей фазе в 0, в фазе спейсера в 1. Полученный результат подтверждает работоспособность предложенного генератора функций двух переменных.

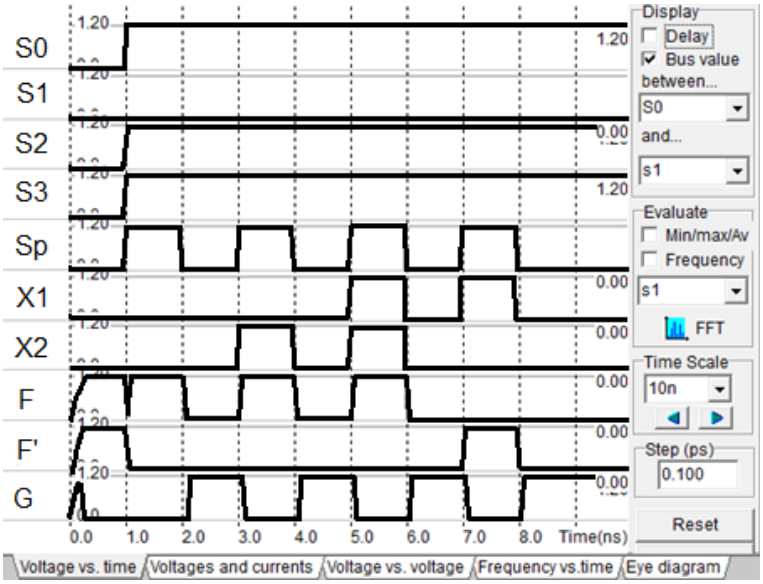



Рисунок 3.6- Результат моделирования генератора функций двух переменных, реализация функции «Импликация».

На рисунке 3.6 показан результат реализации функции «Обратная импликация», который соответствует таблице истинности 3.3. Нулевой такт длительностью 1 нс – установочный, в анализе не учитывается. В рабочей фазе выходы F, F’ инверсны друг другу, в фазе спейсера F=F’=0. Выход Г-триггера в рабочей фазе в 0, в фазе спейсера в 1. Полученный результат подтверждает работоспособность предложенного генератора функций двух переменных.

Полученные результаты топологического моделирования соответствуют результатам схемотехнического моделирования, что подтверждает правильность функционирования разработанного генератора функций на основе стандартных КМОП элементов.

## 3.2. Моделирование разработанного самосинхронного генератора логических функций по принципу LUT

### 3.2.1 Моделирование элемента LUT-ST на логическом уровне

Выполним моделирование разработанной модели элемента LUT-ST [92,96-97] на две переменные, изображённой на рисунке 2.8, в системе схемотехнического моделирования NI Multisim фирмы National Instruments Electronics Workbench Group [58], используя модель транзисторов BSIM 4.8.0 (Level 14).

Результат моделирования показан на рисунке 3.7. Блоки X1, neX1, X2, neX2 реализуют блоки входных наборов. Схема блока показана на рисунке 3.8. Блоки Ch1_1, Ch2_1, Ch1_2, Ch2_2, Ch1_3, Ch2_3 – реализуют адаптированный LUT с цепочкой спейсера. Схема блока показана на рисунке 3.9. Значения настройки S0-S3 и переменные X1, X2 моделируются генератором слов (XWG1). В схеме присутствуют блоки инверсии (NOT), блоки индикаторов входа (I11, I2), блоки индикаторов выходов (I12, I13, I1) и блоки G-триггеров (G1, G2) – рисунок 3.10.

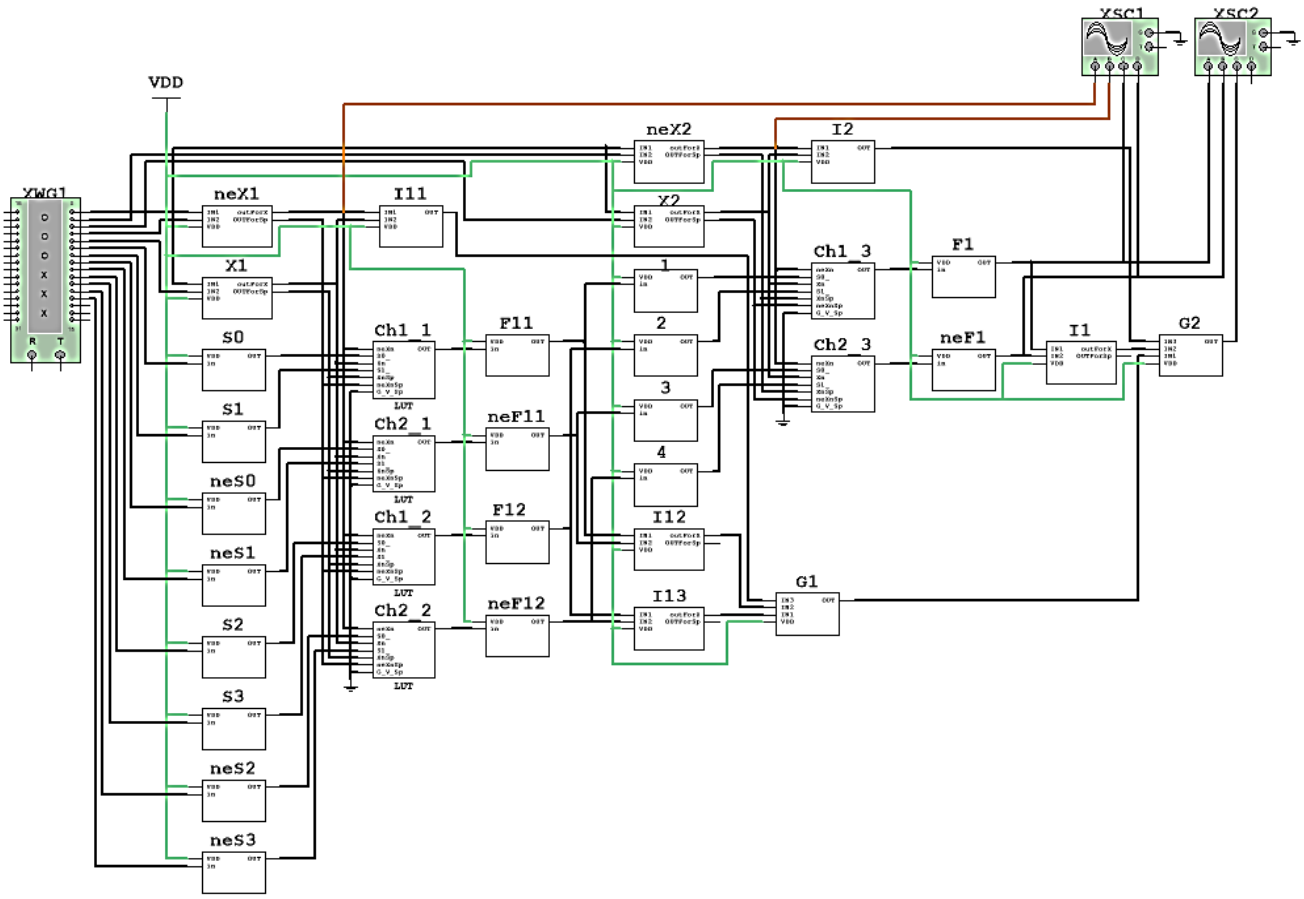



Рисунок 3.7 –Моделирование элемента 2LUT-ST с разомкнутой обратной связью.

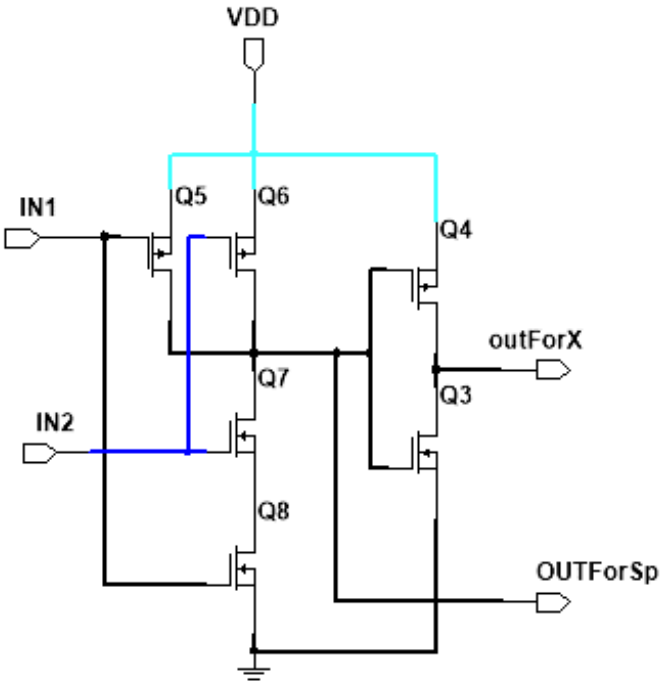



Рисунок 3.8 – Блок входного набора.

Блок входного набора первого слоя реализует функцию 2И-НЕ+НЕ. Транзисторы Q5-Q8 реализуют функцию 2И-НЕ, с выхода которой сигнал передается на затворы транзисторов, реализующих фазу спейсера. Выход OUTForSp активен только в фазе спейсера. Транзисторы Q3-Q4 реализуют инвертор, выход outForX подключается к затворам транзисторов, реализующих рабочую фазу.

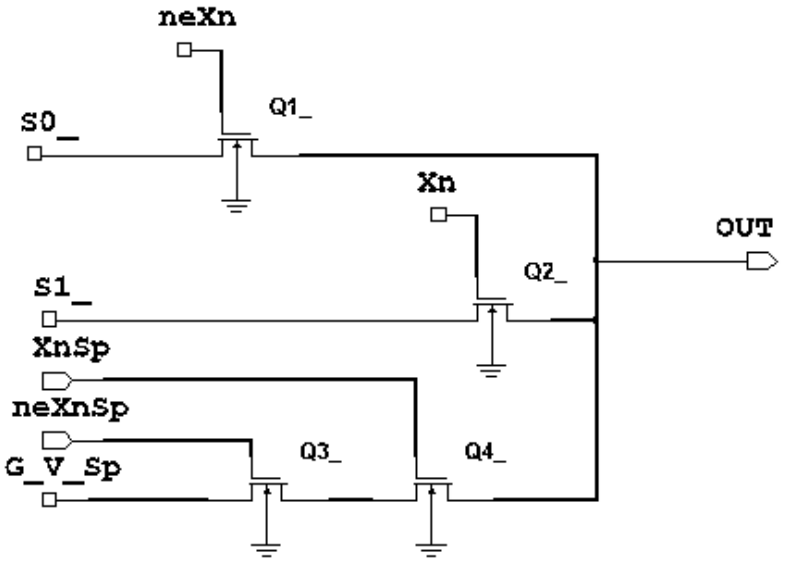



Рисунок 3.9 – Блок LUT одной переменной с цепочкой спейсера.

На рисунке 3.9 транзисторы Q1_, Q2_ реализуют дерево передающих транзисторов, как в известном LUT, а также реализуют рабочую фазу. Транзисторы Q3-Q4 реализуют фазу спейсера.

Входы S0 и S1 предназначены для настройки SRAM, входы neXn, Xn – для подключения переменных, которые поступают через выход outForX, входы neXnSp, XnSp предназначены для подключения выхода OUTForSp.

Значение со входа S1_ передается на выход только когда Xn=1, тогда транзистор Q2 открыт, транзистор Q1 – закрыт. Значение со входа S0_ передается на выход при neXn=1. Если возникла ситуация, когда Xn= neXn=1 это означает, что возникла неисправность, т.к. данный набор является запрещенным.

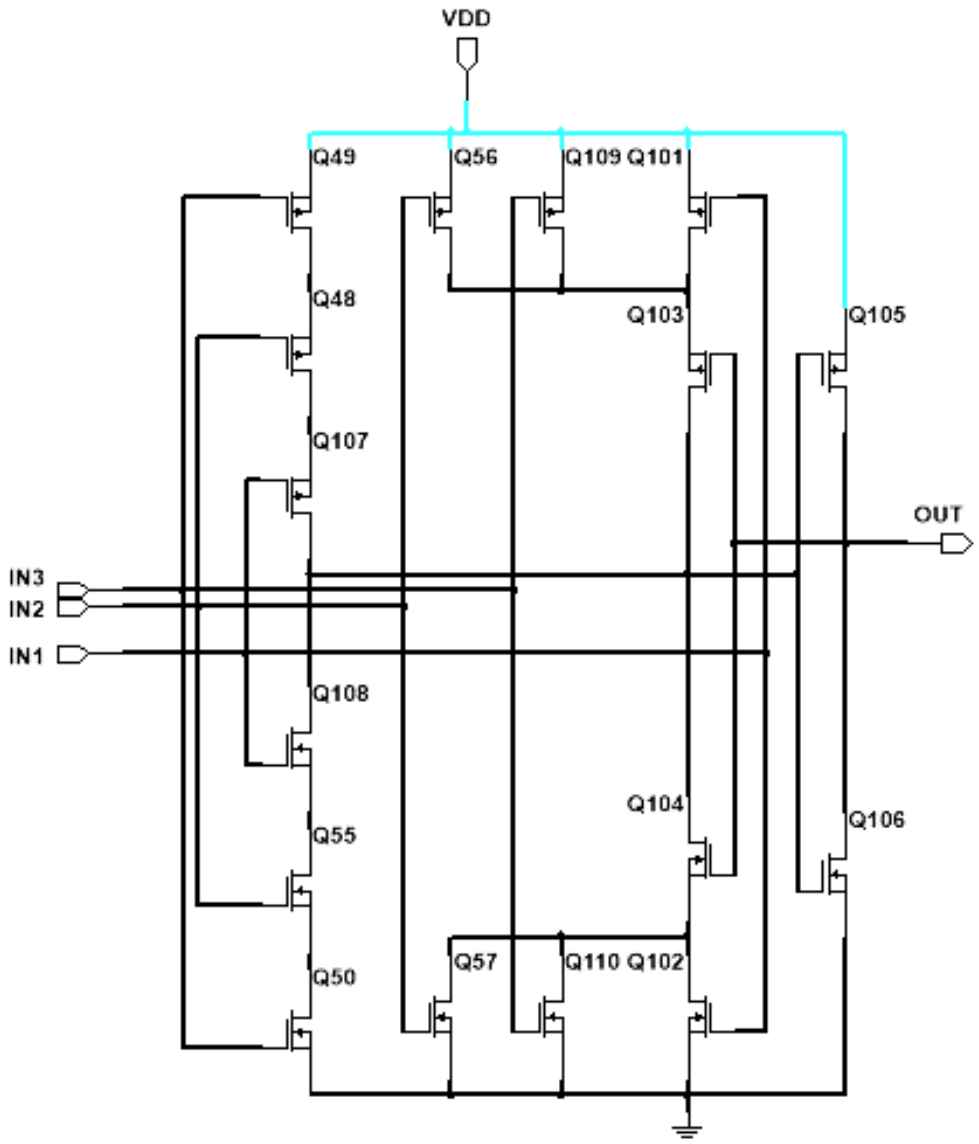



Рисунок 3.10 – Блок трехвходового G-триггера.

Выход OUT устанавливается в 1, когда все входы IN1, IN2, IN3 в логической 1, если все входы в логическом 0, то OUT=0, во всех остальных состояниях входов IN1, IN2, IN3 OUT хранит предыдущее состояние.

Для анализа работы схемы на рисунке 3.7 с помощью генератора XWG1 задаются значения входных переменных X1 и X2, а с помощью осциллографов XSC1 и XSC2 снимаются выходные показатели схемы. На осциллографе XSC2 отражается зависимость выходов основного и инверсного каналов от входных переменных X1 и X2 (рисунок 3.11), а на XSC1 общий индикатор (G-триггер), который формирует сигнал окончания переходных процессов в схеме (рисунок 3.12).

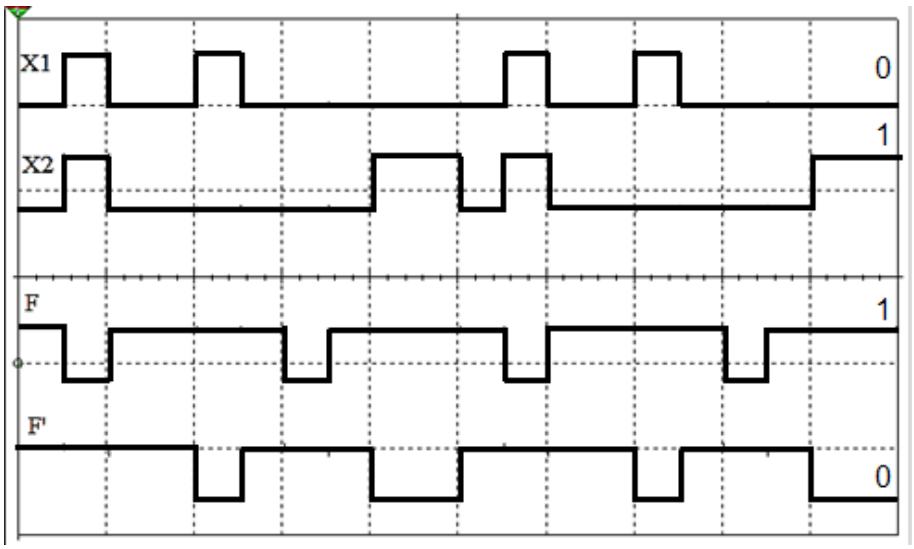



Рисунок 3.11– Диаграммы зависимости основного и двойственного каналов от входных переменных, настройка «Исключающее ИЛИ».

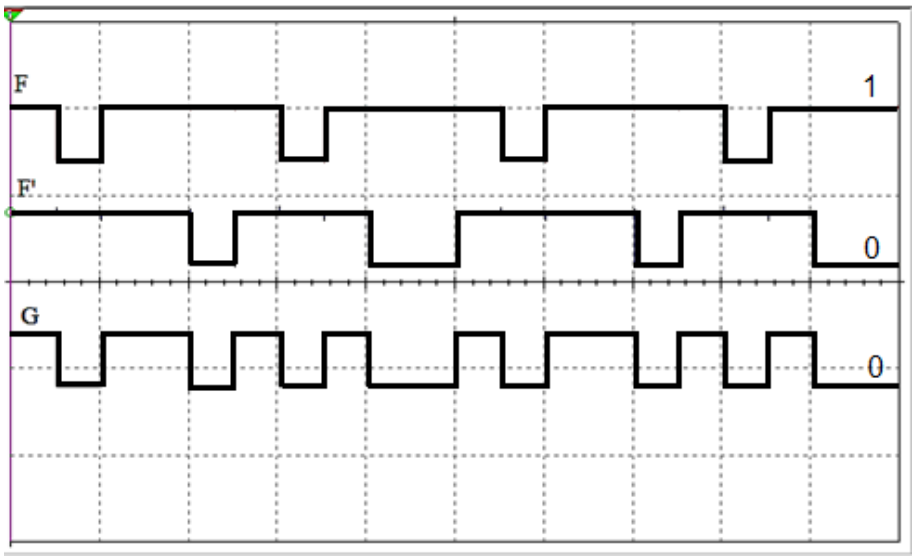



Рисунок 3.12 – Диаграммы зависимости выхода общего индикатора от основного и двойственного каналов, настройка «Исключающее ИЛИ».

На рисунке 3.11 видно, что в рабочей фазе выходы основного и двойственного каналов инверсны друг друга, а в фазе спейсера одинаковы. Каждая рабочая фаза {10} и {01} разделены фазой спейсера, при которой F=F'=1. На рисунке 3.12 выход общего индикатора в фазе спейсера равен 1, а рабочей фазе 0. Функция «исключающее ИЛИ» реализуется верно, поэтому полученные диаграммы подтверждают работоспособность предложенного элемента.

Выполним реализацию разных функций, результат анализа сведем в таблицы истинности для каждой из реализованных функций. Результаты анализа

элемента на всех входных значениях SRAM приведены в приложении №6 таблица П6.2.

Таблица 3.6. – Таблица истинности генератора функций двух переменных. Функция «исключающее ИЛИ».

| D0D1D2D3 | X1X2 | F | F’ |
|---|---|---|---|
| 0110 | 00 | 0 | 1 |
| | 01 | 1 | 0 |
| | 11 | 0 | 0 |
| | 10 | 1 | 1 |

Таблица 3.7. – Таблица истинности генератора функций двух переменных. Функция «Конъюнкция».

| D0D1D2D3 | X1X2 | F | F’ |
|---|---|---|---|
| 1000 | 00 | 0 | 1 |
| | 01 | 0 | 1 |
| | 11 | 1 | 0 |
| | 10 | 0 | 1 |

Таблица 3.8. – Таблица истинности генератора функций двух переменных. Функция «Обратная импликация».

| D0D1D2D3 | X1X2 | F | F’ |
|---|---|---|---|
| 1011 | 00 | 1 | 0 |
| | 01 | 0 | 1 |
| | 11 | 1 | 0 |
| | 10 | 1 | 0 |

Таблица 3.9. – Таблица истинности генератора функций двух переменных. Функция «стрелка Пирса».

| D0D1D2D3 | X1X2 | F | F’ |
|---|---|---|---|
| 0001 | 00 | 1 | 0 |
| | 01 | 0 | 1 |
| | 11 | 0 | 1 |
| | 10 | 0 | 1 |

Таблица 3.10. – Таблица истинности генератора функций двух переменных. Функция «Эквивалентность».

| D0D1D2D3 | X1X2 | F | F’ |
|---|---|---|---|
| 1001 | 00 | 1 | 0 |
| | 01 | 0 | 1 |
| | 11 | 1 | 0 |
| | 10 | 0 | 1 |

Таким образом, выполняя конфигурирование генератора функций на основе LUT, можно получить любую функцию, при этом самосинхронность разработанного элемента сохраняется. Другие разработанные модели и их анализ показаны в приложении №6.

### 3.2.2. Моделирование элемента LUT-ST на топологическом уровне

Выполним построение топологии элемента 2LUT-ST, показанного на рисунке 2.8. Топология элемента показана на рисунке 3.13.

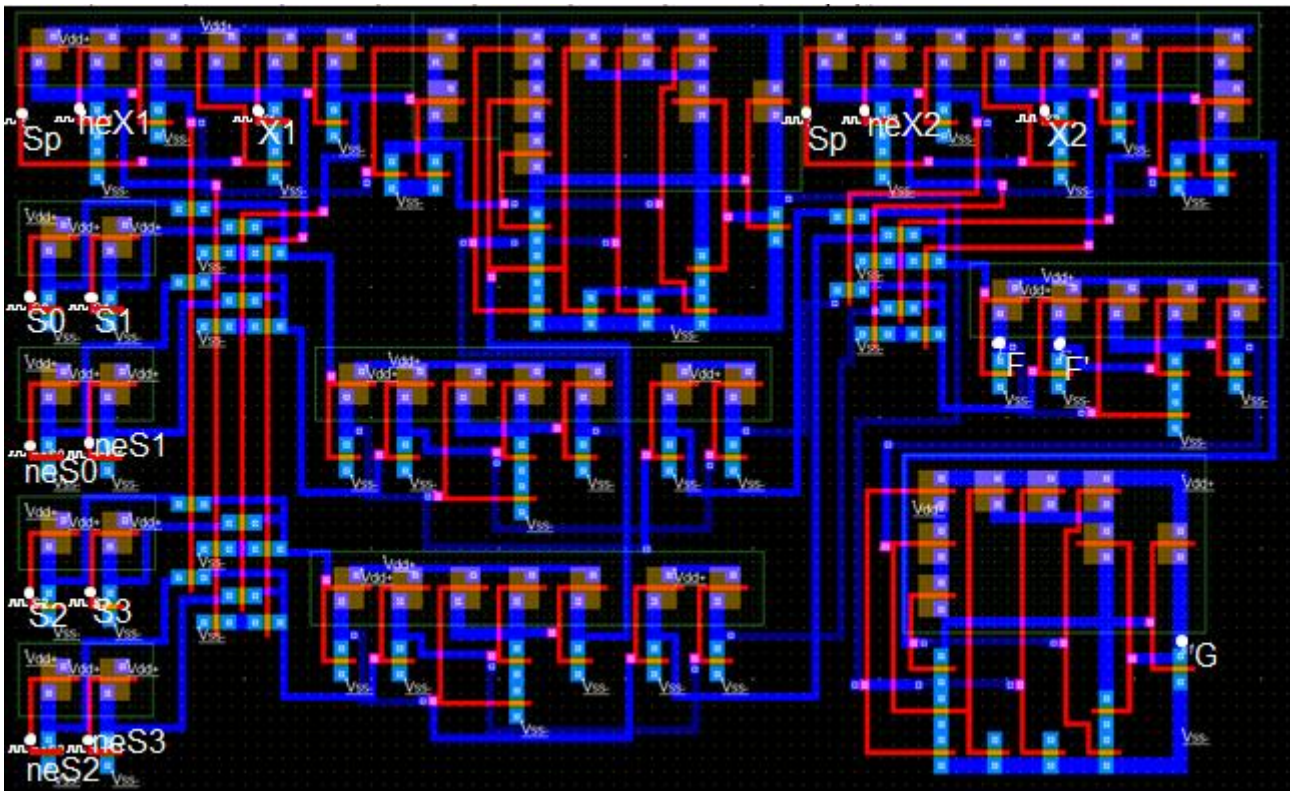



Рисунок 3.13 – Топология элемента 2LUT-ST без восстановления уровня сигналов.

Вход Sp – реализует фазу спейсера и служит для подключения обратной связи с выхода последнего Г-триггера. X1, neX1, X2, neX2 – входы переменных первого каскада и второго соответственно. S0-S3 – входы настройки прямого канала, neS0-neS3 – входы настройки двойственного канала, которые позволяют выполнить конфигурацию элемента константами или выполнить подключение к ячейками SRAM с загруженными таблицами истинности логической функции. F – выход прямого канала, F’ – выход двойственного канала. G – выход последнего гистерезисного триггера схемы.

Выполним проверку реализации функции «исключающее ИЛИ» - таблица истинности 3.6

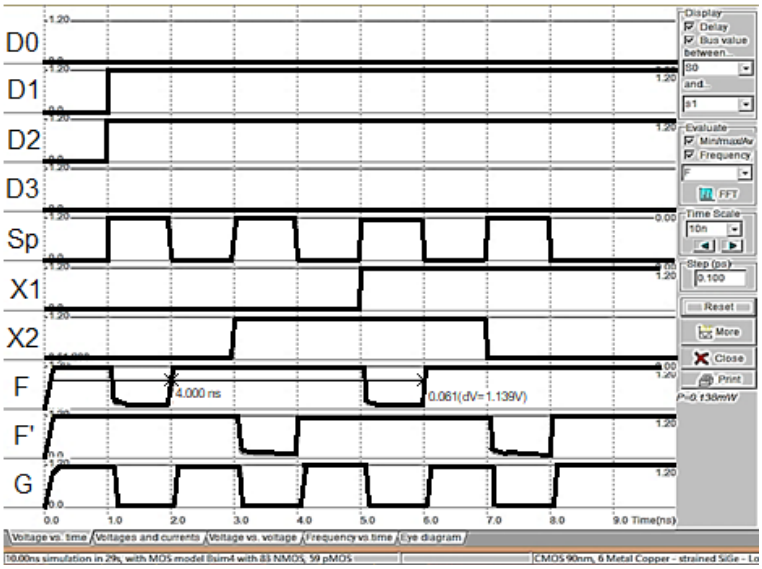



Рисунок 3.14 - Результат моделирования элемента 2LUT-ST без восстановления уровня сигналов. Настройка функции «исключающее ИЛИ».

На рисунке 3.14 представлен результат работы элемента 2LUT-ST, соответствующий таблице истинности 3.6, но каналы F и F’ не устанавливаются в логический 0, а остаются на уровне 0,061В – это означает, что в схеме присутствуют токи утечки. Для решения данной проблемы в [97] предложено несколько вариантов восстановителей. Используем один из них – восстановитель на базе инвертора с обратной связью (рисунок 3.15). Транзисторы Q1-Q2 реализуют известный инвертор. Транзистор Q3 – расчетный.

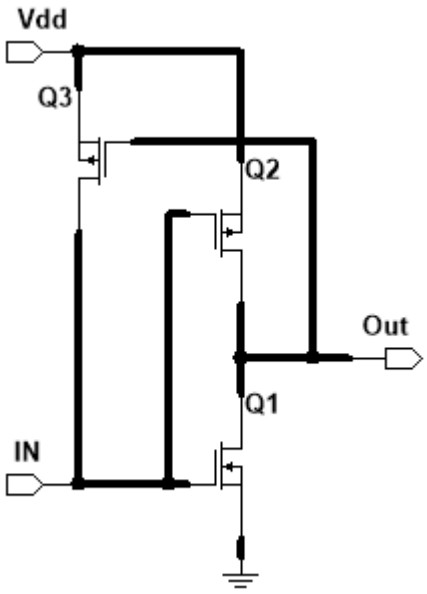



Рисунок 3.15- Схема инвертора с обратной связью.

Выполним моделирование топологии с использованием инвертора с обратной связью.

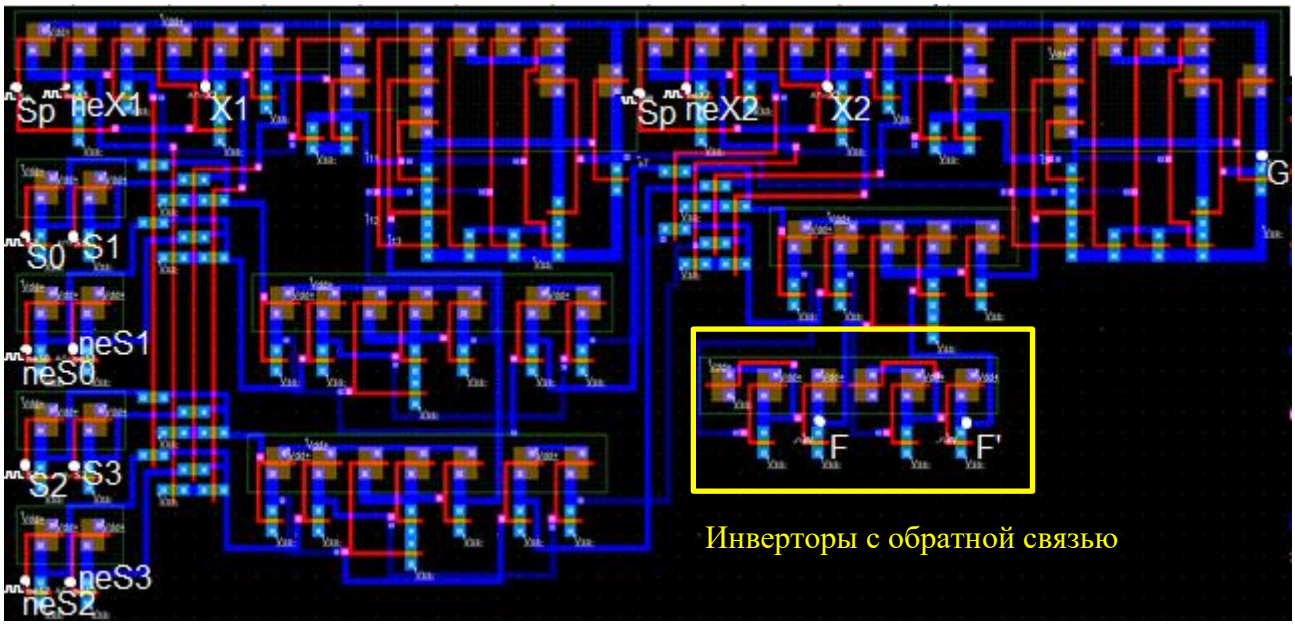



Рисунок 3.16 – Топология элемента 2LUT-ST с восстановителем.

На рисунке 3.17 видно, что, в отличии от результата, представленного на рисунке 3.15, в канале F остаточное напряжение равно 0. Поэтому выбранный восстановитель уровня решает проблему, а также не нарушает свойство полумодулярности. Результат реализации функции «исключающее ИЛИ» соответствует таблице истинности 3.6. Нулевой такт длительностью 1 нс – установочный, в анализе не учитывается. В рабочей фазе выходы F, F’ инверсны

друг другу, в фазе спейсера F=F'=1. Выход Г-триггера в рабочей фазе в 0, в фазе спейсера в 1. Полученный результат подтверждает работоспособность разработанного генератора функций на основе LUT.

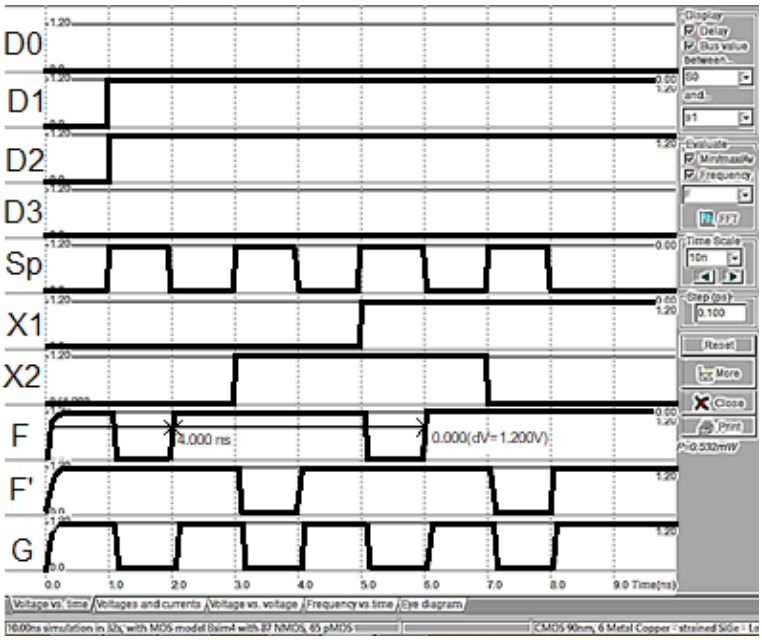



Рисунок 3.17- Результат моделирования элемента 2LUT-ST с восстановителем. Настройка функции «исключающее ИЛИ».

Выполним реализацию функции «Эквивалентность» - таблица истинности 3.10.

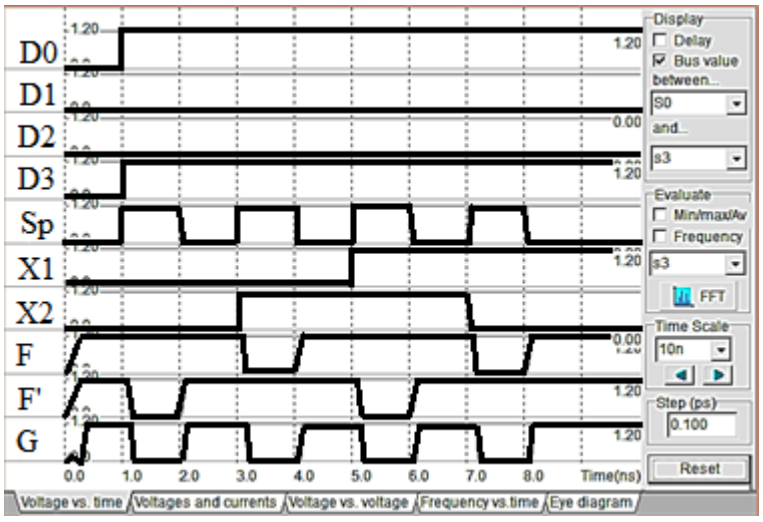



Рисунок 3.18 - Результат моделирования элемента 2LUT-ST с восстановителем. Настройка функции «Эквивалентность».

На рисунке 3.18 показан результат реализации функции «Эквивалентность», который соответствует таблице истинности 3.10. Нулевой такт длительностью 1

нс – установочный, в анализе не учитывается. В рабочей фазе выходы F, F’ инверсны друг другу, в фазе спейсера F=F’=1. Выход Г-триггера в рабочей фазе в 0, в фазе спейсера в 1.

Полученные результаты топологического моделирования соответствуют результатам схемотехнического моделирования, что подтверждает правильность функционирования разработанного генератора функций на основе LUT.

## 3.3. Моделирование разработанного элемента DC-LUT-ST

### 3.3.1 Моделирование DC-LUT-ST на логическом уровне

Выполним моделирование схемы, изображённой на рисунке 2.11 [99], в системе схемотехнического моделирования NI Multisim [58], используя модель транзисторов BSIM 4.8.0 (Level 14).

Для построения схемы будут использованы следующие иерархические блоки:

1. Блок входного набора: neX1, X1, neX2, X2 – принципиальная электрическая схема показана на рисунке 3.9;
2. Блоки инверторов: F0, neF0, F1, neF1, 1, 2, 3, 4, F2, neF2, F3, neF3, F4, neF4, F5, neF5 – реализуют функцию НЕ;
3. Блоки индикаторов входов: I1, I2 –реализуют функцию 2ИЛИ-НЕ;
4. Блоки индикаторов выходов: If0, If1, If2, If3, If4, If5 – реализуют функцию 2И;
5. Блоки трех входовых G-треггеров: G1, G2, G3 – принципиальная электрическая схема показана на рисунке 3.11;
6. Блоки двух входовых G-треггеров: НВ1 – принципиальная электрическая схема показана на рисунке 3.19;
7. Блоки каналов: Ch1_1, Ch2_1, Ch1_2, Ch2_2, Ch1_3, Ch2_3 – принципиальная электрическая схема показана на рисунке 3.20.

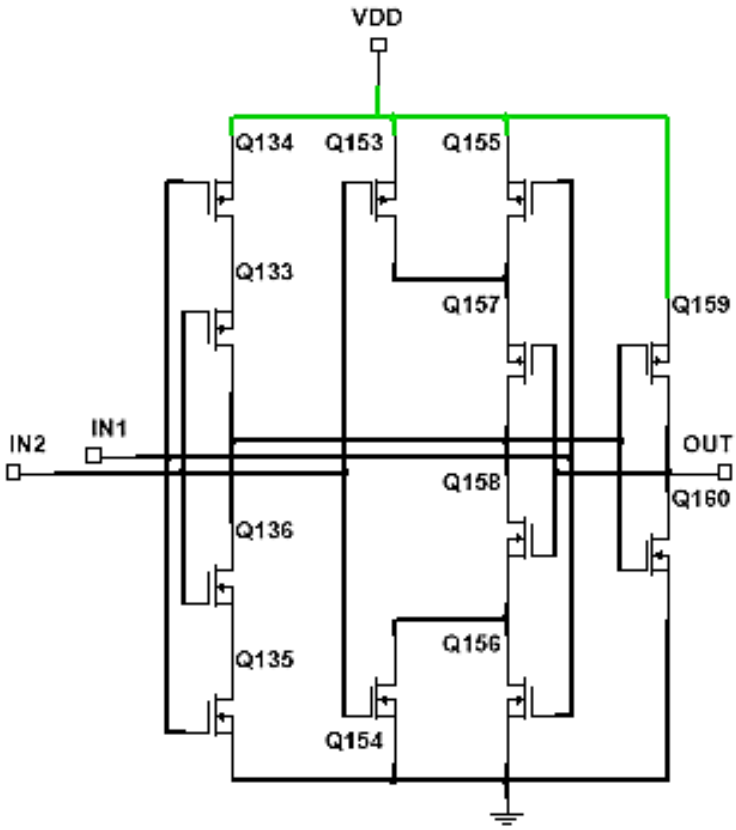



Рисунок 3.19 – Блок трех входового G-триггера.

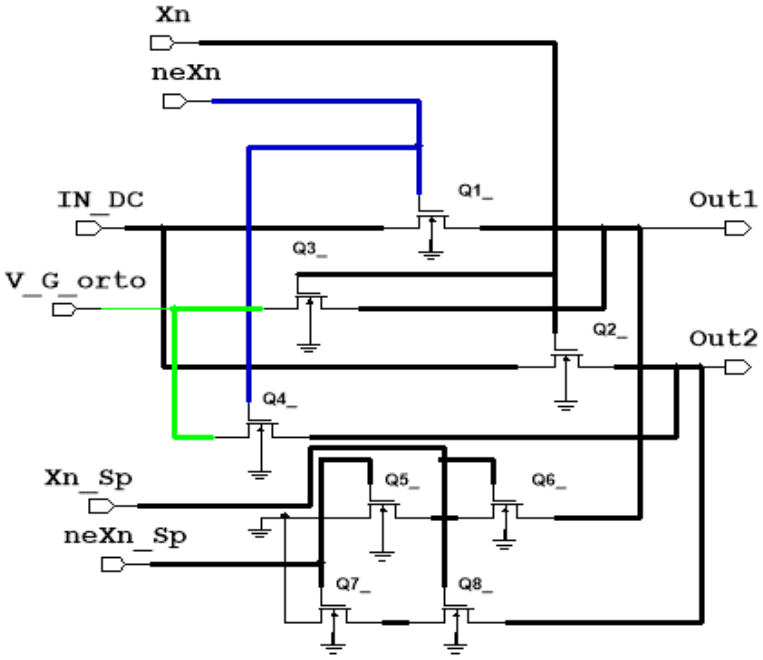



Рисунок 3.20 – Блок каналов.

На рисунке 3.20 транзисторы Q1, Q2 образуют канал DC LUT, транзисторы Q3, Q4 предназначены для обеспечения ортогональности сигналов по каждому выходу элемента. Транзисторы Q5, Q6 и Q7, Q8 реализуют цепочки спейсеров каждого канала. Входы neXn, Xn предназначены для ввода значения переменной

с блока входного набора. Входы neXn_Sp и Xn_Sp – входы для организации спейсера. Вход IN_DC служит для подключения канала к шине питания или шине земли в зависимости от типа канала, или к выходу предыдущего слоя для построения элемента на большее число переменных. Вход V_G_orto предназначен для подключения транзисторов Q3, Q4 к шине питания или шине земли в зависимости от типа канала. На рисунке 3.20 реализован известный DC LUT [8], адаптированный к условиям работы в ССС при помощи цепочек спейсера.

Кроме иерархических блоков для анализа работы элемента в схеме присутствуют генератор слов XWG1 и осциллографы XSC1- XSC5. Разработанный элемент DC LUT ST двух переменных показан на рисунке 3.21.

На рисунке 3.22 видно, что каналы работают правильно, в рабочей фазе все каналы попарно инверсны друг другу, а на выходе общего индикатора схемы логический 0. В фазе спейсера все выходы каналов и выход индикатора в логической 1, что подтверждает правильность работы предложенного элемента.

Сведем полученные результаты анализа в таблицу 3.11.

Таблица 3.11 – Таблица истинности элемента 2DC LUT-ST с нулевым спейсером

| № | X1X2 | F0 | F0’ | F1 | F1’ | F2 | F2’ | F3 | F3’ | G |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 00 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 2 | 01 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 3 | 11 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 4 | 10 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |

Результаты, представленные в таблице 3.11, подтверждают работоспособность предложенного элемента.

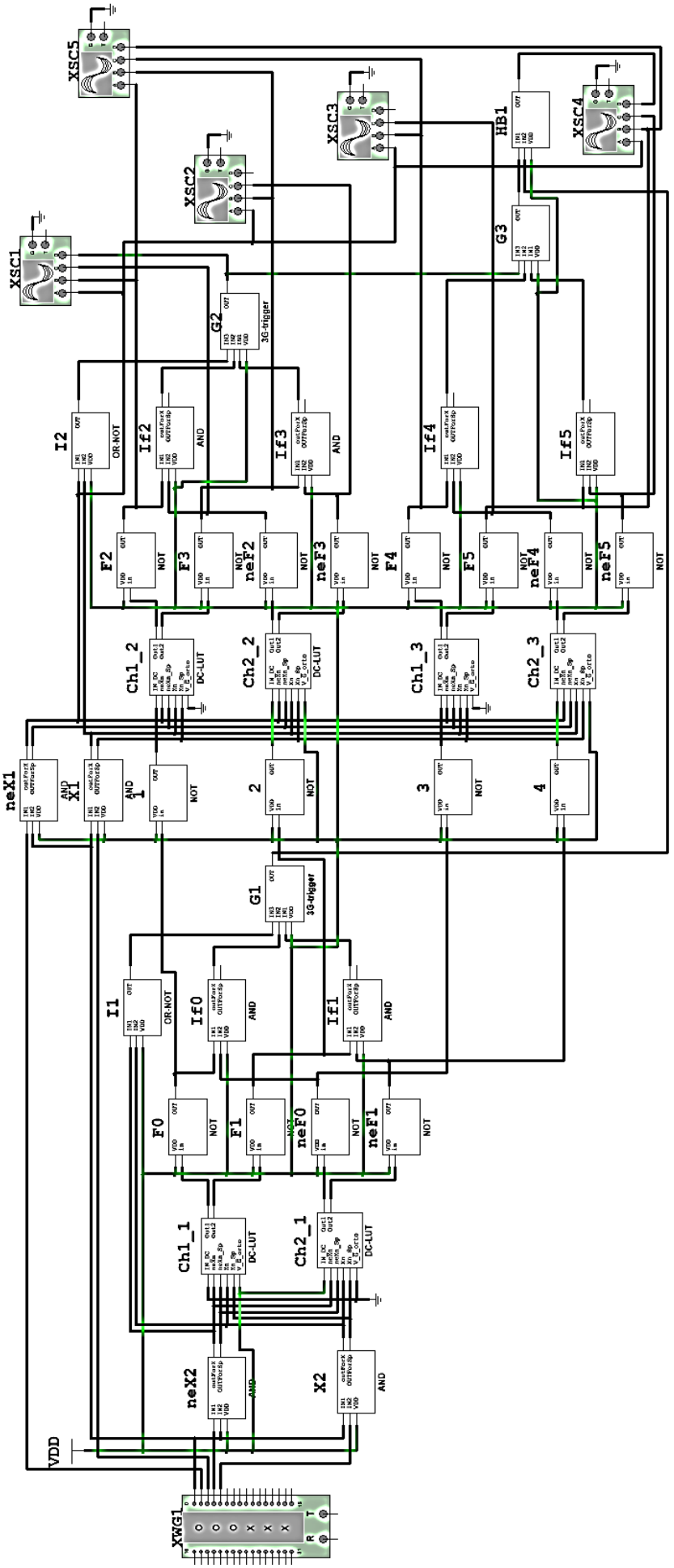
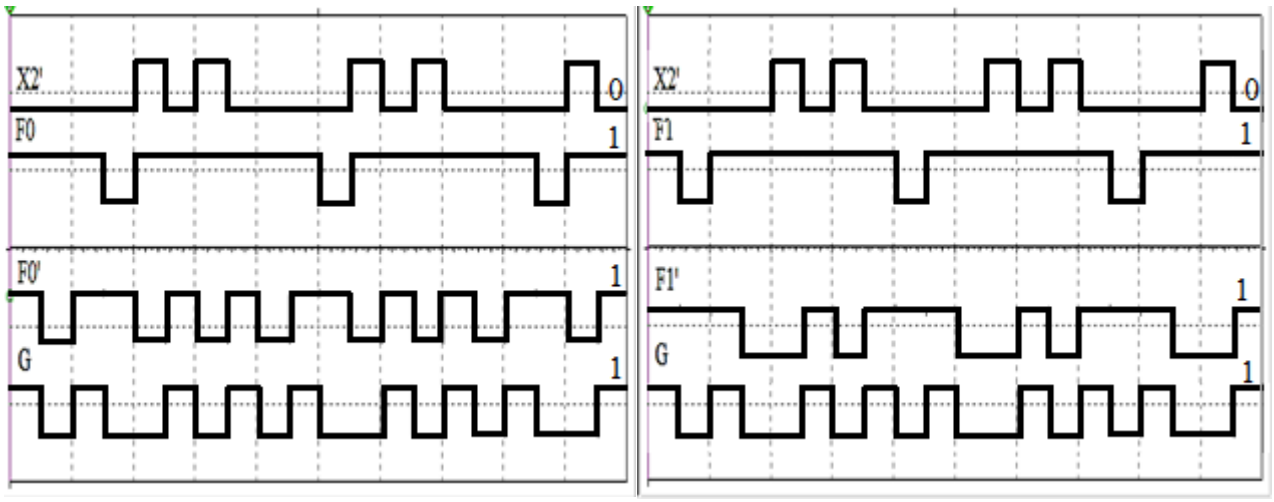
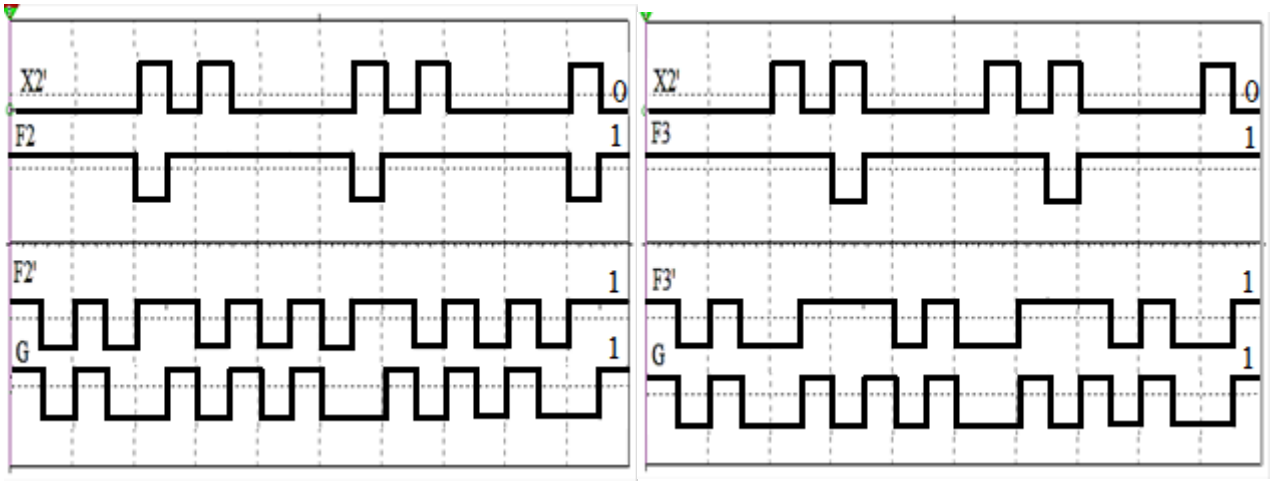



Рис.3.21 – Моделирование элемента 2DC LUT-ST.




а) б)




в) г)

Рисунок 3.22 – Результат моделирования элемента 2DC LUT-ST:

а) зависимость выходов F0, F0’, G от входных переменных;

б) зависимость выходов F1, F1’, G от входных переменных;

в) зависимость выходов F2, F2’, G от входных переменных;

г) зависимость выходов F3, F3’, G от входных переменных.

Другие разработанные модели и их анализ показаны в приложении №6.

### 3.3.2. Моделирование элемента DC-LUT-ST на топологического уровне

Выполним построение топологии элемента DC LUT-ST двух переменных (рисунок 2.11) с восстановлением уровня сигналов, топология элемента без настройки на реализацию функции показана на рисунке 3.24. Вход Sp реализует фазу спейсера и служит для подключения обратной связи с выхода последнего

Г-триггера. X1, neX1, X2, neX2 – входы переменных первого каскада и второго соответственно. F0-F3 – выходы прямых каналов, F0’-F3’ – выходы двойственных каналов, G – выход последнего Г-триггера схемы.

На рисунке 3.23 каждый из выходов F0-F3 активен только при одном из наборов входных переменных, причем нет наборов, которые активируют сразу несколько выходов. Выходы двойственных каналов F0’-F3’ инверсны прямым каналам. Выход G гистерезисного триггера в рабочей фазе в 0, в фазе спейсера в 1. Полученный результат соответствует логике работы CCC, что подтверждает работоспособность предложенного элемента.

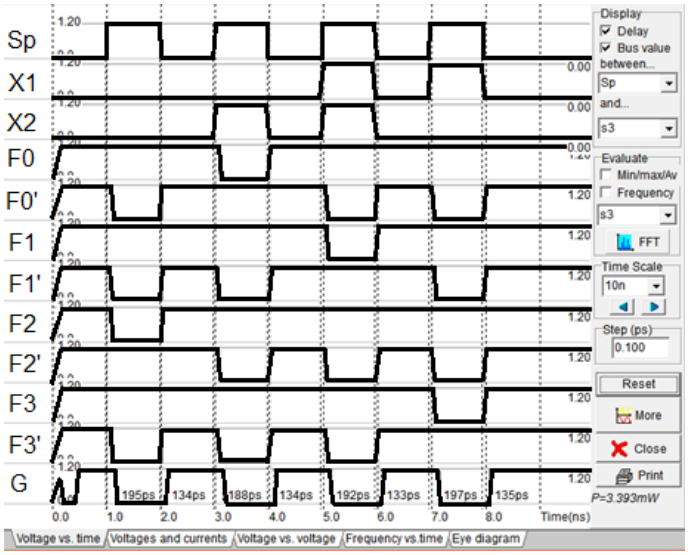



Рисунок 3.23 Результат моделирования топологии элемента 2DC-LUT-ST.

Результат моделирования, показанный на рисунке 3.23, соответствует результатам схемотехнического моделирования (рисунок 3.22) и таблице истинности 3.11 - это подтверждает правильность работы предложенного элемента.

Выполненное топологическое моделирование [74] подтверждает работоспособность разработанного логического элемента DC LUT-ST.

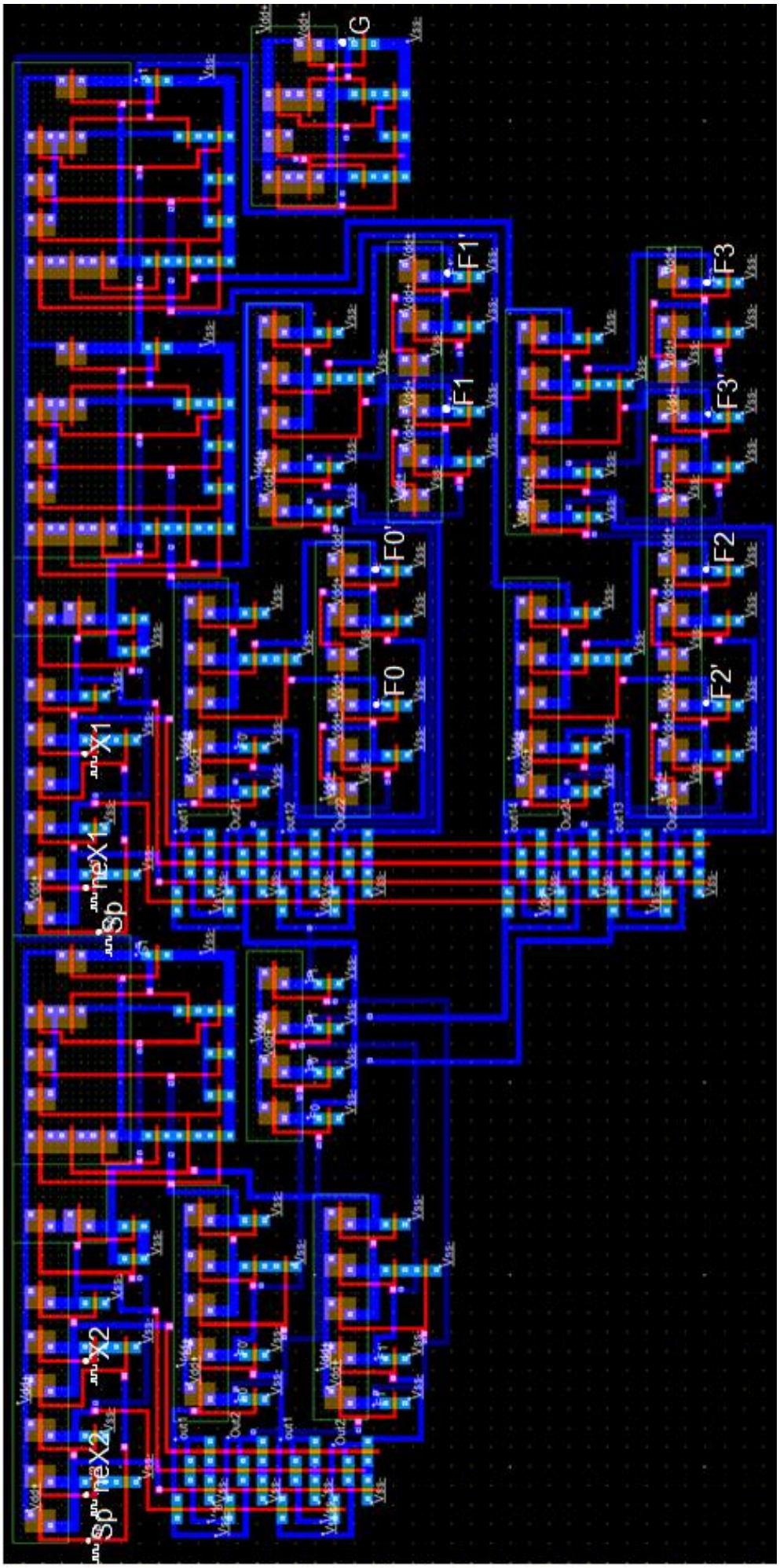



Рис. 3.24 Топология элемента 2DC-LUT-ST.

Выполним настройку работы элемента DC LUT-ST на реализацию функции «исключающее ИЛИ». Для этого подключим к элементу блок настройки, показанный на рисунке 2.13. Для реализации функции «исключающее ИЛИ» блок настройки требуется настроить следующим образом: прямой канал - D0=0, D1=1, D2=1, D3=0, двойственный канал - D0’=1, D1’=0, D2’=0, D3’=1. Топология элемента показана в приложении №5 на рисунке П5.12.

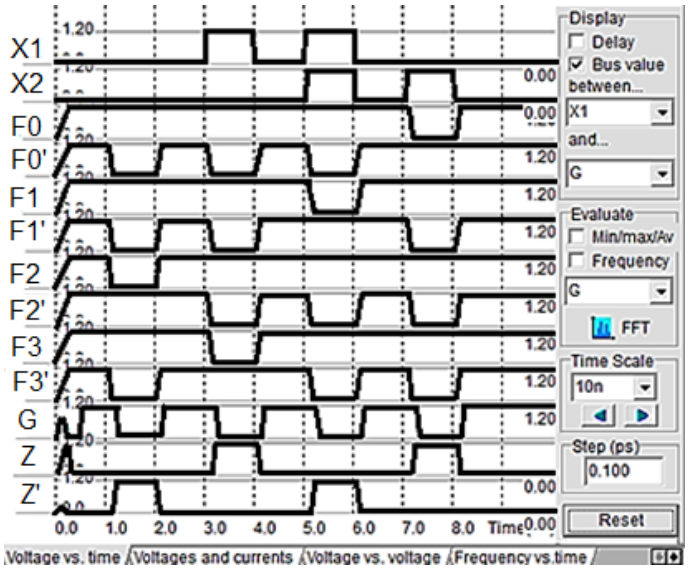



Рисунок 3.25 Результат реализации функции «исключающее ИЛИ» элемента 2DC-LUT-ST.

На рисунке 3.25 показан результат реализации функции «исключающее ИЛИ», где видно, что функция вычисляется верно: при X1=X2=0 и X1=X2=1 значение выхода Z=0, при X1=1, X2=0 и X1=0, X2=1 выхода Z=1, значения двойственного канала Z’ инверсны значениям выхода Z в рабочих фазах, в фазе спейсера Z=Z’=0, что подтверждает правильность работы элемента.

## 3.4. Моделирование блока настройки

### 3.4.1 Моделирование блока настройки на логическом уровне

Выполним моделирование блока настройки, показанного на рисунке 2.13. В качестве средства моделирования использован САПР «Ковчег» [57]. Для реализации прямого канала используются библиотечные элементы OR3B1, в которых один вход инверсный, что позволяет без дополнительного внешнего инвертора реализовать сигнал разрешения Sp. В качестве элементов,

реализующих двойственный канал, используются библиотечные элементы A21O3, которые также имеют один инверсный вход. Кроме библиотечных элементов используются стандартные КМОП элементы, а также элементы, входящие в самосинхронную библиотеку GI2 – двухвходовые Г-триггеры.

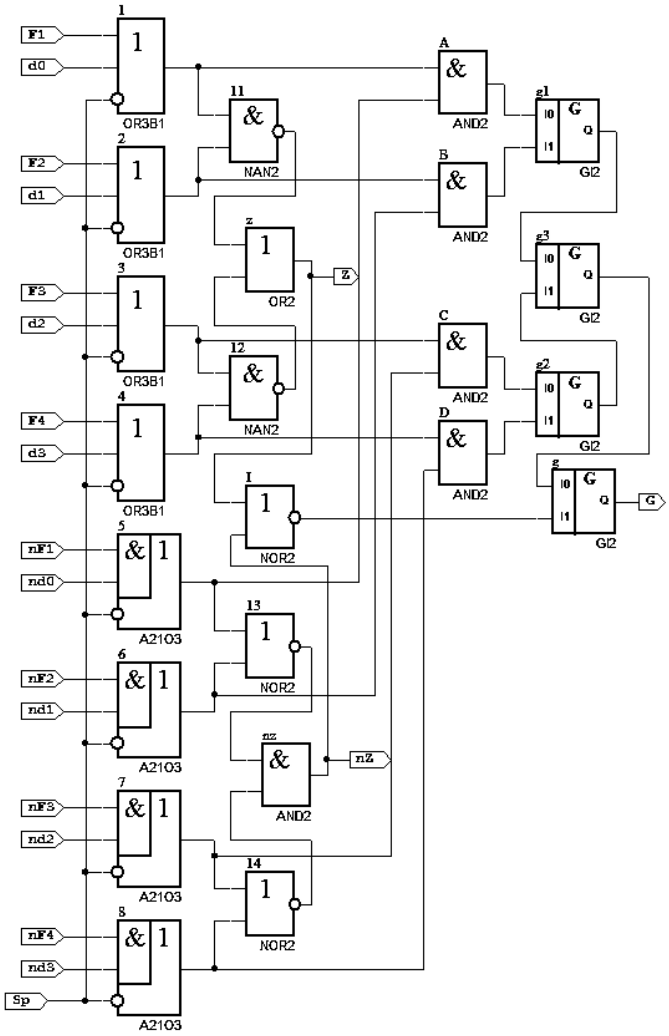



Рисунок 3.26 Моделирование блока настройки для подключения четырех выходов.

Выполним анализ схемы на примере функции «Исключающее ИЛИ», для этого выполним конфигурацию элемента константами: D0=1, D1=0, D2=0, D3=1.


Рисунок 3.27 Результат моделирования блока настройки, для подключения четырех выходов.

На рисунке 3.27 показан результат работы блока настройки на реализацию функции «исключающее ИЛИ»

### 3.4.2 Моделирование блока настройки на топологическом уровне

Выполним моделирование блока настройки, показанного на рисунке 2.13, на топологическом уровне.

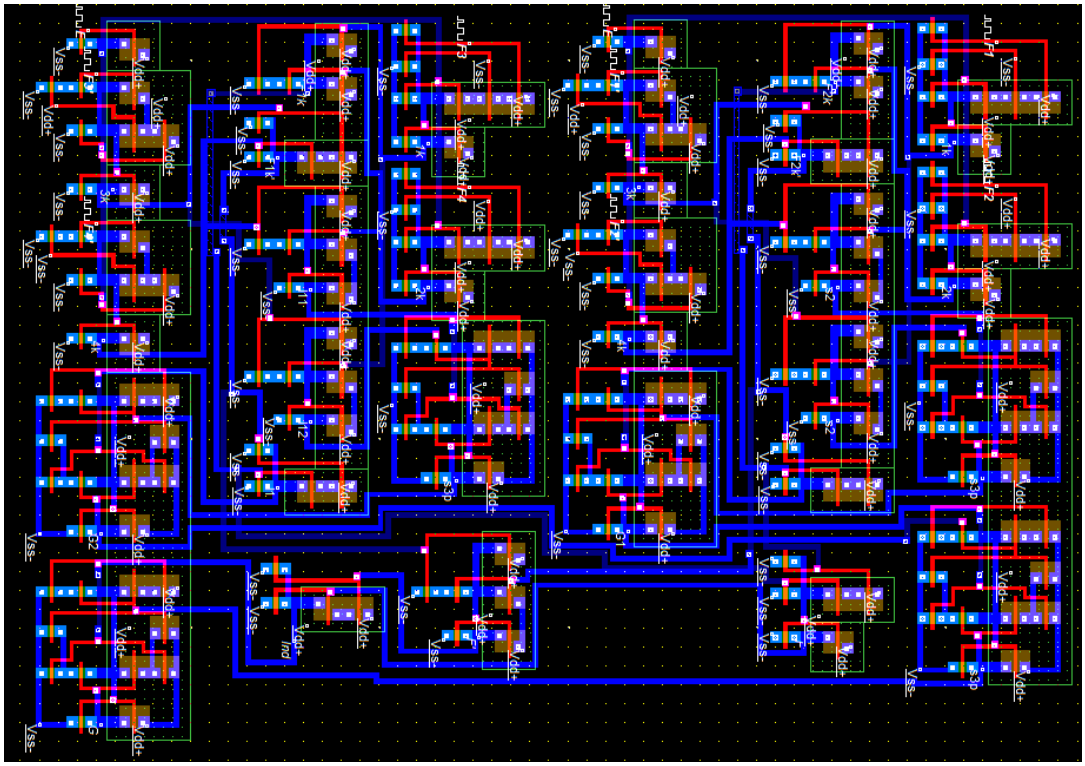


Рисунок 3.28 Топология блока настройки, для подключения четырех выходов.

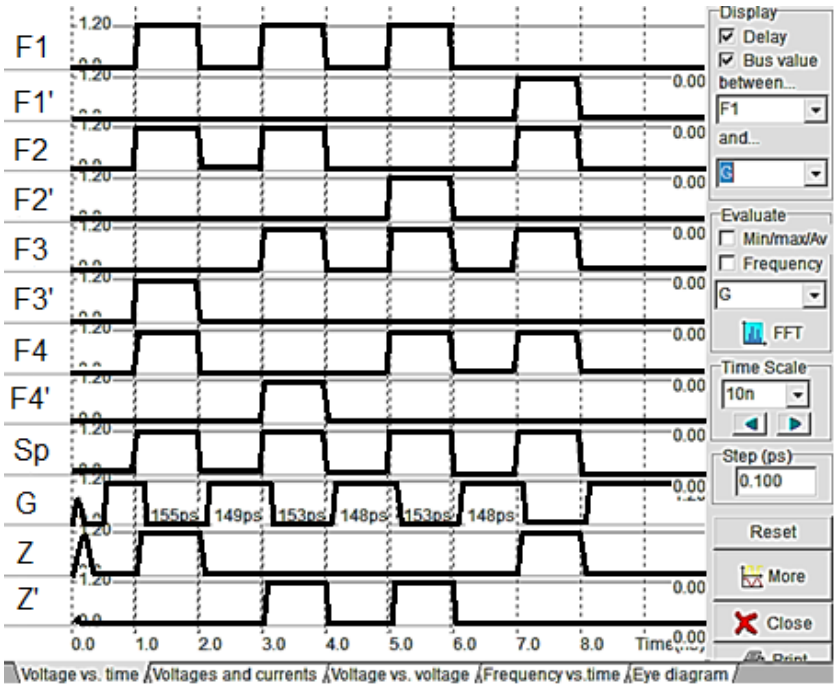



Рисунок 3.29 Результат моделирования блока настройки для подключения четырех выходов.

На рисунке 3.29 результат моделирования подтверждает правильность работы схемы.

## 3.5. Моделирование разработанных самосинхронных генераторов систем логических функций, заданных в ДНФ

### 3.5.1 Моделирование блока конъюнкций с использованием стандартных элементов на логическом уровне

Выполним моделирование разработанного элемента, показанного на рисунке 2.16. Для реализации прямого канала используются библиотечные элементы A21O, в которых один вход инверсный, что позволяет без дополнительного внешнего инвертора реализовать сигнал разрешения Sp. В качестве элементов, реализующих двойственный канал, используются библиотечные элементы OR3, которые также имеют один инверсный вход. Кроме библиотечных элементов используются стандартные КМОП элементы, а также элементы, входящие в самосинхронную библиотеку GI2 – двухвходовые Г-триггеры.

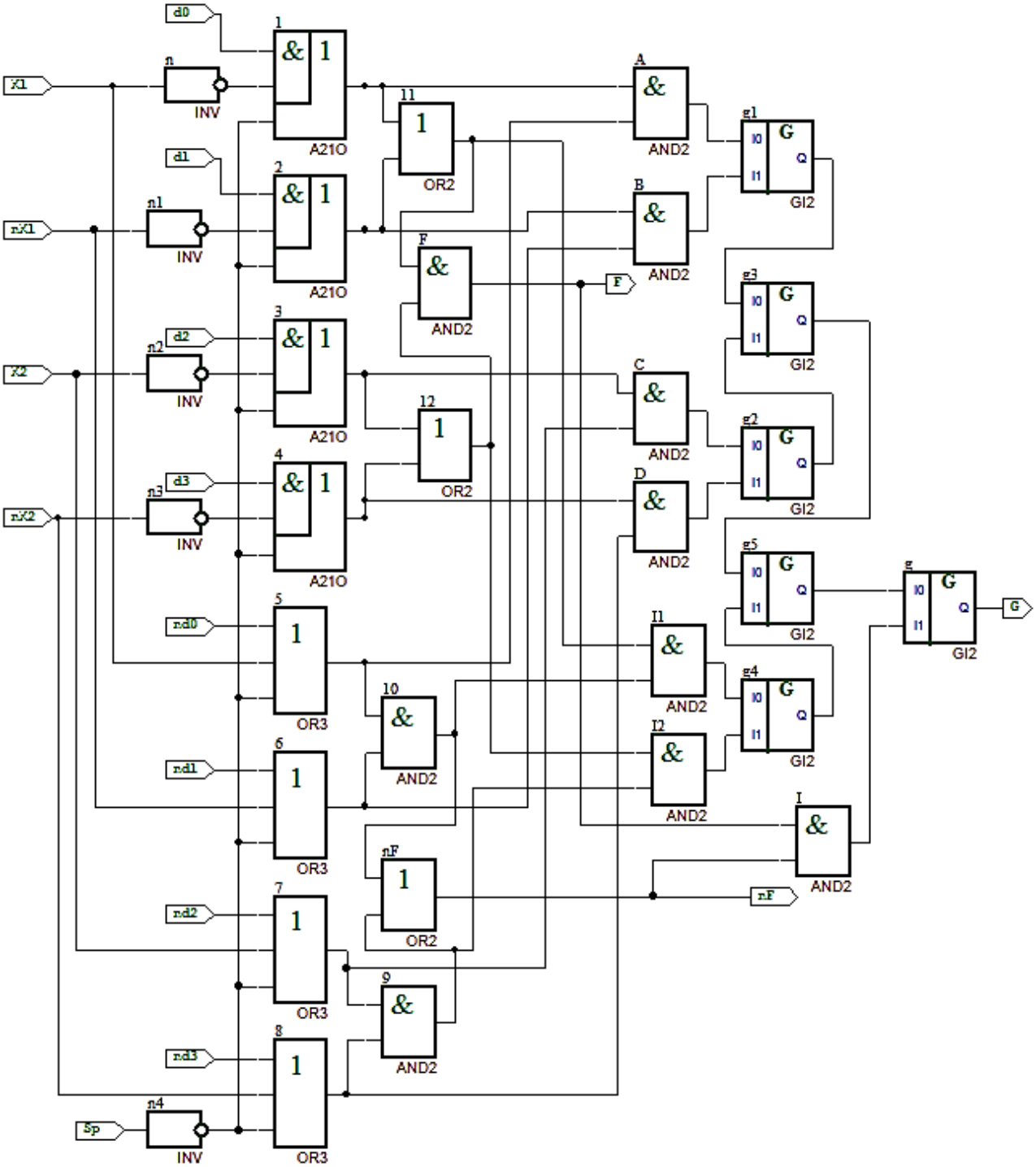



Рисунок 3.30 Схема блока конъюнкций двух разрядов на основе библиотечных элементов.

Выполним анализ схемы на примере функции «Исключающее ИЛИ», для этого выполним конфигурацию элемента константами: D0=0, D1=1, D2=1, D3=0.

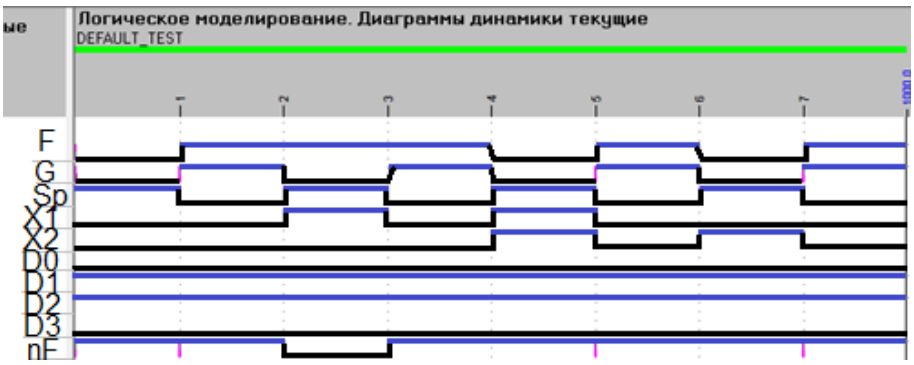



Рисунок 3.31 Результат моделирования блока конъюнкций двух разрядов на основе стандартных элементов.

На рисунке 3.31 буквами обозначены: F – выход прямого канала, nF – выход двойственного канала, G – выход последнего Г-триггера. Реализация функции «исключающее ИЛИ» выполняется верно, например, при X1=X2=1 значение выхода F=0, а канал nF=1, поэтому результат моделирования подтверждает правильность работы схемы.

### 3.4.2 Моделирование блока конъюнкций с использованием стандартных элементов на топологическом уровне

Выполним топологическое моделирование разработанного элемента, показанного на рисунке 2.16. Топология элемента показана на рисунке 3.33. Результат моделирования топологии показан на рисунке 3.32.

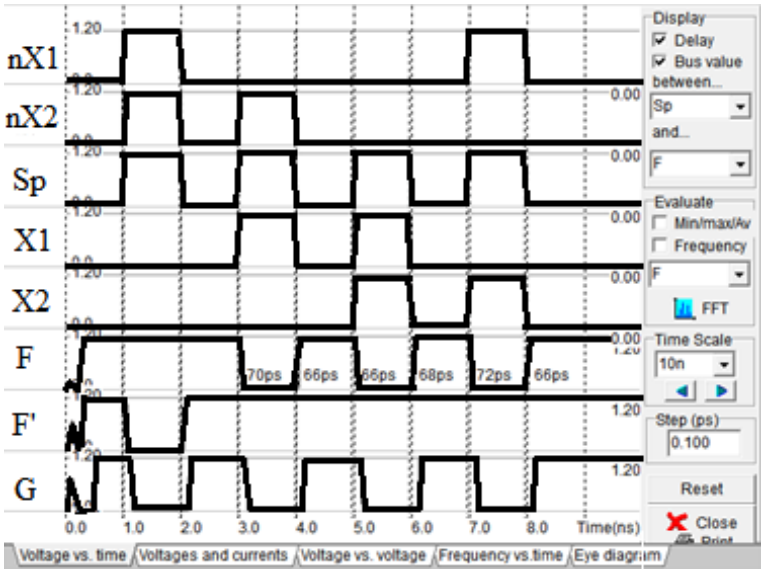



Рисунок 3.32 Результат моделирования блока конъюнкций двух разрядов на основе стандартных элементов.

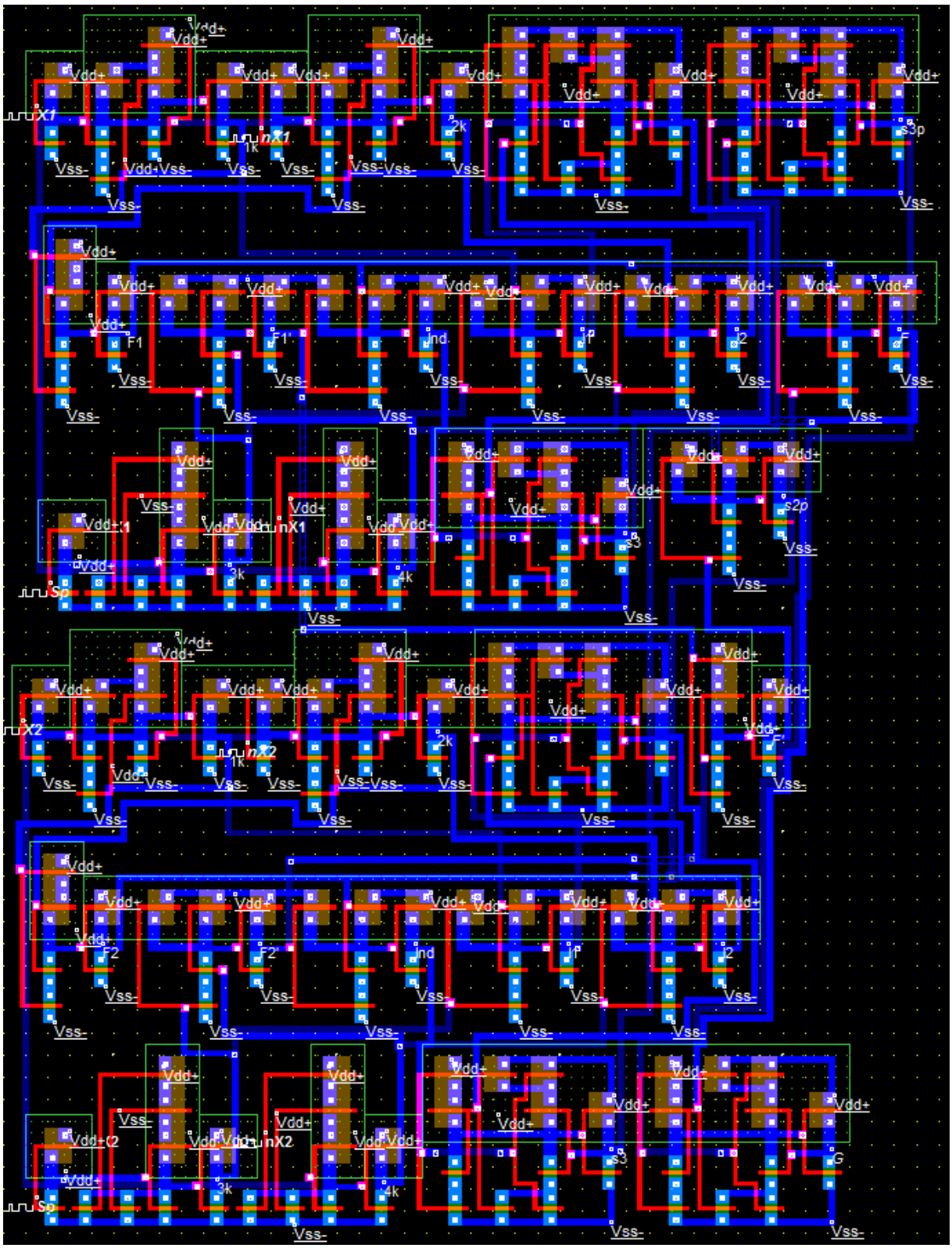



Рисунок 3.33 Топология блока конъюнкций двух разрядов на основе библиотечных элементов.

Результат на рисунке 3.32 соответствует результату моделирования на рисунке 3.31, что подтверждает правильность работы разработанного элемента. Разработанные топологии показаны в приложении №5.

### 3.4.3 Моделирование блока конъюнкций с использованием подтягивающих резисторов на топологическом уровне

Выполним моделирование разработанного элемента, показанного на рисунке 2.20.

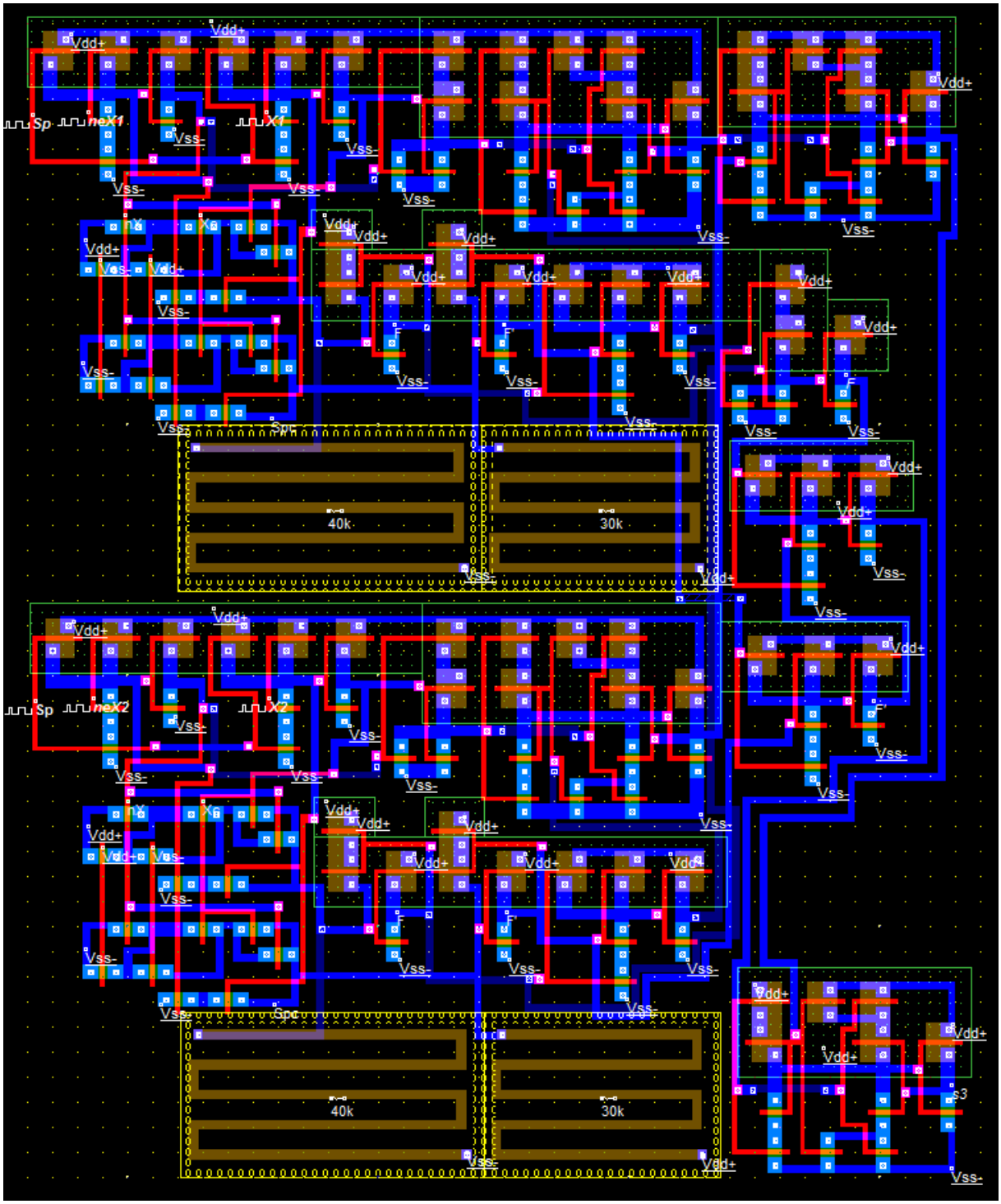



Рисунок 3.34 Топология блока конъюнкций двух разрядов с использованием подтягивающих резисторов.

На рисунке 3.34 видно, что резисторы занимают существенную площадь на кристалле, что является недостатком данного элемента.

Выполним моделирование существенных переменных X2’(nX2) и X1:

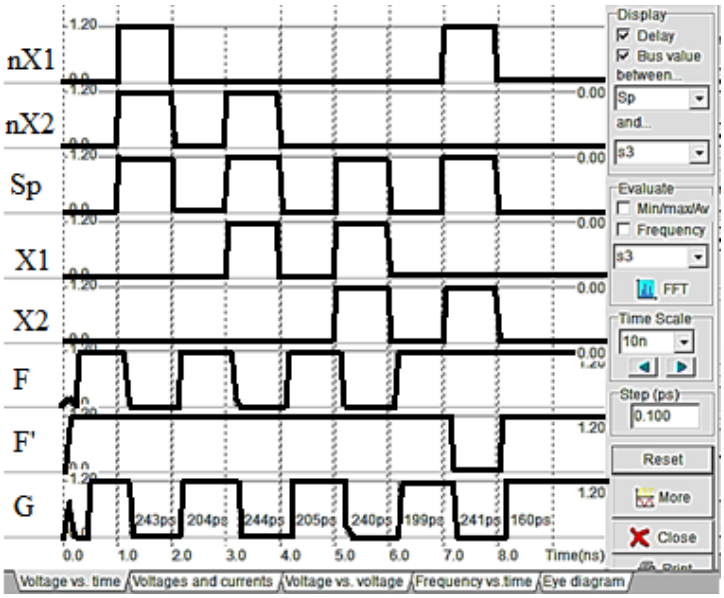



Рисунок 3.35 Результат моделирования существенных переменных X2’ и X1.

На рисунке 3.35 показано, что если хотя бы одна из существенных переменных истина, то выход прямого канала (F) в 0, если обе переменных ложны, то выход прямого канала в 1.

Выполним моделирование несущественной переменной, для этого настройка выполняется константами «1».

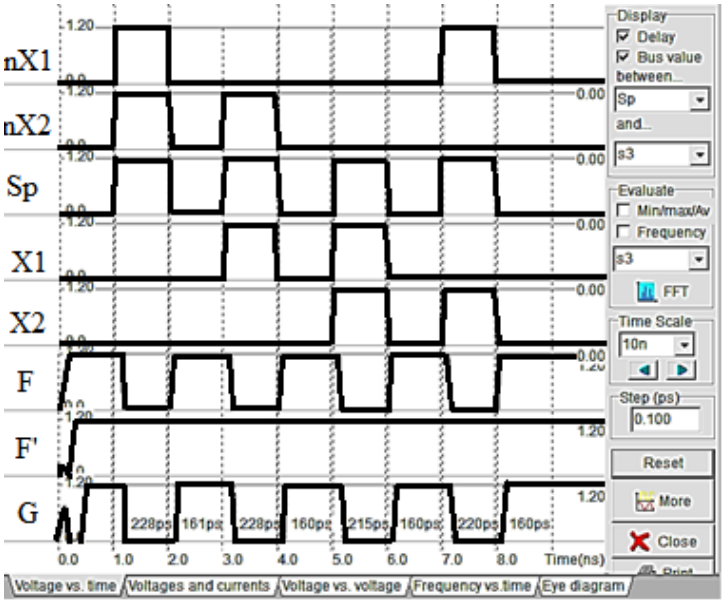



Рисунок 3.36 Результат моделирования несущественности переменных.

На рисунке 3.36 показан результат моделирования несущественности переменных при котором видно, что значения переменных не влияют на работу каналов, прямой канал (F) всегда в логическом 0, а двойственный (F’) в 1.

Полученные результаты подтверждают правильность работы разработанного элемента. Разработанные топологии показаны в приложении №5.

### 3.4.5 Моделирование блока конъюнкций с использованием транзисторов ортогональности на топологическом уровне

Выполним моделирование разработанного элемента, показанного на рисунке 2.23.

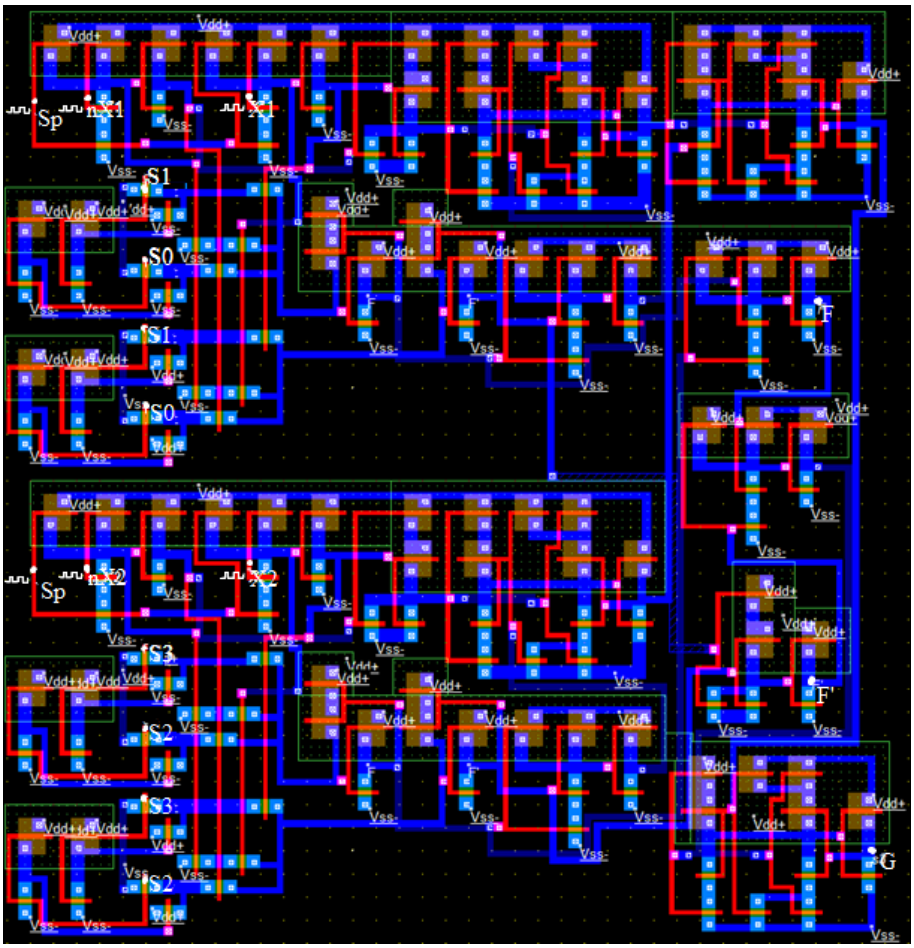



Рисунок 3.37 Топология блока конъюнкций двух разрядов с использованием транзисторов ортогональности.

На рисунке 3.37 показана топология блока конъюнкций, где вход Sp реализует фазу спейсера и служит для подключения обратной связи с выхода последнего Г-триггера. X1, neX1, X2, neX2 – входы переменных. S0-S3 – входы

настройки конъюнкции. F – выход прямого канала, F’ – выход двойственного канала. G – выход последнего гистерезисного триггера схемы.

Выполним моделирование существенных переменных X1 и X2, для этого выполним настройку констант следующим образом: S0=0, S1=1, S2=0, S3=1.

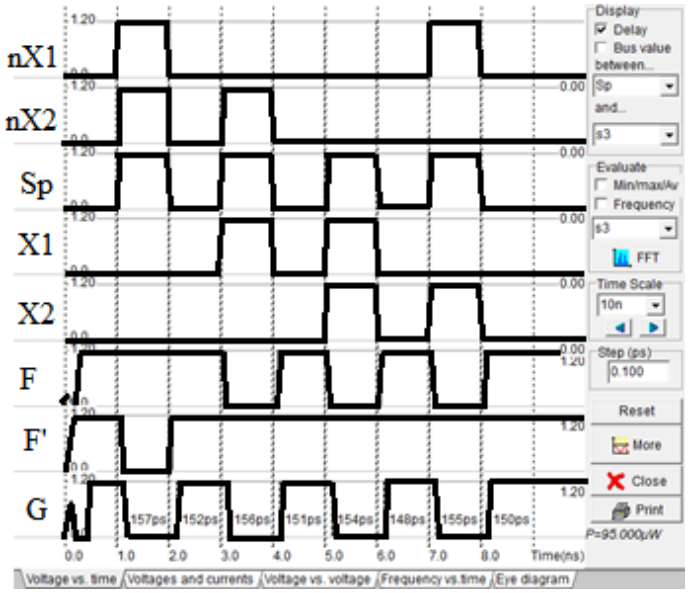



Рисунок 3.38 Результат моделирования существенных переменных X1 и X2.

На рисунке 3.38 показан результат моделирования, если X1=1 и X2=1 - истины, то выход прямого канала F в 0, если значения переменных X1=X2=0 - ложны, то выход прямого канала F=1.

Выполним моделирование существенных переменных X1 и X2’, для этого выполним настройку констант следующим образом: S0=0, S1=1, S2=1, S3=0.

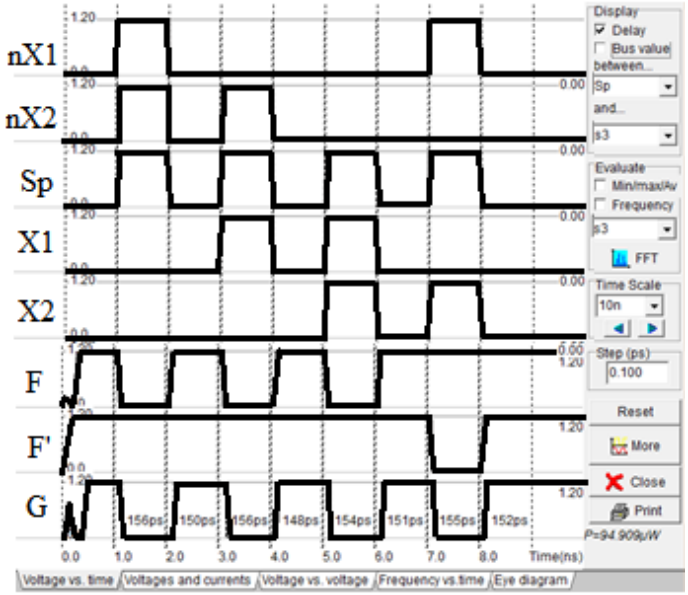



Рисунок 3.39 Результат моделирования существенных переменных X1 и X2’.

На рисунке 3.39 показан результат моделирования, если X1=1 и X2’=1 - истинны, то выход прямого канала F в 0, если значения переменных X1=X2’=0 - ложны, то выход прямого канала F=1.

Выполним моделирование существенных переменных X1’ и X2’, для этого выполним настройку констант следующим образом: S0=1, S1=0, S2=1, S3=0.

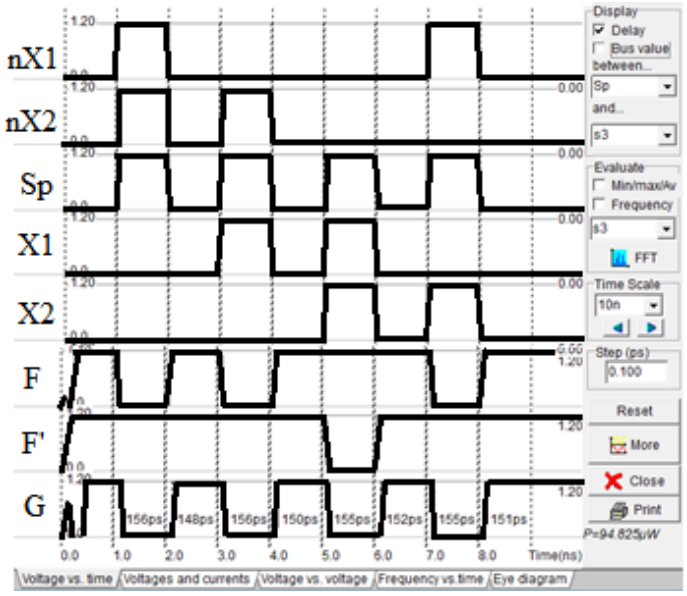



Рисунок 3.40 Результат моделирования существенных переменных X1’ и X2’.

На рисунке 3.40 показан результат моделирования, если X1’=1 и X2’=1 - истинны, то выход прямого канала F в 0, если значения переменных X1’=X2’=0 - ложны, то выход прямого канала F=1.

Выполним моделирование существенных переменных X1’ и X2, для этого выполним настройку констант следующим образом: S0=1, S1=0, S2=0, S3=1.

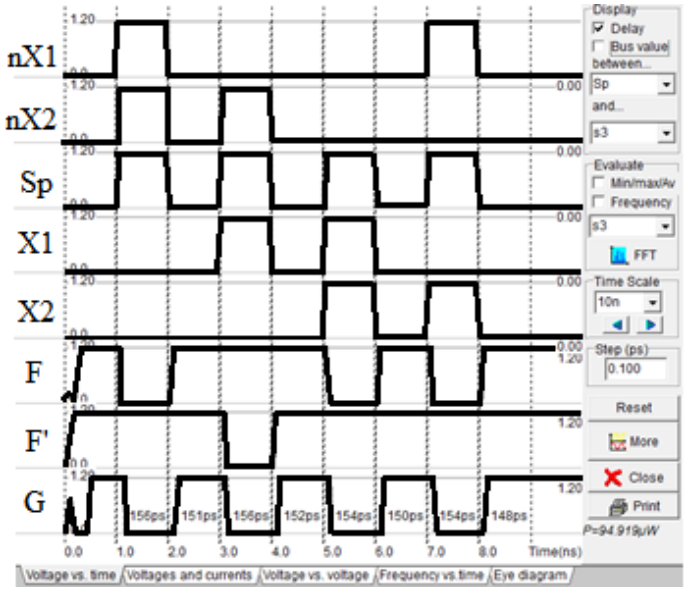



Рисунок 3.41 Результат моделирования существенных переменных X1’ и X2.

На рисунке 3.41 показан результат моделирования, если X1’=1 и X2=1 - истинны, то выход прямого канала F в 0, если значения переменных X1’=X2=0 - ложны, то выход прямого канала F=1.

Выполним проверку несущественности переменных, для этого требуется настроить каналы константами «1».

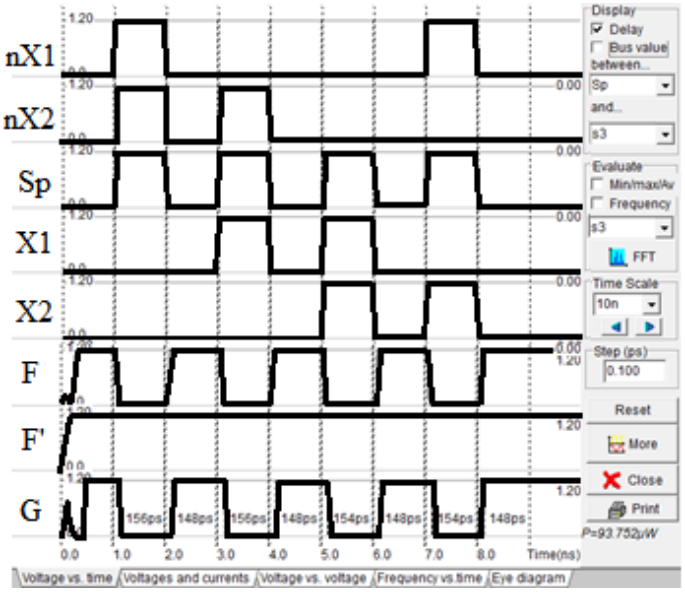



Рисунок 3.42 Результат моделирования несущественности переменных.

На рисунке 3.42 показано, что значения переменных не влияют на работу каналов.

Выполним настройку разработанного элемента на реализацию функции «исключающее ИЛИ» с помощью блока настройки, показанного на рисунке 2.13. Топология разработанного элемента показана в приложении №5 рисунок П5.19.

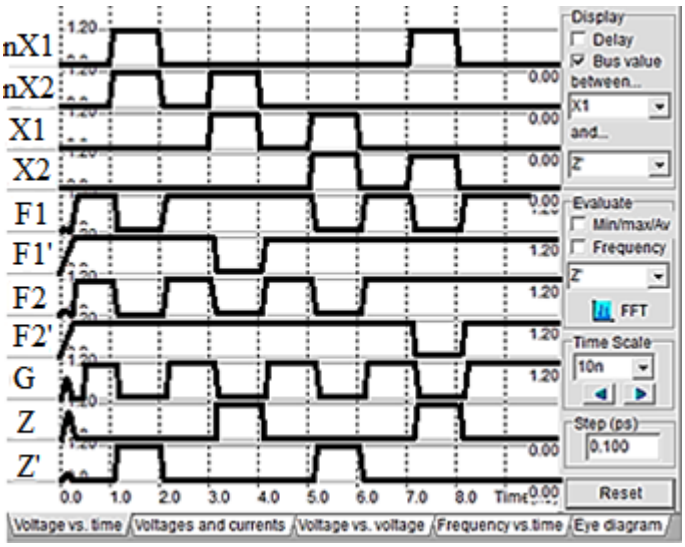



Рисунок 3.43 Результат моделирования функции «исключающее ИЛИ».

На рисунке 3.43 представлен результат моделирования блока конъюнкций двух разрядов с блоком настройки на функцию «исключающее ИЛИ». Один разряд настроен на конъюнкцию X2X1’ – выход F1, другой на X2’X1 – выход F2. Реализация функции «исключающее ИЛИ» выполняется верно, например, при X1=0 и X2=1 Z=1, а при X1=0 и X2=0 Z=0 – логика работы ССС соблюдается, что подтверждает правильность работы предложенного элемента. Разработанные топологии показаны в приложении №5.

### 3.5. Выводы по главе

В главе отражены следующие результаты:

1. Выполнено моделирование элементов (ГФ) на основе библиотечного элемента 2И-2ИЛИ-НЕ в системе автоматизированного проектирования «КОВЧЕГ», подтвердившее работоспособность предложенных элементов. Также выполнено топологическое моделирование ГФ для оценки занимаемой площади на кристалле и других характеристик элемента. Полученные результаты не противоречат друг другу, что подтверждает работоспособность разработанных элементов.

2. Выполнено моделирование элементов LUT-ST и DC LUT-ST в системе схемотехнического моделирования NI Multisim фирмы National Instruments Electronics Workbench Group с использованием модели транзисторов BSIM 4.8.0. Результат моделирования подтверждает работоспособность разработанных элементов. Также выполнено топологическое моделирование элементов LUT-ST и DC LUT-ST, в результате которого выяснилось, что на выходах каналов необходимо устанавливать специальные восстановители уровней сигналов, т.к. при использовании простых инверторов логический 0 восстанавливается не полностью, в схеме протекают токи утечки.

3. Выполнено моделирование блока настройки на логическом уровне и топологическом, подтвердившее работоспособность разработанного блока настройки.

4. Выполнено топологическое моделирование разработанных блоков конъюнкций. Моделирование подтвердило работоспособность всех блоков, на основании данного моделирования сделан вывод, что блок конъюнкций с использованием транзисторов ортогональности имеет выигрыш по занимаемой площади, быстродействию и энергопотреблению перед другими блоками конъюнкций.

# ГЛАВА 4. ОЦЕНКА ЭФФЕКТИВНОСТИ ПРЕДЛОЖЕННЫХ КОНФИГУРИРУЕМЫХ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ ДЛЯ САМОСИНХРОННЫХ СХЕМ ПО РЕЗУЛЬТАТАМ МОДЕЛИРОВАНИЯ

## 4.1. Исследование сложности разработанных логических элементов

Выполним оценку сложности в количестве транзисторов разработанных логических элементов от $n$ переменных для реализации $m$ функций, $w$ конъюнкций, $y$ систем. Для этого выполним оценку сложности каждого блока схемы [100-102]:

Сложность каналов в синхронном ***ГФ*** на основе восьмитранзисторного элемента 2И-2ИЛИ-НЕ для n переменных описывается выражением:

$$Lgf = 8 \cdot (2^n - 1) \tag{4.1}$$

Выражение (4.1) не учитывает принадлежность переменных к множеству натуральных чисел, что необходимо для построения графиков. Для исключения данного недостатка предлагается выполнять округление переменных в нижнюю сторону.

Предложена самосинхронная реализация ГФ, поэтому сложность в количестве элементов 2И-2ИЛИ-НЕ возрастает:

$$Lgf.st.k = 2 \cdot (8 \cdot (2^{\lfloor n \rfloor} - 1)) \tag{4.2}$$

где $\lfloor \ldots \rfloor$-округление в нижнюю строну (floor).

Для реализации блоков входных данных 2n входных переменных требуется $2\lfloor n \rfloor$ блоков. Сложность одного блока составляет восемь транзисторов.

Для индицирования 2n входных переменных требуется $\lfloor n \rfloor$ блоков индикаторов входов. Сложность одного индикатора входов составляет шесть транзисторов.

Для индицирования выходов $2 \cdot (2^{\lfloor n \rfloor} - 1)$ элементов 2И-2ИЛИ-НЕ требуется $2^{\lfloor n \rfloor} - 1$ блоков индикаторов выходов. Сложность одного индикатора выходов составляет шесть транзисторов.

Для фиксации окончания переходных процессов в $2 \cdot (2^{\lfloor n \rfloor} - 1)$ элементах 2И-2ИЛИ-НЕ используется $(2^{\lfloor n \rfloor} + \lfloor n \rfloor - 2)$ двухвходовых Г-триггеров

сложностью в двенадцать транзисторов. Тогда для реализации одной функции сложность элемента ГФ описывается выражением:

$$Lgf.st = 8 \cdot 2\lfloor n \rfloor + 6 \cdot \left(2^{\lfloor n \rfloor} + \lfloor n \rfloor - 1\right) + 2 \cdot (8 \cdot \left(2^{\lfloor n \rfloor} - 1\right)) + 12 \cdot (2^{\lfloor n \rfloor} + \lfloor n \rfloor - 2) \quad (4.3)$$

Для реализации m функций требуется m ГФ, тогда сложность в количестве транзисторов:

$$Lgf.st(m) = m(8 \cdot 2\lfloor n \rfloor + 6 \cdot \left(2^{\lfloor n \rfloor} + \lfloor n \rfloor - 1\right) + 2 \cdot (8 \cdot \left(2^{\lfloor n \rfloor} - 1\right)) + 12 \cdot (2^{\lfloor n \rfloor} + \lfloor n \rfloor - 2)) \quad (4.4)$$

Сложность элемента ***LUT ST*** отличается от ГФ тем, что для реализации каналов используется адаптированное дерево передающих транзисторов известного LUT. Для реализации n переменных в LUT k деревьев передающих транзисторов сложностью в два транзистора каждое. Кроме дерева передающих транзисторов в LUT нужно учесть сложность инверторов на входе каждой ветви и восстановители на выходе дерева. Тогда сложность известного LUT описывается выражением:

$$Llut = 2 \cdot (2^{n+1} - 1) + 3 \cdot (2^n - 1) \quad (4.5)$$

Предложенный самосинхроннвый вариант LUT ST отличается тем, что используются дополнительные цепочки спейсера и двойственный канал, поэтому сложность элемента возрастает:

$$Llut.st = 2 \cdot \left(2 \cdot (2^{\lfloor n \rfloor + 1} - 1) + 3 \cdot (2^{\lfloor n \rfloor} - 1)\right) + 2 \cdot \sum_{i=1}^{n} 2^{\lfloor i \rfloor} \quad (4.6)$$

где $\sum_{i=1}^{n} 2^{\lfloor i \rfloor}$– сложность цепочки спейсера.

Сложность элемента для реализации m логических функций с учетом блоков входного набора, индикаторов входов и выходов и Г-триггеров оценивается выражением:

$$Llut.st(m) = m[8 \cdot 2\lfloor n \rfloor + 6 \cdot \left(2^{\lfloor n \rfloor} + \lfloor n \rfloor - 1\right) + 2 \cdot (2 \cdot (2^{\lfloor n \rfloor + 1} - 1) + +3 \cdot (2^{\lfloor n \rfloor} - 1)) + 2 \cdot \sum_{i=1}^{n} 2^{\lfloor i \rfloor} + 12 \cdot (2^{\lfloor n \rfloor} + \lfloor n \rfloor - 2)] \quad (4.7)$$

Реализация обратного дерева передающих транзисторов известного DC LUT отличается от дерева LUT использованием дополнительных транзисторов, которые обеспечивают ортогональность. Сложность ***DC LUT ST*** кроме этого отличается от ***LUT ST*** количеством инверторов и восстановителей, тогда:

$$Ldclut.st = 2[2 \cdot (2^{\lfloor n \rfloor} - 1) + 2 \cdot \sum_{r=1}^{n} (2^{\lfloor r \rfloor} - 2) + 3 \cdot \sum_{j=1}^{n} 2 \cdot 2^{\lfloor j \rfloor} + 2 \cdot \sum_{i=1}^{n} 2^{\lfloor i \rfloor}] \quad (4.8)$$

где $\sum_{i=1}^{n} 2^{\lfloor i \rfloor}$– сложность цепочки спейсера, $\sum_{r=1}^{n}(2^{\lfloor r \rfloor} - 2)$ – сложность в количестве инверторов, $\sum_{j=1}^{n} 2 \cdot 2^{\lfloor j \rfloor}$ - сложность в количестве восстановителей, $\sum_{j=1}^{n} 2 \cdot 2^{\lfloor j \rfloor}$ - сложность транзисторов ортогональности.

Также DC LUT ST отличается от LUT ST тем, что увеличивается количество выходов каналов, что влечет за собой увеличение индикаторов выходов и G-триггеров. Тогда выражение, описывающее сложность элемента DC LUT-ST одной функции:

$$Ldclut.st = 8 \cdot 2\lfloor n \rfloor + 6 \cdot \left(2^{\lfloor n+1 \rfloor} + \lfloor n \rfloor - 2\right) + 2[2 \cdot (2^{\lfloor n \rfloor} - 1) + 2 \cdot \sum_{r=1}^{n} (2^{\lfloor r \rfloor} - 2)$$

$$+3 \cdot \sum_{j=1}^{n} 2 \cdot 2^{\lfloor j \rfloor} + 2 \cdot \sum_{i=1}^{n} 2^{\lfloor i \rfloor}] + 12 \cdot \left(2^{\lfloor n+1 \rfloor} + \lfloor n \rfloor - 3\right) \quad (4.9)$$

В отличии от ГФ и LUT-ST, для реализации m функций в DC LUT-ST можно использовать блоки дизъюнкций [8], для реализации m функций требуется m блоков. Сложность одного блока, зависящего от n, можно описать:

$$LblokOR = 30 \cdot 2\lfloor n \rfloor - 38 \quad (4.10)$$

Тогда сложность элемента DC LUT-ST для реализации m функций:

$$Ldclut.st = 8 \cdot 2\lfloor n \rfloor + 6 \cdot \left(2^{\lfloor n+1 \rfloor} + \lfloor n \rfloor - 2\right) + 2[2 \cdot (2^{\lfloor n \rfloor} - 1) + 2 \cdot \sum_{r=1}^{n} (2^{\lfloor r \rfloor} - 2) +$$

$$+3 \cdot \sum_{j=1}^{n} 2 \cdot 2^{\lfloor j \rfloor} + 2 \cdot \sum_{i=1}^{n} 2^{\lfloor i \rfloor}] + 12 \cdot \left(2^{\lfloor n+1 \rfloor} + \lfloor n \rfloor - 3\right) + m[30 \cdot 2\lfloor n \rfloor - 38] \quad (4.11)$$

Сложность элемента ***DNF-ST*** на основе библиотечного элемента БМК для реализации одной конъюнкции и одной функции можно описать выражением:

$$Ldnf.st1 = 8 \cdot 4\lfloor n \rfloor + 6 \cdot \left(2^{\lfloor n \rfloor} \cdot (\lfloor n + 1 \rfloor) + 1\right) + 12 \cdot \left(2^{\lfloor n+1 \rfloor} - 2\right) + 2 \cdot 2\lfloor n \rfloor + 2 \quad (4.12)$$

Для реализации w конъюнкций требуется w блоков конъюнкций. С учетом этого выражение (4.12) примет вид:

$$Ldnf.st1(w) = w \cdot [8 \cdot 4\lfloor n \rfloor + 6 \cdot \left(2^{\lfloor n \rfloor} \cdot (\lfloor n + 1 \rfloor) + 1\right) + 12 \cdot \left(2^{\lfloor n+1 \rfloor} - 2\right) + 2 \cdot 2\lfloor n \rfloor + 2] \quad (4.13)$$

Для реализации m функций требуется m блоков дизъюнкций (4.10). С учетом (4.13) получим выражение:

$$Ldnf.st1(w,m) = w \cdot [8 \cdot 4\lfloor n \rfloor + 6 \cdot \left(2^{\lfloor n \rfloor} \cdot (\lfloor n + 1 \rfloor) + 1\right) + 12 \cdot \left(2^{\lfloor n+1 \rfloor} - 2\right) + +2 \cdot 2\lfloor n \rfloor + 2] + m[30 \cdot 2\lfloor n \rfloor - 38] \quad (4.14)$$

Сложность элемента ***DNF-ST*** с использованием подтягивающих резисторов для реализации одной конъюнкции и одной функции можно описать выражением:

$$Ldnf.st2 = 12 \cdot 2\lfloor n \rfloor + 4 \cdot 3\lfloor n \rfloor + 6 \cdot (4\lfloor n \rfloor - 3) + 12 \cdot (3\lfloor n \rfloor - 2) \quad (4.15)$$

Рассуждая аналогичным образом, получим выражение, описывающее работу элемента для реализации w конъюнкций и m функций:

$$Ldnf.st2(w,m) = w[12 \cdot 2\lfloor n \rfloor + 4 \cdot 3\lfloor n \rfloor + 6 \cdot (4\lfloor n \rfloor - 3) + 12 \cdot (3\lfloor n \rfloor - 2) + +m[30 \cdot 2\lfloor n \rfloor - 38] \quad (4.16)$$

Сложность элемента ***DNF-ST*** с использованием транзисторов ортогональности для реализации одной конъюнкции и одной функции можно описать выражением:

$$Ldnf.st3 = 14 \cdot 2\lfloor n \rfloor + 4 \cdot 3\lfloor n \rfloor + 6 \cdot (4\lfloor n \rfloor - 3) + 12 \cdot (3\lfloor n \rfloor - 2) \quad (4.17)$$

Рассуждая аналогичным образом, получим выражение, описывающее работу элемента для реализации w конъюнкций и m функций:

$$Ldnf.st3(w,m) = w[14 \cdot 2\lfloor n \rfloor + 4 \cdot 3\lfloor n \rfloor + 6 \cdot (4\lfloor n \rfloor - 3) + 12 \cdot (3\lfloor n \rfloor - 2) + +m[30 \cdot 2\lfloor n \rfloor - 38] \quad (4.18)$$

Выполним сравнение сложности в количестве транзисторов для заданных характеристик систем:

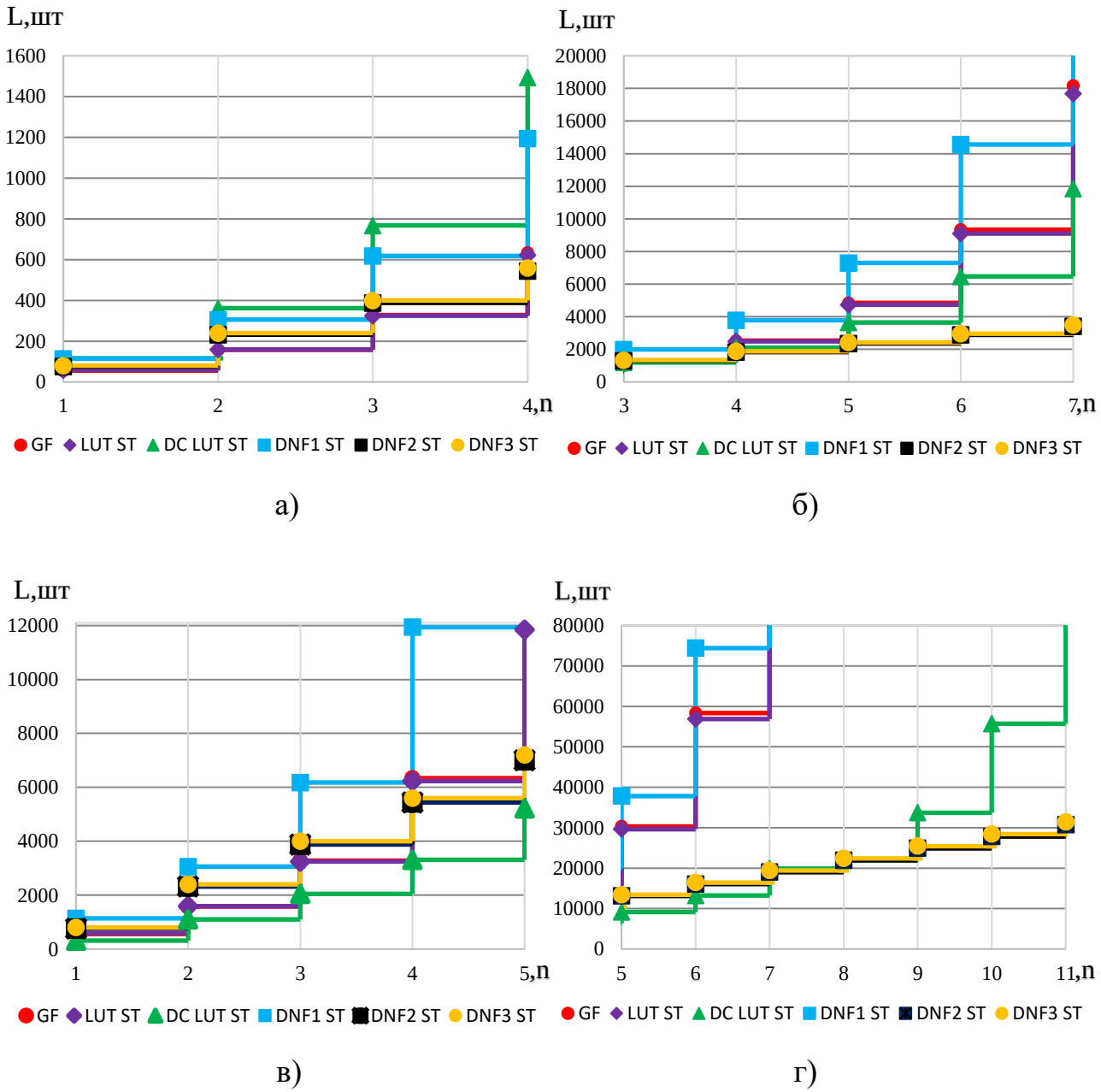



Рисунок 4.1 – Графики сравнения сложности реализации разработанных элементов, где GF – генератор функций на ЛЭ, LUT-ST – элемент LUT ST, DC LUT-ST – дешифратор DC LUT-ST, DNF1 ST – блок конъюнкции на ЛЭ, DNF2 ST - блок конъюнкции с подтягивающим резистором, DNF3 ST - блок конъюнкции с транзистором ортогональности;

а) – при m=1, w=1; б) – при m=4, w=3;

в) – при m=10, w=10; г) – при m=25, w=15

На рисунке 4.1. видно, что элемент LUT ST имеет преимущество в реализации небольшого количества функций и небольшого числа переменных до n=4, при этом ГФ имеет проигрыш всего в 1-3%.

Для количества переменных до n=7 и большого числа функций выигрыш в количестве транзисторов имеет элемент DC LUT-ST, выигрыш по сравнению с самым худшим вариантом составляет от 13% до 43%.

Для реализации большого количества функций и числа переменных от n=7 преимуществом обладает элемент блока конъюнкций с подтягивающим резистором – DNF 2 ST, при этом элемент блока конъюнкций с транзистором ортогональности имеет проигрыш в количестве транзисторов 2%.

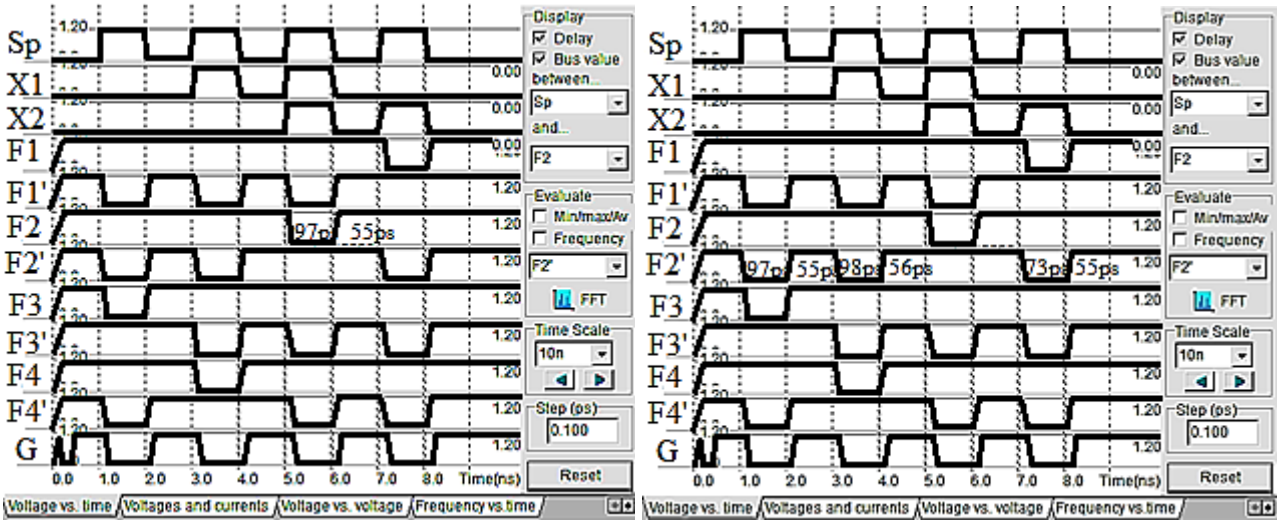
Таким образом, на основании полученных результатов нельзя сделать однозначный вывод, какой из разработанных элементов лучше использовать при других ограничениях систем. Для этого требуется дополнительный анализ, например, быстродействия элемента.

### 4.2. Анализ быстродействия разработанных элементов

На основании моделирования выполним *анализ задержки (T)* разработанных элементов. Для этого используется средство анализа в САПР MicroWind. Оценка временной задержки переключения выходов элемента измеряется относительно изменения рабочей фазы на спейсерную.

Для оценки временных параметров в MicroWind имеется встроенная функция: на диаграмме входов и выходов требуется выбрать, между какими сигналами будет измеряться задержка в переключении, и на диаграмме будет отображаться задержка по фонту и срезу.

Выполним оценку быстродействия элементов на примере элемента DC LUT ST двух переменных. Анализ других разработанных элементов определяются таким же способом, результаты занесем в таблицу 4.1.




а) б)

Рисунок 4.2 – Анализ значения задержек в каналах элемента 2-DC LUT ST:

а) значение задержки второго прямого канала;

б) значение задержки второго двойственного канала;

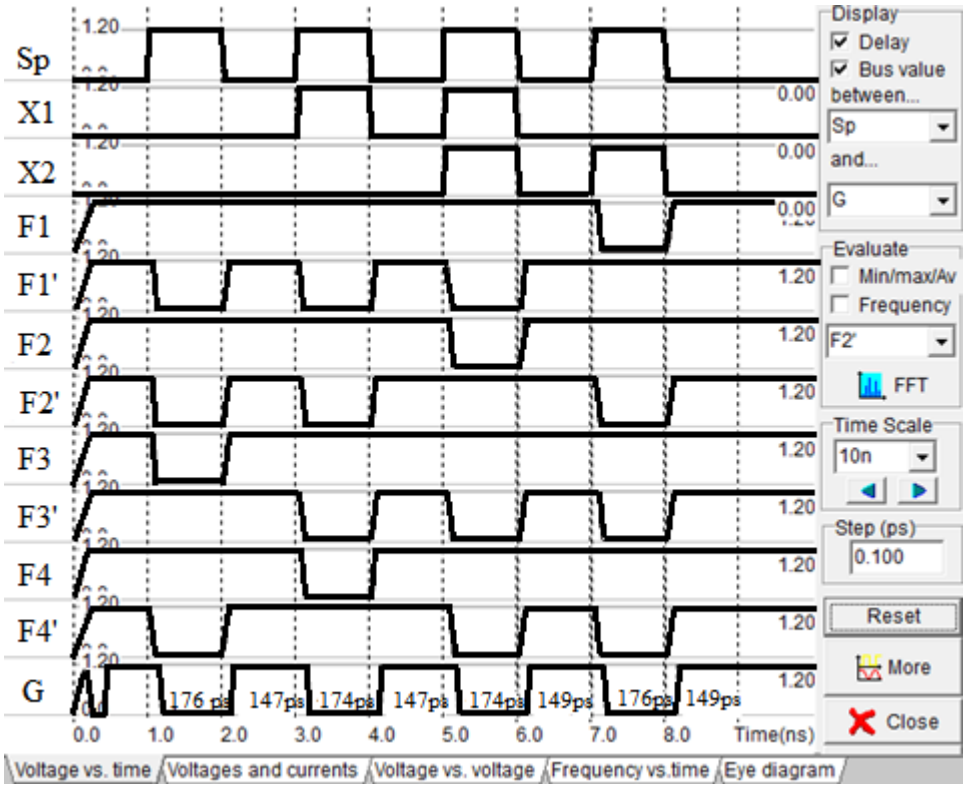



Рисунок 4.3 – Анализ значения задержки последнего элемента схемы.

Результаты анализа задержки для разработанных элементов при n=1,2,3 и входном напряжении питания 1,2 В представлены в таблице 4.1

Таблица 4.1 Значения задержек разработанных элементов:

| Разработанный элемент | Название выхода | n=1 | | n=2 | | n=3 | | n=4 | |
|---|---|---|---|---|---|---|---|---|---|
| | | $T_{срез}$, пс | $T_{фронт}$, пс | $T_{срез}$, пс | $T_{фронт}$, пс | $T_{срез}$, пс | $T_{фронт}$, пс | $T_{срез}$, пс | $T_{фронт}$, пс |
| ГФ | F | 34 | 13 | 35 | 20 | 40 | 28 | - | - |
| | F’ | 32 | 11 | 36 | 21 | 41 | 29 | - | - |
| | G | 65 | 59 | 123 | 119 | 217 | 212 | - | - |
| LUT ST | F | 52 | 51 | 99 | 51 | 154 | 51 | - | - |
| | F’ | 50 | 52 | 98 | 52 | 153 | 52 | - | - |
| | G | 83 | 89 | 152 | 125 | 220 | 156 | - | - |
| DC LUT ST | F0 | 51 | 52 | 99 | 54 | 151 | 57 | - | - |
| | F1 | 58 | 53 | 97 | 55 | 149 | 58 | - | - |
| | F0' | 59 | 53 | 100 | 56 | 153 | 60 | - | - |
| | F1' | 57 | 53 | 98 | 56 | 150 | 60 | - | - |
| | F2 | - | - | 100 | 53 | 153 | 57 | - | - |
| | F3 | - | - | 99 | 55 | 151 | 58 | - | - |
| | F2' | - | - | 101 | 57 | 154 | 62 | - | - |
| | F3' | - | - | 100 | 57 | 153 | 62 | - | - |
| | F4 | - | - | - | - | 155 | 61 | - | - |
| | F5 | - | - | - | - | 154 | 61 | - | - |
| | F4' | - | - | - | - | 158 | 65 | - | - |
| | F5' | - | - | - | - | 151 | 63 | - | - |
| | F6 | - | - | - | - | 153 | 64 | - | - |
| | F7 | - | - | - | - | 152 | 66 | - | - |
| | F6' | - | - | - | - | 154 | 62 | - | - |
| | F7' | - | - | - | - | 153 | 60 | - | - |
| | G | 99 | 99 | 119 | 123 | 308 | 308 | - | - |
| DNF 1 ST | F | 60 | 143 | 73 | 181 | - | - | 155 | 193 |
| | F’ | 94 | 154 | 135 | 168 | - | - | 157 | 180 |
| | G | 132 | 185 | 205 | 225 | - | - | 267 | 308 |
| DNF 2 ST | F | 49 | 60 | 63 | 89 | - | - | 65 | 92 |
| | F’ | 51 | 63 | 82 | 85 | - | - | 85 | 88 |
| | G | 88 | 94 | 148 | 150 | - | - | 215 | 221 |
| DNF 3 ST | F | 45 | 54 | 68 | 72 | - | - | 88 | 92 |
| | F’ | 41 | 52 | 62 | 72 | - | - | 88 | 94 |
| | G | 95 | 99 | 148 | 153 | - | - | 218 | 226 |
| Блок настройки | Z | - | - | 25 | 49 | - | - | 66 | 69 |
| | Z’ | - | - | 33 | 50 | - | - | 78 | 43 |
| | G | - | - | 91 | 97 | - | - | 149 | 155 |

На основании полученных данных найдем зависимость задержки в последнем компоненте схемы (Г-триггер) от n. Для этого предлагается выполнить полиномиальную аппроксимацию с наименьшей погрешностью

между экспериментальными данными и расчетными. Для этого используем инструмент Microsoft Excel, который предоставляет возможность экономико-статистических расчетов, графический инструмент и возможность макропрограммирования на языке Visual Basic for Application (VBA) [103].

Экспериментальные данные, представленные в таблице 4.1, показывают, что все разработанные элементы, за исключением блока настройки, можно описать полиномом второго порядка, который вносит наименьшую погрешность в вычислении.

Выражение, описывающее зависимость задержки от n и m в разработанном элементе ГФ:

$$Tgf.st(m) = m(18 \cdot \lfloor n \rfloor^2 + 4 \cdot \lfloor n \rfloor + 43) \quad (4.19)$$

Выражение, описывающее зависимость задержки от n и m в разработанном элементе LUT ST:

$$Tlut.st(m) = m(2.5 \cdot \lfloor n \rfloor^2 + 55.5 \cdot \lfloor n \rfloor + 31) \quad (4.20)$$

Для нахождения задержки разработанного элемента DC LUT ST требуется учесть задержку блока настройки, которая увеличивается линейно в зависимости от n:

$$TblokOR = 29 \cdot \lfloor n \rfloor + 39 \quad (4.21)$$

Тогда зависимость задержки от n и m в разработанном элементе DC LUT ST:

$$Tdclut.st(m) = 27.5 \cdot \lfloor n \rfloor^2 - 5.5 \cdot \lfloor n \rfloor + 77 + m(29 \cdot \lfloor n \rfloor + 39) \quad (4.22)$$

Для нахождения задержки разработанных элементов ДНФ кроме того, что требуется учесть задержку блока настройки, также требуется учесть число конъюнкций. Тогда выражение, описывающее задержку элемента ДНФ на основе библиотечного элемента БМК:

$$Tdnf.st1(w,m) = w(0.5 \cdot \lfloor n \rfloor^2 + 38.5 \cdot \lfloor n \rfloor + 146) + m(29 \cdot \lfloor n \rfloor + 39) \quad (4.23)$$

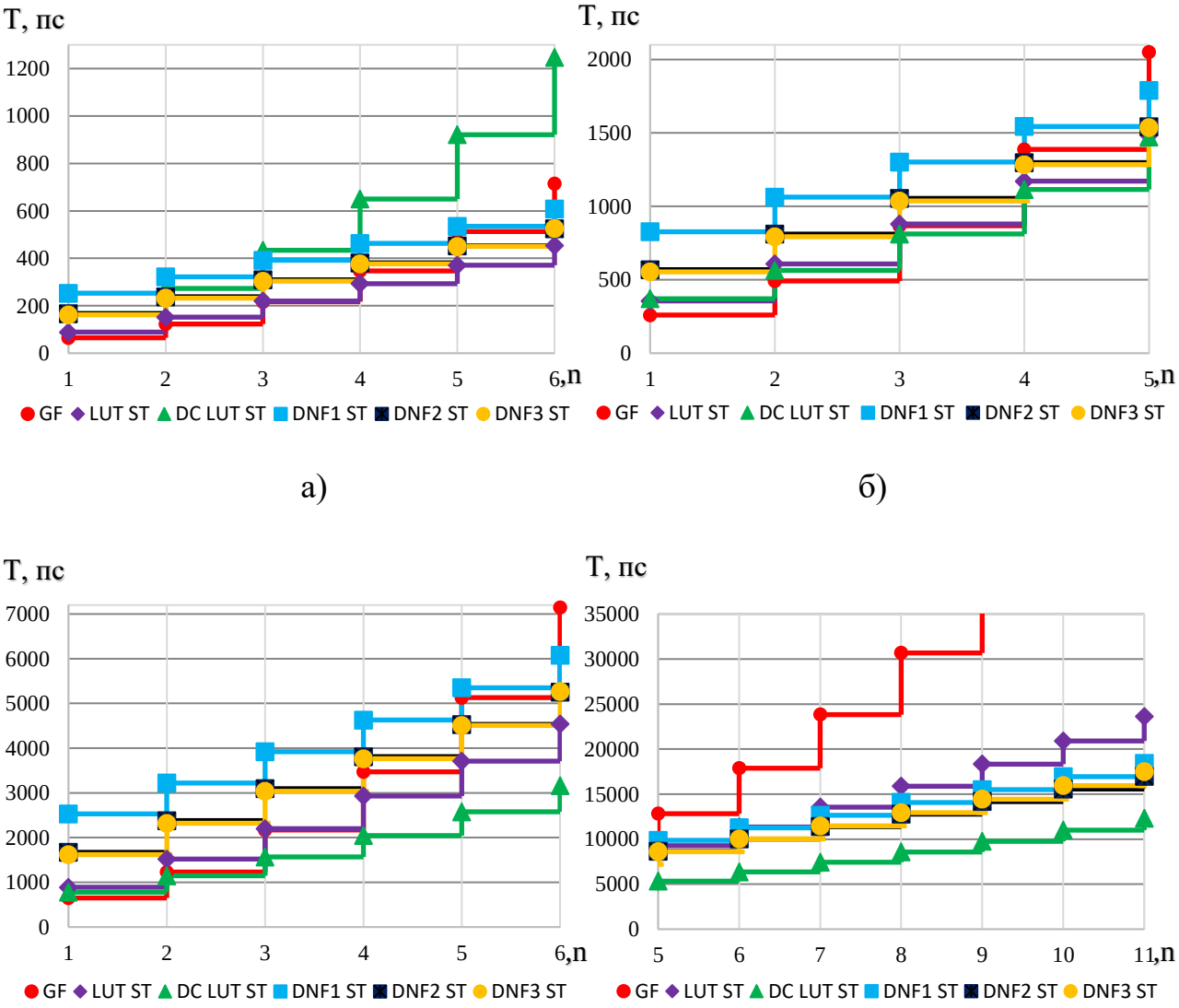
Выражение, описывающее задержку элемента ДНФ с использованием подтягивающих резисторов:

$$Tdnf.st2(w,m) = w(0.2 \cdot \lfloor n \rfloor^2 + 41.5 \cdot \lfloor n \rfloor + 57.3) + m(29 \cdot \lfloor n \rfloor + 39) \quad (4.24)$$

Выражение, описывающее задержку элемента ДНФ с использованием транзисторов ортогональности:

$$Tdnf.st3(w,m) = w(0.7 \cdot \lfloor n \rfloor^2 + 39 \cdot \lfloor n \rfloor + 54.3) + m(29 \cdot \lfloor n \rfloor + 39) \quad (4.25)$$

На основании полученных формул построим графики зависимости от числа переменных (n), числа функций (m) и числа конъюнкций (w).




а) б)

в) г)

Рисунок 4.4 – Графики сравнения задержки разработанных элементов, где GF – генератор функций на ЛЭ, LUT-ST – элемент LUT ST, DC LUT-ST – дешифратор DC LUT-ST, DNF1 ST – блок конъюнкции на ЛЭ, DNF2 ST - блок конъюнкции с подтягивающим резистором, DNF3 ST - блок конъюнкции с транзистором ортогональности; а) – при m=1, w=1; б) – при m=4, w=3; в) – при m=10, w=10; г) – при m=25, w=15.

На рисунке 4.4 видно, что при небольшом количестве функций до m=2 LUT ST имеет преимущество по быстродействию, но при небольшом числе переменных до n=3 выигрыш в быстродействии имеет элемент ГФ от 1-37%, причем задержка увеличивается пропорционально при увеличении числа переменных. Если требуется реализовать небольшое число функций, зависящее от малого числа переменных (до n=5), однозначно выбрать, какой элемент выигрывает, сложно. Если требуется реализовать большое число функций с большим числом переменных, то явным преимуществом обладает элемент DC LUT ST, быстродействие которого в 140-240% выше, чем у самого худшего варианта.

### 4.3. Исследование площади, занимаемой на кристалле логических элементов генератора функций, LUT ST, DC LUT-ST

В главе 3 было выполнено топологическое моделирование разработанных элементов, по результату которого можно оценить площадь S, занимаемую на кристалле разработанными самосинхронными элементами. Так, например, ГФ на две переменные занимает 418,8 мкм$^2$:

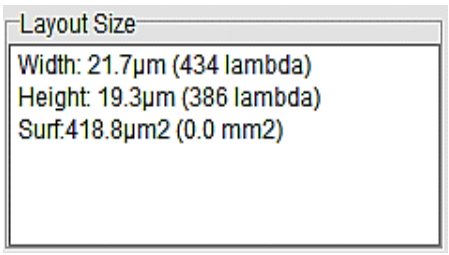



Рисунок 4.5 - Размеры и занимаемая площадь генератора функций двух переменных.

Остальные результаты занимаемой площади разработанных элементов сведем в таблицу 4.2.

Таблица 4.2 Зависимость занимаемой площади от количества переменных:

Начало таблицы 4.2

| Элемент | Количество переменных | | | |
|---|---|---|---|---|
| | n=1, S, µm2 | n=2, S, µm2 | n=3, S, µm2 | n=4, S, µm2 |
| ГФ | 159,7 | 418,8 | 930,2 | - |
| LUT-ST | 123,7 | 402,4 | 1092 | - |

Окончание таблицы 4.2

| Элемент | Количество переменных | | | |
|---|---|---|---|---|
| | n=1, S, μm2 | n=2, S, μm2 | n=3, S, μm2 | n=4, S, μm2 |
| DC LUT-ST | 173,8 | 630,8 | 1699,9 | - |
| DNF 1 ST | 159,2 | 435,2 | - | 1129 |
| DNF 2 ST | 225,1 | 564,4 | - | 1397,2 |
| DNF 3 ST | 141,3 | 383 | - | 987,8 |
| Блок настройки | - | 189,7 | - | 525,7 |

На основании полученных данных найдем зависимость площади разработанных элементов от n, для этого предлагается выполнить полиномиальную аппроксимацию с наименьшей погрешностью между экспериментальными данными и расчетными. Для этого используем инструмент Microsoft Excel, который предоставляет возможность экономико-статистических расчетов, графический инструмент и возможность макропрограммирования на языке Visual Basic for Application (VBA) [103].

Экспериментальные данные, представленные в таблице 4.2, показывают, что ГФ, LUT ST и DC LUT ST имеют экспоненциальную зависимость, элементы ДНФ можно описать полиномом второго порядка, а блок настройки можно описать линейной функцией. Такие зависимости вносят наименьшую погрешность в вычислении.

Выражение, описывающее зависимость площади от n и m в разработанном элементе ГФ:

$$Sgf.st(m) = m(68e^{0.88\lfloor n \rfloor}) \quad (4.26)$$

Выражение, описывающее зависимость площади от n и m в разработанном элементе LUT ST:

$$Slut.st(m) = m(42.9e^{1.08\lfloor n \rfloor}) \quad (4.27)$$

Для нахождения площади разработанного элемента DC LUT ST требуется учесть площадь блока настройки, которая увеличивается линейно в зависимости от n:

$$SblokOR = 168 \cdot \lfloor n \rfloor - 146.3 \quad (4.28)$$

Тогда зависимость площади от n и m в разработанном элементе DC LUT ST:

$$Sdclut.st(m) = 58{,}4e^{1.14\lfloor n \rfloor} + m(168 \cdot \lfloor n \rfloor - 146.3) \quad (4.29)$$

Для нахождения площади разработанных элементов ДНФ кроме того, что требуется учесть площадь блока настройки, также требуется учесть число конъюнкций. Тогда выражение, описывающее площадь элемента ДНФ на основе библиотечного элемента БМК:

$$Sdnf.st1(w,m) = w(23.6 \cdot \lfloor n \rfloor^2 + 205.1 \cdot \lfloor n \rfloor - 69{,}5) + m(168 \cdot \lfloor n \rfloor - 146.3) \quad (4.30)$$

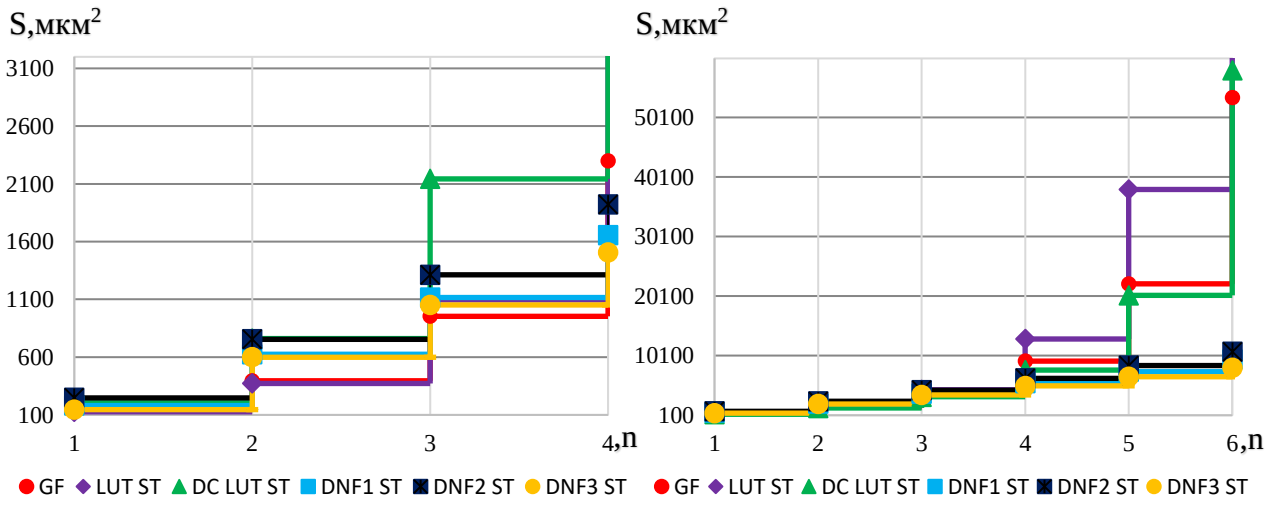
Выражение, описывающее площадь элемента ДНФ с использованием подтягивающих резисторов:

$$Sdnf.st2(w,m) = w(25.7 \cdot \lfloor n \rfloor^2 + 262.2 \cdot \lfloor n \rfloor - 62.8) + m(168 \cdot \lfloor n \rfloor - 146.3) \quad (4.31)$$

Выражение, описывающее площадь элемента ДНФ с использованием транзисторов ортогональности:

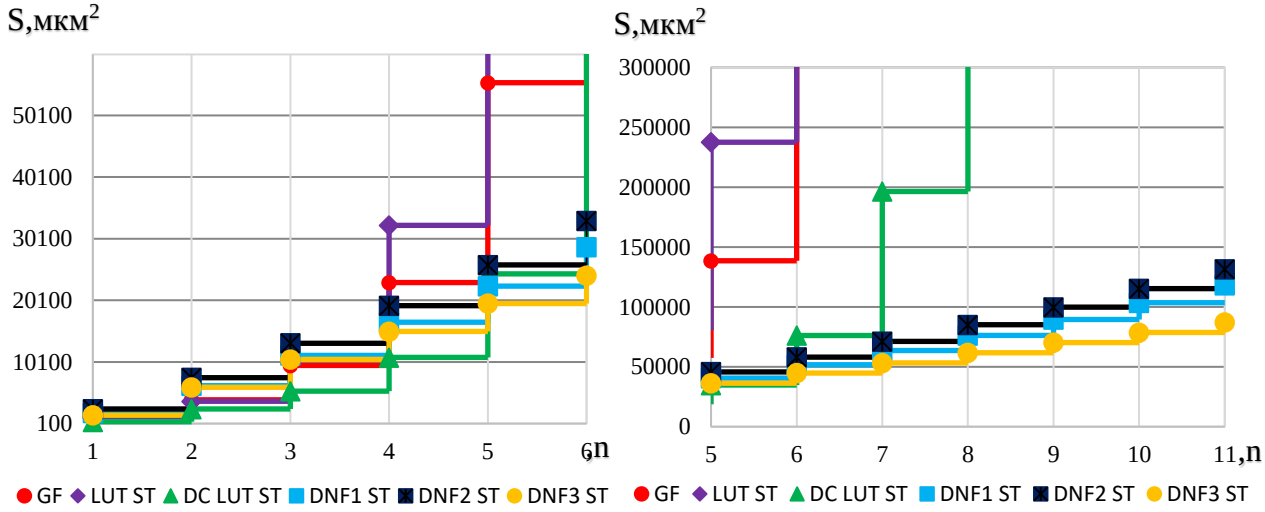
$$Sdnf.st3(w,m) = w(285.06 \cdot \lfloor n \rfloor - 161.1) + m(168 \cdot \lfloor n \rfloor - 146.3) \quad (4.32)$$

На основании полученных формул построим графики зависимости площади от числа переменных (n), числа функций (m) и числа конъюнкций (w).




а) б)

Рисунок 4.6 – Графики сравнения задержки разработанных элементов, где GF – генератор функций на ЛЭ, LUT-ST – элемент LUT ST, DC LUT-ST – дешифратор DC LUT-ST, DNF1 ST – блок конъюнкции на ЛЭ, DNF2 ST - блок конъюнкции с подтягивающим резистором, DNF3 ST - блок конъюнкции с транзистором ортогональности; а) – при m=1, w=1; б) – при m=4, w=3.




а) б)

Рисунок 4.7 – Графики сравнения задержки разработанных элементов, где GF – генератор функций на ЛЭ, LUT-ST – элемент LUT ST, DC LUT-ST – дешифратор DC LUT-ST, DNF1 ST – блок конъюнкции на ЛЭ, DNF2 ST - блок конъюнкции с подтягивающим резистором, DNF3 ST - блок конъюнкции с транзистором ортогональности; а) – при m=10, w=10; б) – при m=25, w=15.

На рисунках 4.6 и 4.7 видно, что элементы LUT ST и ГФ имеют преимущество при реализации небольшого количества функций и небольшого числа переменных до n=4. Для количества переменных до n=7 и большого числа функций преимущество по площади имеет элемент DC LUT-ST, выигрыш по сравнению с самым худшим вариантом составляет от 1,5 до 4,5 раз.

Для реализации большого количества функций и числа переменных от n=7 преимуществом обладает элемент блока конъюнкций с транзистором ортогональности, выигрыш по сравнению с самым худшим вариантом составляет в 1770 раз.

### 4.4. Анализ электропотребления разработанных элементов

На основании выполненного моделирования, выполним анализ потребляемой мощность разработанных элементов. Для оценки параметра используется САПР MicroWind. Результаты анализа сведем в таблицу 4.3.

Таблица 4.3 Зависимость потребляемой мощности от количества переменных:

| Элемент | Количество переменных | | | |
|---|---|---|---|---|
| | n=1, P, мкВт | n=2, P, мкВт | n=3, P, мкВт | n=4, P, мкВт |
| ГФ | 159,7 | 418,8 | 930,2 | - |
| LUT-ST | 35,64 | 100 | 207 | - |
| DC LUT-ST | 56,75 | 171 | 325 | - |
| DNF 1 ST | 81,75 | 194 | - | 383 |
| DNF 2 ST | 65,97 | 141 | - | 327 |
| DNF 3 ST | 52,15 | 94,86 | - | 231 |
| Блок настройки | - | 48,38 | - | 129 |

На основании полученных данных найдем зависимость потребляемой мощности от n. Для этого предлагается выполнить полиномиальную аппроксимацию, которая дает наименьшей погрешность между экспериментальными данными и расчетными. Для этого воспользуемся Microsoft Excel, который предоставляет возможность экономико-статистических расчетов, графический инструмент и возможность макропрограммирования на языке Visual Basic for Application (VBA) [103].

Экспериментальные данные, представленные в таблице 4.3, показывают, что все разработанные элементы, за исключением блока настройки, можно описать полиномом второго порядка, который вносит наименьшую погрешность в вычислении.

Выражение, описывающее зависимость энергопотребления от n и m в разработанном элементе ГФ:

$$Pgf.st(m) = m(41.2 \cdot \lfloor n \rfloor^2 - 78.6 \cdot \lfloor n \rfloor + 60.8) \quad (4.33)$$

Выражение, описывающее зависимость энергопотребления от n и m в разработанном элементе LUT ST:

$$Plut.st(m) = m(21.3 \cdot \lfloor n \rfloor^2 + 0.4 \cdot \lfloor n \rfloor + 13.9) \quad (4.34)$$

Для нахождения энергопотребления разработанного элемента DC LUT ST требуется учесть энергопотребление блока настройки, которое увеличивается линейно, в зависимость от n:

$$PblokOR = 40.3 \cdot \lfloor n \rfloor - 32.2 \quad (4.35)$$

Тогда зависимость энергопотребления от n и m в разработанном элементе DC LUT ST:

$$Pdclut.st(m) = 19.9 \cdot \lfloor n \rfloor^2 + 54.6 \cdot \lfloor n \rfloor - 17.8 + m(40.3 \cdot \lfloor n \rfloor - 32.2) \quad (4.36)$$

Для нахождения энергопотребления разработанных элементов ДНФ кроме того, что требуется учесть энергопотребление блока настройки, также требуется учесть число конъюнкций. Тогда выражение, описывающее энергопотребление элемента ДНФ на основе библиотечного элемента БМК:

$$Pdnf.st1(w,m) = w(-5.9 \cdot \lfloor n \rfloor^2 + 129.9 \cdot \lfloor n \rfloor - 42.3) + m(40.3 \cdot \lfloor n \rfloor - 32.2) \quad (4.37)$$

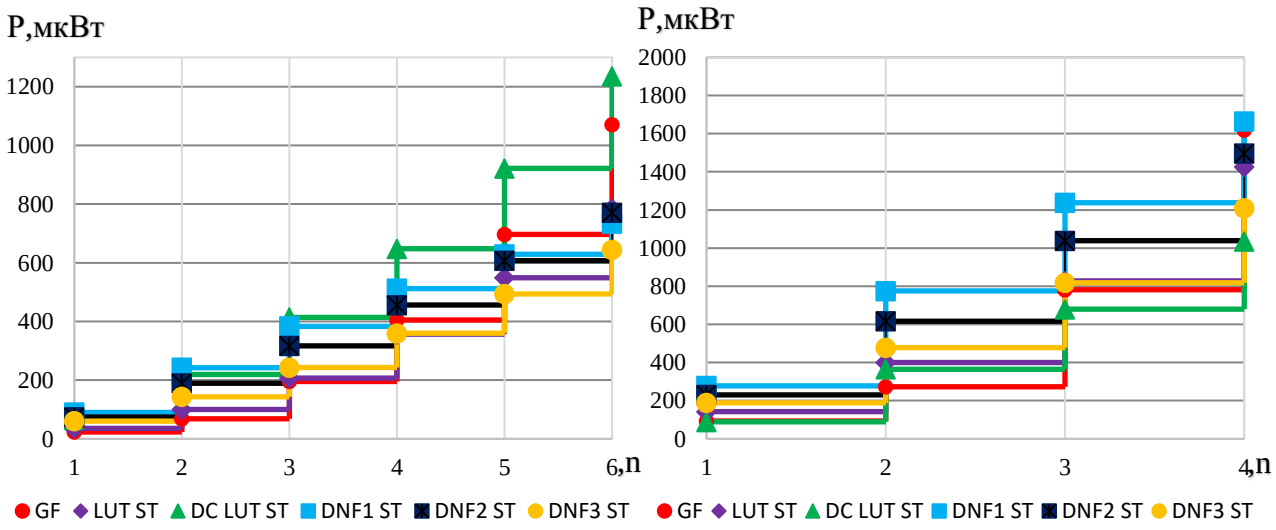
Выражение, описывающее энергопотребление элемента ДНФ с использованием подтягивающих резисторов:

$$Pdnf.st2(w,m) = w(6 \cdot \lfloor n \rfloor^2 + 57.1 \cdot \lfloor n \rfloor + 2.92) + m(40.3 \cdot \lfloor n \rfloor - 32.2) \quad (4.38)$$

Выражение, описывающее энергопотребление элемента ДНФ с использованием транзисторов ортогональности:

$$Pdnf.st3(w,m) = w(8.5 \cdot \lfloor n \rfloor^2 + 17.4 \cdot \lfloor n \rfloor + 26.3) + m(40.3 \cdot \lfloor n \rfloor - 32.2) \quad (4.39)$$

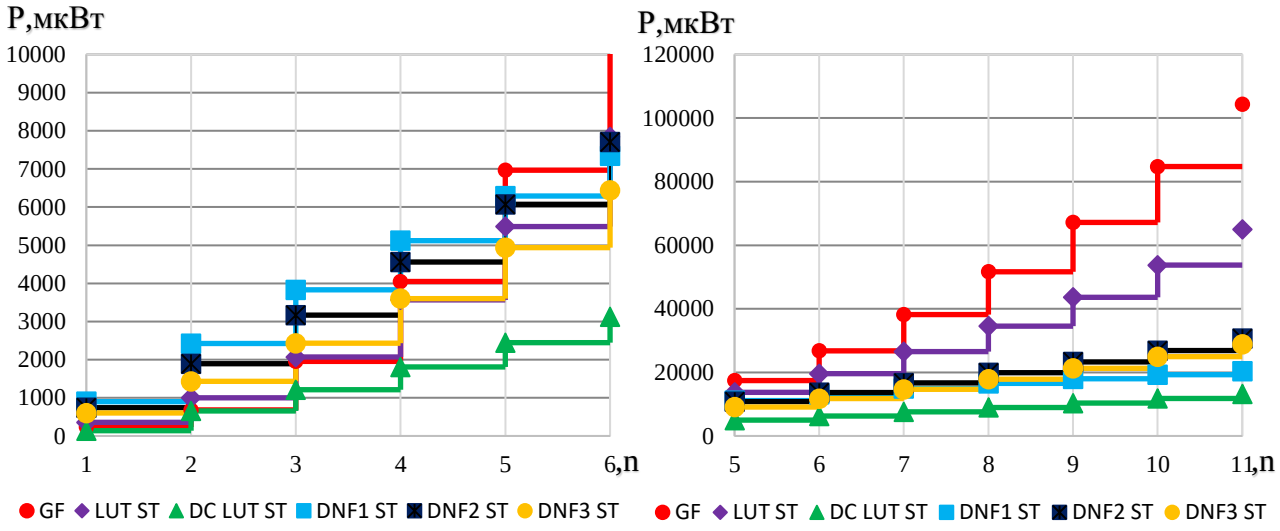
На основании полученных формул построим графики зависимости энергопотребления от числа переменных (n), числа функций (m) и числа конъюнкций (w).




а) б)

Рисунок 4.8 – Графики сравнения задержки разработанных элементов, где GF – генератор функций на ЛЭ, LUT-ST – элемент LUT ST, DC LUT-ST –

дешифратор DC LUT-ST, DNF1 ST – блок конъюнкции на ЛЭ, DNF2 ST - блок конъюнкции с подтягивающим резистором, DNF3 ST - блок конъюнкции с транзистором ортогональности; а) – при m=1, w=1; б) – при m=4, w=3.




а) б)

Рисунок 4.9 – Графики сравнения задержки разработанных элементов, где GF – генератор функций на ЛЭ, LUT-ST – элемент LUT ST, DC LUT-ST – дешифратор DC LUT-ST, DNF1 ST – блок конъюнкции на ЛЭ, DNF2 ST - блок конъюнкции с подтягивающим резистором, DNF3 ST - блок конъюнкции с транзистором ортогональности; а) – при m=10, w=10; б) – при m=25, w=15.

На рисунках 4.8 и 4.9 видно, что элементы LUT ST и ГФ имеют преимущество при реализации небольшого количества функций и небольшого числа переменных до n=4. Для количества переменных и большого числа функций преимущество по потребляемой мощности имеет элемент DC LUT-ST, выигрыш по сравнению с самым худшим вариантом составляет от 2 до 8 раз.

### 4.5. Разработка алгоритма

В качестве вариантов реализации систем логических функций рассматриваются разработанные логические элементы:

1. Генератор функций на основе библиотечных элементов - ГФ;
2. Генератор функций на основе универсального элемента LUT – LUT ST;
3. Генератор функций на основе элемента DC LUT – DC LUT ST;

4. Генератор функций на основе усовершенствованного элемента для ПЛМ – DNF 3 ST.

Задачу выбора оптимального набора конфигурируемых логических элементов для реализации заданных характеристик систем логических функций предлагается свести к задаче оптимизации назначений. Однако варианты наборов требуется сравнивать не по одному параметру, а по нескольким, таким как: сложности в количестве транзисторов, площади, занимаемой на кристалле, быстродействию и энергопотреблению. Данные параметры получены на стадии функционального проектирования. Поэтому предлагается решение многокритериальной задачи с дальнейшим построением Парето-оптимального множества из которого выбираются оптимальные наборы конфигурируемых логических элементов для реализации заданных характеристик систем логических функций.

Таким образом, для решения поставленной задачи предлагается разработать алгоритм выбора оптимального набора конфигурируемых логических элементов [104].

В первом этапе алгоритма формируется абстрактное множество на основании заданных характеристик системы.

$$M = \{m_i\}_{i=1,l} \tag{4.40}$$

где $m_i$ – выбранные конфигурируемые генераторы функций, $l$ – количество конфигурируемых генераторов функций.

Для формирования абстрактного множества вводятся следующие параметры:

1. Множество разработанных конфигурируемых генераторов функций и количество допустимых логических элементов в системе (N);

2. Параметры необходимые для вычисления значений функций: $m_{dnf}$ – разрядность ДНФ-LUT-ST, $w$ – количество конъюнкций ДНФ-LUT-ST и $c$ – количество столбцов таблицы сложности, для каждого из которых задается строка ($n_i$, $m_i$, $y_i$), где $n_i$ – количество переменных, $m_i$ – количество функций, $y_i$ – количество систем, $i=1,c$.

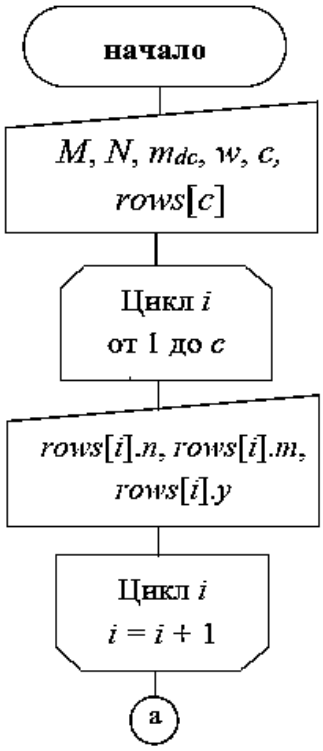



Рисунок 4.10 – Схема этапа I алгоритма, где rows[c] – множество строк ($n_i$, $m_i$, $y_i$), а – конец этапа.

На втором этапе после того, как было сформировано множество *M*, находится булеан *B:*

$$B(M) = \{b | b \subseteq M\} \backslash \{\emptyset\} \tag{4.41}$$

Далее необходимо найти множество всевозможных наборов логических элементов *E*. Для этого по каждому элементу $b \in B$, определяются подмножества $E_i \subset E$, *i=1,|E|*.

Элементами подмножества $E_i = \{e_j\}_{j=1,|E_i|}$ являются наборы ЛЭ, которые должны определяться по следующим правилам:

1. Сумма элементов в наборе не должна превышать число допустимых логических элементов в системе (N);
2. Генераторы функций, использующиеся в наборе, не дублируются.

Эти правила можно описать выражением:

$$E_i = \left\{ \bigcup_{j=1}^{|b|} (a_j * f_j) \middle| \sum_{t=1}^{|M|} a_t = N, f_t \in M \right\}. \tag{4.42}$$

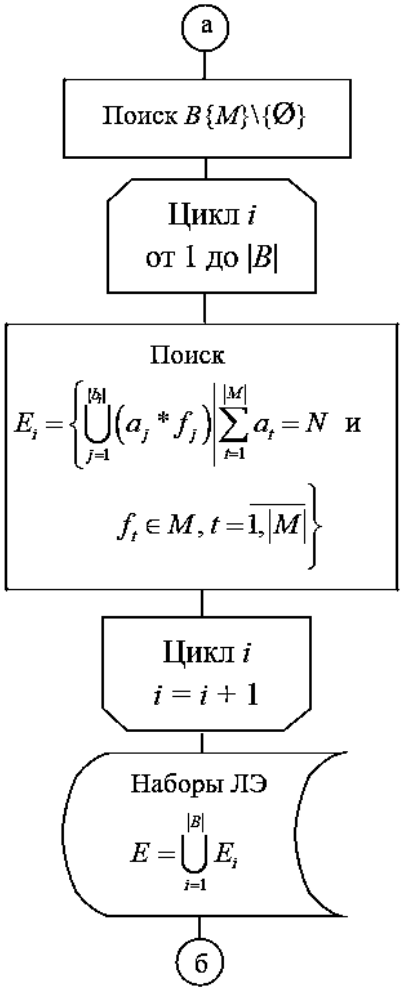



Рисунок 4.9 – Схема этапа II алгоритма, где $E_i \subset E$ – подмножество множества $E$, полученное для каждого $b_i \in B$,
а и б – начало и конец этапа.

На третьем этапе происходит формирование таблицы сложности по типу таблицы представления в венгерском методе, отличающейся тем, что учитывается несколько критериев.

Таблица 4.4. Назначение вариантов наборов ЛЭ для реализации различных заданных характеристик систем

Начало таблицы 4.4

| Варианты наборов ЛЭ | Характеристики различных систем логических функций | | | |
|---|---|---|---|---|
| | $n_0$, $m_0$, $y_0$, $m_{dnf}$, $w$ | $n_1$, $m_1$, $y_1$, $m_{dnf}$, $w$ | …. | $n_c$, $m_c$, $y_c$, $m_{dnf}$, $w$ |
| $N*m_0$ | $V_{1.0}$ | $V_{1.1}$ | | $V_{1.c}$ |
| … | … | … | | … |
| $N*m_l$ | $V_{l.0}$ | $V_{l.1}$ | | $V_{l.c}$ |

Окончание таблицы 4.4

| Варианты наборов ЛЭ | Характеристики различных систем логических функций | | | |
|---|---|---|---|---|
| | $n_0, m_0, y_0, m_{dnf}, w$ | $n_1, m_1, y_1, m_{dnf}, w$ | …. | $n_c, m_c, y_c, m_{dnf}, w$ |
| $(m_0 \ (N\text{-}1)^* \ m_1)$ | $V_{l+1.0}$ | $V_{l+1.1}$ | | $V_{l+1.c}$ |
| … | … | … | | … |

Размерность таблицы 4.4 $|E|\times|C|$, где $C$ – множество столбцов. Для каждого элемента $c_i \in C$ задаются параметры:

$$n_i, m_i, y_i, m_{dnf}, w, i=1,|C| \tag{4.43}$$

где $n_i$ – количество переменных, $m_i$ – количество функций, $y_i$ - количество систем, $m_{dnf}$ – разрядность DC LUT ST, $w$ – число конъюнкций ДНФ ST.

Каждой строке таблицы 4.4 соответствуют наборы разработанных конфигурируемых ЛЭ $e_i \in E$, $i=1,|E|$. Элементы в строке заполняются согласно следующему алгоритму:

1. Для соответствующего набора ЛЭ $e_i \in E$ находится минимальный поднабор ЛЭ $e_i' \subset e_i$, такой, что значение логической функции является минимальным:

$$e_i' = \left\{ e_i' \subset e_i \left| \sum_{j=1}^{|e_i'|} a_j \cdot f_j(n_i, m_i, y_i, m_{dnf}, w) \rightarrow min \right. \right\} \tag{4.44}$$

где $a_j$ – количество ЛЭ соответствующей логической функции $f_j$.

2. Вычисление следующего значения в строке находятся, как и в п.1, но вместо набора ЛЭ $e_i \in E$ берется набор $e_j' = e_i \cap e_i' \cap \ldots \cap e_{j-1}'$, $i=1,|E|$, $j=1,|C|$, т.е. набор, который получается пересечением исходного набора ЛЭ с минимальными поднаборами, полученными в предыдущих столбцах.

Вычисление значений элементов в строке выполняется до тех пор, пока в исходном наборе $e_i \in E$, $i=1,|E|$ достаточно логических элементов. Если при вычислении следующего значения элемента в строке недостаточно ЛЭ, то расчет прекращается и алгоритм переходит к следующему набору логических элементов.

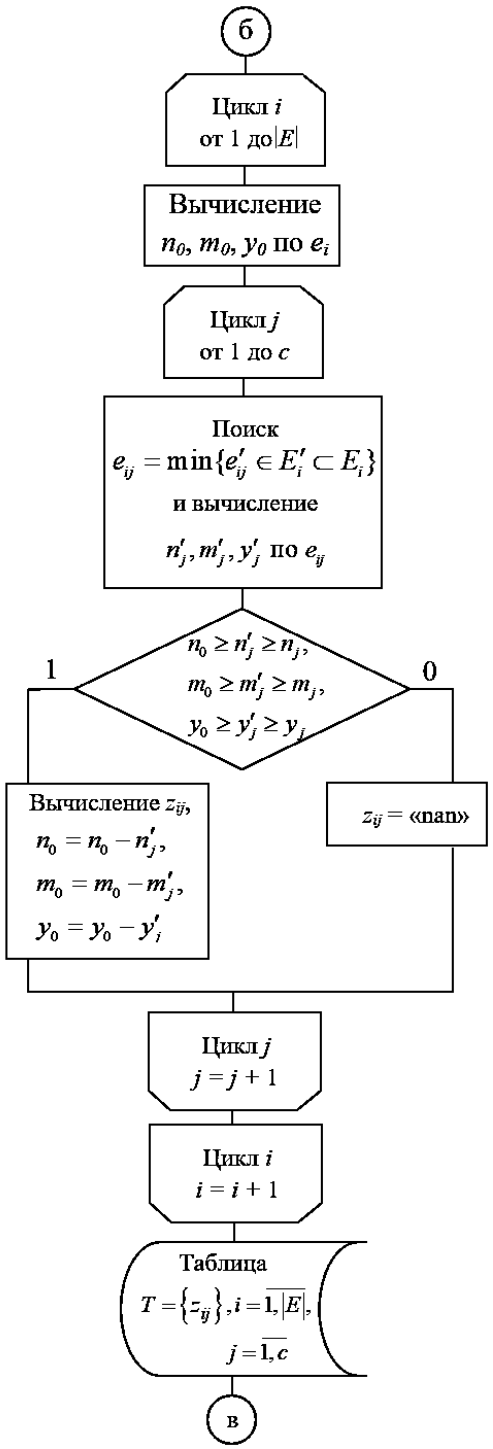



Рисунок 4.10 – Схема этапа III алгоритма, где $E$ – множество всевозможных наборов ЛЭ, ($n_0$, $m_0$, $y_0$) – параметры для текущего набора из $E$, $T$ – таблица сложности, $z_{ij}$ – минимальное значение логической функции, б и в – начало и конец этапа.

На четвертом этапе происходит поиск оптимальных наборов логических элементов по сформированной таблице сложности. Поиск множества оптимальных наборов происходит по каждому из параметров. По сложности в

количестве транзисторов ($L$), по площади, занимаемой на кристалле ($S$) и по потребляемой мощности ($P$) идет поиск минимального покрытия, т.е. находится покрытие с минимальным значением сложности каждого из параметров. По критерию быстродействия ($T$) идет поиск покрытия, в котором содержится минимальный элемент из максимальных значений логических функций каждой строки таблицы сложности. Далее по найденным покрытиям формируются оптимальные наборы логических элементов.

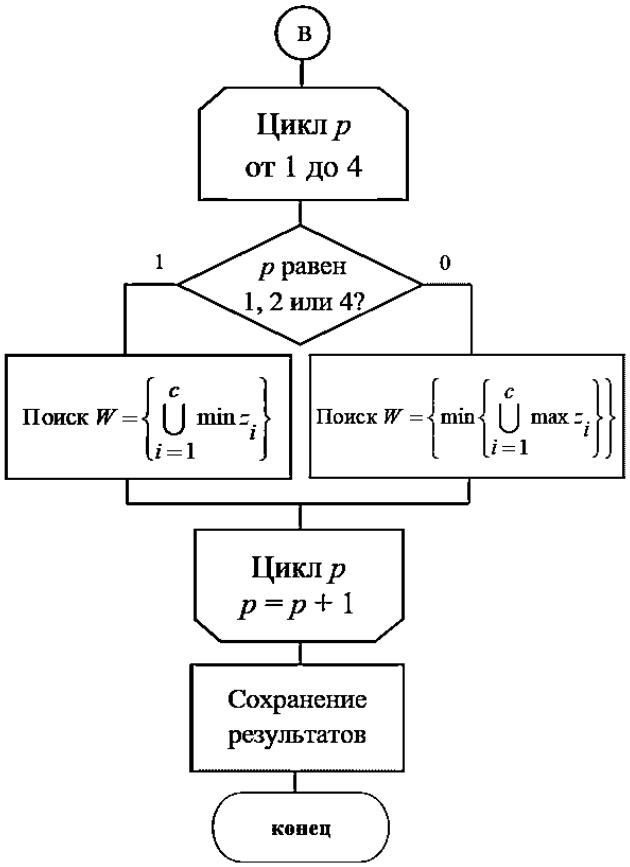



Рисунок 4.11 – Схема этапа IV, где W (V) – минимальное покрытие, z – значение логической функции таблицы сложности, $c$ – количество столбцов в таблице сложности, в – начало этапа.

Для более удобного использования данного алгоритма была разработана портативная программа на языке C++ – «Программа выбора оптимального набора ССС ЛЭ». Листинг программы показан в приложении 1.

При запуске программы появляется главное окно, имеющее вид, показанный на рисунке 4.12.

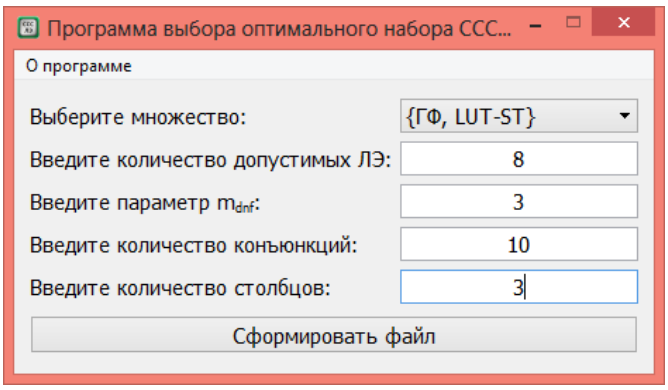



Рисунок 4.12 – Главное окно программы – «Программа выбора оптимального набора ССС ЛЭ».

Пользователю предлагается выбор типов логических элементов, на основании которых будет сформировано непустое множество M, задание количества допустимых логических элементов в проекте и др.

После того, как пользователь введет требуемые параметры проекта и нажмет на кнопку «Сформировать файл», откроется окно ввода критериев для проекта.

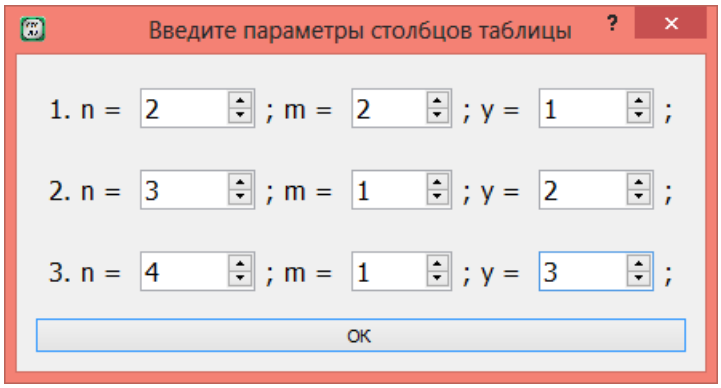



Рисунок 4.13 - Окно ввода критериев, которые формируют таблицу сложности, где n – количество переменных, m – количество функций, y – количество систем.

При нажатии на кнопку «ОК» сформируется файл в формате *.xlsx с вычисленными значениями таблицы сложности и оптимальными наборами ЛЭ. А также отобразится диалоговое окно с графическим представлением найденных значений в плоскости (S, T, P).

Рассмотрим применение алгоритма с использованием разработанной программы.

Пусть задано множество генераторов функций: ГФ, LUT-ST, DC LUT ST и ДНФ ST. Количество допустимых ЛЭ равно 8. Количество проектов=1, число переменных =4, количество функций =1, число систем =1.

При помощи программы, разработанной с использованием алгоритма, описанного выше, получены оптимальные наборы:

1. 1*LUT-ST,
2. 1*ДНФ ST

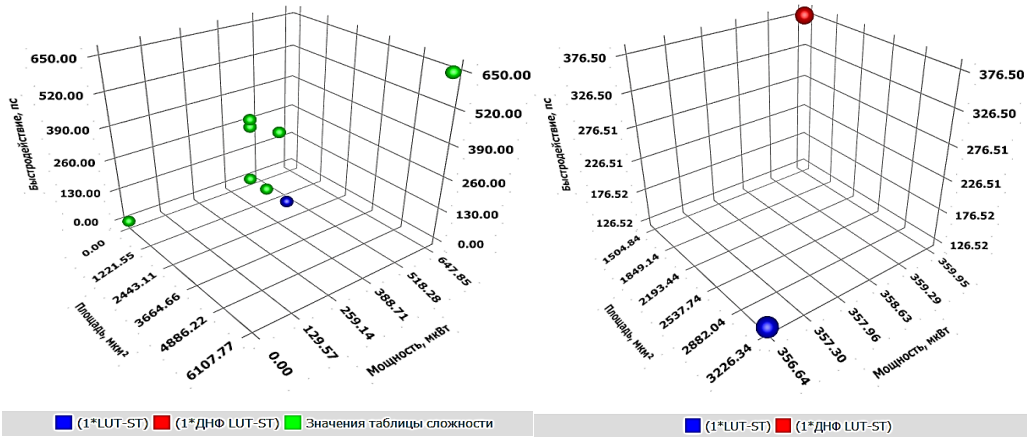
Количественные характеристики для полученных оптимальных наборов ЛЭ определены таблицей 4.5. Цветом выделены минимальные значения для каждого параметра.

Таблица 4.5. Значения количественных характеристик для наборов ЛЭ.

| № | *L*, шт | *S*, мкм$^2$ | *T*, пс | *P*, мкВт |
|---|---|---|---|---|
| 1*LUT-ST | 596 | 3226.34 | 126.52 | 356.64 |
| 1*ДНФ ST | 1454 | 1504.84 | 376.5 | 359.95 |

Из таблицы 4.5 видно, что для реализации логической функции с заданными характеристиками требуется всего один элемент, причем элемент LUT-ST имеет выигрыш в отношении ДНФ ST почти по всем критериям, уступая только в площади.

Графическое представление вычисленных данных отражено на Рис. 4.14




а) б)

Рисунок 4.14 – Результат работы алгоритма по заданных условиям,

а) результат работы алгоритма, б) Парето-оптимальное множество.

Пусть задано множество генераторов функций: ГФ, LUT-ST, DC LUT ST и ДНФ ST. Количество допустимых ЛЭ равно 8. Количество проектов =4. Для каждого из проектов заданы следующие параметры:

1. n=5, m=7, y=1, $m_{dc}$=1, w=1;
2. n=15, m=8, y=1, $m_{dc}$=1, w=1;
3. n=8, m=3, y=1, $m_{dc}$=1, w=1;
4. n=25, m=15, y=1, $m_{dc}$=1, w=1.

При помощи программы, получен оптимальный набор:

1. 4*ДНФ ST.

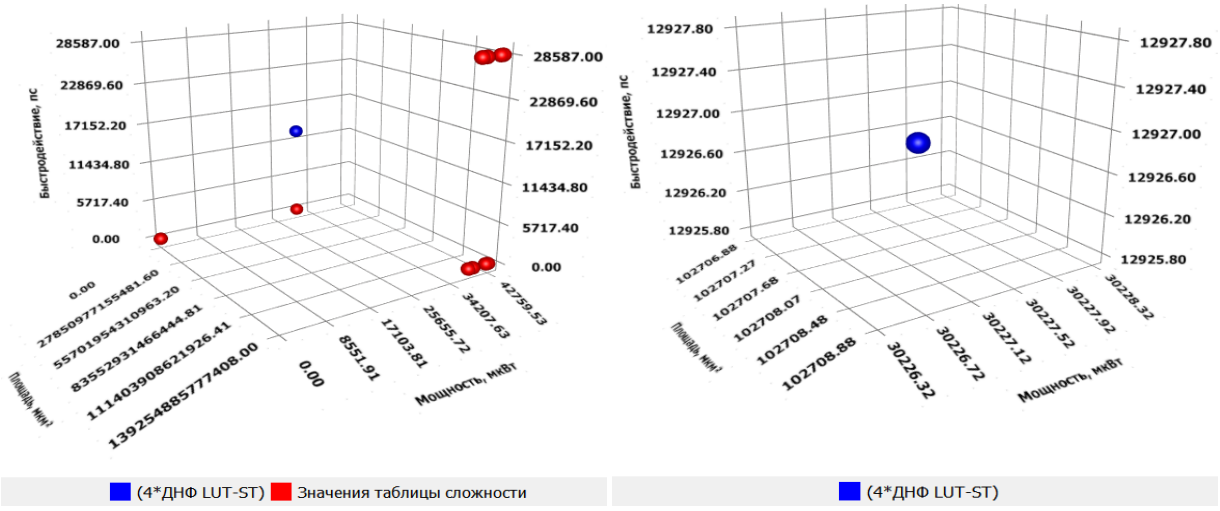
Вычисленные значения характеристик представлены в таблице 4.6. Цветом выделены минимальные значения для каждого параметра.

Таблица 4.6. Значения количественных характеристик для наборов.

| № | *L*, шт | *S*, мкм$^2$ | *T*, пс | *P*, мкВт |
|---|---|---|---|---|
| 4*ДНФ ST | 2.88386e+10 | 102708 | 12926.8 | 30227.3 |

Кроме таблицы с результатом оптимального набора ЛЭ, выводится таблица сложности 4.7.

Графическое представление вычисленных данных отражено на Рис. 4.15




а) б)

Рисунок 4.15 – Результат работы алгоритма по заданным условиям,

а) результат работы алгоритма, б) Парето-оптимальное множество.

Таблицы 4.7 Фрагмент таблицы сложности

| **Варианты наборов ЛЭ** | **n=5 m=7 y=1 m1=1 k=1** | | | | **n=15 m=8 y=1 m1=1 k=1** | | | | **n=8 m=3 y=1 m1=1 k=1** | | | | **n=25 m=15 y=1 m1=1 k=1** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *L*, шт | *S*, мкм$^2$ | *T*, пс | *P*, мкВт | *L*, шт | *S*, мкм$^2$ | *T*, пс | *P*, мкВт | *L*, шт | *S*, мкм$^2$ | *T*, пс | *P*, мкВт | *L*, шт | *S*, мкм$^2$ | *T*, пс | *P*, мкВт |
| (8*DC LUT-ST) | 15024 | 22309.76 | 2025 | 1937.57 | 17301950 | 1559027506 | 9974 | 9853.98 | 63566 | 537147 | 2606 | 2562.33 | 3087007888 | 1.39253327749e+14 | 28587 | 28405.65 |
| (8*ДНФ LUT-ST) | 13458 | 6120.1 | 1554.8 | 1509.52 | 15305908 | 23104.4 | 4588.8 | 6767.13 | 48810 | 5712.48 | 1224.1 | 1576.67 | 2882326616 2 | 67770.9 | 12926.8 | 20374 |
| (8*LUT-ST) | 7798 | 66503.85 | 1043.14 | 3842.44 | nan | nan | nan | nan | nan | nan | nan | nan | nan | nan | nan | nan |
| (8*ГФ) | 59388 | 271514 | 25137 | 34138.3 | nan | nan | nan | nan | nan | nan | nan | nan | nan | nan | nan | nan |
| (1*LUT-ST) (7*ДНФ LUT-ST) | 13458 | 6120.1 | 1554.8 | 1509.52 | 15305908 | 23104.4 | 4588.8 | 6767.13 | 48810 | 5712.48 | 1224.1 | 1576.67 | 2882326616 2 | 67770.9 | 12926.8 | 20374 |
| (3*LUT-ST) (5*ДНФ LUT-ST) | 13458 | 6120.1 | 1554.8 | 1509.52 | 15305908 | 23104.4 | 4588.8 | 6767.13 | 23844 | 5712.48 | 739.56 | 1576.67 | nan | 67770.9 | nan | 20374 |
| (5*LUT-ST) (3*ДНФ LUT-ST) | 13458 | 6120.1 | 1554.8 | 1509.52 | 15305908 | 23104.4 | 4588.8 | 6767.13 | 23844 | 5712.48 | 739.56 | 1576.67 | nan | nan | nan | nan |

Из фрагмента таблицы сложности видно, что для реализации заданных проектов у некоторых наборов логических элементов заканчиваются ресурсы: например, элементы ГФ и LUT ST могут реализовать только по одному проекту из четырех, а у набора (3*LUT-ST) + (5*ДНФ LUT-ST) заканчивается ресурс после реализации трех проектов из четырех.

Кроме этого, наряду с набором ДНФ ST набор DC LUT ST тоже может реализовать заданные условия, но для этого потребуется намного больше аппаратных затрат.

Таким образом, при небольшом количестве переменных и функций оптимально использовать наборы, состоящие преимущественно из элементов ГФ и LUT ST. Если требуется реализовывать проекты с большим числом переменных и большим количеством функций, то рекомендуется использовать наборы элементов DC LUT ST и ДНФ ST.

### 4.6. Выводы по главе

1. Выполненное исследование сложности в количестве транзисторов разработанных логических элементов показало, что если требуется реализовать небольшое количество логических функций, зависящих от небольшого числа переменных, то элемент LUT ST имеет преимущество в количестве транзисторов перед элементами DC LUT-ST и ДНФ, при этом ГФ имеет проигрыш всего в 1-3%. Если требуется реализовать большое число логических функций от числа переменных до n=7, то преимущество имеет элемент DC LUT-ST, выигрыш по сравнению с самым худшим вариантом составляет от 13% до 43%. Если требуется реализовать большое число логических функций и большое число переменных, то преимуществом обладают элементы ДНФ 2 ST (блок конъюнкций с подтягивающим резистором) и ДНФ 3 ST (блок конъюнкций с транзистором ортогональности), который проигрывает ДНФ 2 ST 2% транзисторов.

2. Выполненный анализ быстродействия разработанных элементов показал, что при небольшом количестве переменных и небольшом числе

логических функций преимуществом обладают элементы ГФ и LUT ST, причем при небольшом числе переменных до n=3 выигрыш в быстродействии имеет элемент ГФ от 1-37%, после задержка увеличивается пропорционально при увеличении числа переменных. Если требуется реализовать небольшое число функций, зависящее от малого числа переменных (до n=5), однозначно выбрать, какой элемент выигрывает сложно. Если требуется реализовать большое число функций с большим числом переменных, то явным преимуществом обладает элемент DC LUT ST, быстродействие которого в 140-240% выше, чем у самого худшего варианта.

3. Выполненный анализ площади, занимаемой на кристалле, показал, что зависимость от числа переменных сравнима с результатами оценки сложности в количестве транзисторов, за исключением элементов, в которых присутствуют резисторы. Так для реализации большого количества функций и числа переменных от n=7 преимуществом обладает элемент блока конъюнкций с транзистором ортогональности (ДНФ 3 ST), который не обладал явным выигрышем в сложности реализации, выигрыш по сравнению с самым худшим вариантом составляет в 1770 раз.

4. Выполненный анализ быстродействия разработанных элементов показал, что что элементы LUT ST и ГФ имеют преимущество при реализации небольшого количества функций и небольшого числа переменных до n=4. Для количества переменных и большого числа функций преимущество по потребляемой мощности имеет элемент DC LUT-ST, выигрыш по сравнению с самым худшим вариантом составляет от 2 до 8 раз.

5. Таким образом, выполненные анализы по отдельным критериям, показали, что нельзя учитывать только один их критериев для выбора оптимального логического элемента.

6. Разработанный алгоритм для решения многокритериальной задачи показал, что при небольшом количестве переменных и функций оптимально использовать наборы, состоящие преимущественно из элементов ГФ и LUT ST. Если требуется реализовывать проекты с большим числом переменных и

большим количеством функций, то рекомендуется использовать наборы элементов DC LUT ST и ДНФ ST.

**ЗАКЛЮЧЕНИЕ**

Выполненная диссертационная работа посвящена решению научно-технической задачи выбора оптимального набора конфигурируемых логических элементов для унифицированной реализации систем логических функций в самосинхронных схемах. В диссертационной работе поставлены и решены следующие задачи исследования:

1. Разработан метод реализации конфигурируемого самосинхронного генератора логических функций на основе библиотечного базиса 2И-2ИЛИ-НЕ, отличающийся тем, что библиотечный элемент адаптирован к условиям работы в ССС. Для этого применяется парафазная дисциплина кодирования сигнала и используется фаза спейсера, причем для согласованности работы блоков схемы при количестве переменных n>1 спейсер каждого слоя блоков изменяется;

2. Разработан метод реализации конфигурируемого самосинхронного генератора логических функций в ССС по принципу LUT (Look Up Table), используемому в программируемых логических интегральных схемах (ПЛИС типа FPGA), отличающийся тем, что используется дополнительная ветвь дерева передающих транзисторов, активируемая в фазе спейсера, а двойственный канал универсального логического элемента настраивается инверсными константами;

3. Разработан метод реализации конфигурируемого самосинхронного генератора систем логических функций, заданных в СДНФ на основе DC LUT FPGA, отличающийся тем, что он адаптирован к работе в ССС;

4. Разработан метод реализации конфигурируемого самосинхронного генератора систем логических функций, заданных в ДНФ на основе блока конъюнкций, отличающийся тем, что он адаптирован к работе в ССС;

5. Получены оценки сложности в количестве транзисторов, площади, задержки реализации логических функций на основе разработанных логических элементов по результатам моделирования;

6. Разработан алгоритм выбора оптимального набора конфигурируемых логических элементов для реализации типовых систем логических функций.

Дальнейшие исследования целесообразны в области самосинхронной реализации матриц коммутации ПЛИС.

## СПИСОК ТЕРМИНОВ И СОКРАЩЕНИЙ

CPLD - Complex Programmable Logic Device;

DC LUT – Decoder Look Up Table;

DCM - Digital Clock Manager;

DI - Delay-insensitive;

DLL - Delay Locked Loop;

FPGA – Field-Programmable Gate Array;

FU- Functional Unit;

GCG - Global Clock Generator;

IEEE - Institute of Electrical and Electronics Engineers;

LUT – Look Up Table;

NCL - Null Convention Logic;

NI – National Instruments;

PLL - Phase Locked Loop;

PUPD - PullUp, Pull Down;

RCM - Regional Clock Manager;

SI - Speed-independent;

SiP - System-in-Package;

ST – Self-timed;

STACC - Self-timed Array of Configurable Cells;

АЛМ - адаптивный логический модуль;

БМК – базовый матричный кристалл;

ГФ – генератор функций;

КНЗ – константная неисправность типа залипание;

ПЛИС – программируемая логическая интегральная схема;

САПР – система автоматизированного проектирования;

СДНФ – совершенная дизъюнктивная нормальная форма;

ССС –самосинхронная схема;

ФПТ – функционально-полный толерантный элемент;

ЭВМ – электронно-вычислительная машина.

## СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Каменских А.Н., Степченков Ю.А, Тюрин С.Ф. Проблема анализа полумодулярности и энергонадежности отказоустойчивых самосинхронных схем / М.: Знак, Электротехника. - 2015. - № 11. - С. 27-31.

2. Каменских А.Н. Комбинированное резервирование самосинхронных схем. Диссертация на соискание учёной степени кандидата технических наук. – URL: https://www.dissercat.com/content/kombinirovannoe-rezervirovanie-samosinkhronnykh-skhem. Научная библиотека диссертаций и авторефератов disserCat http://www.dissercat.com (дата обращения 21.02.2020)

3. Плеханов Л.П. Основы самосинхронных электронных схем. / М.: БИНОМ. Лаборатория знаний, 2013. - 208 с.

4. Сурков А.В. Устойчивые к параметрическим отказам самосинхронные сопроцессоры конвейерного типа для встраиваемых микропроцессоров. Автореферат на соискание учёной степени кандидата технических наук. / URL: https://www.dissercat.com/content/samosinkhronnye-soprotsessory-konveiernogo-tipa-dlya-ekstremalnykh-uslovii-ekspluatatsii Научная библиотека диссертаций и авторефератов disserCat http://www.dissercat.com (дата обращения 21.02.2020)

5. Степченков Ю. А. и др. Библиотека элементов для проектирования самосинхронных полузаказных микросхем серий 5503/5507 и 5508/5509 / М.: ИПИ РАН, 2012. – 238 с.

6. S. Pontarelli, M. Ottavi, V. Vankamamidi, A. Salsano, F. Lombardi. Reliability Evalution of Repairable / Reconfigurable FPGAs / 21st IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT’06), October, 2006. – pp. 227-235.

7. Строгонов А., Цыбин С. Программируемая коммутация ПЛИС: взгляд изнутри. – URL: http://www.kit-e.ru/articles/plis/2010_11_56.php (дата обращения: 16.12.2019)

8. Вихорев Р.В. Логические элементы ПЛИС FPGA для реализации систем функций. Диссертация на соискание учёной степени кандидата

технических наук. – URL: https://www.dissercat.com/content/logicheskie-elementy-plis-fpga-dlya-realizatsii-sistem-funktsii. Научная библиотека диссертаций и авторефератов disserCat http://www.dissercat.com (дата обращения: 28.03.2020)

9. Smith S. C. Design of an FPGA Logic Element for Implementing Asynchronous NULL Convention Logic Circuits. – URL: https://ieeexplore.ieee.org/document/4231891 (дата обращения: 10.01.2020).

10. Tyurin S. F. LUT based Fredkin gate / Radio Electronics, Computer Science, Control, 2020 – №1. – P.44-53.

11. Improving of a Circuit Checkability and Trustworthiness of Data Processing Results in LUT-based FPGA Components of Safety-Related Systems / A.V. Drozd, M. Drozd, O. Martynyuk, M. Kuznietsov. – CEUR Workshop Proceedings, 2017. – vol. 1844. – P. 654–661.

12. Drozd A.V. Use of Natural LUT Redundancy to Improve Trustworthiness of FPGA Design / A.V. Drozd, M. Drozd, M. Kuznietsov. – CEUR Workshop Proceedings, 2016. – vol. 1614. – P. 322–331.

13. Speedster22i Configuration User Guide. – URL: http://www.achronix.com/wp-content/uploads/docs/Speedster22i_Configuration_User_Guide_UG033.pdf (дата обращения: 10.01.2020).

14. C. G. Wong, A. J. Martin and P. Thomas. An architecture for asynchronous FPGAs / IEEE Int. Conference on Field-Programmable Technology (FPT), 2003. – P. 170—177.

15. D. Shang, F. Xia, A. Yakovlev. Asynchronous FPGA architecture with distributed control / IEEE Int. Symposium on Circuits and Systems (ISCAS), 2010. – P. 1436—1439.

16. Y. Komatsu, M. Hariyama, M. Kameyama. Architecture of an Asynchronous FPGA for Handshake-Component-Based Design / IEICE Transactions on Information and Systems, 2013 – vol. E96-D. – № 8. – P. 1632-1644.

17. P. S. K. Siegel. Automatic Technology Mapping for Asynchronous Designs. PhD dissertation / Stanford University, 1995. – 159 p.

18. S. W. Moore, P. Robinson. Rapid prototyping of self-timed circuits / IEEE Int. Conference on Computer Design (ICCD), 1998. – P. 360—365.

19. Hauck, S., Borriello, G., Burns, S., & Ebeling, C. MONTAGE: An FPGA for synchronous and asynchronous circuits / Field-Programmable Gate Arrays: Architecture and Tools for Rapid Prototyping, 1993 – P. 44-51.

20. Асинхронный арбитр / URL: https://vak.dreamwidth.org/405837.html?thread=4038477&style=light (дата обращения: 01.12.2019).

21. Muller D. E., Bartky W.S. A Theory of Asynchronous Circuits / Proceedings of the International Symposium on the Theory of Switching. – Part I. – Harvard University Press, 1959. – P. 204-243

22. Апериодические автоматы / Под ред. В.И. Варшавского. - М.: Наука, 1976. - 424 с.

23. Hauck S. Asynchronous design methodologies: An overview / Proceedings of the IEEE, 1995. – vol. 83. – №. 1. – P. 69-93.

24. Marakhovsky V.B. Globally asynchronous system of interactive Moore state machines / V.B. Marakhovsky, A.V. Surkov. – IET Computers and Digital Techniques, 2016. – vol. 10, Issue 4. – P. 186-192.

25. Yakovlev A.V., Koelmans A.M., Lavango L. High-level modeling and design of asynchronous interface logic / IEEE Design & Test, 2010. - vol. 12. - № 1. - P. 32-40.

26. Филин А.В. Исследование нетрадиционных подходов к созданию компьютеров гарантированно высокой надежности / Филин А.В., Степченков Ю.А., Петрухин В.С., Гринфельд Ф.И. – Системы и средства информатики. – М.: Наука, 1993. – № 5. – С.181-196.

27. Цирлин Б.С. Базисные реализации полумодулярных схем / Техническая кибернетика, 1983. - №5. - 194 с.

28. Плеханов Л.П., Денисов А.Н., Дьяченко Ю.Г., Степченков Ю.А., Мамонов Д.И., Степченков Д.Ю. Синтез самосинхронных схем в базисе БМК / Наноиндустрия, 2020. – № S96-2. – с. 460-470.

29. Денисов А., Коняхин В. Перспективная элементная база для аппаратуры с жесткими условиями эксплуатации / Наноиндустрия, 2016. – № 8 (70). – с. 22-31.

30. Bobkov S.G. Self-timed Circuits for Low Power and Highly Reliable Microprocessors / Bobkov S.G., Gorbunov M.S., Diachenko Yu.G., Rozhdestvenskij Yu.V., Stepchenkov Yu.A., Surkov A.V. – Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС), 2015. – № 1. – С. 12-13.

31. Бобков С.Г. Опыт разработки и производства микросхем промышленного назначения / Наноиндустрия, 2018. – № S (82). – С. 24-27.

32. J. Brady J. Di. Radiation-Hardened Delay Insensitive Asynchronous Circuits / Single Event Effects Symposium, May 2014.

33. P. Shepherd. Wide-Temperature Data Transmission System for Space Environments / P. Shepherd, S. C. Smith, J. Holmes, A. M. Francis, N. Chiolino, H. A. Mantooth. A Robust. – IEEE Aerospace Conference. – Big Sky, Mt, 2013.

34. J. Brady. An Asynchronous Cell Library for Operation in Wide-Temperature & Ionizing-Radiation Environments / J. Brady, A. M. Francis, J. Holmes, J. Di, H. A. Mantooth. – IEEE Aerospace Conference. – Big Sky, Mt, 2015.

35. K. M. Fant, S. A. Brantd. NULL Convention Logic: a complete and consistent logic for asynchronous digital circuit synthesis / ASAP 96. – Chicago, IL, Aug. 1996.

36. S. C. Smith, J. Di. Designing Asynchronous Circuits using NULL Convention Logic (NCL) / Morgan Claypool Publishers, 2009.

37. Варшавский В.И, Мелешина М.В, Цетлин М.Л. Поведение автоматов в периодических случайных средах и задача синхронизации при наличии помех / Проблемы передачи информации, 1965. - Т. 1. - №1. - С. 65-71

38. Махаровский В.Б. КМОП-реализация обучаемого порогового логического элемента. Часть 1: Проектирование и схема обучения / Информационно-управляющие системы, 2014. - № 3 (70). - С. 47-56.

39. GALA (Globally Asynchronous - Locally Arbitrary) Design", LNCS 2549, Concurrency and Hardware Design, Advances in Petri Nets/ [eds.: Jordi Cortadella, Alex Yakovlev, and Grzegorz Rozenberg].– Berlin: Springer, 2002.– P. 61-107.

40. Степченков Ю.А., Петрухин В.С., Дьяченко Ю.Г. Опыт разработки самосинхронного ядра микроконтроллера на базовом матричном кристалле / Нано- и микросистемная техника, 2006. - №5. - С. 29-36.

41. Дьяченко Ю. Г., Степченков Ю. А., Бобков С. Г. Квазисамосинхронный вычислитель: методологические и алгоритмические аспекты / Проблемы разработки перспективных микро-и наноэлектронных систем (МЭС), 2008. – №1. – с. 441-446.

42. Тюрин С. Ф. Программируемый логический элемент для самосинхронных схем / Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии, 2016. – № 3. – С. 106-110.

43. Плеханов Л.П. Проектирование самосинхронных схем: функциональный подход / Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС-2010): Сб. трудов IV Всеросс. научно-технич. конф. – М.: ИППМ РАН, 2010. – Вып. 1. – С. 26-31.

44. Yakovlev A. Energy-modulated computing. In Design, Automation & Test in Europe Conference & Exhibition (DATE) / IEEE, 2011.– P. 1-6.

45. Automating the Design of Asynchronous Logic Control for AMS Electronics / [Danil Sokolov, Victor Khomenko, Andrey Mokhov, et al.] – IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020. – Vol. 39, Issue 5. – P 952-965.

46. Speed-independent floating point coproceccor / [Y.A. Stepchenkov, V.N. Zakharov, Y.V. Rogdestvenski, et al.] – IEEE East-West Design & Test Symposium:

International, 26-29 Sept. 2015: proceedings. – Batumi, Georgia: IEEE, 2015. – P. 7493110.

47. Степченков Ю. А. и др. Самосинхронный умножитель с накоплением: практическая реализация / Системы и средства информатики, 2014. - Т.24. - №3. - С.63-77.

48. Hollosi B. at al. Delay-insensitive asynchronous ALU for cryogenic temperature environments / 51st Midwest Symposium on Circuits and Systems. - IEEE, 2008. - P. 322-325.

49. Степченков Ю. А. и др. Самосинхронные схемы-ключ к построению эффективной и надежной аппаратуры долговременного действия / Наукоемкие технологии, 2007. – Том 8. – №. 5-6. – С. 73-89.

50. Self-timed Circuits / URL: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-004-computation-structures-spring-2017/c7/c7s2/c7s2v5/self-timed-circuits-6-21 (дата обращения: 10.01.2020).

51. Walker A. An Approach for Self-Timed Synchronous CMOS Circuit Design / URL: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20040068174.pdf (дата обращения: 10.01.2020).

52. Nielsen L. S. Low-power operation using self-timed circuits and adaptive scaling of the supply voltage / URL: https://backend.orbit.dtu.dk/ws/files/4151035/Nielsen.pdf (дата обращения: 10.01.2020).

53. Masashi I. Performance Comparison between Self-timed Circuits and Synchronous Circuits Based on the Technology Roadmap of Semiconductors / URL: http://webhost.laas.fr/TSF/WDSN08/2ndWDSN08(LAAS)_files/Texts/WDSN08-05-Imai.pdf (дата обращения: 10.01.2020).

54. Kenny R. The Breakthrough Advantage for FPGAs with Tri-Gate Technology / URL: https://www.altera.com/en_US/pdfs/literature/wp/wp-01201-fpga-tri-gate-technology.pdf (дата обращения: 10.01.2020).

55. Плеханов Л.П. Проектирование самосинхронных схем: структурные методы в иерархическом анализе / Информатика и ее применение, 2014. – Том 8. – №3. – С.105-113.

56. Плеханов Л.П. Иерархический метод анализа самосинхронных электронных схем / Системы и средства информатики, 2012. – Том 22. – №1. – С.62-73.

57. САПР "КОВЧЕГ". / URL: http://www.tcen.ru/rus/products/radiatsionnyy-kontrol/sapr-kovcheg (дата обращения: 21.02.2020)

58. Сайт разработчика National Instruments / URL: http://www.ni.com/multisim/ (дата обращения: 10.01.2020).

59. Коноплев Б.Г. Проектирование интегральных схем / Коноплев Б.Г., Рындин Е.А., Приступчик Н.К., Денисенко М.А. – Таганрог: Изд-во ТТИ ЮФУ, 2010. – 76 с.

60. Payne, R. Self-timed FPGA systems / Field-programmable Logic and Applications. – Springer, Berlin, Heidelberg, 1995. – P. 21-35.

61. Teife, J., & Manohar, R. (2003, September). Programmable asynchronous pipeline arrays / International Conference on Field Programmable Logic and Applications. – Springer, Berlin, Heidelberg, 2003 – P.345-354.

62. Maheswaran K., Akella V. Hazard-free implementation of the self-timed cell set in a xilinx fpga / Univ. Calif. Davis, Davis, CA, Tech. Report. – 1994.

63. Dugganapally, I. P. Design and Implementation of FPGA Configuration Logic Block Using Asynchronous Static NCL / Dugganapally, I. P., Al-Assadi, W. K., Tammina, T., Smith, S. – Region 5 Conference, IEEE, 2008 – P. 1-6.

64. Березняков С.В., Аверкиев М.А. Самосинхронные элементы и устройства / Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления, 2016. – № 1 (17). – С.80-94.

65. Ronald.C. SEE Mitigation Technique for Self-Timed Circuits and Rad-Hard, Self-Timed Configurable Memory / URL: https://www.techbriefs.com/component/content/article/tb/techbriefs/semiconductors-and-ics/21058 (дата обращения: 30.11.2019).

66. Carlos Gómez Osuna, Pablo Ituero, Marisa López-Vallejo. A Self-Timed Multipurpose Delay Sensor for Field Programmable Gate Arrays (FPGAs) / URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3926550 (дата обращения: 29.11.2019).

67. Mickael Fiorentino, Omar Al-Terkawi, Yvon Savaria, Claude Thibeault Self-timed circuits FPGA implementation flow / URL: http://ieeexplore.ieee.org/document/7182063 (дата обращения: 29.11.2019).

68. Bahram Rashidi. FPGA Based A New Low Power and Self-Timed AES 128-bit Encryption Algorithm for Encryption Audio Signal / URL: http://www.mecs-press.org/ijcnis/ijcnis-v5-n2/IJCNIS-V5-N2-2.pdf (дата обращения: 30.11.2019).

69. Phillip David Ferguson. Optimizing Self-Timed FPGA Circuits / Phillip David Ferguson, Aristides Efthymiou, Tughrul Arslan, Danny Hume. - URL: http://ieeexplore.ieee.org/document/5615544/ (дата обращения: 30.11.2019).

70. C. Traver, R.B. Reese, M.A. Thornton. Cell designs for self-timed FPGAs / URL: http://ieeexplore.ieee.org/document/954693/ (дата обращения: 30.11.2019).

71. INTEL FPGA / URL: https://www.altera.com/ (дата обращения: 23.11.2019).

72. SiP Products / URL: https://www.altera.com/products/sip/overview.html (дата обращения: 24.11.2019).

73. Understanding How the New Intel®HyperFlex™ FPGA Architecture Enables NextGeneration High-Performance Systems / URL: https://www.altera.com/products/fpga/stratix-series/stratix-10/features.html#hyperflexarchitecture (дата обращения: 20.02.2020).

74. Скорнякова А.Ю., Тюрин С.Ф. Синтез самосинхронных генераторов логических функций / Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления, 2020. - № 34. – С. 168-185.

75. Городилов А.Ю. Методы и алгоритмы диагностирования и реконфигурации логики высоконадёжных ПЛИС. Автореферат диссертации на соискание ученой степени кандидата технических наук / Пермский национальный исследовательский политехнический университет. Пермь, 2016.

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

77. Тюрин С.Ф., Скорнякова А.Ю. Универсальный логический элемент для самосинхронной схемы / Вестник Рязанского государственного радиотехнического университета, 2017 – № 61. – С. 41-45.

78. Plotnikova A. Yu. Fault-Tolerant Self-Timed Indicator / В сборнике: Proceedings of the 2016 IEEE North West Russia Section Young Researchers in Electrical and Electronic Engineering Conference, EIConRusNW 2016, 2016. – P. 308-312. (Scopus, Web of Science)

79. Плотникова А.Ю. Отказоустойчивый самосинхронный индикатор на основе самосинхронного базисного элемента / Авиация и космонавтика – 2015. тез. – Москва, 2015. – С. 196-197. (РИНЦ)

80. Плотникова А. Ю. Радиационно-стойкий индикатор для самосинхронной схемы / Элементная база отечественной радиоэлектроники: импортозамещение и применение: тр. II рос.-белорус. науч.-техн. конф. им. О. В. Лосева. – Нижний Новгород, 2015. – С. 85-90. (РИНЦ)

81. Патент № 2601145 Российская Федерация, МПК G11C 17/00 (2006.01). Программируемое логическое устройство : № 2015117840/08 : заявл. 12.05.2015 : опубл. 27.10.2016 / Тюрин С.Ф., Каменских А.Н., Скорнякова А.Ю. (Плотникова А.Ю.); заявитель ПНИПУ-12 с. : ил. – Текст : непосредственный.

82. Carver A. Mead, Lynn Conway. Introduction to VLSI Systems / URL: 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 (дата обращения: 21.10.2017).

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

84. Золотуха Р., Комолов Д. Stratix III — новое семейство FPGA фирмы Altera / URL: http://kit-e.ru/assets/files/pdf/2006_12_30.pdf (дата обращения 14.03.2020).

85. Logic Array Blocks and Adaptive Logic Modules in Stratix III Devices / URL: https://www.altera.com.cn/content/dam/altera-www/global/zh_CN/pdfs/literature/hb/stx3/stx3_siii51002.pdf (дата обращения: 29.06.2019).

86. Обзор архитектуры ПЛИС семейства Virtex-5 / URL: http://elektors.ru/radioelektronika/mikroshemy/2759-obzor-arhitektury-plis-semeystva-virtex-5.html (дата обращения: 27.08.2019).

87. Как объективно оценить параметры FPGA разных производителей? / URL: http://www.russianelectronics.ru/leader-r/review/2189/doc/40317/ (дата обращения: 21.01.2020).

88. Тюрин С.Ф., Набатов А.В., Громов О.А., Греков А.В., Карлов Д.А. Программируемое логическое устройство. Патент РФ № 2503993. Опубл. 10.01.2014. Бюлл. №1.

89. Тюрин С.Ф., Сулейманов А.А., Плотникова А.Ю. Реконфигурируемый логический элемент DC LUT для ПЛИС типа FPGA / Вестник Пермского университета. Серия: Математика. Механика. Информатика, 2015. – № 2 (29). – С. 67-71.

90. Тюрин С.Ф., Скорнякова А.Ю. Самосинхронный универсальный логический элемент для реализации систем функций / Инженерный вестник Дона, 2017. – № 1.

91. Патент № 2653304 Российская Федерация, МПК G06F 7/57 (2006.01), H03K 19/173 (2006.01). СПК G06F 7/57 (2006.01), H03K 19/173 (2006.01), G06F 12/0831 (2006.01) Программируемое логическое устройство : № 2017131825 : заявл. 11.09.2017 : опубл. 07.05.2018 / Тюрин С.Ф., Скорнякова А.Ю. заявитель ПНИПУ-20 с. : ил. – Текст : непосредственный.

92. Патент № 2653301 Российская Федерация, МПК G06F 7/57 (2006.01). СПК G06F 7/57 (2006.01), G06F 9/3887 (2006.01), G06F 15/8007 (2006.01). Программируемое логическое устройство : № 2017134253: заявл. 02.10.2017 : опубл. 07.05.2018 / Тюрин С.Ф., Скорнякова А.Ю.; заявитель ПНИПУ-21 с. : ил. – Текст : непосредственный.

93. Тюрин С.Ф., Плотникова А.Ю. Концепция «зеленой» логики / Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления, 2013. – № 8. – С. 61-72.

94. Каменских А. Н., Плотникова А.Ю. Разработка подходов к созданию отказоустойчивой элементной базы для самосинхронной схемотехники / Микроэлектроника и информатика, 2015. тез. – Москва, 2015. – С. 83.

95. Skornyakova A.Y. Statement of the Problem of Finding an Optimal Set of Functionally Complete Tolerant Boolean Functions in the Synthesis of Self-Timed Circuits / В сборнике: Proceedings of the 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, ElConRus 2018, 2018. – P. 244-246. (Scopus, Web of Science)

96. Вихорев Р.В., Прохоров А.С., Скорнякова А.Ю., Тюрин С.Ф. Усовершенствованные методы реализации программируемой логики / В сборнике: Управление большими системами. УБС-2017 материалы XIV

Всероссийской школы-конференции молодых ученых. – Пермь, 2017. – С. 306-315.

97. Вихорев Р.В., Скорнякова А.Ю. Моделирование усовершенствованных устройств программируемой логики / Вестник Пермского университета. Серия: Математика. Механика. Информатика. – Пермь. 2017. – №3(38). – С. 77-81.

98. Рабаи Ж.М., Чандракасан А., Николич Б. Цифровые интегральные схемы. Методология проектирования, 2-е изд.: Пер. с англ. / Рабаи Ж.М., Чандракасан А., Николич Б. – М.: ООО «И.Д.Вильямс», 2007. – 912 с.

99. Тюрин С.Ф., Плотникова А.Ю., Вихорев Р.В. Анализ надёжности логических элементов с избыточным базисом при учёте резервирования входов / Вестник Пермского университета. Серия: Математика. Механика. Информатика, 2014. – № 4 (27). – С. 104-110.

100. Скорнякова А.Ю. Оценка сложности самосинхронных логических элементов FPGA / Вестник Пермского университета. Серия: Математика. Механика. Информатика, 2019. – № 4 (47). – С. 86-89. (РИНЦ)

101. Никитин А.С., Вихорев Р.В., Скорнякова А.Ю. Оптимизация LUT FPGA на основе модифицированного венгерского метода / В сборнике: Управление большими системами. УБС-2017 материалы XIV Всероссийской школы-конференции молодых ученых, 2017. – С. 563-572. (РИНЦ)

102. Тюрин С.Ф., Никитин А.С., Вихорев Р.В., Скорнякова А.Ю. Выбор набора конфигурируемых логических элементов с использованием венгерского метода. Вестник Пермского университета. Серия: Математика. Механика. Информатика. 2017. № 2 (37). С. 65-68. (РИНЦ)

103. Microsoft Excel / URL: https://ru.wikipedia.org/wiki/Microsoft_Excel (дата обращения: 09.09.2020).

104. Иванова К.М., Скорнякова А.Ю. Алгоритм оптимизации комплекта конфигурируемых строго самосинхронных генераторов логических функций для заданных параметров систем функций / Наноиндустрия, 2020. – №S4(99) т.13. – С. 334-336.

**Приложение №1 Листинг программы выбора оптимального набора ССС ЛЭ**

***Класс для вычисления значений функций:***

```
#include "extmath.h"
#include <QDebug>
ExtMath::ExtMath()
{ }
double ExtMath::calcValueFunctionTranzists(int m, int n, int type, int m1, int k)
{    double value = 0.0;
   switch (type)
   {    case 1: value = calcLgfTranzists(m, n); break;
   case 2: value = calcLlutTranzists(m, n); break;
   case 3: value = calcLdclutTranzists(m1, n); break;
   case 4: value = calcDnfTranzists(n, m1, k); break;    }
   return value; }
double ExtMath::calcValueFunctionArea(int m, int n, int type, int m1, int k)
{    double value = 0.0;
   switch (type)
   {    case 1: value = calcLgfArea(n, m); break;
   case 2: value = calcLlutArea(n, m); break;
   case 3: value = calcLdclutArea(n, m1); break;
   case 4: value = calcDnfArea(n, m1, k); break;    }
   return value;}
double ExtMath::calcValueFunctionFast(int m, int n, int type, int m1, int k)
{    double value = 0.0;
   switch (type)
   {    case 1: value = calcLgfFast(n, m); break;
   case 2: value = calcLlutFast(n, m); break;
   case 3: value = calcLdclutFast(n, m1); break;
```

```
    case 4: value = calcDnfFast(n, m1, k); break;    }
    return value;}
double ExtMath::calcValueFunctionPower(int m, int n, int type, int m1, int k)
{   double value = 0.0;
    switch (type)
    {   case 1: value = calcLgfPower(n, m); break;
    case 2: value = calcLlutPower(n, m); break;
    case 3: value = calcLdclutPower(n, m1); break;
    case 4: value = calcDnfPower(n, m1, k); break;    }
    return value;}
double ExtMath::calcLgfTranzists(int m, int n)
{   double value = 0.0;
    value = (double)(m*(16*n+6*n+6*(pow(2, n)-1)+16*(pow(2,n)-
1)+12*(pow(2,n)+n-2)));
    return value;}
double ExtMath::calcLlutTranzists(int m, int n)
{   double value = 0.0;
    value = (double)(m*(16*n+6*(pow(2,n)+n-1)+2*(2*(pow(2,n+1)-
1)+3*(pow(2,n)-1))+2*calcSumPow(2,n)+12*(pow(2,n)+n-2)));
    return value;}
double ExtMath::calcLdclutTranzists(int m, int n)
{   double value = 0.0;
    value = (double)(16*n+6*(pow(2,n+1)+n-2)+2*(2*(pow(2,n)-
1)+2*(calcSumPow(2,n)-2*n)+ 6*calcSumPow(2,n)+ 2*calcSumPow(2,n))+
12*(pow(2,n+1) +n-3)+m*(60*n-38));
    return value;}
double ExtMath::calcDnfTranzists(int n, int m, int k)
{   double value = 0.0;
```

```
    value = (double)(k*(28*n+12*n+6*(4*n-3)+12*(3*n-2)+m*(60*n-38)));
    return value;}
double ExtMath::calcLgfArea(int n, int m)
{    double value = 0.0;
    value = (double)(m*68.03*exp(0.88*n));
    return value;}
double ExtMath::calcLlutArea(int n, int m)
{    double value = 0.0;
    value = (double)(m*42.91*exp(1.08*n));
    return value;}
double ExtMath::calcLdclutArea(int n, int m)
{    double value = 0.0;
    value = (double)(58.4*exp(1.14*n)+m*(168*n-146.3));
    return value;}
double ExtMath::calcDnfArea(int n, int m, int k)
{    double value = 0.0;
    value = (double)(k*(285.06*n-161.1)+m*(168*n-146.3));
    return value;}
double ExtMath::calcLgfFast(int n, int m)
{    double value = 0.0;
    value = (double)(m*(18*n*n+4*n+43));
    return value;}
double ExtMath::calcLlutFast(int n, int m)
{    double value = 0.0;
    value = (double)(m*(2.5*n*n+55.52+31));
    return value;}
double ExtMath::calcLdclutFast(int n, int m)
{    double value = 0.0;
```

```
    value = (double)(27.5*n*n-5.5*n+77+m*(29*n+39));
    return value;}
double ExtMath::calcDnfFast(int n, int m, int k)
{    double value = 0.0;
    value = (double)((0.7*n*n+39*n+54.3)*k+m*(29*n+39));
    return value;}
double ExtMath::calcLgfPower(int n, int m)
{    double value = 0.0;
    value = (double)(m*(41.15*n*n-78.56*n+60.75));
    return value;}
double ExtMath::calcLlutPower(int n, int m)
{    double value = 0.0;
    value = (double)(m*(21.32*n*n+0.4*n+13.92));
    return value;}
double ExtMath::calcLdclutPower(int n, int m)
{    double value = 0.0;
    value = (double)(19.88*n*n+54.63*n-17.75+m*(40.31*n-32.24));
    return value;}
double ExtMath::calcDnfPower(int n, int m, int k)
{    double value = 0.0;
    value = (double)(k*(8.45*n*n+17.35*n+26.35)+m*(40.31*n-32.24));
    return value;}
double ExtMath::calcDnfSFTranzists(int k, int m, int n)
{    double del = static_cast<double>(floor(n)/floor(k));
    double up = calcUpElementSum(n, k);
    double N = floor(static_cast<double>(floor(up/2)));
    double sum = 0.0;
    for (int i = 0; i < N; i++)
```

```
        sum += pow(2, floor(up)-floor(i)*floor(2));
    double value = 2*(floor(k)*(20*floor(n)+2*ceil(static_cast<double>(n/2)))
     +6*floor(m)*(floor(k)+2)+4*floor(n))+12*(sum + ceil(del) - floor(del));
    return value;}
double ExtMath::calcSumPow(int val, int n)
{   double sum = 0.0;
    for (int i = 0; i < n; i++)
        sum += pow(val, i+1);
    return sum;}
double ExtMath::calcUpElementSum(int n, int k)
{   double del = static_cast<double>(floor(n)/floor(k));
    double N = floor(del);
    double sum = 0.0;
    for (int i = 0; i < N; i++)
        sum += pow(2, floor(n)-floor(i)*floor(k));
    return floor(sum + ceil(del)-floor(del));}
```

***Класс алгоритма поиска Парето-оптимальных решений:***

```
#include "fileprocessor.h"
#include <QDateTime>
#include <QApplication>
#include <QDir>
#include <QDebug>
#include <QTextCodec>
#include "graphicplot.h"
#define TO_INT   10000
const QString nameType[] = {"Транзисторы, шт", "Площадь, мкм²",
"Быстродействие, пс", "Мощность, мкВт"};
```

```
const QString nameFun[] = {"ГФ", "LUT-ST", "DC LUT-ST", "ДНФ LUT-ST"};
const QColor  colorType[] = {Qt::red, Qt::blue, Qt::green, Qt::yellow};
//Функция сравнения элементов листа
template <typename T> bool compare(const T& a,  T& b)
{    return a.length() < b.length();}
template <typename T> bool compareSum(const T& a,  T& b)
{    return a.M < b.M;}
template <typename T> bool compareVal(const T& a,  T& b)
{    return a.value.toFloat() < b.value.toFloat();}
FileProcessor::FileProcessor(QObject *parent) : QObject(parent)
{    m_values.clear();
   m_listGraph.clear();
   m_count = 0;
   m_sum = 0;
   m_m1 = 0;
   m_k = 0;
   m_set = "";}
void FileProcessor::process()
{    m_listGraph.clear();
   if(m_values.size() == 0)
      return;
   m_xlsx = new XlsxFile;
   m_xlsx->setHeaderTextTable(m_values, m_m1, m_k);
   //Вычисление наборов ЛЭ
   QStringList listRow = calcRowTable(calcBolianSet());
   for(int i = 0; i < listRow.size(); i++)
      m_listGraph.append(new QVector3D());
```

```
m_xlsx->setListRow(listRow);
//Тут будут записываться результирующие наборы
QList<Results*> resultsList;
for(int i = 0; i < NUMBER_PARAM; i++)
{       Results* result = new Results;
   result->reset();
   result->naborsList = saveFile(i, listRow, &result->table);
   resultsList.append(result);   }
//После того как записали все найденные значение таблицы сложности
//нужно найти минимальные покрытия без nan в строке
calcResultsString(&resultsList, listRow);
//Провека, все ли покрытия найдены
bool isGood = false;
for(int i = 0; i < resultsList.size(); i++)
   if(resultsList.at(i)->nabor.isEmpty())
      isGood = true;
if(isGood)
{       emit signalsShowMessage("Не найдено ни одного покрытия, задайте
другие данные!");
   signalsIsReady(File::Fail);
   delete m_xlsx;
   return;
} else {       //Сохранение результатов
   saveResuls(resultsList);
   //Чтобы при открытии файла, была выбрана первая страница
   if(m_xlsx->saveFile())
      emit signalsShowMessage("Файл успешно сохранен:\n" + m_xlsx-
>getNameFile());
```

```
        signalsIsReady(File::Success);
        delete m_xlsx;    }}
    QStringList FileProcessor::saveFile(int typeFun, QStringList listRow, QStringList* table)
    {    m_listUsed.clear();
      QStringList naborTable;  //таблица наборов которые используются по факту
        naborTable.clear();
        //Заполним списки
        for(int i = 0; i < listRow.size(); i++)
        {       VariantFunc* variant = new VariantFunc;
          variant->reset();
          m_listUsed.append(variant);  //лист для проверки достаточности данных для вычисления значений
          table->append("");
          naborTable.append("");    }
        //Вычисление всех значений в таблицу
        for(int i = 0; i < m_values.size(); i++)
        {       //Вычисление столбца данных
          QStringList list = calcColumn(listRow, m_values.at(i), &naborTable, typeFun, *table);
          m_xlsx->setData(list, 2+typeFun+i*NUMBER_PARAM, 0);
          for(int k = 0; k < list.size(); k++)
          {           QString str = table->at(k) + list.at(k) + ";";
            table->replace(k, str);  //заполнение таблицы       }    }
        calcAllCompexity(typeFun, *table); // для построение графика
        return naborTable;}
    QStringList FileProcessor::calcBolianSet()
```

```
{   //получение множества Булиана
   QStringList outSet;
   int count = m_count; //количество элементов
   double initSet[count]; //элементы, которые будут использоваться в множестве
   QStringList list = m_set.split(",", QString::SkipEmptyParts);
   for(int i = 0; i < count; i++)
      initSet[i] = list.at(i).toInt();
   //Используется бинарный метод нахождения
   int N = pow(2, count);
   for(int i = 1; i < N; i++)
   {       QString str = "";
      for(int j = 0; j < count; j++)
         if( i & (1 << j))
            str += QString::number(initSet[j]) + " ";
      outSet.append(str);    }
   //Сортировка значений
   qSort(outSet.begin(), outSet.end(), compare<QString>);
   return outSet;}
QStringList FileProcessor::calcRowTable(QStringList boolian)
{   //Вычисление наборов ЛЭ
   QStringList listRow;
   listRow.clear();
   for(int i = 0; i < boolian.size(); i++)
   {       QStringList countList;
      countList.clear();
      //Элементы которые будут повторятся
      QStringList list = boolian.at(i).split(" ", QString::SkipEmptyParts);
```

Продолжение приложения №1

```
        //тут надо получим все комбинации при которых сумма элементов матрицы будет = 8
        int M = list.size(); //Количество элементов
        int N = m_sum - M + 1; //Какие элементы будут повторятся
        int *mass = new int[M];
        for(int k = 0; k < M; k++)
           mass[k] = 1;
        while(nextRow(mass, N, M))
        {  int sum = 0;
           QString strSum = "";
           for(int k = 0; k < M; k++)
           {              sum += mass[k];
              strSum += QString::number(mass[k]) + " ";          }
           if(sum == m_sum)
              countList.append(strSum);       }
        delete[] mass;
        //Получение строк таблицы
        for(int k = 0; k < countList.size(); k++)
        {          //Сколько раз будут повторятся элементы
           QStringList cList = countList.at(k).split(" ", QString::SkipEmptyParts);
           //причем list.size() == cList.size()
           QString str = "";
           for(int j = 0; j < list.size(); j++)
              str += cList.at(j) + "*" + list.at(j) + " ";
           str = str.left(str.length()-1);
           listRow.append(str); }}
     return listRow;}
  bool FileProcessor::nextRow(int* mass, int n, int m, int val)
```

```
{    //получение всевозможных комбинаций из чисел N, длины M
    int j = m - 1;
    while((j >=0) && (mass[j] == n))
        j--;
    if(j < 0)
        return false;
    if(mass[j] >= n)
        j--;
    mass[j]++;
    if(j == m - 1)
        return true;
    for(int k = j + 1; k < m; k++)
        mass[k] = val;
    return true;}
void FileProcessor::setValues(QList <Values> values)
{    QMutexLocker locker(&m_mutex);
    m_values.clear();
    //проверка на нулевые значения
    bool isNull = false;
    for(int i = 0; i < values.size(); i++)
        if(values.at(i).y == 0)
            isNull = true;
    if(isNull)
    {        emit signalsShowMessage("Неверно заданы параметры!");
        emit signalsIsReady(File::Fail);
        return;    }
 else {        m_values = values;
        process();}}
```

```
QStringList FileProcessor::calcColumn(QStringList listRow, Values value, QStringList* nabors, int typeFun, QStringList table)
{   QStringList outList;
    outList.clear();
    //value.y - столько функций нужно для набора
    for(int k = 0; k < listRow.size(); k++)
    {       //Определим сколько функций есть для текущего набора ЛЭ
        QStringList list = listRow.at(k).split(" ", QString::SkipEmptyParts); //лист наборов
        int fact[NUMBER_FUN]; //фактические количество функций, сколько их есть в наборе ЛЭ для каждого типа функций, даже если он не выбран
        for(int i = 0; i < NUMBER_FUN; i++)
            fact[i] = 0;
        for(int i = 0; i < list.size(); i++)
        {               int type = list.at(i).split("*").last().toInt()-1;//Типа варианта функции 0-ГФ, 1-ЛУТ, 2-ДС ЛУТ и 3-ДНФ
            int count = list.at(i).split("*").first().toInt();//Количество систем
            fact[type] = count - m_listUsed.at(k)->fact[type]; }
        for(int i = 0; i < 2; i++) // у ГФ и LUT нужно уменьшить количество элементов
            fact[i] = (int) (fact[i]/value.m);
        //теперь нужно найти все варианты удовлетворяющие, то что нужно, из того что есть
        QList<CalcNabor> values; //список: вариант + значение функции, потом будем находить минимальное значение
        //Найдем максимальное значение в фактических данных
        int N = 0; //Какие элементы будут повторятся
        int sum = 0; //Проверка хватает функций или нет
```

```
for(int i = 0; i < NUMBER_FUN; i++)
{           if(fact[i] > N)
      N = fact[i];
   sum += fact[i];  }
//!Ищем все значения функций и комбинации
if(sum >= value.y)
{ int* mass = new int[NUMBER_FUN]; //набор
   for(int i = 0; i < NUMBER_FUN; i++)
      mass[i] = 0;
   while(nextRow(mass, N, NUMBER_FUN, 0))
   {  int sumY = 0;
      int count = 0; //счетчик проверки подходит набор или нет
      for(int j = 0; j < NUMBER_FUN; j++)
      {    sumY += mass[j];
         if(fact[j] >= mass[j])
            count ++;     }
      if((sumY == value.y) && (count == NUMBER_FUN))
      {//условие что набор подходит
         QString nabor = "";
         double znach = 0.0;
         //получим набор значений, вычислим значение и запишем в
строку
         for(int i = 0; i < NUMBER_FUN; i++)
         {  if(mass[i] != 0)
            {  //можитель для вычисления значения функций
               int param = mass[i];
               if(i < 2)
                  param *= value.m;
```

Продолжение приложения №1

```
nabor += QString("%1*%2").arg(param).arg(i+1) + " "; //полученный набор
int coef = ceil(value.m/m_m1); //!условие на m1 (пока тут: m1<m, то нужно добавить чтобы покрывало m)
switch (typeFun) { //вычисление значения функции
case TRANZISTS:
znach += param*ExtMath::calcValueFunctionTranzists(value.m, value.n, i+1, coef*m_m1, m_k);
break;
case AREAS:
znach += param*ExtMath::calcValueFunctionArea(value.m, value.n, i+1, coef*m_m1, m_k);
break;
case FAST:
znach += param*ExtMath::calcValueFunctionFast(value.m, value.n, i+1, coef*m_m1, m_k);
break;
case POWER:
znach += param*ExtMath::calcValueFunctionPower(value.m, value.n, i+1, coef*m_m1, m_k);
break; } } }
nabor = nabor.left(nabor.length()-1); //убрать пробел в конце
QString s = QString::number(znach);
values.append(CalcNabor(nabor, QString::number(znach, 'g', s.length())));}}
delete[] mass;}
if((!values.isEmpty()) && (!table.at(k).contains(STR_NAN, Qt::CaseInsensitive)))
{//Найдем набор при котором значение функции минимально и запишем в лист
```

```
qSort(values.begin(), values.end(), compareVal<CalcNabor>);
outList.append(values.first().value);
QString str = nabors->at(k) + values.first().nabor + ";";
nabors->replace(k, str); //запись набора в лист
//Учитываем сколько мы функций и систем уже использовали, и учитываем это в m_listUsed
QStringList outNabor = values.first().nabor.split(" ", QString::SkipEmptyParts);
for(int i = 0; i < outNabor.size(); i++)
{QStringList vars = outNabor.at(i).split("*");
  int fun = vars.first().toInt(); //количество функций
 int type = vars.last().toInt()-1;//тип функции
m_listUsed.at(k)->fact[type] += fun;}} else
  {//Если значение не удалось вычислить то NAN
 outList.append(STR_NAN);
QString strN = nabors->at(k) + "nan;";
nabors->replace(k, strN); }}
return outList;}
void FileProcessor::setCount(QString set)
{    QString elements = set;
    elements.replace("{", "");
    elements.replace("}", "");
    elements.replace("ГФ", "1");
    elements.replace("DC LUT-ST", "3");
    elements.replace("ДНФ LUT-ST", "4");
    elements.replace("LUT-ST", "2");
    elements.replace(" ", "");
    m_set = elements;  //Получаем элементы в виде: 2,3,4
    m_count = m_set.split(",", QString::SkipEmptyParts).size();}
```

Продолжение приложения №1

```
void FileProcessor::setSum(int sum)
{m_sum = sum;}
void FileProcessor::setM1(int m1)
{m_m1 = m1;}
void FileProcessor::setK(int k)
{m_k = k;}
int FileProcessor::getIndexRowMinSum(QStringList list, double* complexity,
double* maxCompl, double* averCompl)
{ int row = -1;
   double min = 0.0;
   double max = 0.0;
   double aver = 0.0;
   //Сначала находим хоть какой то элемент из столбца не равный nan
   for(int i = 0; i < list.size(); i++)
      if(!list.at(i).contains(STR_NAN, Qt::CaseInsensitive))
      {min = list.at(i).toDouble();
  row = i;
break; }
   //потом уже ищем минимальный, т.к. заданный min = 0 может оказаться
минимальным
   int count  = 0;
   for(int i = 0; i < list.size(); i++)
   {if(!list.at(i).contains(STR_NAN, Qt::CaseInsensitive))
      {aver += list.at(i).toDouble();
         count ++;
         if(list.at(i).toDouble() < min)
         { min = list.at(i).toDouble();
            row = i; } else if(list.at(i).toDouble() > max)
```

```
                max = list.at(i).toDouble();}}
        *complexity += (double) min;
        *maxCompl += (double) max;
        *averCompl += (double)(aver/count);
         return  row; }
        int FileProcessor::getIndexRowMaxSum(double* complexity, QStringList table,
QList <int> noNanTable,  double *maxCompl, double *averCompl)
        {//Тут поиск минимального значения из максимальных
           QStringList maxList;
           maxList.clear();
           //Сначала найдем все максимальные значения из всех столбцов
           int row = -1;
           for(int i = 0; i < table.size(); i++)
           {double max = 0.0;
              if(noNanTable.at(i) >= 0)
              { QStringList list = table.at(i).split(";", QString::SkipEmptyParts);
                 for(int k = 0; k < list.size(); k++)
                 { double value = list.at(k).toDouble();
                    if(value > max)
        max = value; }}
          if(max*TO_INT != 0)
        maxList.append(QString::number(max));
         else
        maxList.append(STR_NAN);}
           double max = 0.0, aver = 0.0, min = 0.0;
           //Найдем индекс строки с минимальным значением из максимальных
           row = getIndexRowMinSum(maxList, &min, &max, &aver);
           //Если все окей, то считаем сложность
```

```
if(row >= 0)
      { min = max = aver = 0.0;
        //Сначала находим хоть какой то элемент из столбца не равный nan
        for(int i = 0; i < maxList.size(); i++)
           if(!maxList.at(i).contains(STR_NAN, Qt::CaseInsensitive))
           {min = maxList.at(i).toDouble();
              break;}
        int count = 0;
        for(int i = 0; i < maxList.size(); i++)
        { if(!maxList.at(i).contains(STR_NAN, Qt::CaseInsensitive))
           {aver += maxList.at(i).toDouble();
   count ++;
    if(max < maxList.at(i).toDouble())
   max = maxList.at(i).toDouble();
   if(min > maxList.at(i).toDouble())
   min = maxList.at(i).toDouble();}}
   *complexity = (double) min;
   *maxCompl = (double) max;
   *averCompl = (double) aver/count;}
      return  row;}
   void FileProcessor::saveResuls(QList <Results*> resultList)
   {    QList <QVector3D*> vectors;
      vectors.clear();
      QStringList nabors;
      nabors.clear();
     //сохранение результатов в файл
      m_xlsx->setResultsData(resultList, &vectors, &nabors, m_values.size());
    //отображение графика
```

```
    GraphicPlot* plot = new GraphicPlot;
    for(int i = 0; i < vectors.size(); i++)
    {       QList <QVector3D*> list;
      list.append(vectors.at(i));
      plot->addData(list, i, nabors.at(i));
      for(int k = 0; k < m_listGraph.size(); k++)
        if(vectors.at(i) == m_listGraph.at(k))
          m_listGraph.removeAt(k);}
    plot->addData(m_listGraph, vectors.size(), "Значения таблицы сложности",
true);
    plot->show();}
  QXlsx::Format FileProcessor::getFormatXslx(QColor color, bool isBold)
  {   QXlsx::Format format;
    format.setHorizontalAlignment(QXlsx::Format::AlignHCenter);
    format.setVerticalAlignment(QXlsx::Format::AlignVCenter);
    format.setBorderStyle(QXlsx::Format::BorderThin);
    format.setPatternBackgroundColor(color);
    format.setFontBold(isBold);
    return format;}
  void FileProcessor::calcAllCompexity(int type, QStringList table)
  {for(int i = 0; i < table.size(); i++)
    { float complexity = 0.0;
      if(!table.at(i).contains(STR_NAN, Qt::CaseInsensitive))
      {QStringList list = table.at(i).split(";", QString::SkipEmptyParts);
        if(type != FAST)
          for(int k = 0; k < list.size(); k++)
            complexity += list.at(k).toFloat();
        else
```

```
{ for(int k = 0; k < list.size(); k++)
      if(complexity < list.at(k).toFloat())
         complexity = list.at(k).toFloat();}
if(complexity > 0.0)
{switch (type) {case AREAS:
      m_listGraph.at(i)->setX(complexity);
      break;
   case FAST:
      m_listGraph.at(i)->setY(complexity);
      break;
   case POWER:
      m_listGraph.at(i)->setZ(complexity);
      break;} } } }}
```

```
void FileProcessor::calcResultsString(QList<Results *> *resultList, QStringList listRow)
{QList <int> indexNoNan = m_xlsx->calcTableNoNanString(resultList->first()->table.size(), m_values.size());
   for(int i = 0; i < resultList->size(); i++)
   { //Поиск минимальных покрытий
      QList <int> indexsRow;
      indexsRow.clear();
      double complexity = 0.0; //Сложность
      double averCompl = 0.0, maxCompl = 0.0;
      for(int j = 0; j < m_values.size(); j++)
      {QStringList list; //столбец
         for(int k = 0; k < resultList->at(i)->table.size(); k++)
         { if(indexNoNan.at(k) >= 0)
```

Продолжение приложения №1

```
                list.append(resultList->at(i)->table.at(k).split(";",
QString::SkipEmptyParts).at(j));  //получение столбца данных из общей таблицы
        else    list.append(STR_NAN); }
                int row = -1;
                if(i != FAST)
                   row = getIndexRowMinSum(list, &complexity, &maxCompl,
&averCompl);  //получение индекса строки при минимальном значении
                else
                   row = getIndexRowMaxSum(&complexity, resultList->at(i)->table,
indexNoNan, &maxCompl, &averCompl);          //получение индекса строки для
быстродействия
                if(row >= 0)
      indexsRow.append(row);  //сохранение индексов минимальных значений}
           //если есть полное покрытие
           if(indexsRow.size() == m_values.size())
           { //Найдем оптимальный набор
              int mass[NUMBER_FUN];
              for(int j = 0; j < NUMBER_FUN; j++)
                 mass[j] = 0;
              for(int j = 0; j < indexsRow.size(); j++)
      {QString varNabor = resultList->at(i)->naborsList.at(indexsRow.at(j));
       if(!varNabor.isEmpty())
      {//Найдем набор ЛЭ соответствующий минимальному элементу
                    QString str = varNabor.split(";", QString::SkipEmptyParts).at(j);
                    QStringList list = str.split(" ", QString::SkipEmptyParts);
                    for(int k = 0; k < list.size(); k++)
                    {QStringList para = list.at(k).split("*", QString::SkipEmptyParts);
                       int type = para.last().toInt()-1;
```

Продолжение приложения №1

```
                int fun = para.first().toInt();
                mass[type] += fun; } }}
        //Проверим есть ли набор, для которого найденный набор mass является поднабором
        QString nabor = "";
        int rowTable = -1;
        for(int j = 0; j < listRow.size(); j++)
    {QStringList list = listRow.at(j).split(" ", QString::SkipEmptyParts); //пары
            int arr[NUMBER_FUN]; //набор из общих наборов
            for(int k = 0; k < NUMBER_FUN; k++)
               arr[k] = 0;
            if(indexNoNan.at(j) >= 0)
            {for(int k = 0; k < list.size(); k++)
    {QStringList para = list.at(k).split("*", QString::SkipEmptyParts);
                arr[para.last().toInt()-1] = para.first().toInt();}
              bool ok = true;
              for(int k = 0; k < NUMBER_FUN; k++)
                 if(mass[k] > arr[k])
                    ok = false;
              if(ok)
              {if(!resultList->at(i)->table.at(j).contains(STR_NAN, Qt::CaseInsensitive))
    { double c = 0.0;
    QStringList complList = resultList->at(i)->table.at(j).split(";", QString::SkipEmptyParts);
                    if(i != FAST)
                       for(int k = 0; k < complList.size(); k++)
                          c += complList.at(k).toFloat();
```

```
                else
                {for(int k = 0; k < complList.size(); k++)
                      if(c < complList.at(k).toDouble())
                         c = complList.at(k).toDouble();}
                   if(QString::number(complexity) == QString::number(c))
                   {  rowTable = j;
                      break; } } }}}
for(int j = 0; j < NUMBER_FUN; j++)
   if(mass[j] != 0)
      nabor += QString("(%1*%2) ").arg(mass[j]).arg(nameFun[j]);
nabor = nabor.left(nabor.length()-1);
if(rowTable >= 0)
{indexsRow.clear();
   for(int k = 0; k < m_values.size(); k++)
      indexsRow.append(rowTable); }
m_xlsx->setStringColorTable(colorType[i], indexsRow, i);
//Запись результатов
resultList->at(i)->nabor = nabor;
resultList->at(i)->complexity = complexity;
resultList->at(i)->averCompl = averCompl;
resultList->at(i)->maxCompl = maxCompl;
resultList->at(i)->rowTable = rowTable+2; }}}
```

## Приложение №2 Акты внедрения

УТВЕРЖДАЮ
Директор ФИЦ ИУ РАН
_____ И.А. Соколов
«28» 07 2020 г.

**Акт о внедрении**
результатов диссертационных исследований
*Скорняковой Александры Юрьевны*

Настоящим актом подтверждается, что в научно-исследовательской работе Института проблем информатики «Федерального исследовательского центра» «Информатика и управление» Российской академии наук: использовались следующие научные результаты, полученные в кандидатской диссертации аспиранта кафедры «Автоматика и телемеханика» ФГБОУ ВО Пермского национального исследовательского политехнического университета Скорняковой Александры Юрьевны: раздел 5 части №3 отчета (направления) за 2017 г.: «Концептуальные и методологические основы создания семейства потоковых самосинхронных процессоров и средств поддержки их проектирования» (№ госрегистрации 117030650038):

1. Метод реализации конфигурируемого самосинхронного генератора логических функций на основе стандартных логических КМОП элементов;
2. Метод реализации конфигурируемого самосинхронного генератора логических функций на основе Look Up Table (LUT), используемого в ПЛИС;
3. Метод реализации конфигурируемого самосинхронного генератора систем логических функций, заданных в СДНФ, на основе дешифратора DC LUT и блоков дизъюнкций;
4. Метод реализации конфигурируемого самосинхронного генератора систем логических функций, заданных в ДНФ, на основе блоков конъюнкций и дизъюнкций;
5. Оценки сложности реализации систем логических функций на основе разработанных методов;
6. Алгоритм выбора оптимального набора конфигурируемых логических элементов для реализации типовых систем логических функций.

Разработанные методы используются при проектировании самосинхронных программируемых логических устройств на основе отечественной элементной базы, при этом обеспечивается нахождение Парето-оптимальных решений по сложности в количестве транзисторов, площади, занимаемой на кристалле, быстродействию и потребляемой мощности.

Руководитель отд. 52 Ю.А. Степченков

Продолжение приложения №2

**УТВЕРЖДАЮ**

Проректор по учебной работе ПНИПУ

_____ / Лобов Н.В. /

«___» _____ 2020 г.

**АКТ**

внедрения в учебный процесс кафедры **«Автоматика и телемеханика» ФГБОУ ВО ПНИПУ** результатов диссертационной работы **Скорняковой Александры Юрьевны** на тему «Конфигурируемые логические элементы для самосинхронных схем»

Комиссия в составе:

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

Члены комиссии: Хижняков Ю. Н. – д.т.н., проф. кафедры «Автоматика и телемеханика»

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

составили настоящий акт о том, что результаты диссертационного исследования Скорняковой Александры Юрьевны 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» и «Цифровая схемотехника» для бакалавриата направления 27.03.04 «Управление в технических системах», дисциплин «Электроника» и «Цифровая схемотехника» для бакалавриата направления 11.03.02 «Инфокоммуникационные технологии и системы связи», а также по спец. предметам аспирантуры направления подготовки 09.06.01 «Информатика и вычислительная техника», научной специальности 05.13.05 «Элементы и устройства вычислительной техники и систем управления».

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

1. Лабораторная работа №1 «Исследование метода реализации конфигурируемого самосинхронного генератора логических функций на основе стандартных логических КМОП элементов»;
2. Лабораторная работа №2 «Исследование метода реализации конфигурируемого самосинхронного генератора логических функций на основе LookUpTable (LUT), используемого в ПЛИС»;
3. Лабораторная работа №3 «Исследование метода реализации конфигурируемого самосинхронного генератора систем логических функций, заданных в СДНФ, на основе дешифратора DC LUT и блоков дизъюнкций»;

Окончание приложения №2

4. Лабораторная работа №4 «Исследование метода реализации конфигурируемого самосинхронного генератора систем логических функций, заданных в ДНФ, на основе блоков конъюнкций и дизъюнкций»;
5. Практическое занятие №1 «Анализ оценок сложности реализации систем логических функций на основе исследованных методов»;
6. Практическое занятие №2«Получение оптимальных наборов с помощью алгоритма выбора оптимального набора конфигурируемых логических элементов для реализации типовых систем логических функций».

Эффект от внедрения результатов диссертационной работы заключается в повышении уровня компетенций в соответствии со стандартом ФГОС ВО 3++ по направлениям 27.03.04 «Управление в технических системах», 11.03.02 «Инфокоммуникационные технологии и системы связи» и 09.06.01 «Информатика и вычислительная техника», научной специальности 05.13.05 «Элементы и устройства вычислительной техники и систем управления».

Председатель:

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

Члены комиссии:

д.т.н., проф. кафедры АТ _____ / Хижняков Ю.Н./

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

**Приложение №3 Анализ полумодулярности в САПР Forcage (подсистема TRANAL)**

Проверка схем на полумодулярность.

Схема рисунок 2.1

1. Состояние SRAM 00

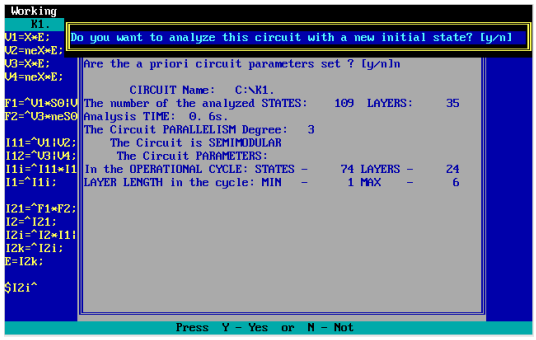



Рисунок П3.1 – Результат анализа генератора одной переменной (SRAM 00) в САПР Forcage (подсистема TRANAL)

2. Состояние SRAM 01

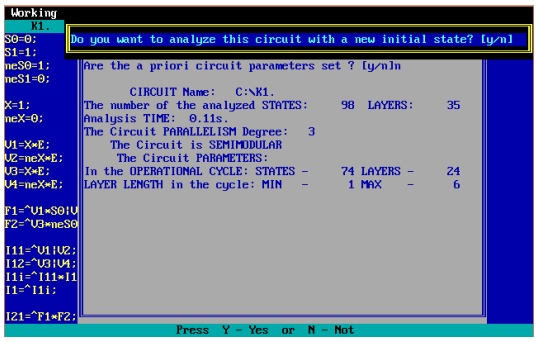



Рисунок П3.2 – Результат анализа генератора одной переменной (SRAM 01) в САПР Forcage (подсистема TRANAL)

3. Состояние SRAM 11

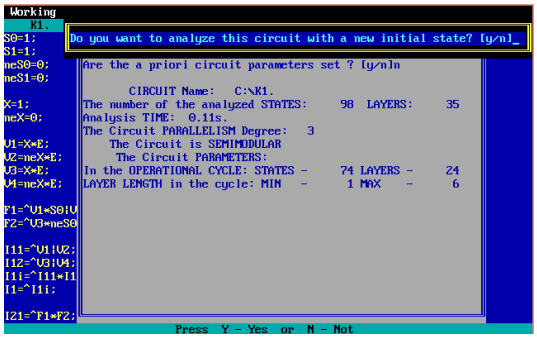



Рисунок П3. 3 – Результат анализа генератора одной переменной (SRAM 11) в САПР Forcage (подсистема TRANAL)

Продолжение приложения №3

Выполним анализ генератора функций двух переменных, показанного на рисунке 2.2., проверку схемы сведем в таблицу П3.1.

Таблица П3.1 ( Начало)

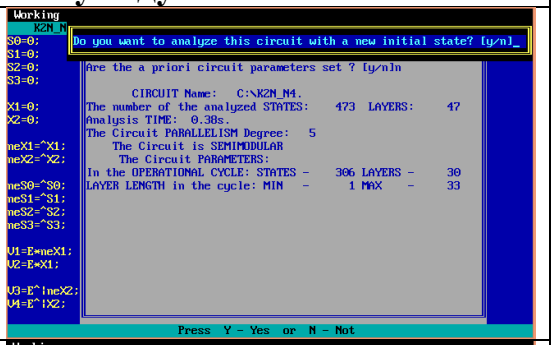
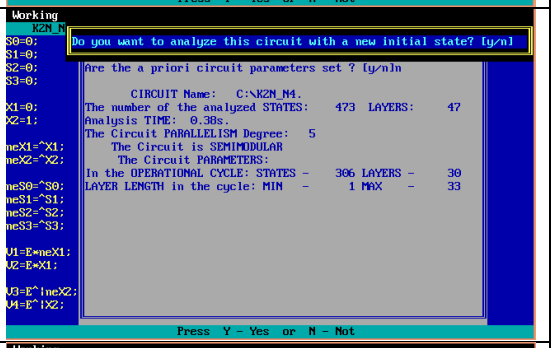
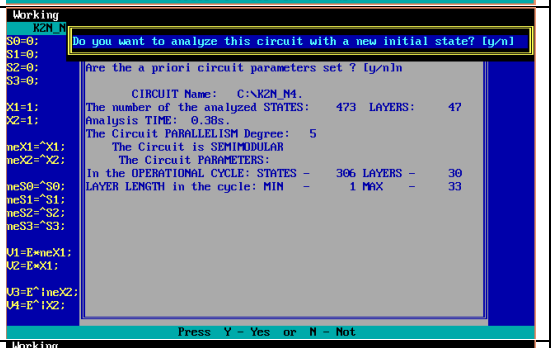
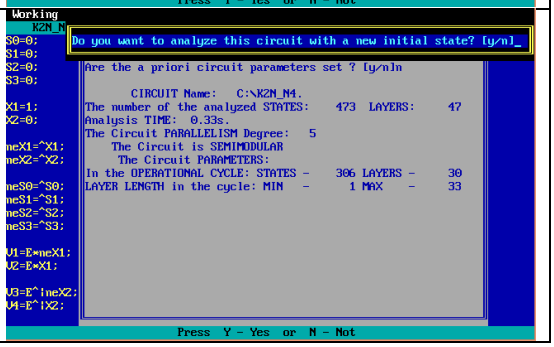
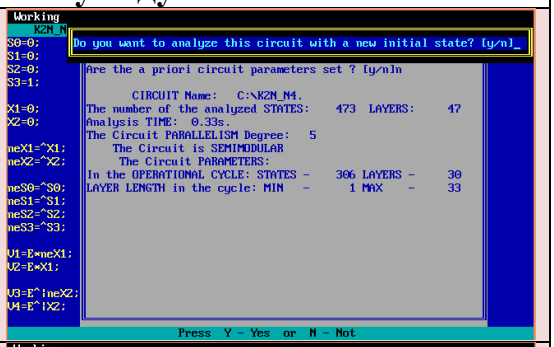
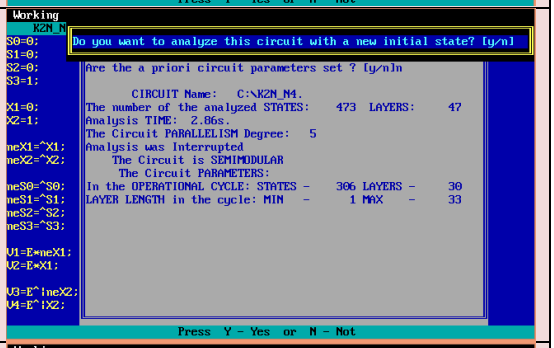
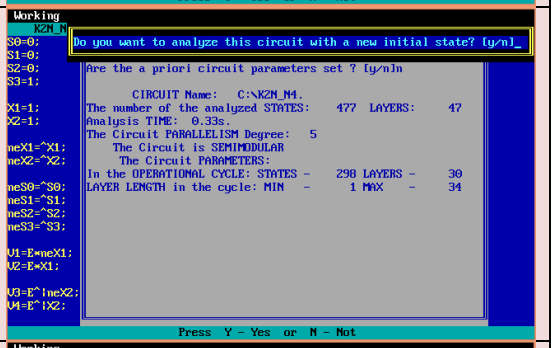
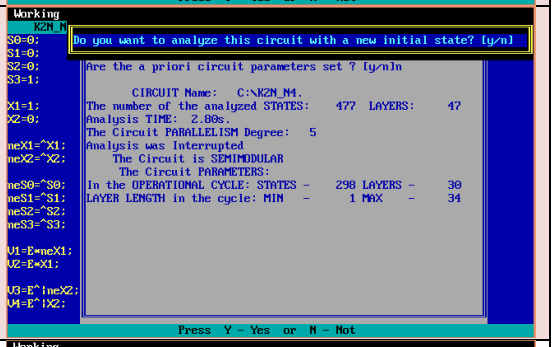
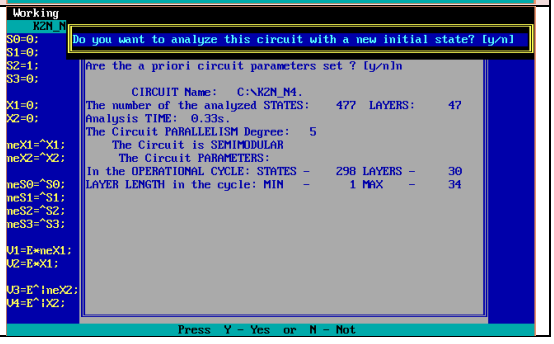
| № | S0S1S2S3 | X1X2 | Полумодулятность |
|---|---|---|---|
| 1 | 0000 | 00 |  |
| | | 01 |  |
| | | 11 |  |
| | | 10 |  |

Таблица П3.1 (Продолжение)

| № | S0S1S2S3 | X1X2 | Полумодулятность |
|---|---|---|---|
| 2 | 0001 | 00 |  |
| | | 01 |  |
| | | 11 |  |
| | | 10 |  |
| 3 | 0010 | 00 |  |

Продолжение приложения №3

Таблица П3.1 (Продолжение)

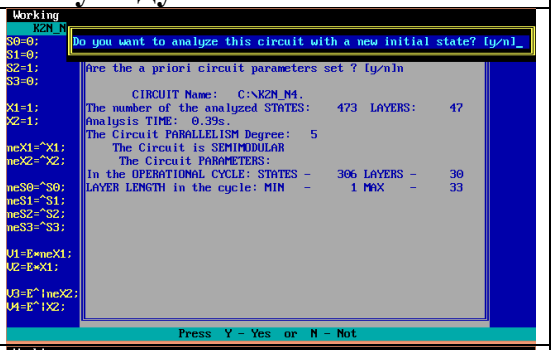
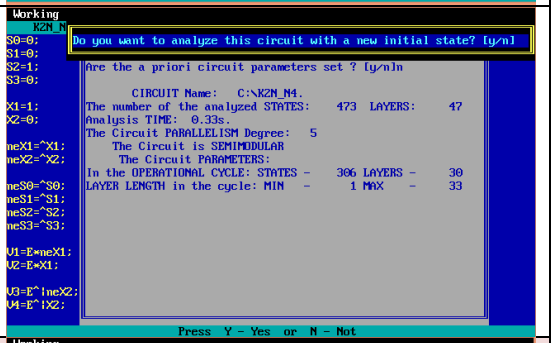
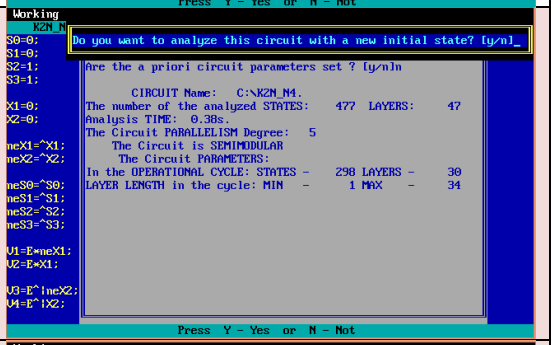
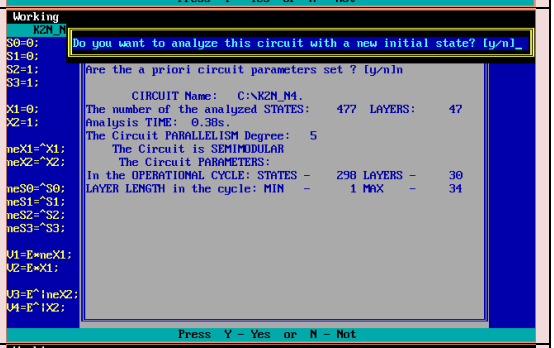
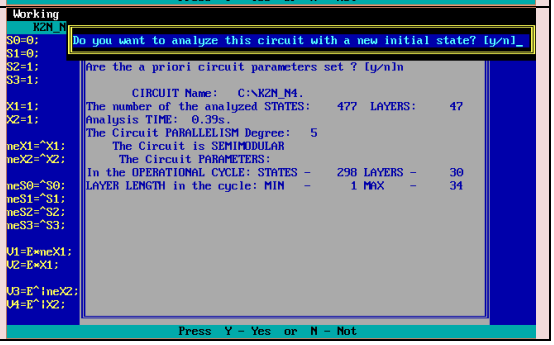
| № | S0S1S2S3 | X1X2 | Полумодулятность |
|---|---|---|---|
| 3 | 0010 | 11 |  |
| | | 10 |  |
| 4 | 0011 | 00 |  |
| | | 01 |  |
| | | 11 |  |

Таблица П3.1 (Продолжение)

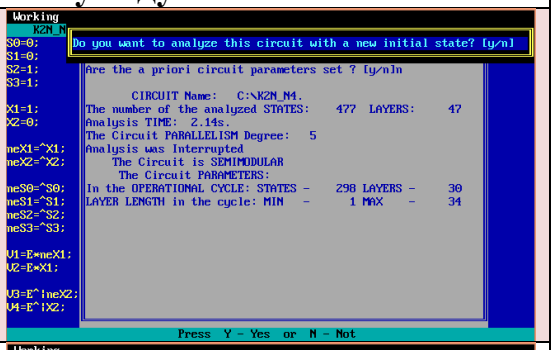
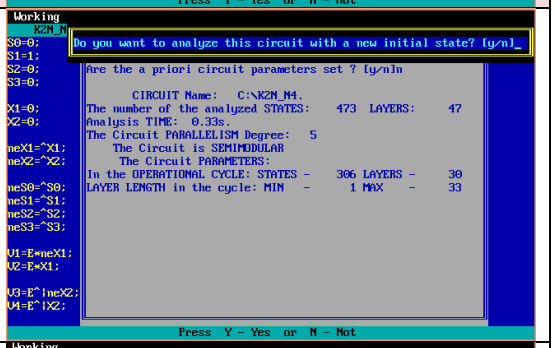
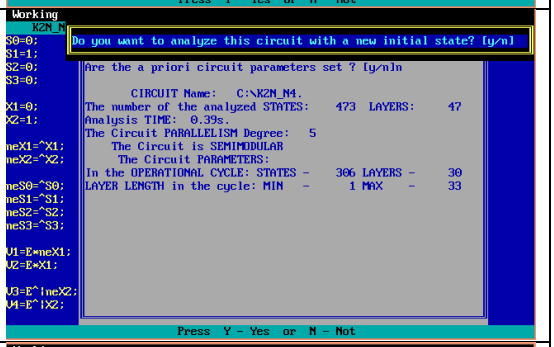
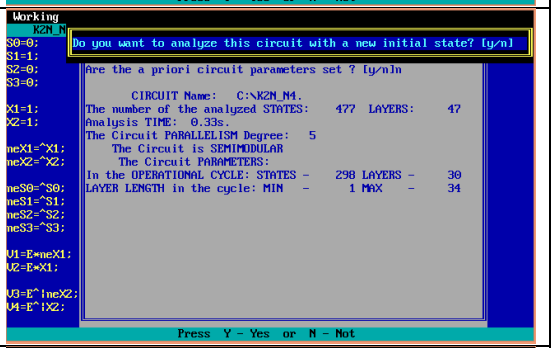
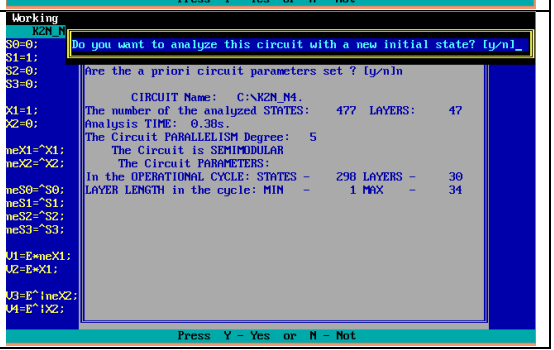
| № | S0S1S2S3 | X1X2 | Полумодулятность |
|---|---|---|---|
| 4 | 0011 | 10 |  |
| 5 | 0100 | 00 |  |
| | | 01 |  |
| | | 11 |  |
| | | 10 |  |

Таблица П3.1 (Продолжение)

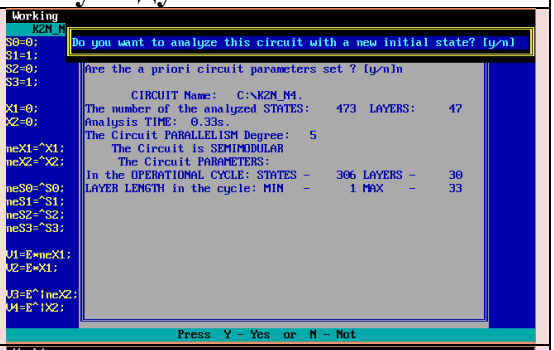
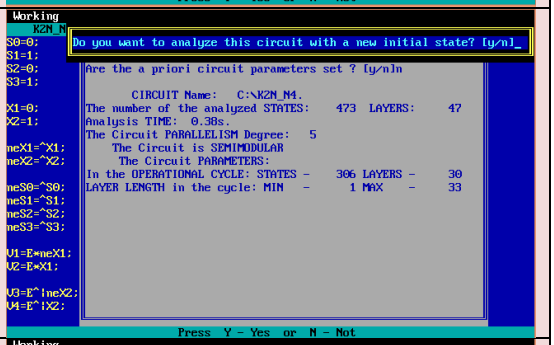
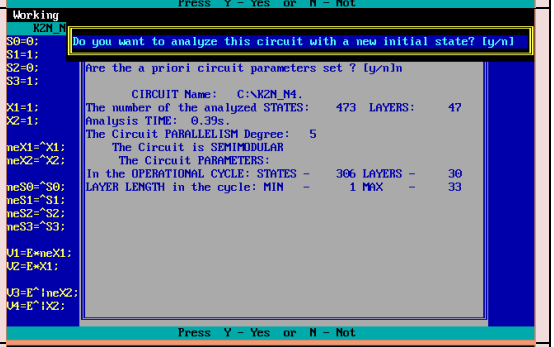
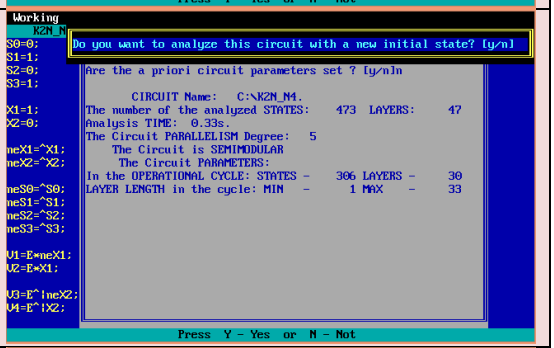
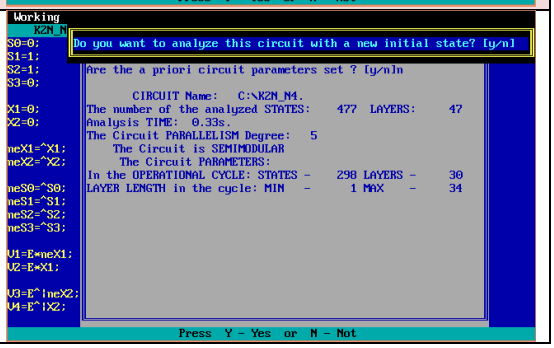
| № | S0S1S2S3 | X1X2 | Полумодулятность |
|---|---|---|---|
| 6 | 0101 | 00 |  |
| | | 01 |  |
| | | 11 |  |
| | | 10 |  |
| 7 | 0110 | 00 |  |

Таблица П3.1 (Продолжение)

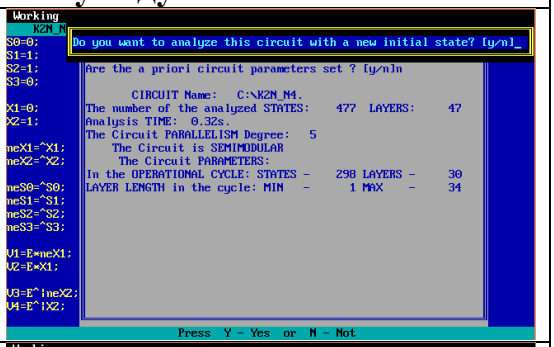
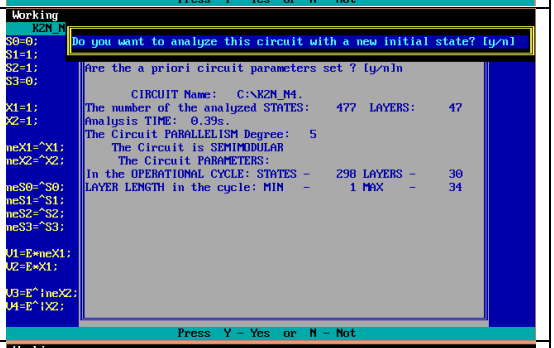
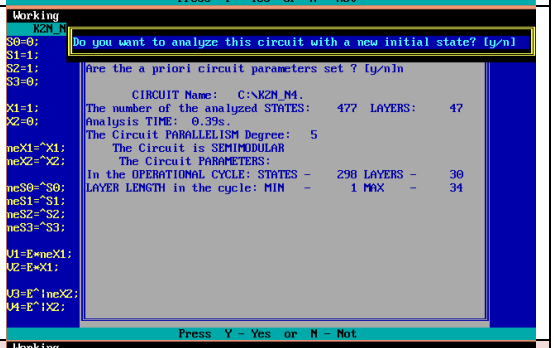
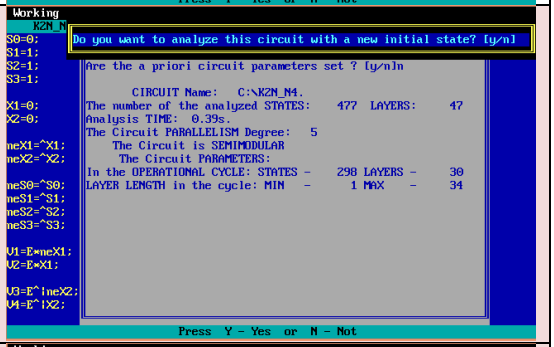
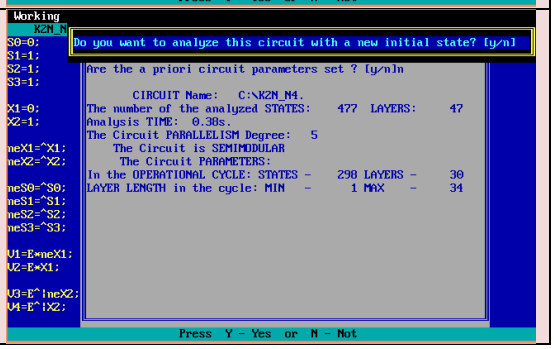
| № | S0S1S2S3 | X1X2 | Полумодулятность |
|---|---|---|---|
| 7 | 0110 | 01 |  |
| | | 11 |  |
| | | 10 |  |
| 8 | 0111 | 00 |  |
| | | 01 |  |

Таблица П3.1 (Продолжение)

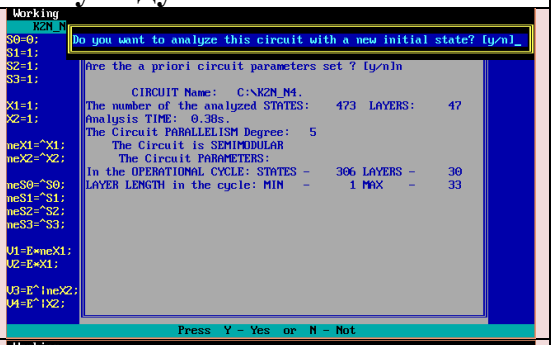
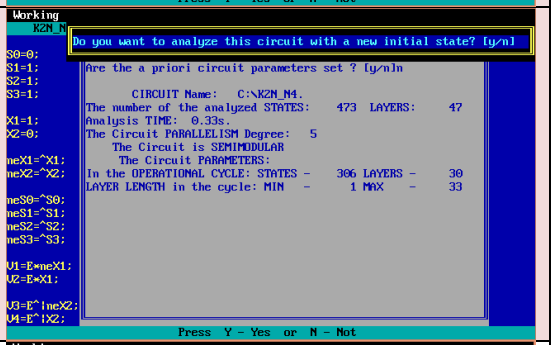
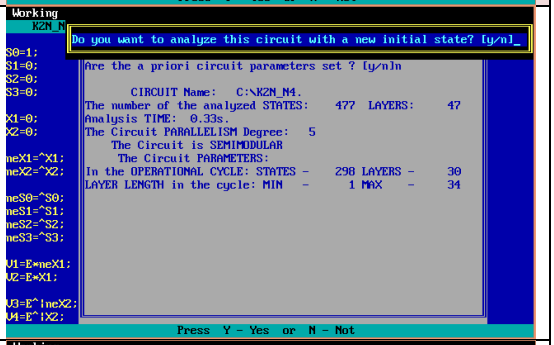
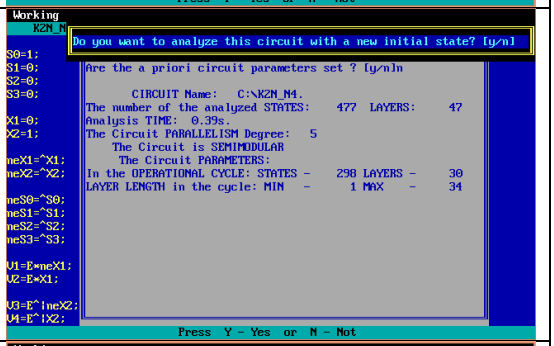
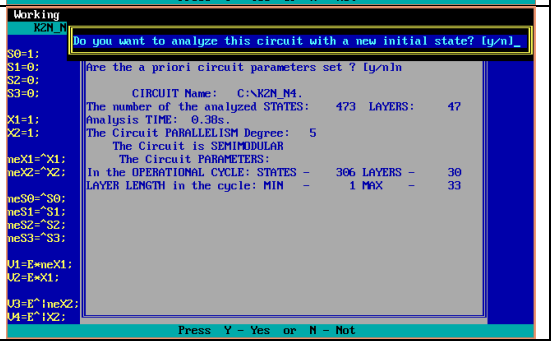
| № | S0S1S2S3 | X1X2 | Полумодулятность |
|---|---|---|---|
| 8 | 0111 | 11 |  |
| | | 10 |  |
| 9 | 1000 | 00 |  |
| | | 01 |  |
| | | 11 |  |

Таблица П3.1 (Продолжение)

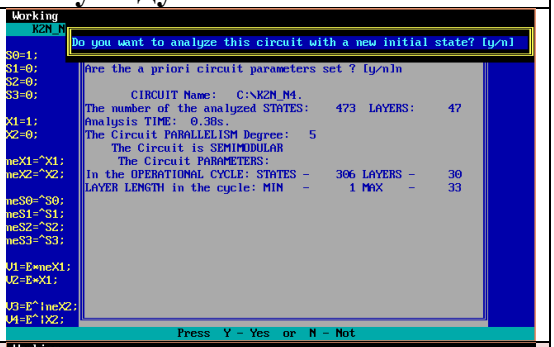
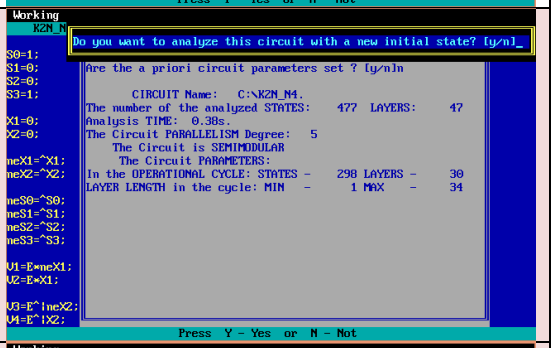
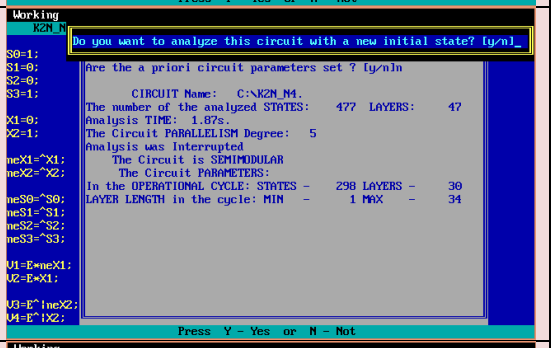
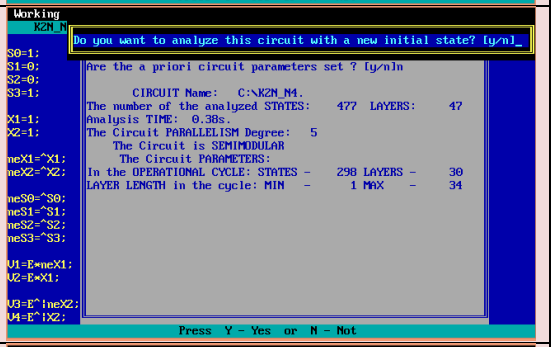
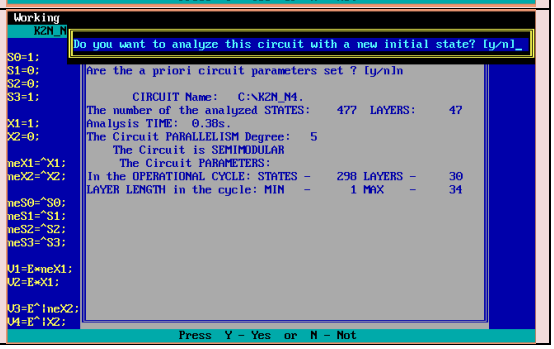
| № | S0S1S2S3 | X1X2 | Полумодулятность |
|---|---|---|---|
| 9 | 1000 | 10 |  |
| 10 | 1001 | 00 |  |
| | | 01 |  |
| | | 11 |  |
| | | 10 |  |

Таблица П3.1 (Продолжение)

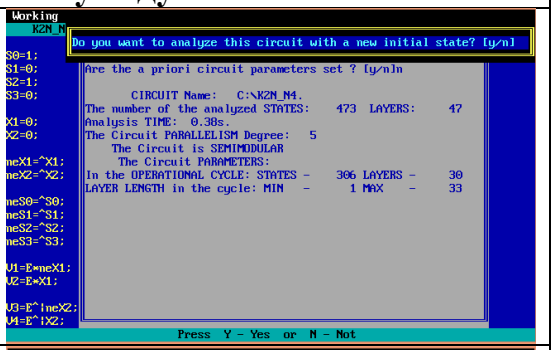
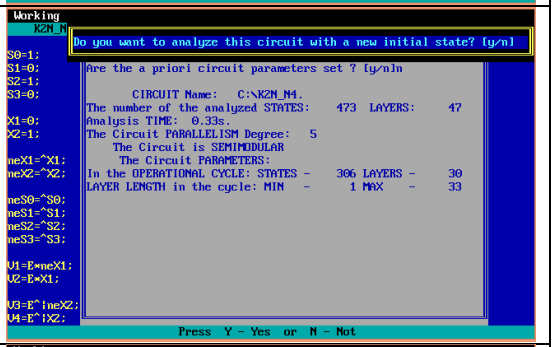
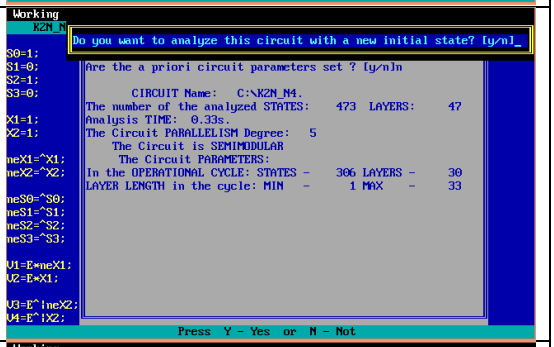
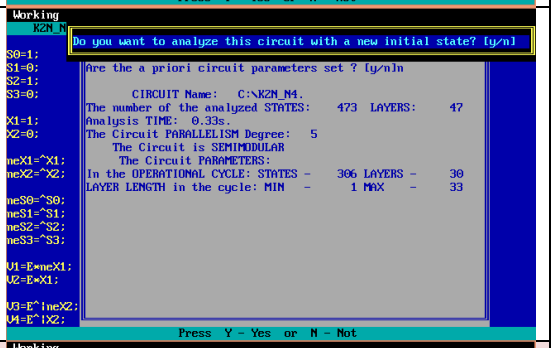
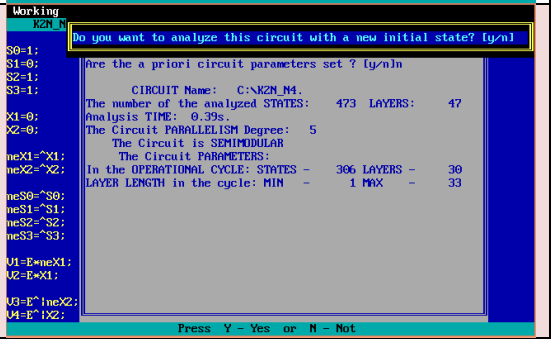
| № | S0S1S2S3 | X1X2 | Полумодулятность |
|---|---|---|---|
| 11 | 1010 | 00 |  |
| | | 01 |  |
| | | 11 |  |
| | | 10 |  |
| 12 | 1011 | 00 |  |

Таблица П3.1 (Продолжение)

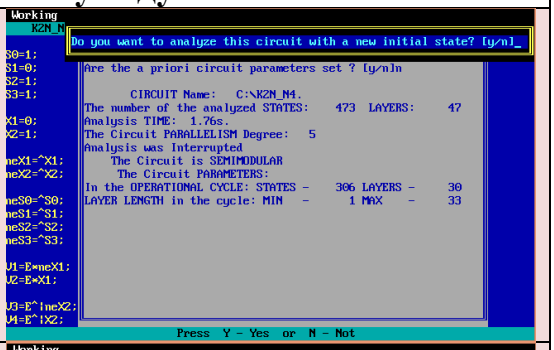
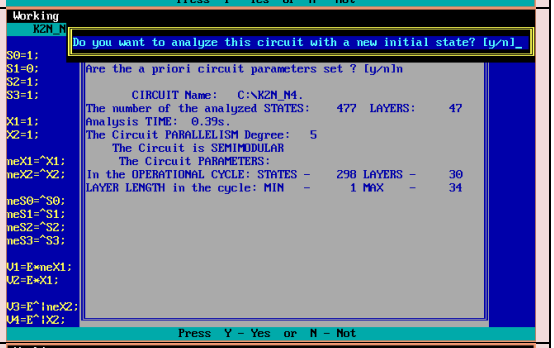
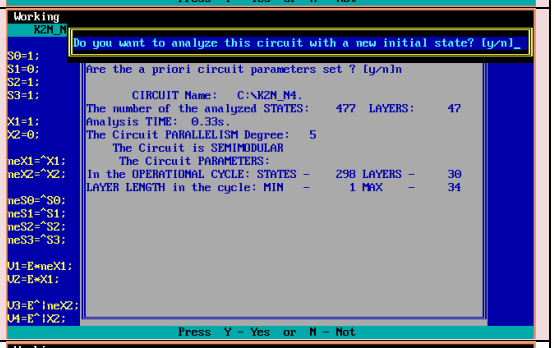
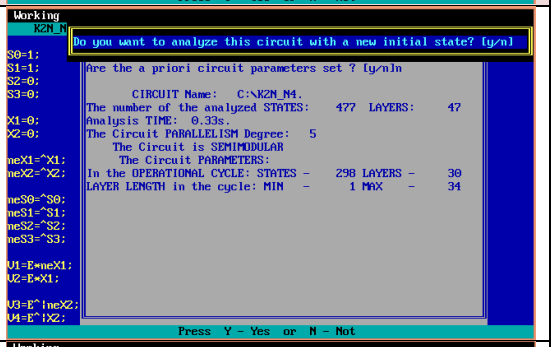
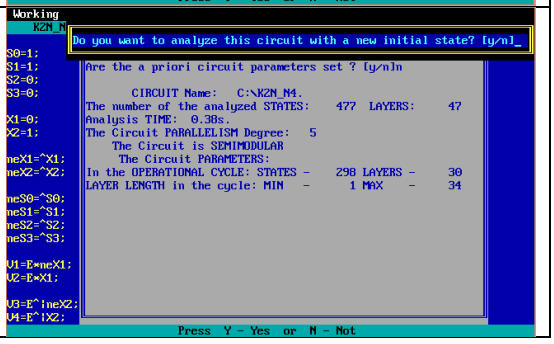
| № | S0S1S2S3 | X1X2 | Полумодулятность |
|---|---|---|---|
| 12 | 1011 | 01 |  |
| | | 11 |  |
| | | 10 |  |
| 13 | 1100 | 00 |  |
| | | 01 |  |

Таблица П3.1 (Продолжение)

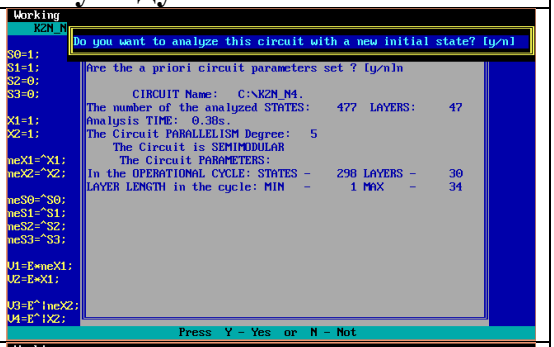
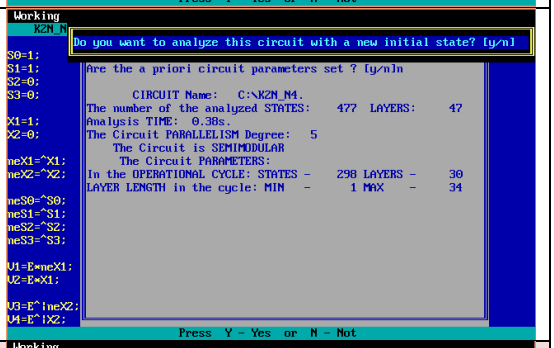
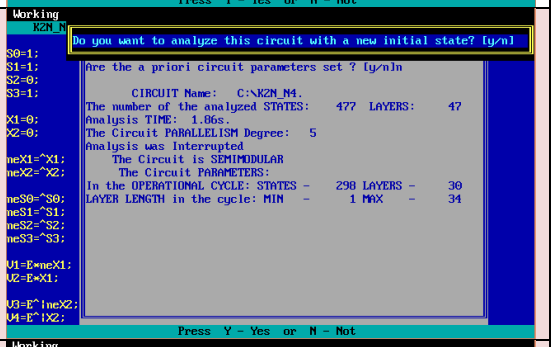
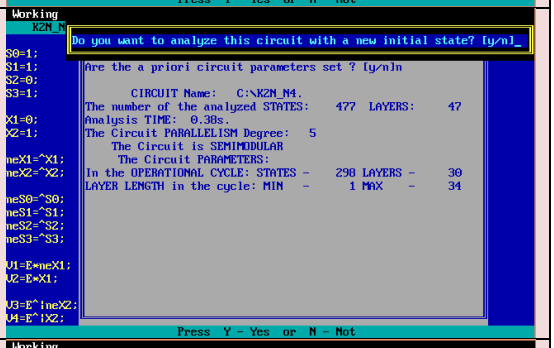
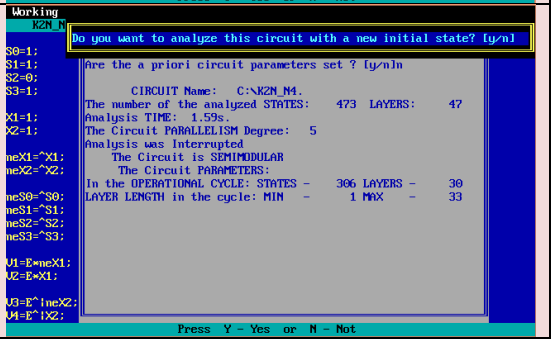
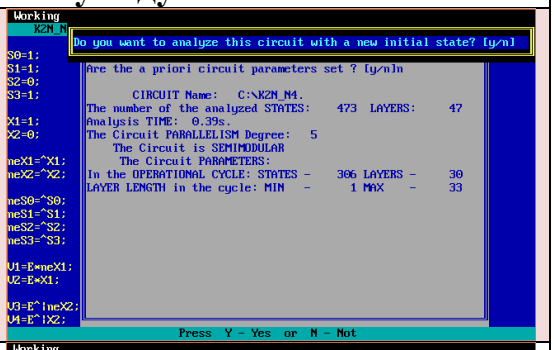
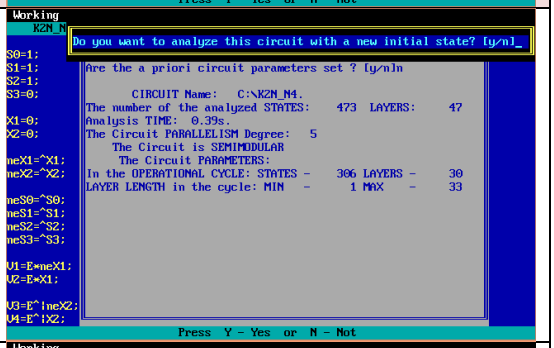
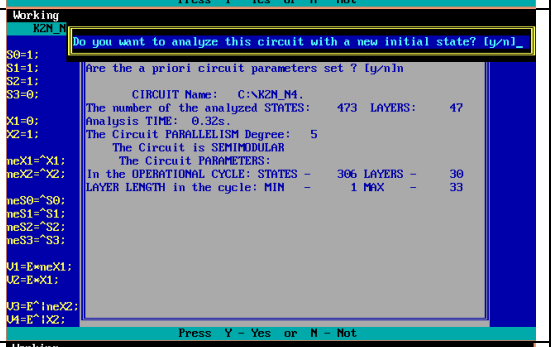
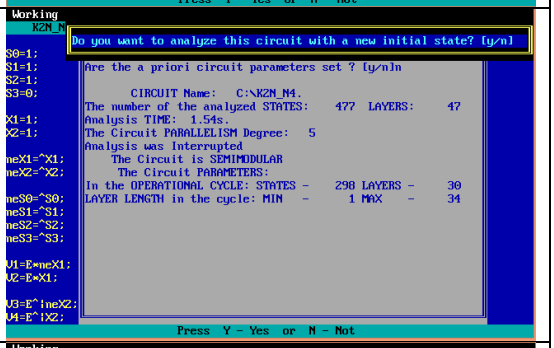
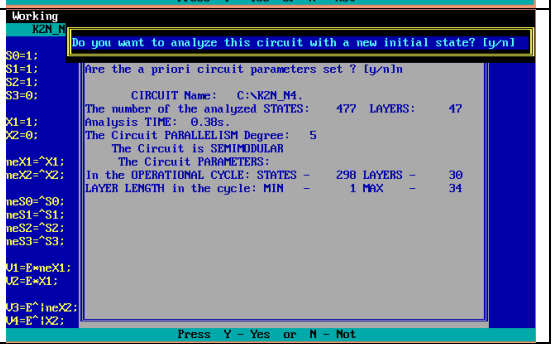
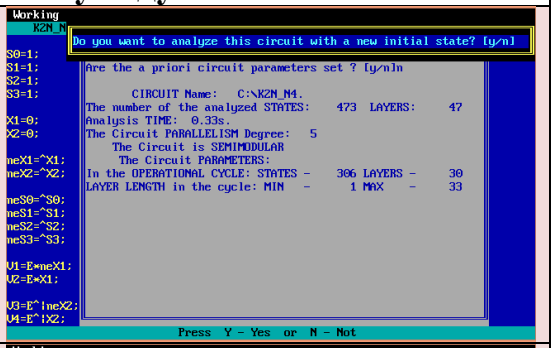
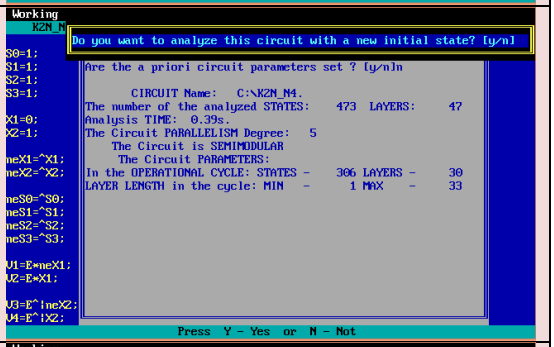
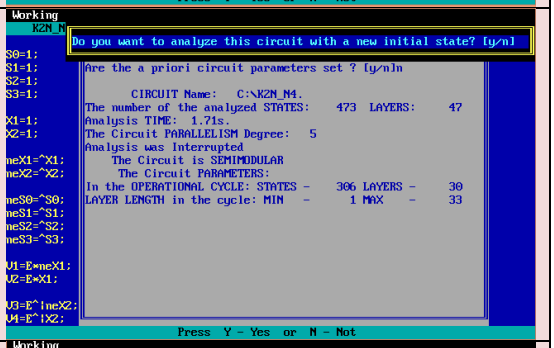
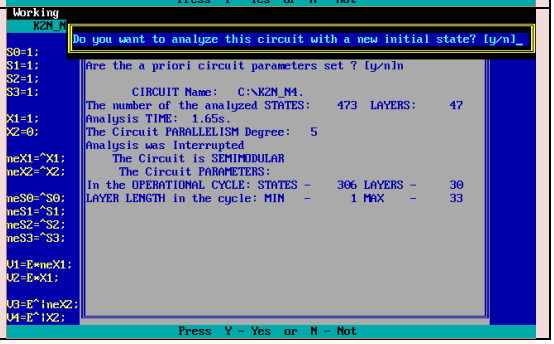
| № | S0S1S2S3 | X1X2 | Полумодулятность |
|---|---|---|---|
| 13 | 1100 | 11 |  |
| | | 10 |  |
| 14 | 1101 | 00 |  |
| | | 01 |  |
| | | 11 |  |

Таблица П3.1 (Продолжение)

| № | S0S1S2S3 | X1X2 | Полумодулятность |
|---|---|---|---|
| 14 | 1101 | 10 |  |
| 15 | 1110 | 00 |  |
| | | 01 |  |
| | | 11 |  |
| | | 10 |  |

Продолжение приложения №3

Таблица П3.1 (Окончание)

| № | S0S1S2S3 | X1X2 | Полумодулятность |
|---|---|---|---|
| 16 | 1111 | 00 |  |
| | | 01 |  |
| | | 11 |  |
| | | 10 |  |









Анализ схемы на рисунке 2.7 на полумодулярность.

S0=0; S1=1;

X=0;

neS0=^S0; neS1=^S1;

neX=^X;

Gr=0;

V1=^neX*E; V2=^X*E;

SP=Gr(V1*V2);

F=^S0^*V1^|S1^*V2^|SP;

neF=^neS0^*V1^|neS1^*V2^|SP;

I1=^V1^|V2^; I2=^F*neF;

Gt=^I1*I2^|Gti(I1|I2^); Gti=^Gt; E=Gti;

$Gt

Результат анализа представлен в таблице П3.2.

Таблица П3.2 – Результаты анализа схемы LUT-ST на полумодулярность.(Начало)

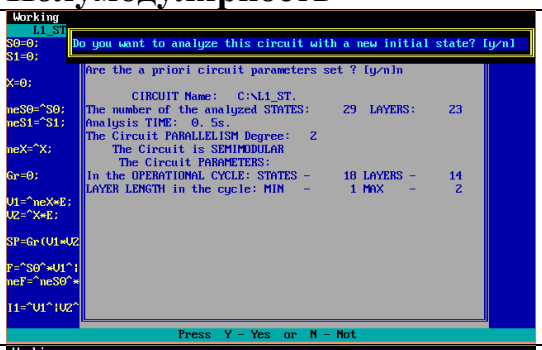
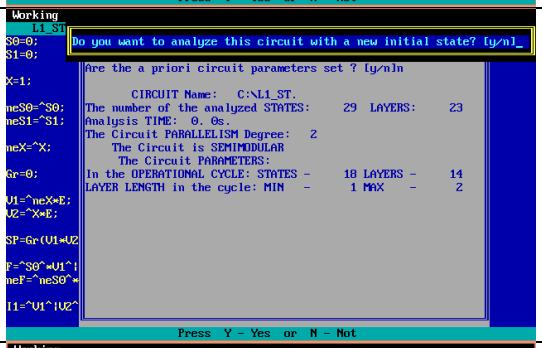
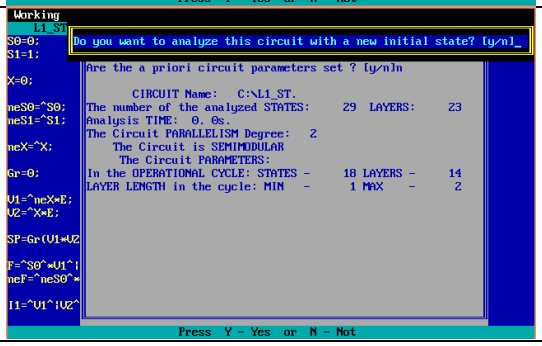
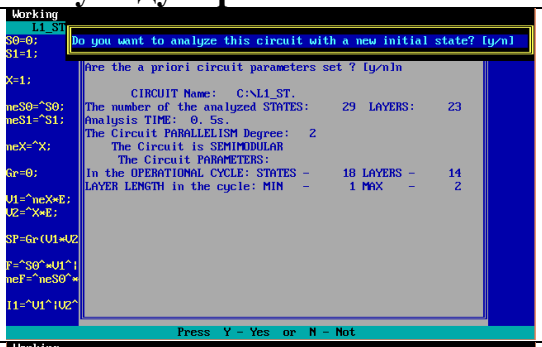
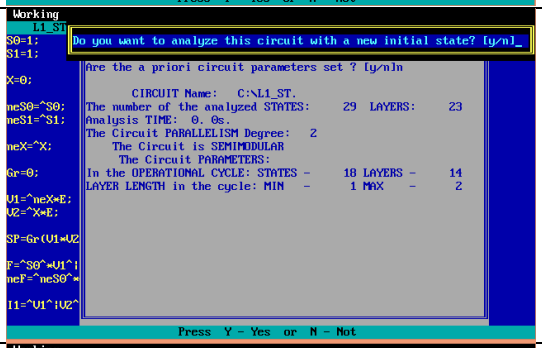
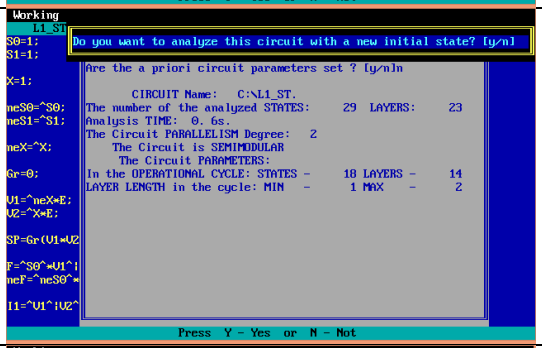
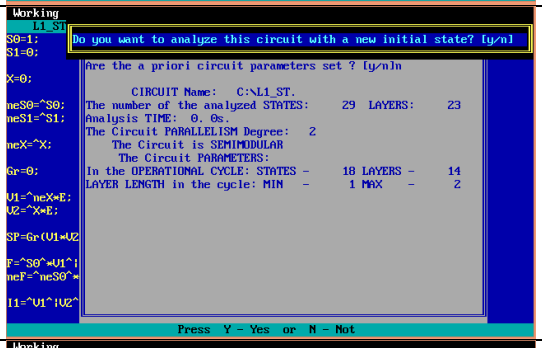
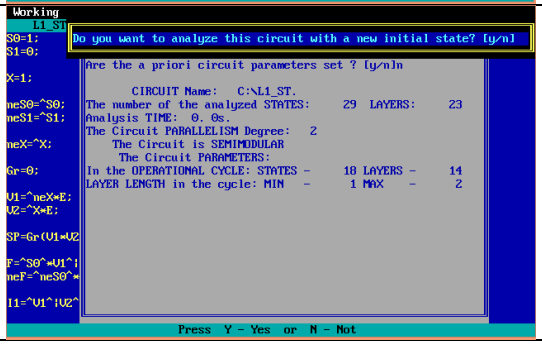
| № | S0S1 | X | Полумодулярность |
|---|---|---|---|
| 1 | 00 | 0 |  |
| | | 1 |  |
| 2 | 01 | 0 |  |

Таблица П3.2 (Окончание)

| № | S0S1 | X | Полумодулярность |
|---|---|---|---|
| 2 | 01 | 1 |  |
| 3 | 11 | 0 |  |
| | | 1 |  |
| 4 | 10 | 0 |  |
| | | 1 |  |

Продолжение приложения №3

Проверка схемы DC-LUT-ST на рисунке 2.9 на полумодулярность

Gr=0; Vc=1;

X=1; neX=^X;

V1=^neX*E; V2=^X*E; SP=Gr(V1*V2);

F0=Gr*V1^|Vc*V2^|SP; neF0=Vc*V1^|Gr*V2^|SP;

F1=Gr*V2^|Vc*V1^|SP; neF1=Vc*V2^|Gr*V1^|SP;

I1=^V1^|V2^; I2=F0^*neF0^; I3=F1^*neF1^;

Gt=^I1*I2*I3|Gti(I1|I2|I3); Gti=^Gt;

E=Gti;

$Gt

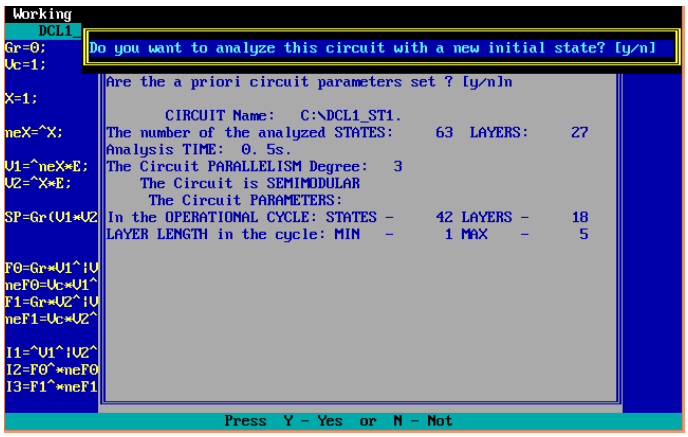



Рисунок П3.4 – Результат анализа элемента DC-LUT-ST в САПР Forcage (подсистема TRANAL)

Выполним проверку блока настройки, показанного на рисунке 2.13.

S0=0; S1=1; neS0=S0^; neS1=S1^;

F1=1; F2=0; neF1=0; neF2=1;

A=S0|F1|E^; B=S1|F2|E^; Z1ch=^A*B;

H=neS0*neF1|E^; P=neS1*neF2|E^; neZ=^H|P;

I=^Z1ch|neZ; I0=A*H; I1=B*P;

g1=^I0*I1|g1i(I0|I1); g1i=^g1;

G=^I*g1i|Gi(I|g1i); Gi=^G;

E=Gi;

$G

Продолжение приложения №3

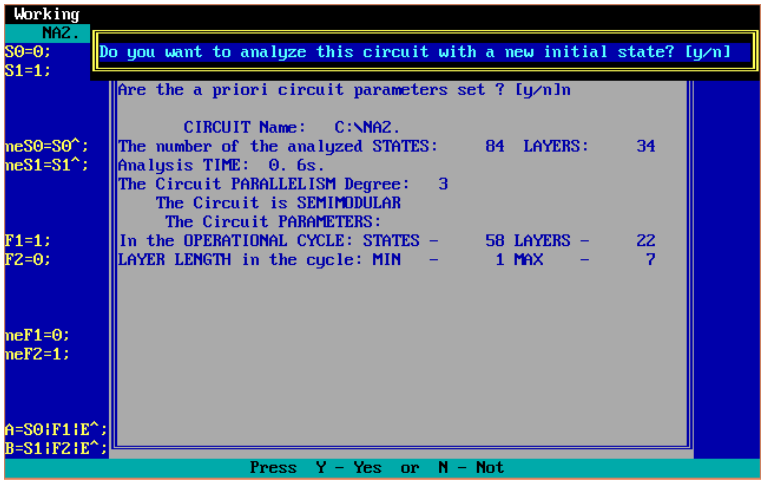



Рисунок П3.5 – Результат анализа блока дизъюнкции настройки функции на два входа каналов

Выполним проверку блока конъюнкций, показанного на рисунке 2.16.

S0=0; S1=1; neS0=1; neS1=0;

X=0; nX=1;

A=S0|X|E^; B=S1|nX^|E^; Z=A*B;

C=neS0*nX|E^; D=neS1*X^|E^; nZ=C|D;

I=Z*nZ; I0=A*C; I1=B*D;

g1=^I0*I1|g1i(I0|I1); g1i=^g1;

G=^I*g1i|Gi(I|g1i); Gi=^G;

E=Gi;

$G

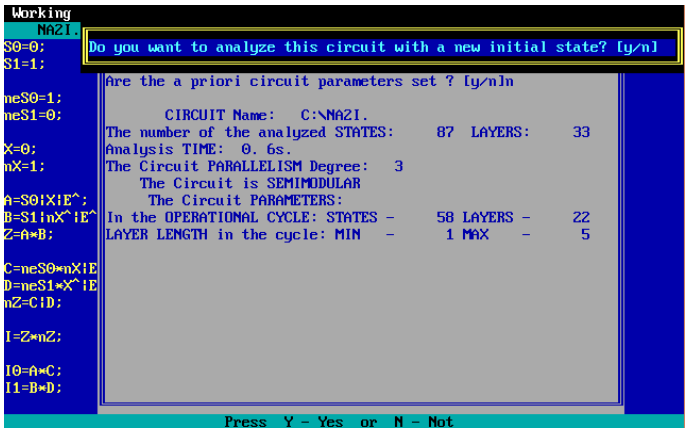



Рисунок П3.6 – Результат анализа блока конъюнкций одного разряда на основе библиотечных элементов

Окончание приложения №3

Выполним проверку блока конъюнкций, показанного на рисунке 2.24.

Gr=0; Vc=1;

X=1; nX=^X;

S0=0; S1=1;

V1=^nX*E; V2=^X*E; Sp=Gr(V1*V2);

F=^Vc(V1^|S0*V2^|S1*V1^|V2^)|Sp;

nF=^Gr(V1^|S0*V2^|S1*V1^|V2^)|Sp;

I=^V1^|V2^; I1=F*nF;

G1=^I*I1|G1i(I|I1); G1i=^G1;

E=G1i;

$G1

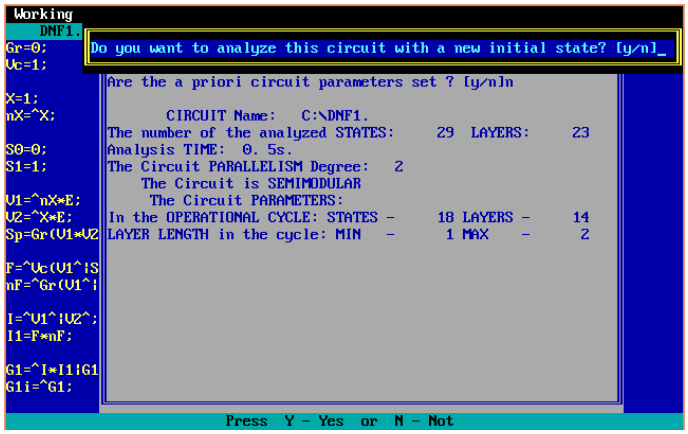



Рисунок П3.7 – Результат анализа блока конъюнкций одного разряда на основе библиотечных элементов с использованием транзисторов ортогональности

**Приложение №4. Анализ работы схем в САПР «Ковчег»**

Анализ работы генератора функций на 1 переменную

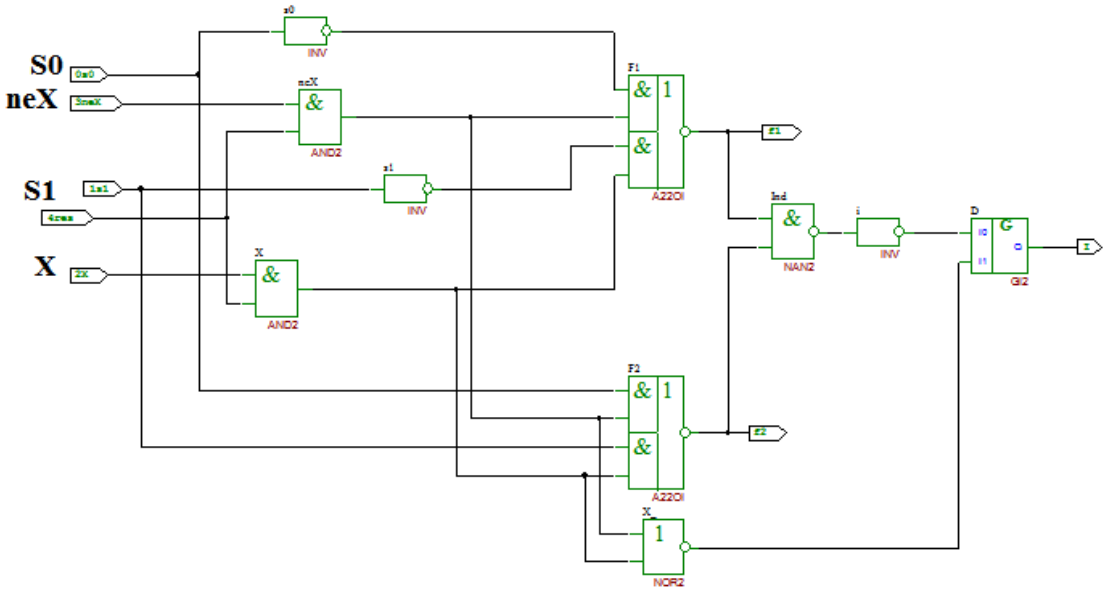



Рисунок П4.1 – Генератор функций одной переменной

Листинг тестов для анализа генератора функций одной переменной:

Test1=4, Test2=4, Test3=4, Test4=4;

Test1:

4res = 0,1;

0s0 = 0;

1s1 = 0;

2X = 0,1,0,1;

3neX = 1,0,1,0;

Test2:

4res = 0,1;

0s0 = 0;

1s1 = 1;

2X = 0,1,0,1;

3neX = 1,0,1,0;

Test3:

4res = 0,1;

0s0 = 1;

1s1 = 0;

2X = 0,1,0,1;

3neX = 1,0,1,0;

Test4:

4res = 0,1;

0s0 = 1;

1s1 = 1;

2X = 0,1,0,1;

3neX = 1,0,1,0;

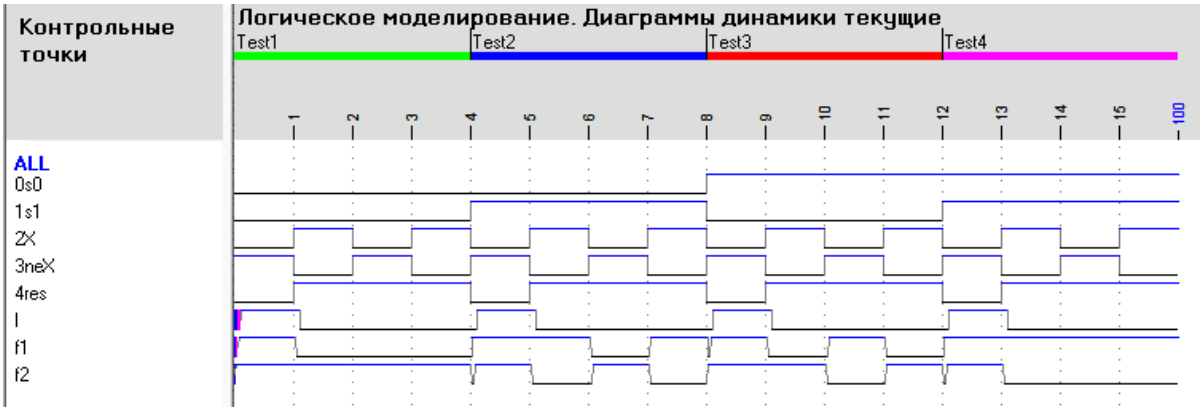



Рисунок П4.2 – Результат моделирования генератора функций одной переменной в САПР «Ковчег»

На рисунке П4.2 видно, что общий индикатор схемы устанавливается в логическую 1, когда схема переходит в фазу спейсера (оба выхода в логической 1). В рабочей фазе выходы инверсны друг другу.

На основании результатов моделирования получим таблицу истинности (таблица П4.1)

Таблица П4.1 – Таблица истинности генератора логических функций одной переменной с нулевым спейсером.

| № | S0S1 | X | F | F’ | I |
|---|---|---|---|---|---|
| 1 | 00 | 0 | 0 | 1 | 0 |
| | | 1 | 0 | 1 | 0 |
| 2 | 01 | 0 | 0 | 1 | 0 |
| | | 1 | 1 | 0 | 0 |
| 3 | 10 | 0 | 1 | 0 | 0 |
| | | 1 | 0 | 1 | 0 |
| 4 | 11 | 0 | 1 | 0 | 0 |
| | | 1 | 1 | 0 | 0 |

Таблица истинности, полученная с помощью моделирования, подтверждает работоспособность генератора функций одной переменной.

Листинг тестов для анализа генератора функций двух переменных

```
T1=8, T2=8, T3=8, T4=8, T5=8, T6=8, T7=8, T8=8, T9=8, T10=8, T11=8,
T12=8, T13=8, T14=8, T15=8, T16=8;
T1:
8R=1,0,1,0,1,0,1,0;
9R=0,1,0,1,0,1,0,1;
0s0 = 0;
1s1 = 0;
2s2 = 0;
3s3 = 0;
4X1 =   0,0,1,0,1,0,0,0;
5neX1 = 1,0,0,0,0,0,1,0;
6X2 =   0,0,0,0,1,0,1,0;
7neX2 = 1,0,1,0,0,0,0,0;

T2:
8R=1,0,1,0,1,0,1,0;
9R=0,1,0,1,0,1,0,1;
0s0 = 0;
1s1 = 0;
2s2 = 0;
3s3 = 1;
4X1 =   0,0,1,0,1,0,0,0;
5neX1 = 1,0,0,0,0,0,1,0;
6X2 =   0,0,0,0,1,0,1,0;
7neX2 = 1,0,1,0,0,0,0,0;

T3:
8R=1,0,1,0,1,0,1,0;
9R=0,1,0,1,0,1,0,1;
0s0 = 0;
1s1 = 0;
2s2 = 1;
3s3 = 0;
4X1 =   0,0,1,0,1,0,0,0;
5neX1 = 1,0,0,0,0,0,1,0;
6X2 =   0,0,0,0,1,0,1,0;
7neX2 = 1,0,1,0,0,0,0,0;

T4:
8R=1,0,1,0,1,0,1,0;
9R=0,1,0,1,0,1,0,1;
0s0 = 0;
1s1 = 0;
2s2 = 1;
3s3 = 1;
4X1 =   0,0,1,0,1,0,0,0;
5neX1 = 1,0,0,0,0,0,1,0;
6X2 =   0,0,0,0,1,0,1,0;
7neX2 = 1,0,1,0,0,0,0,0;

T5:
8R=1,0,1,0,1,0,1,0;
9R=0,1,0,1,0,1,0,1;
0s0 = 0;
1s1 = 1;
2s2 = 0;
3s3 = 0;
4X1 =   0,0,1,0,1,0,0,0;
5neX1 = 1,0,0,0,0,0,1,0;
6X2 =   0,0,0,0,1,0,1,0;
7neX2 = 1,0,1,0,0,0,0,0;

T6:
8R=1,0,1,0,1,0,1,0;
9R=0,1,0,1,0,1,0,1;
0s0 = 0;
```

Продолжение приложения №4

```
1s1 = 1;
2s2 = 0;
3s3 = 1;
4X1 =   0,0,1,0,1,0,0,0;
5neX1 = 1,0,0,0,0,0,1,0;
6X2 =   0,0,0,0,1,0,1,0;
7neX2 = 1,0,1,0,0,0,0,0;

T7:
8R=1,0,1,0,1,0,1,0;
9R=0,1,0,1,0,1,0,1;
0s0 = 0;
1s1 = 1;
2s2 = 1;
3s3 = 0;
4X1 =   0,0,1,0,1,0,0,0;
5neX1 = 1,0,0,0,0,0,1,0;
6X2 =   0,0,0,0,1,0,1,0;
7neX2 = 1,0,1,0,0,0,0,0;

T8:
8R=1,0,1,0,1,0,1,0;
9R=0,1,0,1,0,1,0,1;
0s0 = 0;
1s1 = 1;
2s2 = 1;
3s3 = 1;
4X1 =   0,0,1,0,1,0,0,0;
5neX1 = 1,0,0,0,0,0,1,0;
6X2 =   0,0,0,0,1,0,1,0;
7neX2 = 1,0,1,0,0,0,0,0;

T9:
8R=1,0,1,0,1,0,1,0;
9R=0,1,0,1,0,1,0,1;
0s0 = 1;
1s1 = 0;
2s2 = 0;
3s3 = 0;
4X1 =   0,0,1,0,1,0,0,0;
5neX1 = 1,0,0,0,0,0,1,0;
6X2 =   0,0,0,0,1,0,1,0;
7neX2 = 1,0,1,0,0,0,0,0;

T10:
8R=1,0,1,0,1,0,1,0;
9R=0,1,0,1,0,1,0,1;
0s0 = 1;
1s1 = 0;
2s2 = 0;
3s3 = 1;
4X1 =   0,0,1,0,1,0,0,0;
5neX1 = 1,0,0,0,0,0,1,0;
6X2 =   0,0,0,0,1,0,1,0;
7neX2 = 1,0,1,0,0,0,0,0;

T11:
8R=1,0,1,0,1,0,1,0;
9R=0,1,0,1,0,1,0,1;
0s0 = 1;
1s1 = 0;
2s2 = 1;
3s3 = 0;
4X1 =   0,0,1,0,1,0,0,0;
5neX1 = 1,0,0,0,0,0,1,0;
6X2 =   0,0,0,0,1,0,1,0;
7neX2 = 1,0,1,0,0,0,0,0;
```

```
T12:
8R=1,0,1,0,1,0,1,0;
9R=0,1,0,1,0,1,0,1;
0s0 = 1;
1s1 = 0;
2s2 = 1;
3s3 = 1;
4X1 =   0,0,1,0,1,0,0,0;
5neX1 = 1,0,0,0,0,0,1,0;
6X2 =   0,0,0,0,1,0,1,0;
7neX2 = 1,0,1,0,0,0,0,0;

T13:
8R=1,0,1,0,1,0,1,0;
9R=0,1,0,1,0,1,0,1;
0s0 = 1;
1s1 = 1;
2s2 = 0;
3s3 = 0;
4X1 =   0,0,1,0,1,0,0,0;
5neX1 = 1,0,0,0,0,0,1,0;
6X2 =   0,0,0,0,1,0,1,0;
7neX2 = 1,0,1,0,0,0,0,0;

T14:
8R=1,0,1,0,1,0,1,0;
9R=0,1,0,1,0,1,0,1;
0s0 = 1;
1s1 = 1;
2s2 = 0;
3s3 = 1;
4X1 =   0,0,1,0,1,0,0,0;
5neX1 = 1,0,0,0,0,0,1,0;
6X2 =   0,0,0,0,1,0,1,0;
7neX2 = 1,0,1,0,0,0,0,0;

T15:
8R=1,0,1,0,1,0,1,0;
9R=0,1,0,1,0,1,0,1;
0s0 = 1;
1s1 = 1;
2s2 = 1;
3s3 = 0;
4X1 =   0,0,1,0,1,0,0,0;
5neX1 = 1,0,0,0,0,0,1,0;
6X2 =   0,0,0,0,1,0,1,0;
7neX2 = 1,0,1,0,0,0,0,0;

T16:
8R=1,0,1,0,1,0,1,0;
9R=0,1,0,1,0,1,0,1;
0s0 = 1;
1s1 = 1;
2s2 = 1;
3s3 = 1;
4X1 =   0,0,1,0,1,0,0,0;
5neX1 = 1,0,0,0,0,0,1,0;
6X2 =   0,0,0,0,1,0,1,0;
7neX2 = 1,0,1,0,0,0,0,0;
```

Таблица П4.2 – Таблица истинности генератора логических функций двух переменных с реализацией возможных функций (Начало)

| № | S0S1S2S3 | X1X2 | F | F’ | G |
|---|---|---|---|---|---|
| 1 | 0000 | 00 | 0 | 1 | 0 |
| | | 01 | 0 | 1 | 0 |
| | | 11 | 0 | 1 | 0 |
| | | 10 | 0 | 1 | 0 |
| 2 | 0001 | 00 | 0 | 1 | 0 |
| | | 01 | 0 | 1 | 0 |
| | | 11 | 1 | 0 | 0 |
| | | 10 | 0 | 1 | 0 |
| 3 | 0010 | 00 | 0 | 1 | 0 |
| | | 01 | 1 | 0 | 0 |
| | | 11 | 0 | 1 | 0 |
| | | 10 | 0 | 1 | 0 |
| 4 | 0011 | 00 | 0 | 1 | 0 |
| | | 01 | 1 | 0 | 0 |
| | | 11 | 1 | 0 | 0 |
| | | 10 | 0 | 1 | 0 |
| 5 | 0100 | 00 | 0 | 1 | 0 |
| | | 01 | 0 | 1 | 0 |
| | | 11 | 0 | 1 | 0 |
| | | 10 | 1 | 0 | 0 |
| 6 | 0101 | 00 | 0 | 1 | 0 |
| | | 01 | 0 | 1 | 0 |
| | | 11 | 1 | 0 | 0 |
| | | 10 | 1 | 0 | 0 |
| 7 | 0110 | 00 | 0 | 1 | 0 |
| | | 01 | 1 | 0 | 0 |
| | | 11 | 0 | 1 | 0 |
| | | 10 | 1 | 0 | 0 |
| 8 | 0111 | 00 | 0 | 1 | 0 |
| | | 01 | 1 | 0 | 0 |
| | | 11 | 1 | 0 | 0 |
| | | 10 | 1 | 0 | 0 |
| 9 | 1000 | 00 | 1 | 0 | 0 |
| | | 01 | 0 | 1 | 0 |
| | | 11 | 0 | 1 | 0 |
| | | 10 | 0 | 1 | 0 |

Таблица П4.2 (Окончание)

| № | S0S1S2S3 | X1X2 | F | F’ | G |
|---|---|---|---|---|---|
| 10 | 1001 | 00 | 1 | 0 | 0 |
| | | 01 | 0 | 1 | 0 |
| | | 11 | 1 | 0 | 0 |
| | | 10 | 0 | 1 | 0 |
| 11 | 1010 | 00 | 1 | 0 | 0 |
| | | 01 | 1 | 0 | 0 |
| | | 11 | 0 | 1 | 0 |
| | | 10 | 0 | 1 | 0 |
| 12 | 1011 | 00 | 1 | 0 | 0 |
| | | 01 | 1 | 0 | 0 |
| | | 11 | 1 | 0 | 0 |
| | | 10 | 0 | 1 | 0 |
| 13 | 1100 | 00 | 1 | 0 | 0 |
| | | 01 | 0 | 1 | 0 |
| | | 11 | 0 | 1 | 0 |
| | | 10 | 1 | 0 | 0 |
| 14 | 1101 | 00 | 1 | 0 | 0 |
| | | 01 | 0 | 1 | 0 |
| | | 11 | 1 | 0 | 0 |
| | | 10 | 1 | 0 | 0 |
| 15 | 1110 | 00 | 1 | 0 | 0 |
| | | 01 | 1 | 0 | 0 |
| | | 11 | 0 | 1 | 0 |
| | | 10 | 1 | 0 | 0 |
| 16 | 1111 | 00 | 1 | 0 | 0 |
| | | 01 | 1 | 0 | 0 |
| | | 11 | 1 | 0 | 0 |
| | | 10 | 1 | 0 | 0 |

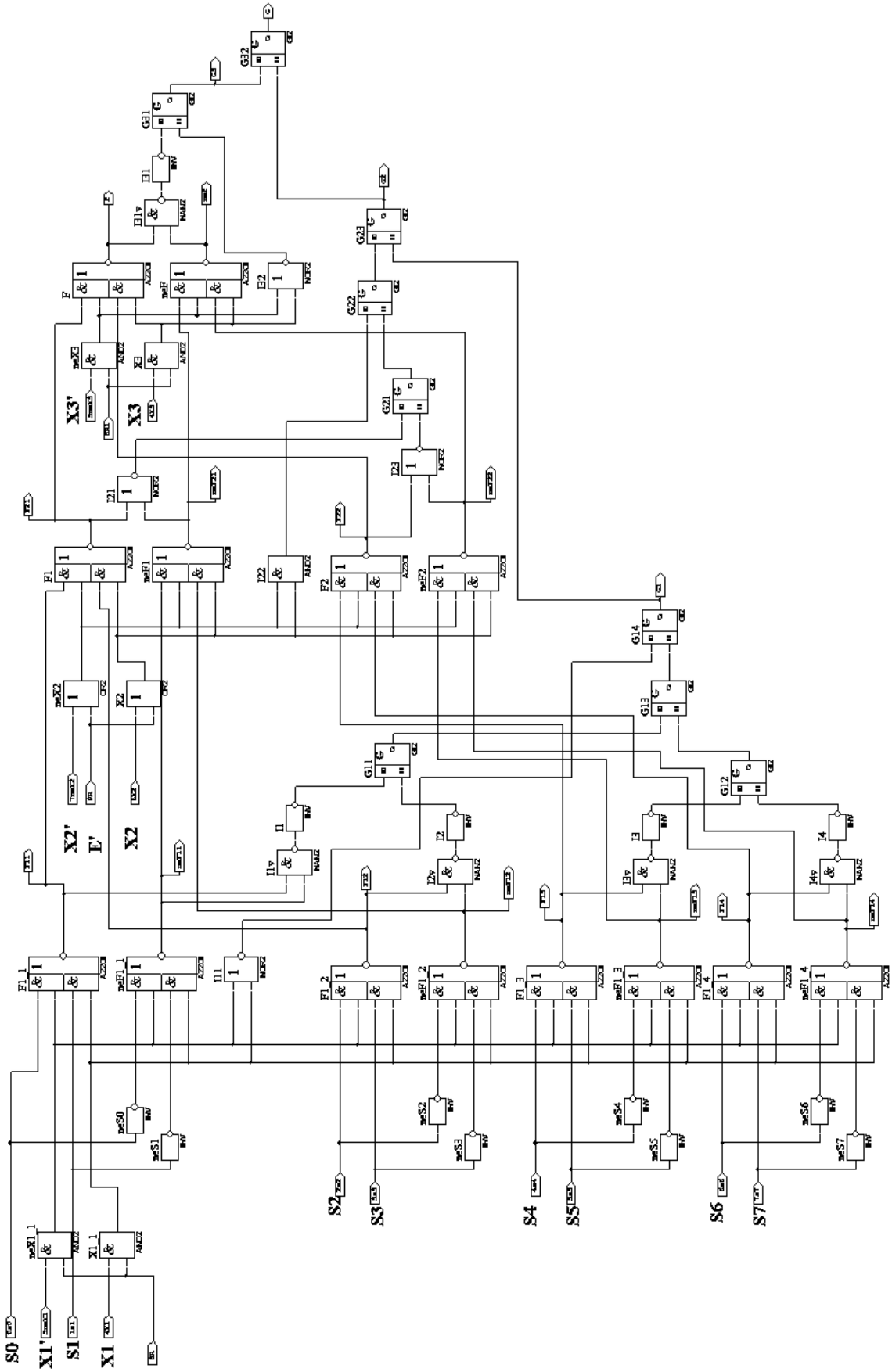



Рис.П4.3 – Моделирование генератора функций трех переменных в САПР «Ковчег»

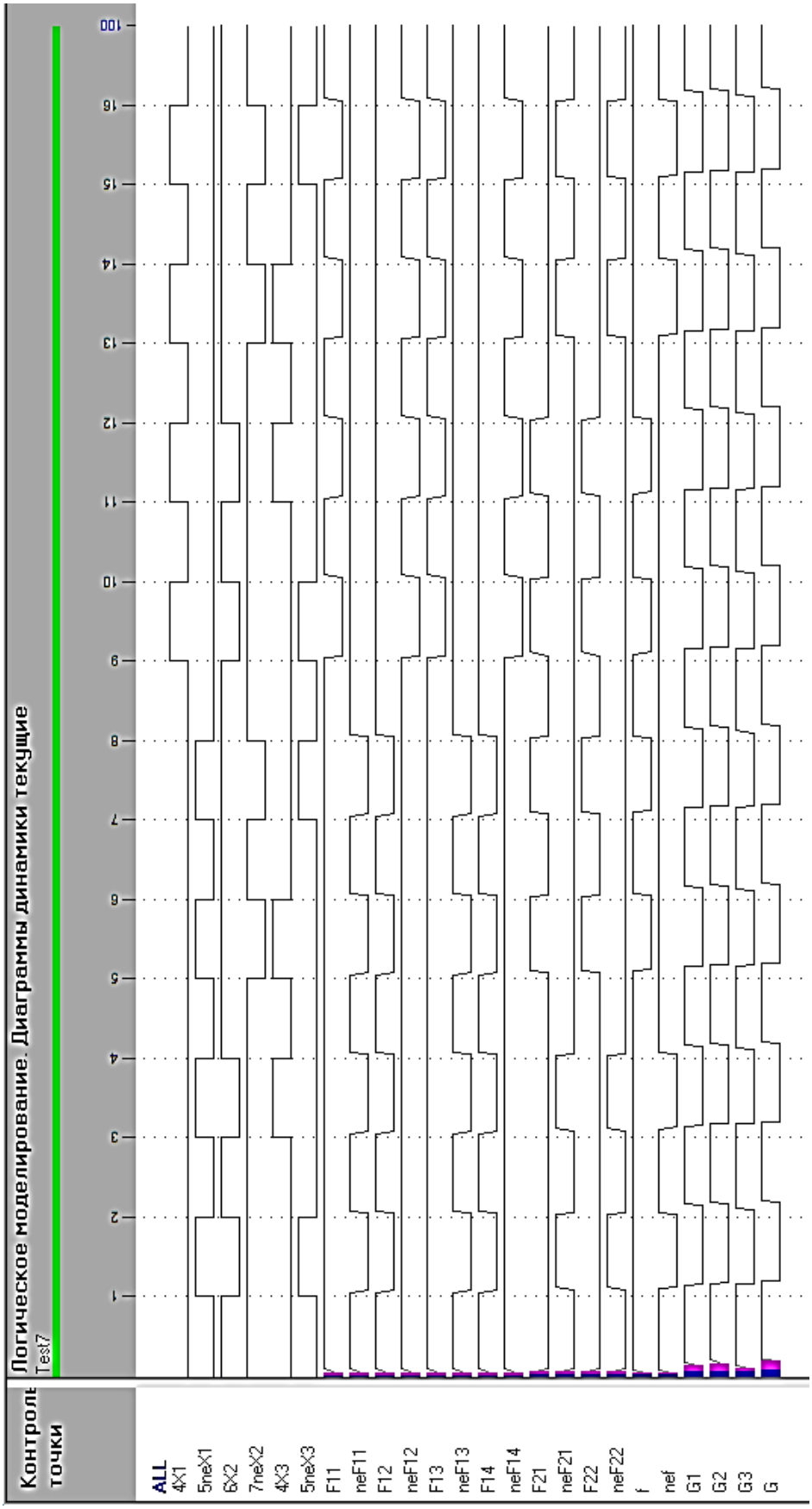



Рис. П4.4 – Результат моделирования генератора функций трех переменных в САПР «Ковчег»

**Приложение №5 Топологии элементов в САПР «Microwind»**

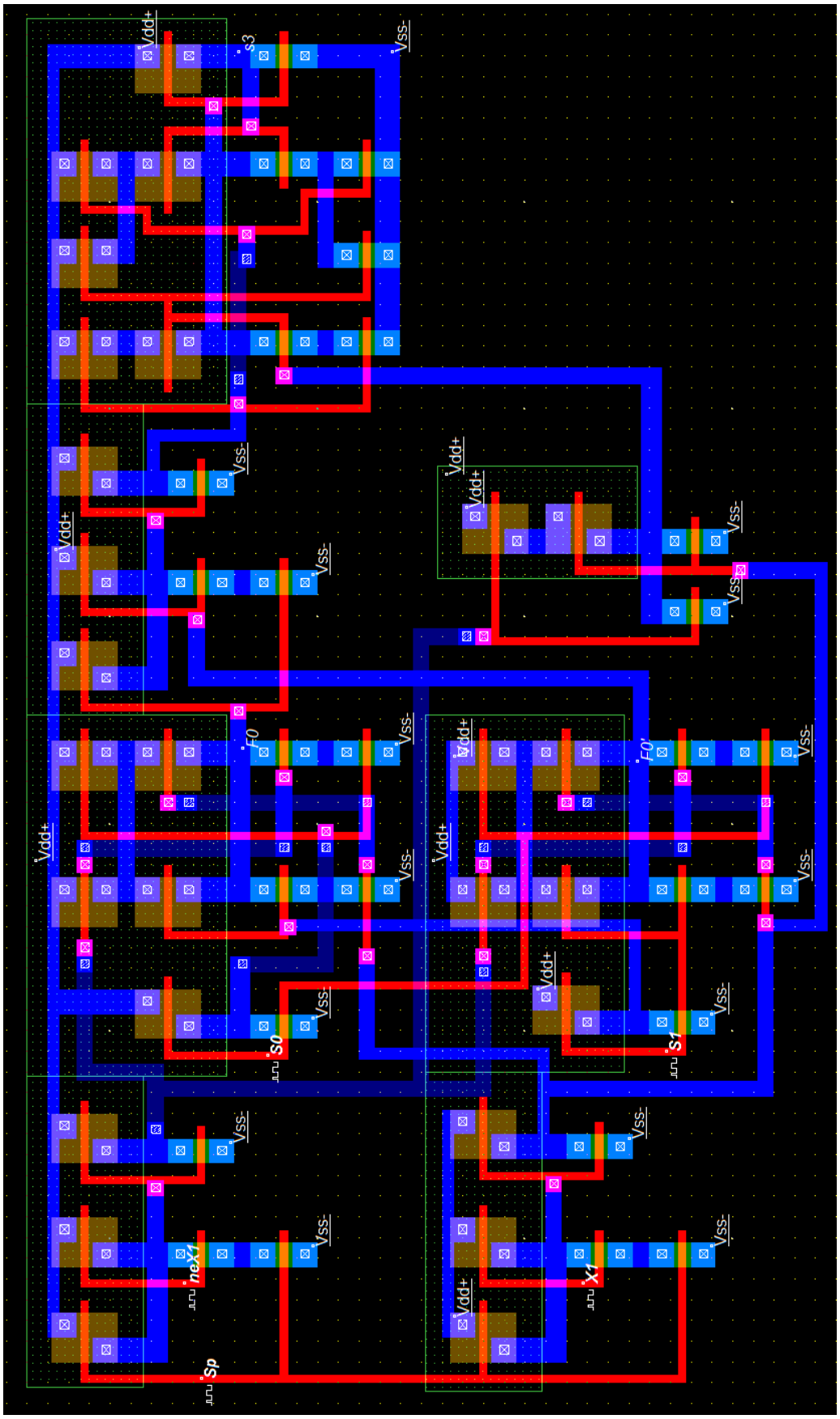



Рис.П5. 1 – Топология генератора функций одной переменной

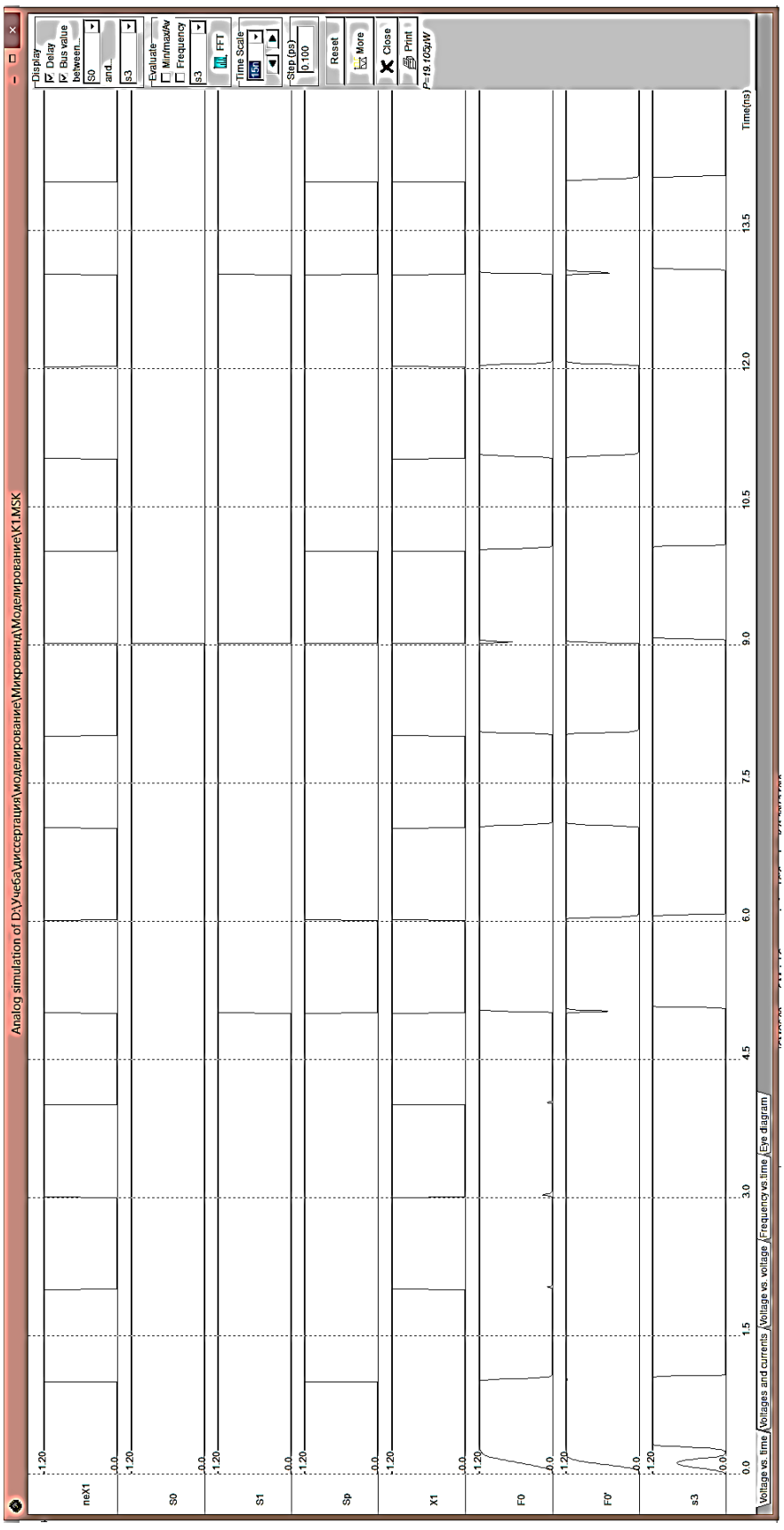



Рис.П5. 2 – Результат моделирования генератора функций одной перемен

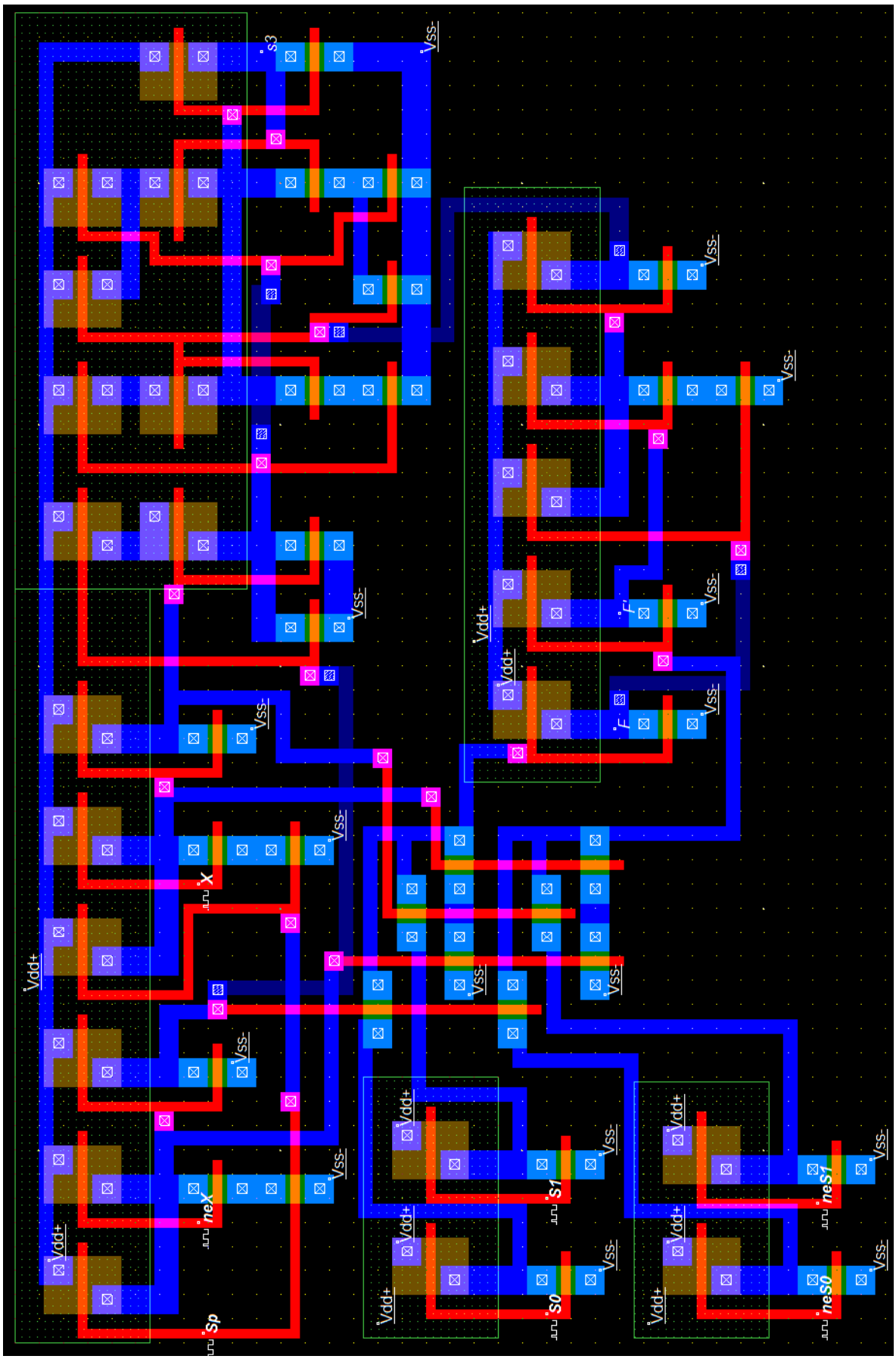



Рис. П5.3 – Топология элемента LUT-ST одной переменной (без восстановления уровня сигналов)

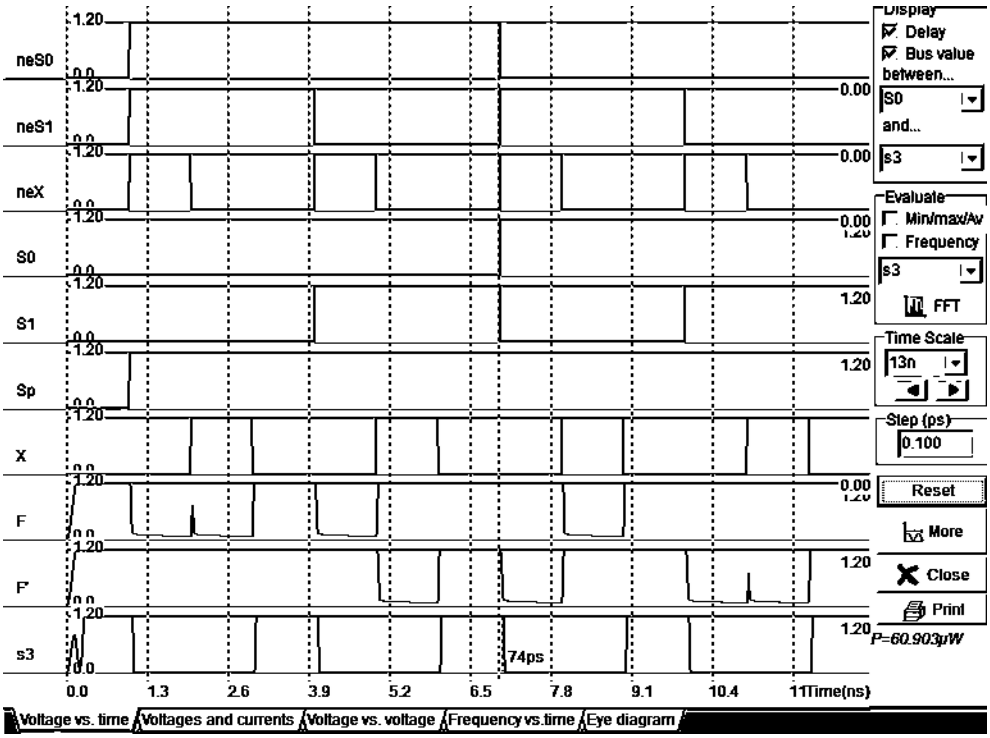



Рисунок П5. 4 – Результат моделирования элемента LUT-ST одной переменной (без восстановления уровня сигналов)

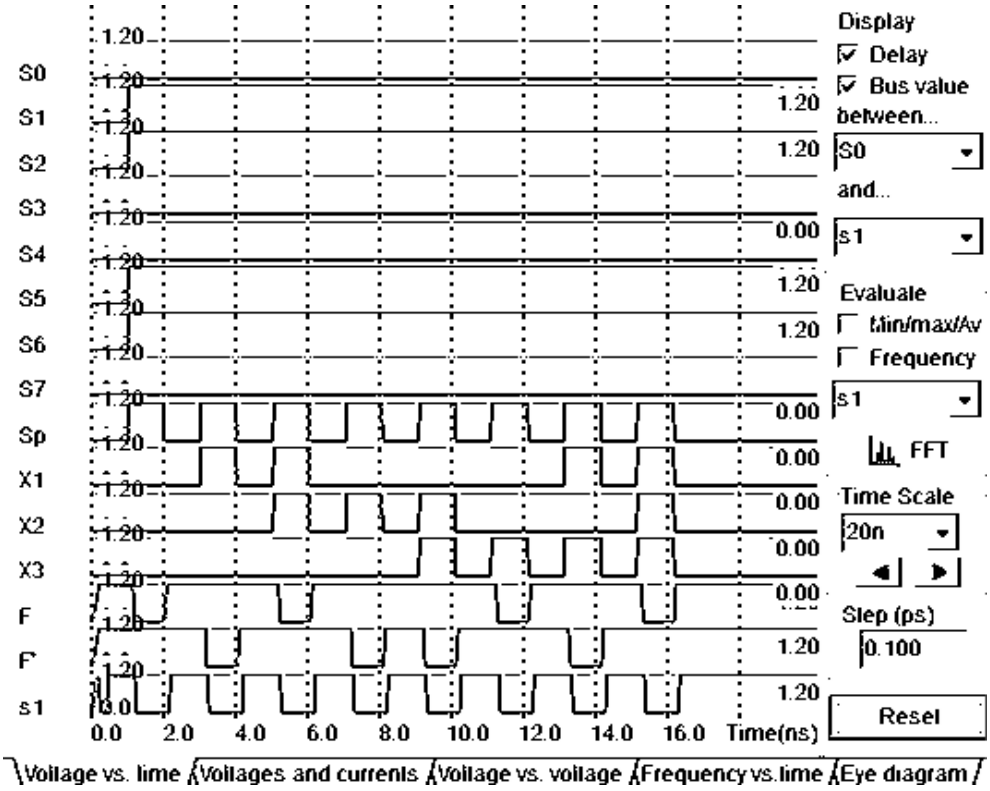



Рисунок П5. 5 – Результат моделирования элемента LUT-ST трех переменных (с восстановлением уровня сигналов)

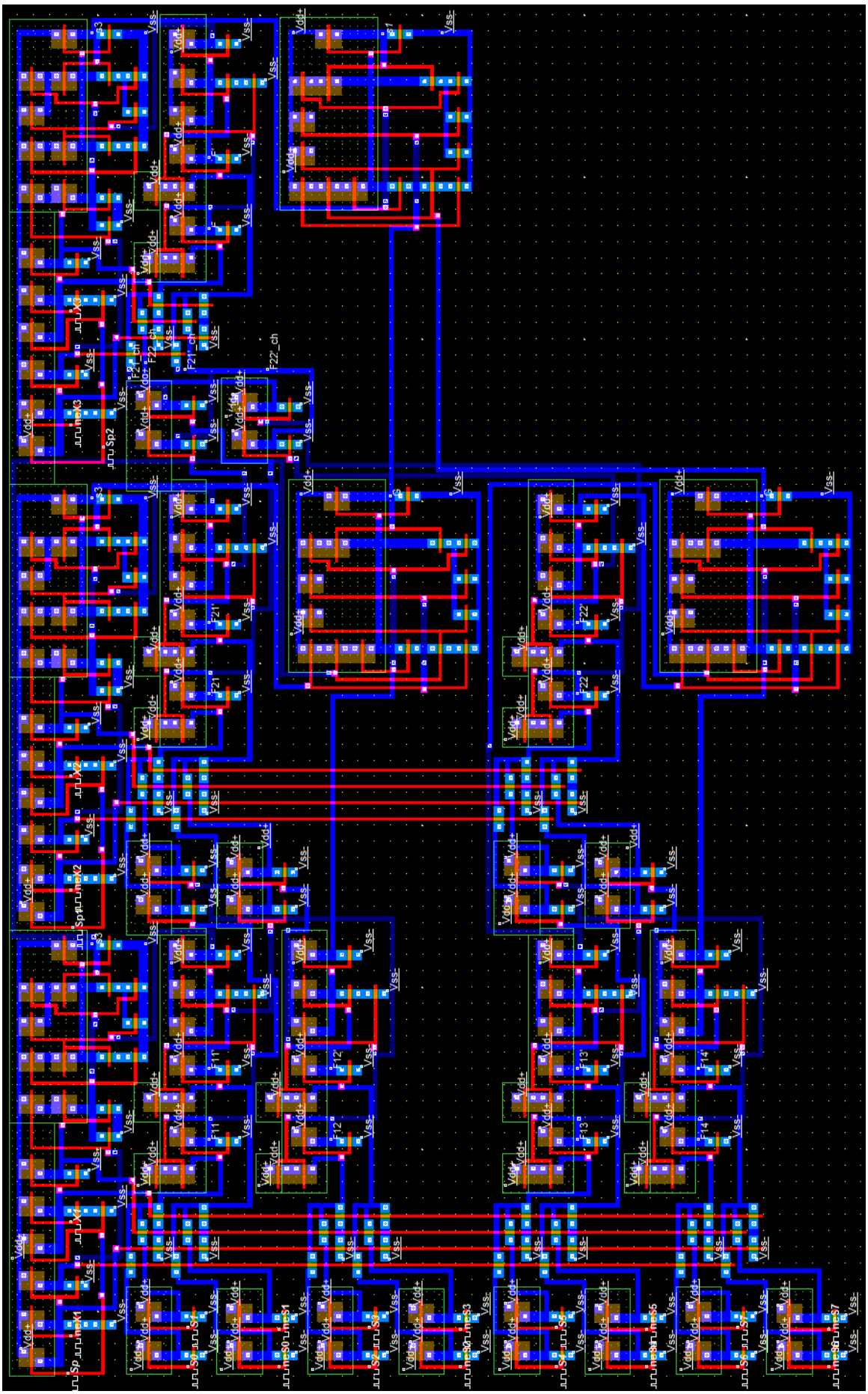


Рис. П5.6 – Топология элемента LUT-ST трех переменных (с восстановлением уровня сигнал

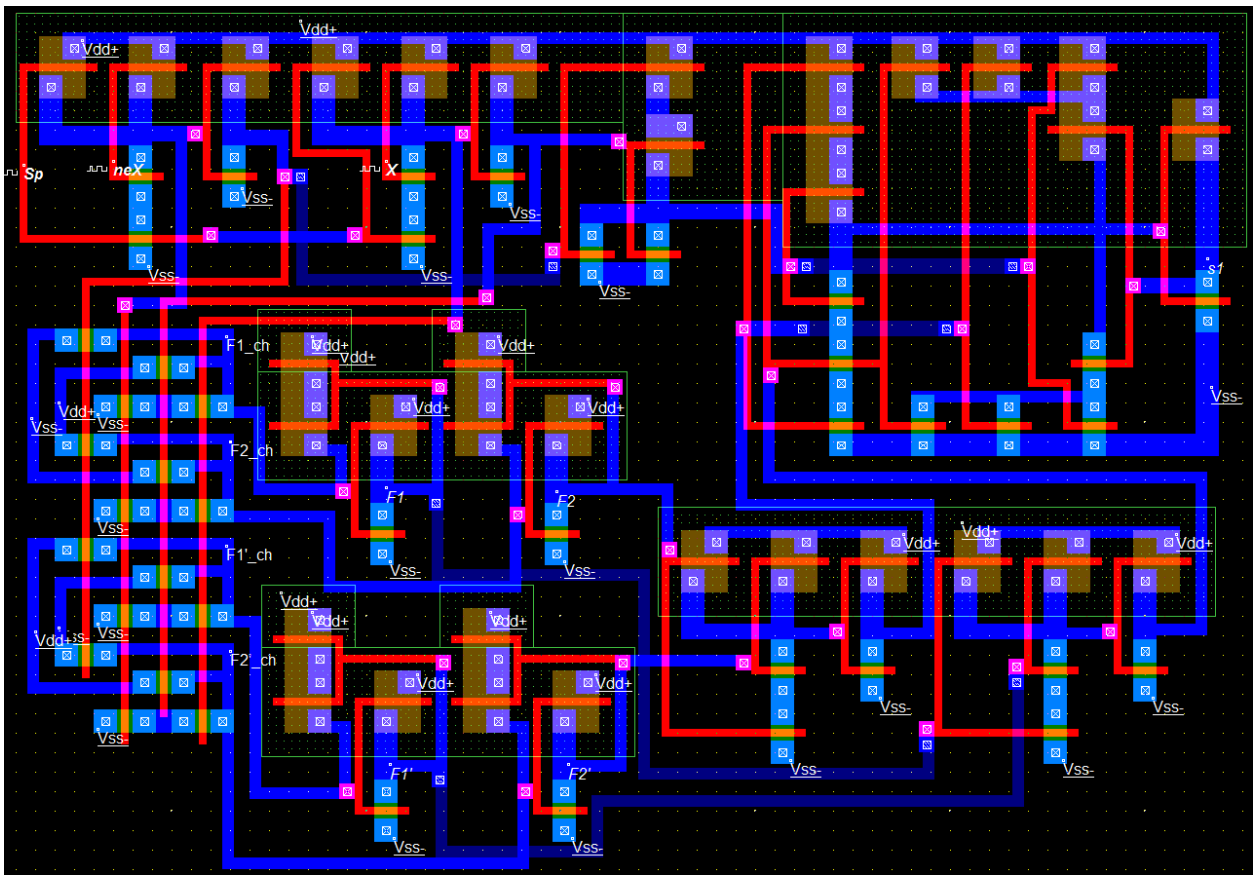



Рисунок П5. 7 – Топология элемента DC LUT-ST одной переменной (с восстановлением уровня сигналов)

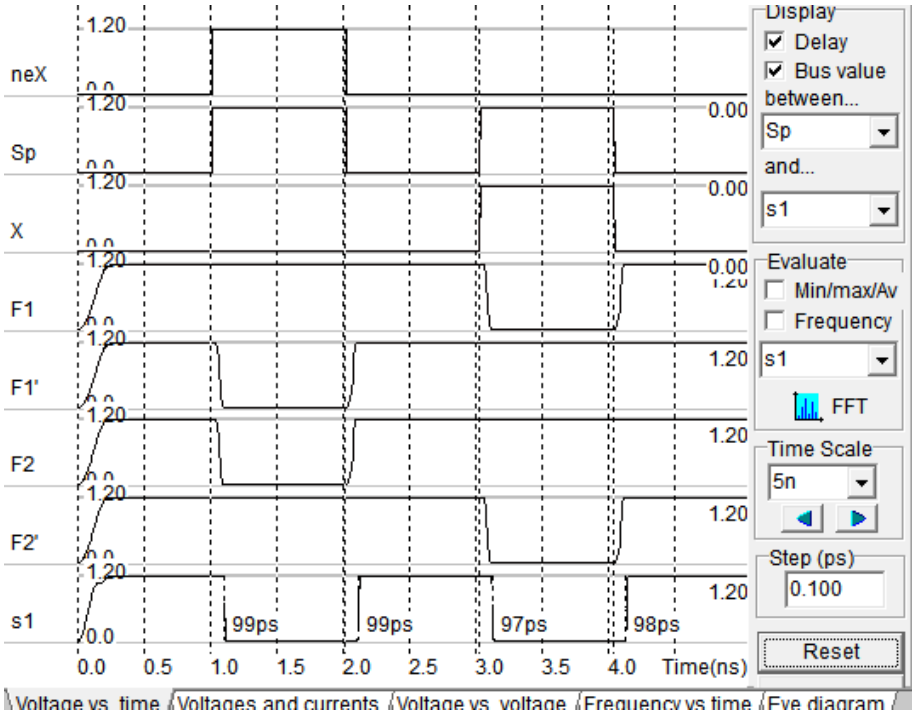



Рисунок П5. 8 – Результат моделирования элемента DC LUT-ST одной переменной (с восстановлением уровня сигналов)

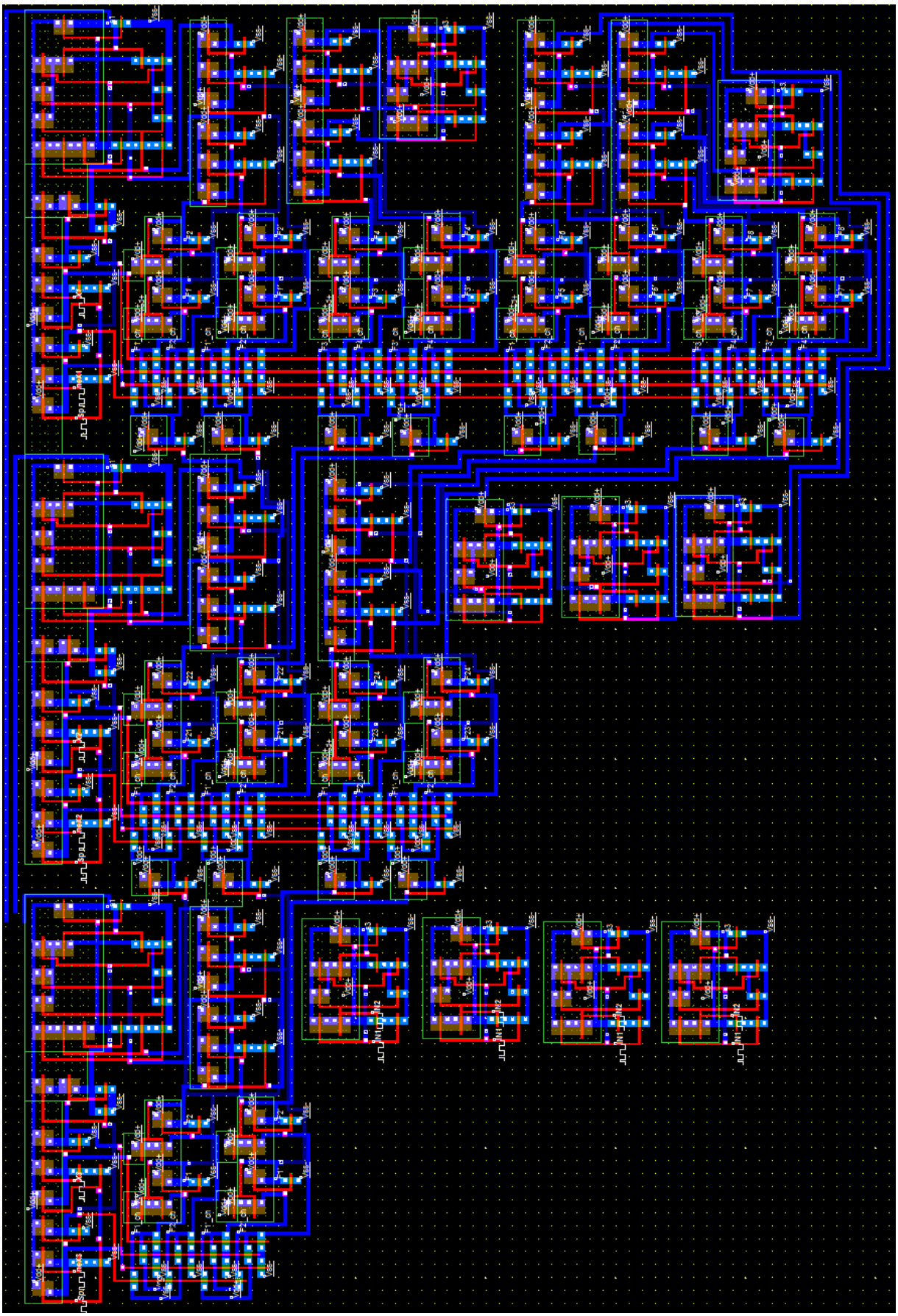


Рис.П5. 9 – Топология элемента DC LUT-ST трех переменных (с восстановлением уровня сигналов)

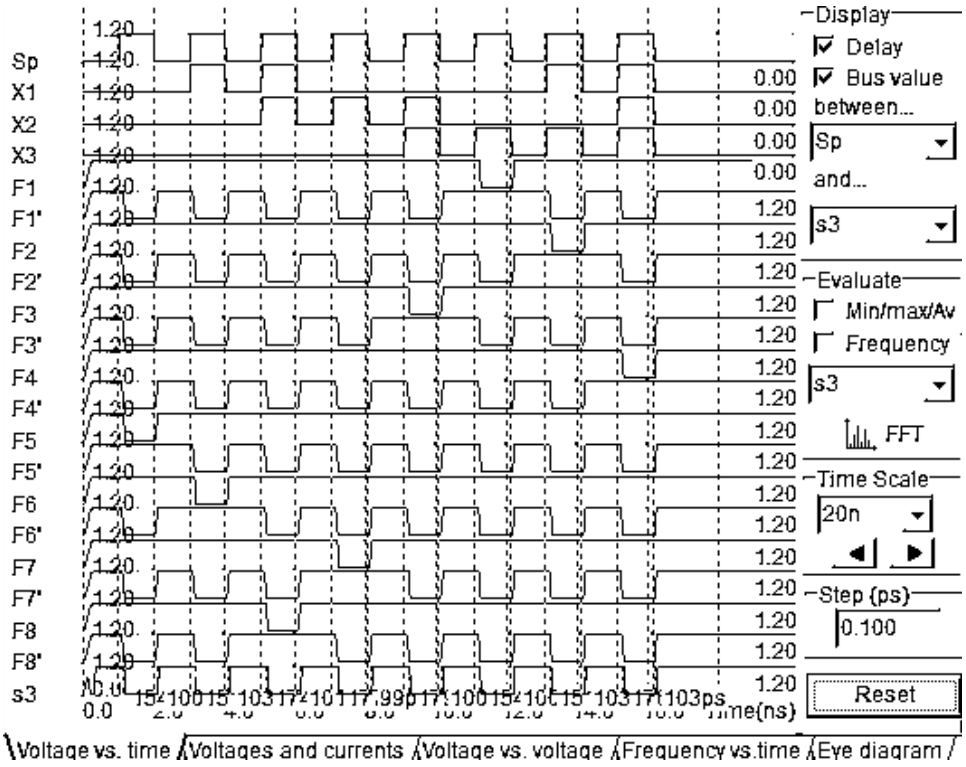



Рисунок П5. 10 – Результат моделирования элемента DC LUT-ST трех переменных (с восстановлением уровня сигналов)

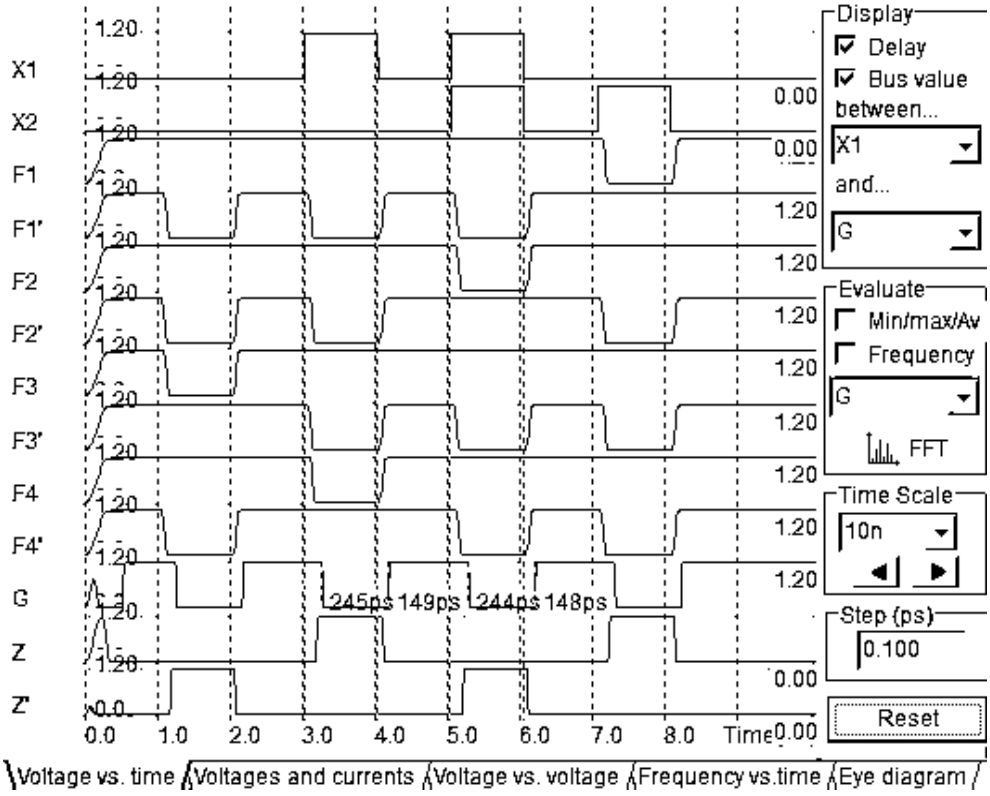



Рисунок П5. 11 – Результат моделирования элемента DC LUT-ST двух переменных с блоком настройки на реализацию функции «исключающее ИЛИ»

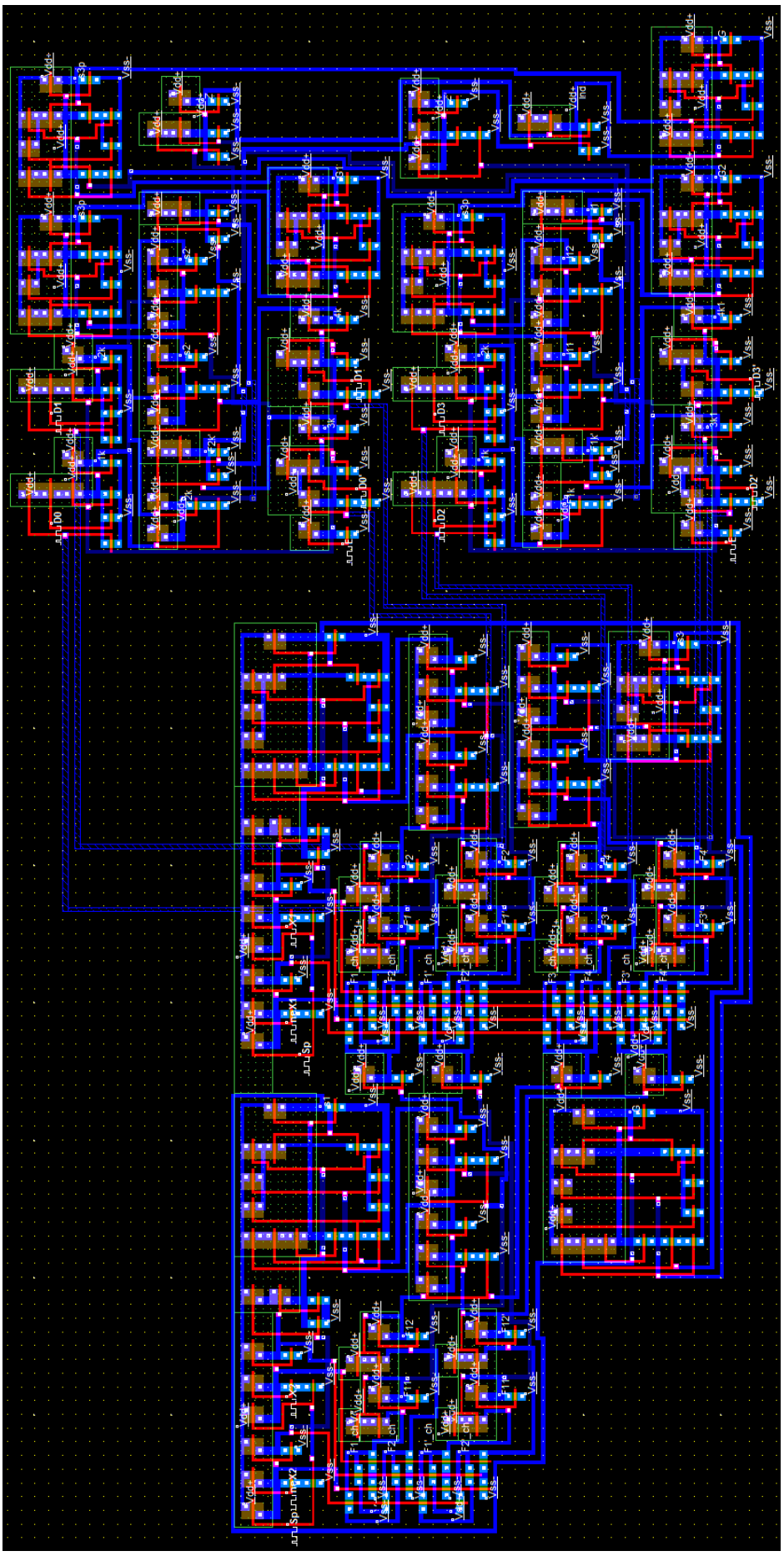


Рис.П5. 12 – Топология элемента DC LUT-ST двух переменных с блоком настройки на реализацию функции «исключающее ИЛИ»

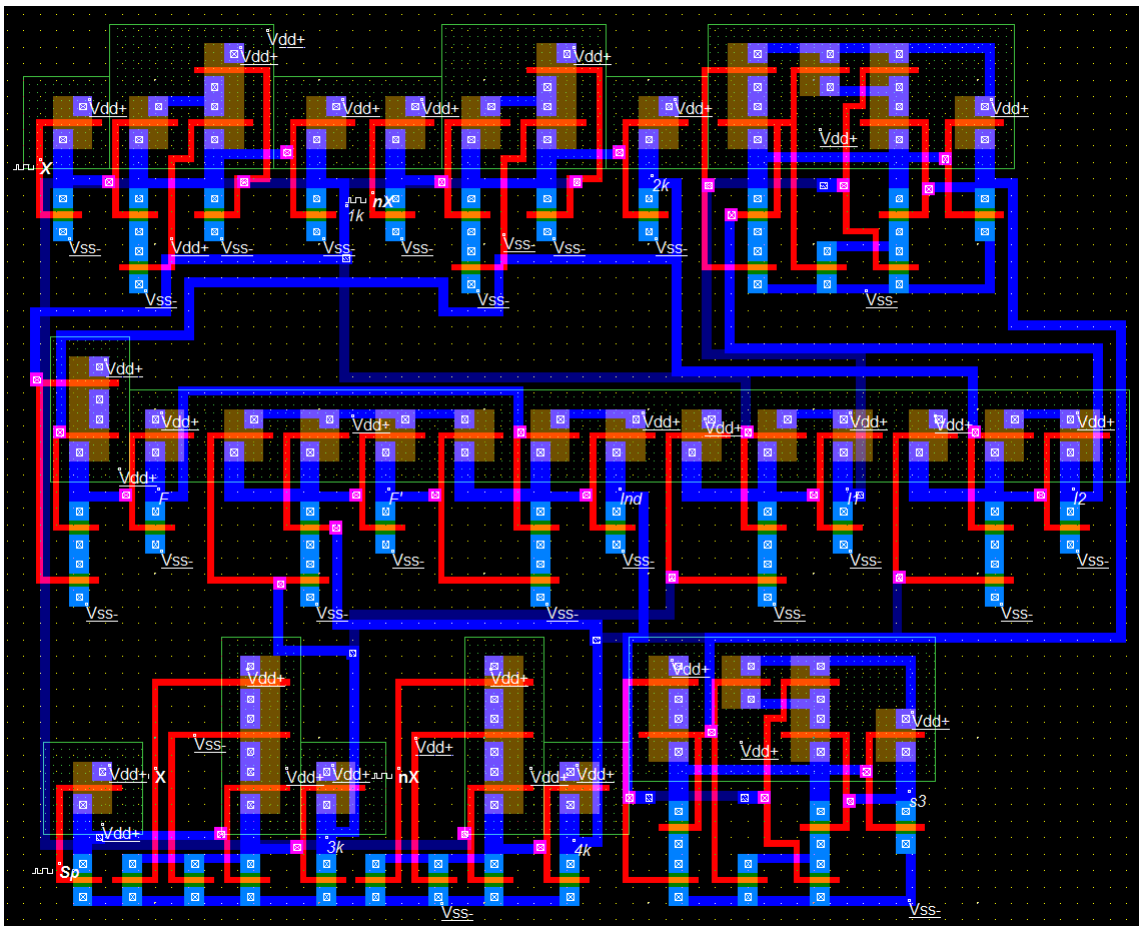



Рисунок П5. 13 – Топология блока конъюнкций одного разряда на основе библиотечных элементов

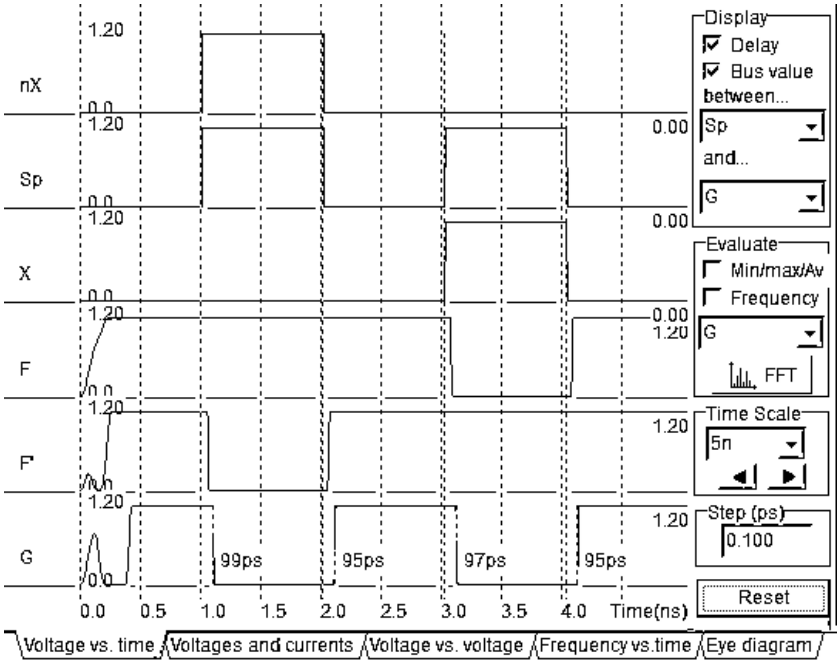



Рисунок П5. 14 – Результат моделирования блока конъюнкций одного разряда на основе библиотечных элементов

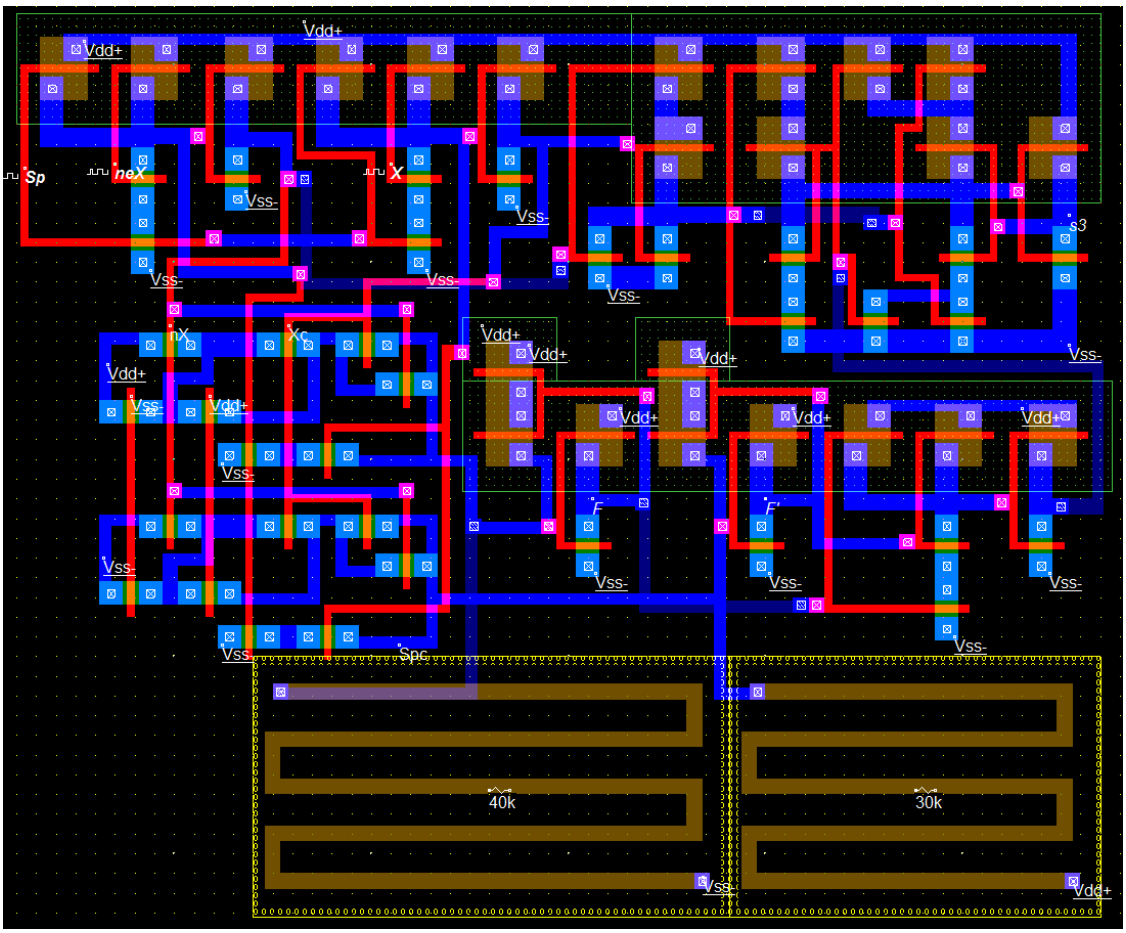



Рисунок П5. 15 – Топология блока конъюнкций одного разряда с подтягивающими резисторами

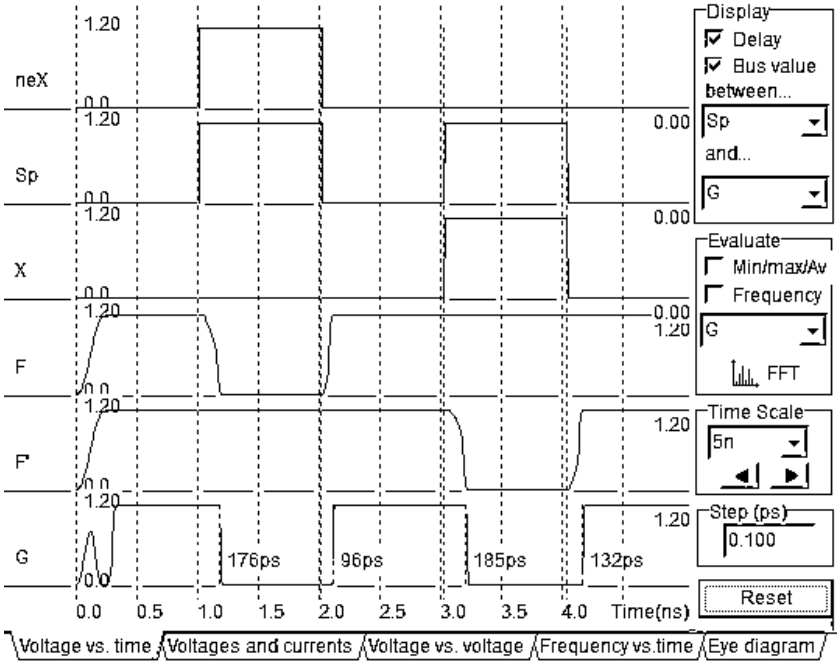



Рисунок П5. 16 – Результат моделирования блока конъюнкций одного разряда с подтягивающими резисторами

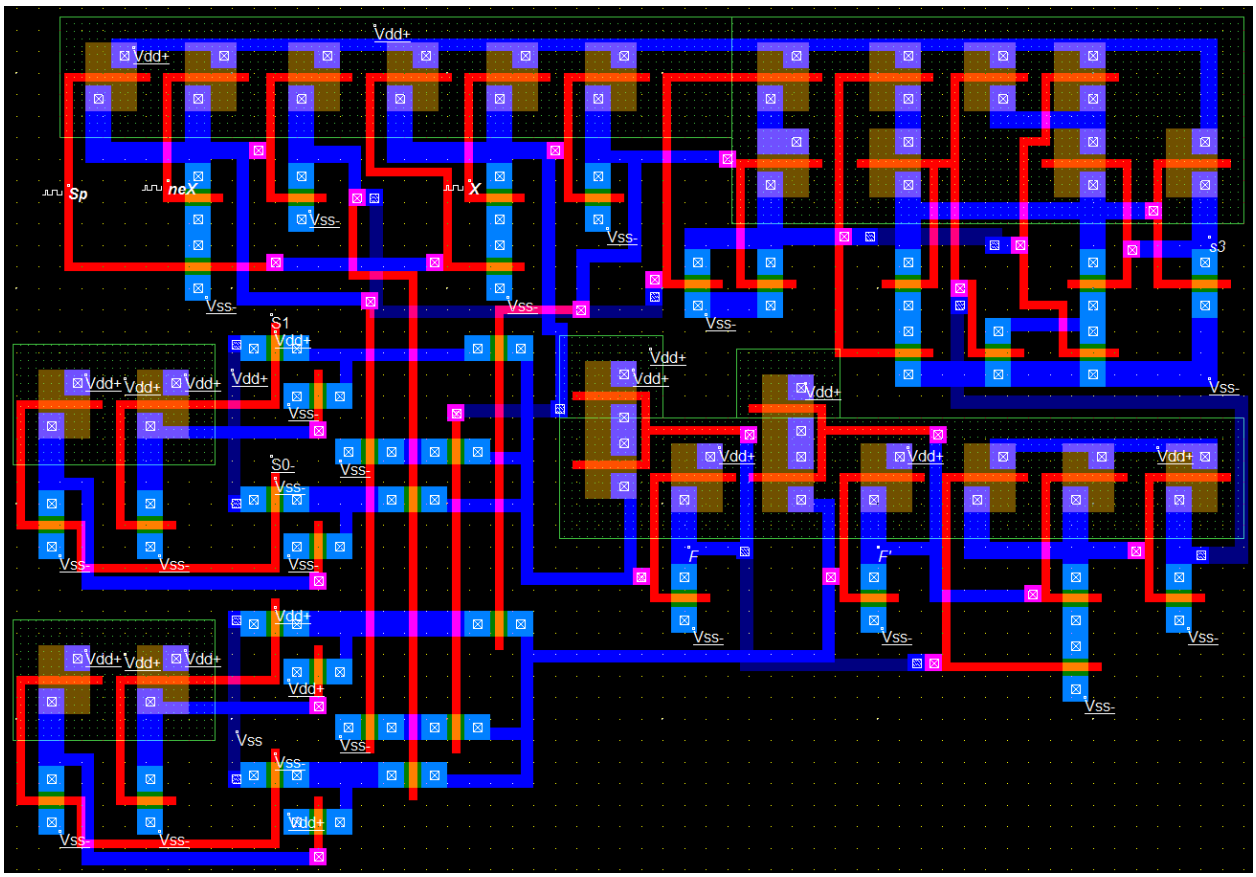



Рисунок П5. 17 – Топология блока конъюнкций одного разряда с транзисторами ортогональности

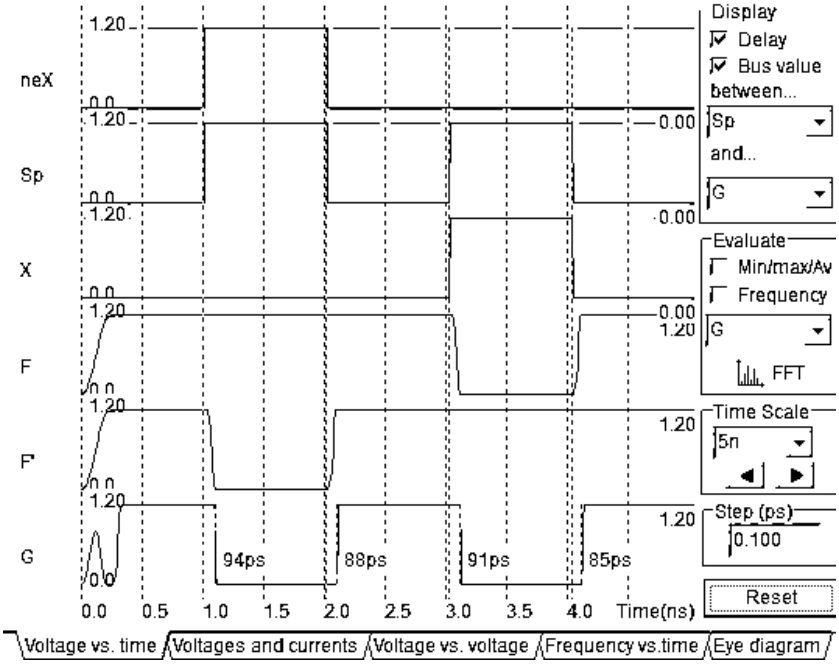



Рисунок. 18 – Результат моделирования блока конъюнкций одного разряда с транзисторами ортогональности

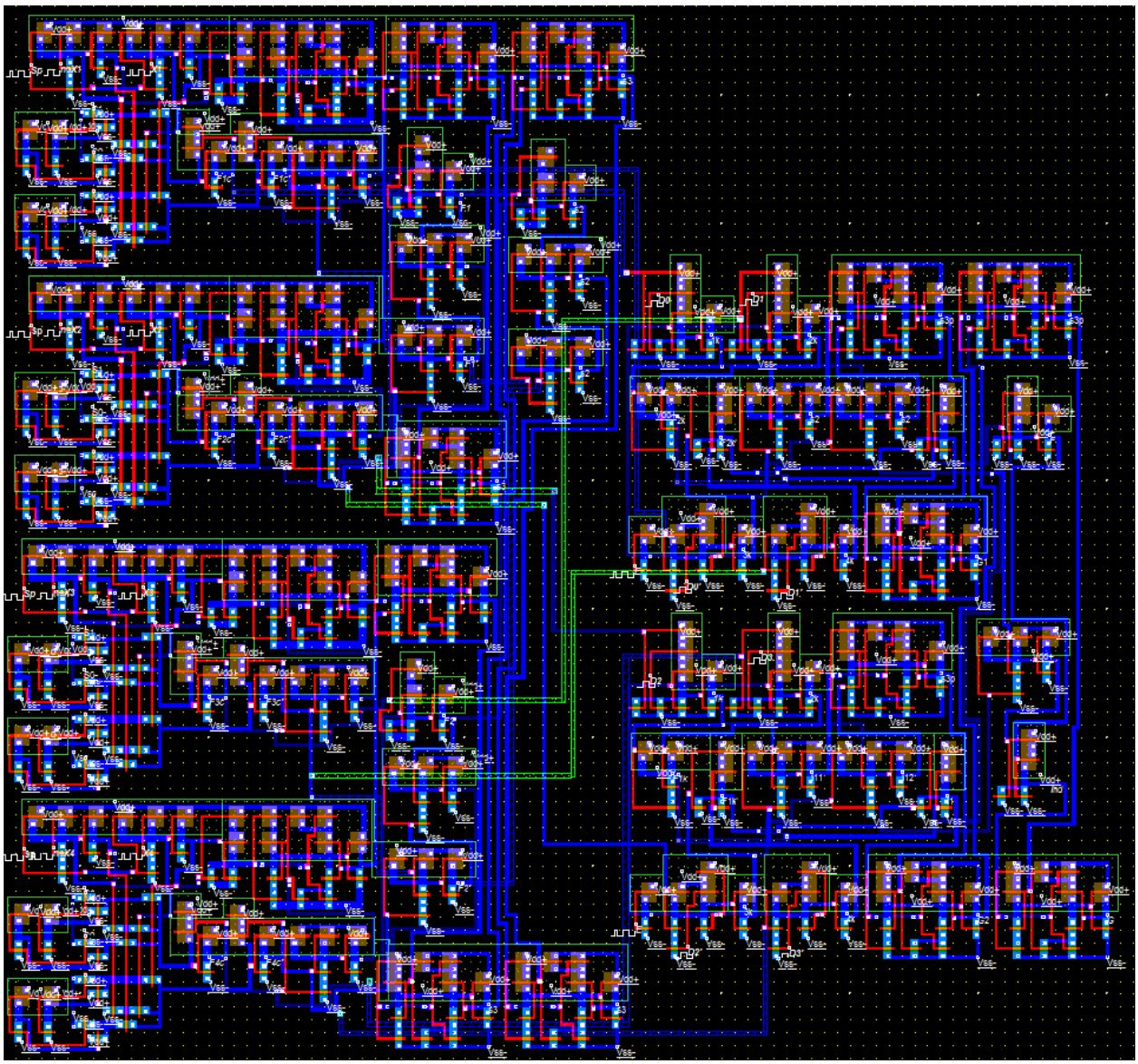


Рисунок П5. 19 – Топология блока конъюнкций двух разрядов с транзисторами ортогональности и блоком настройки на реализацию функции «исключающее ИЛИ»

Результат моделирования показан на рисунке 3.43.

**Приложение №6. Схемы электрические принципиальные в САПР «Multisim»**

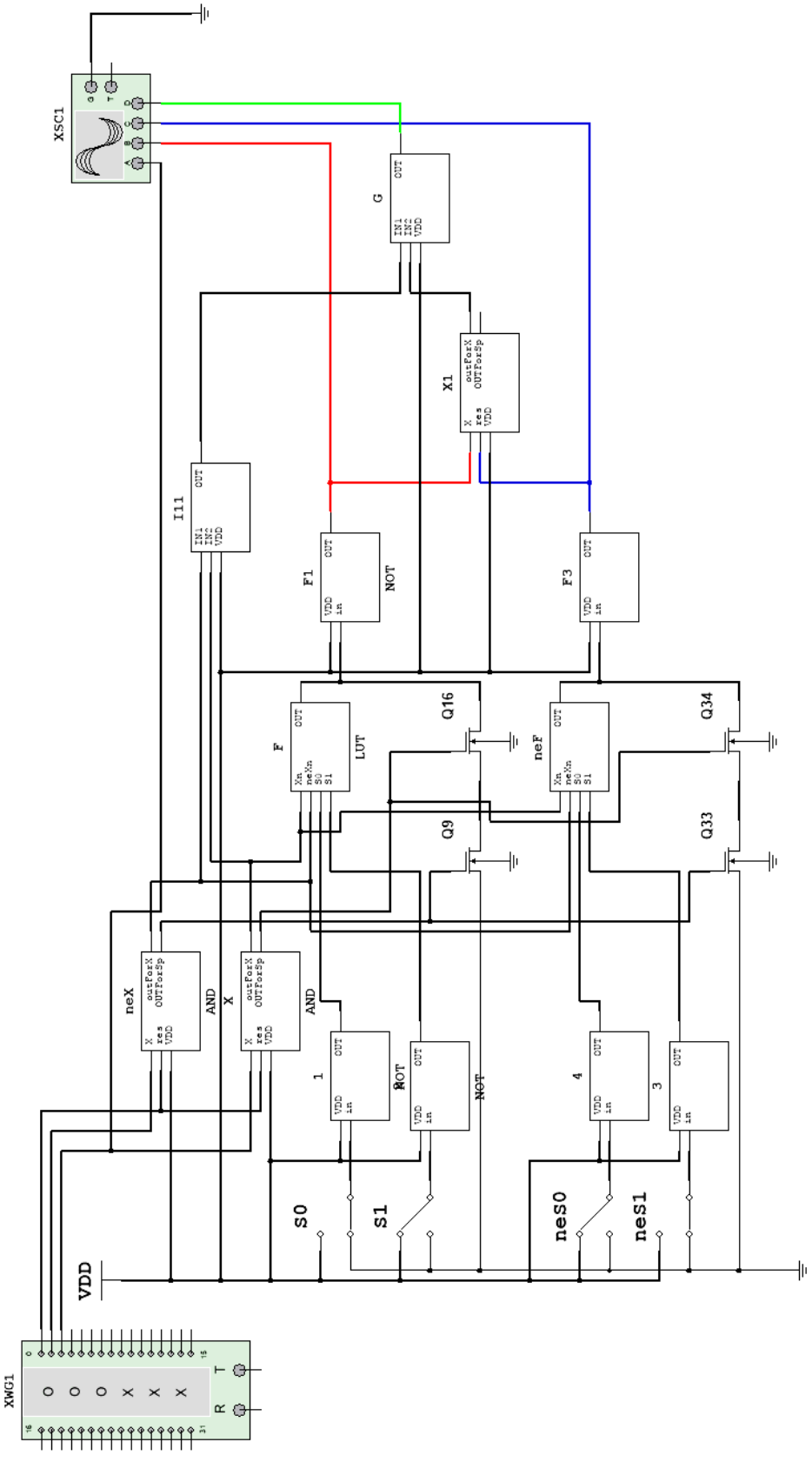



Рис. П6.1 – Схема электрическая принципиальная LUT-ST одной переменной

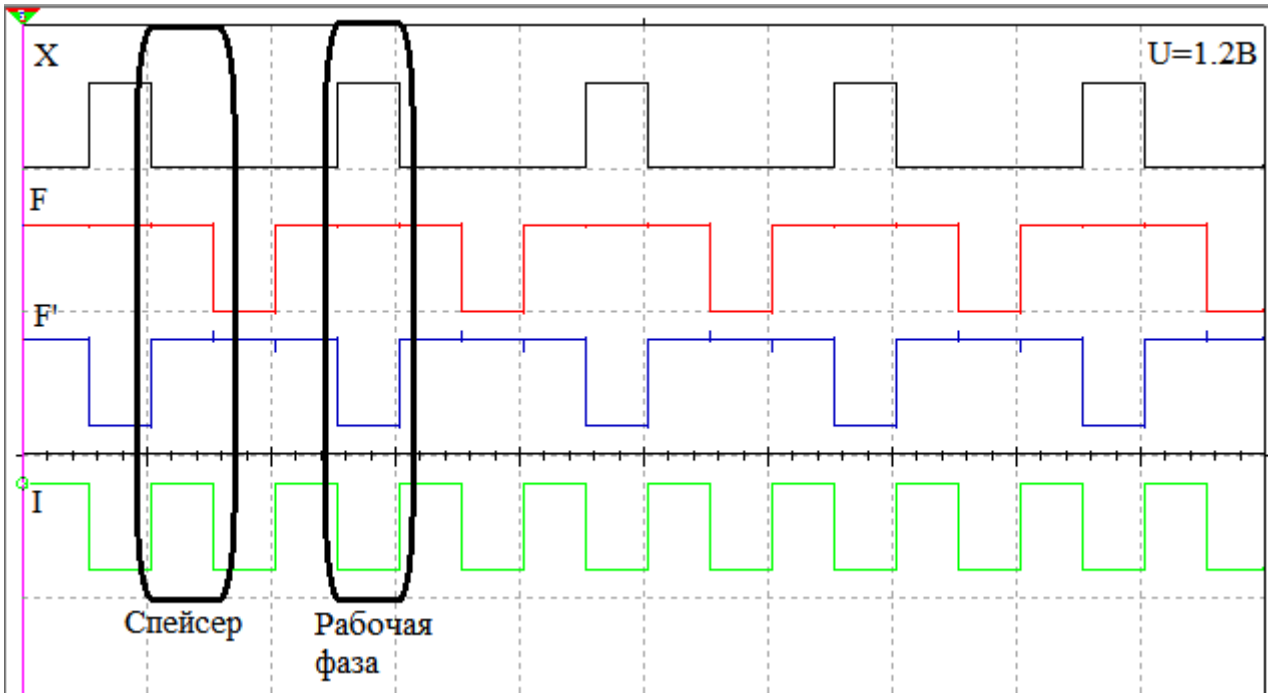



Рисунок П6.2 – Результат моделирования элемента LUT-ST одной переменной

Таблица П6.1 – Таблица истинности элемента LUT-ST одной переменной

| № | S0S1 | X | F | F’ | I |
|---|---|---|---|---|---|
| 1 | 00 | 0 | 0 | 1 | 0 |
| | | 1 | 0 | 1 | 0 |
| 2 | 01 | 0 | 0 | 1 | 0 |
| | | 1 | 1 | 0 | 0 |
| 3 | 10 | 0 | 1 | 0 | 0 |
| | | 1 | 0 | 1 | 0 |
| 4 | 11 | 0 | 1 | 0 | 0 |
| | | 1 | 1 | 0 | 0 |

Продолжение приложения №6

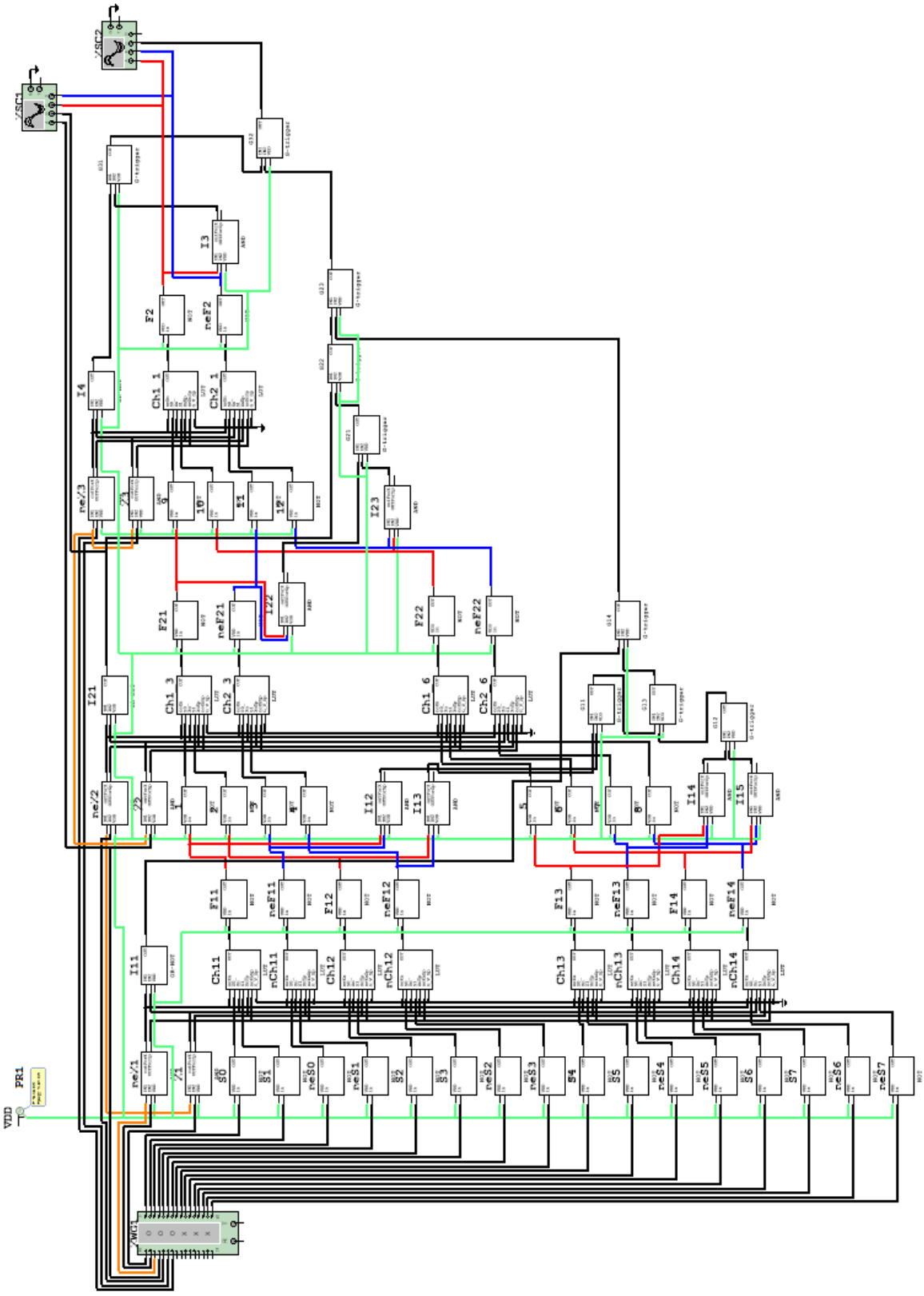


Рис. П6.3 – Схема электрическая принципиальная LUT-ST трех переменных

Таблица П6.2 – Таблица истинности элемента 2LUT-ST с нулевым спейсером (Начало)

| № | S0S1S2S3 | X1X2 | F | F’ | G |
|---|---|---|---|---|---|
| 1 | 0000 | 00 | 0 | 1 | 0 |
| | | 01 | 0 | 1 | 0 |
| | | 11 | 0 | 1 | 0 |
| | | 10 | 0 | 1 | 0 |
| 2 | 0001 | 00 | 1 | 0 | 0 |
| | | 01 | 0 | 1 | 0 |
| | | 11 | 0 | 1 | 0 |
| | | 10 | 0 | 1 | 0 |
| 3 | 0010 | 00 | 0 | 1 | 0 |
| | | 01 | 0 | 1 | 0 |
| | | 11 | 0 | 1 | 0 |
| | | 10 | 1 | 0 | 0 |
| 4 | 0011 | 00 | 1 | 0 | 0 |
| | | 01 | 0 | 1 | 0 |
| | | 11 | 0 | 1 | 0 |
| | | 10 | 1 | 0 | 0 |
| 5 | 0100 | 00 | 0 | 1 | 0 |
| | | 01 | 1 | 0 | 0 |
| | | 11 | 0 | 1 | 0 |
| | | 10 | 0 | 1 | 0 |
| 6 | 0101 | 00 | 1 | 0 | 0 |
| | | 01 | 1 | 0 | 0 |
| | | 11 | 0 | 1 | 0 |
| | | 10 | 0 | 1 | 0 |
| 7 | 0110 | 00 | 0 | 1 | 0 |
| | | 01 | 1 | 0 | 0 |
| | | 11 | 0 | 1 | 0 |
| | | 10 | 1 | 0 | 0 |
| 8 | 0111 | 00 | 1 | 0 | 0 |
| | | 01 | 1 | 0 | 0 |
| | | 11 | 0 | 1 | 0 |
| | | 10 | 1 | 0 | 0 |
| 9 | 1000 | 00 | 0 | 1 | 0 |
| | | 01 | 0 | 1 | 0 |
| | | 11 | 1 | 0 | 0 |
| | | 10 | 0 | 1 | 0 |
| 10 | 1001 | 00 | 1 | 0 | 0 |
| | | 01 | 0 | 1 | 0 |

Таблица П6.2 (Окончание)

| № | S0S1S2S3 | X1X2 | F | F’ | G |
|---|---|---|---|---|---|
| 10 | 1001 | 11 | 1 | 0 | 0 |
| | | 10 | 0 | 1 | 0 |
| 11 | 1010 | 00 | 0 | 1 | 0 |
| | | 01 | 0 | 1 | 0 |
| | | 11 | 1 | 0 | 0 |
| | | 10 | 1 | 0 | 0 |
| 12 | 1011 | 00 | 1 | 0 | 0 |
| | | 01 | 0 | 1 | 0 |
| | | 11 | 1 | 0 | 0 |
| | | 10 | 1 | 0 | 0 |
| 13 | 1100 | 00 | 0 | 1 | 0 |
| | | 01 | 1 | 0 | 0 |
| | | 11 | 1 | 0 | 0 |
| | | 10 | 0 | 1 | 0 |
| 14 | 1101 | 00 | 1 | 0 | 0 |
| | | 01 | 1 | 0 | 0 |
| | | 11 | 1 | 0 | 0 |
| | | 10 | 0 | 1 | 0 |
| 15 | 1110 | 00 | 0 | 1 | 0 |
| | | 01 | 1 | 0 | 0 |
| | | 11 | 1 | 0 | 0 |
| | | 10 | 1 | 0 | 0 |
| 16 | 1111 | 00 | 1 | 0 | 0 |
| | | 01 | 1 | 0 | 0 |
| | | 11 | 1 | 0 | 0 |
| | | 10 | 1 | 0 | 0 |

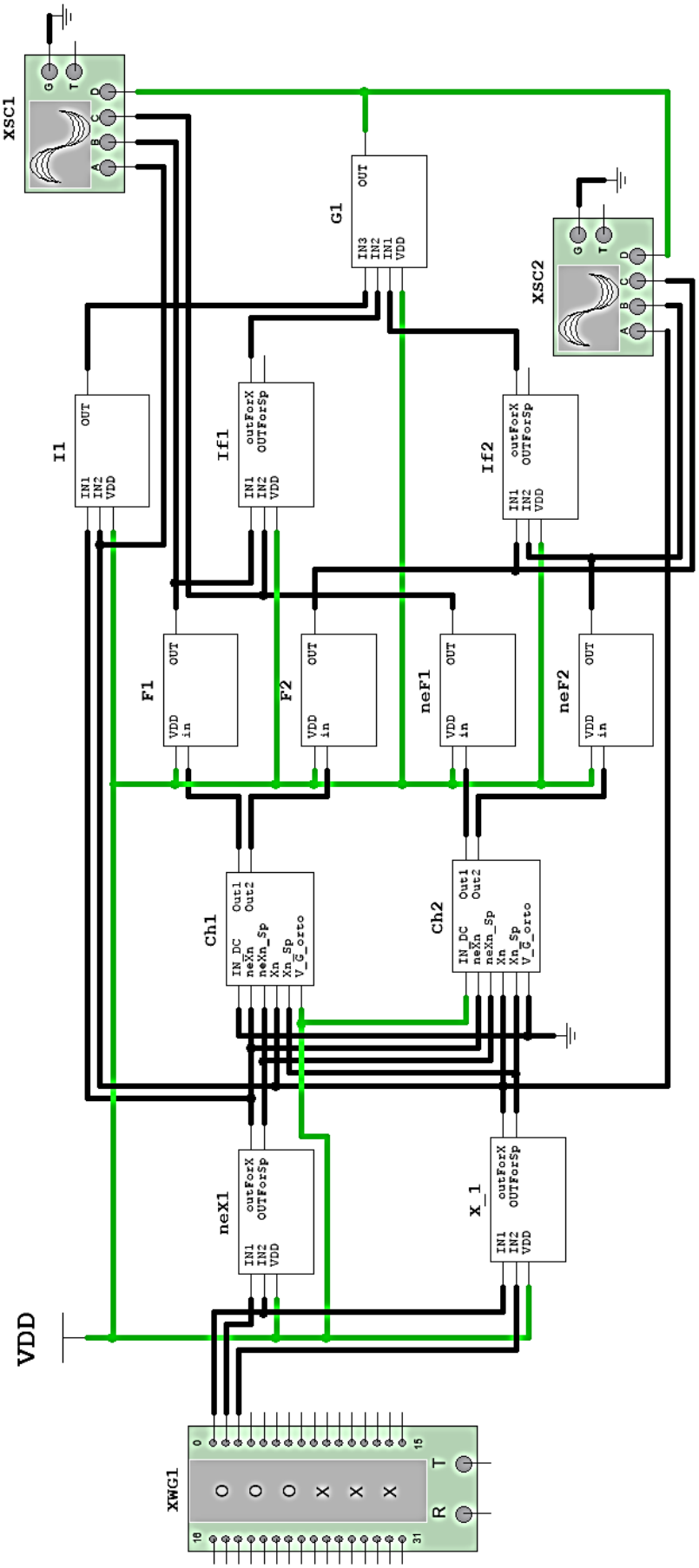



Рис. П6.4 – Схема электрическая принципиальная DC LUT-ST одной переменной

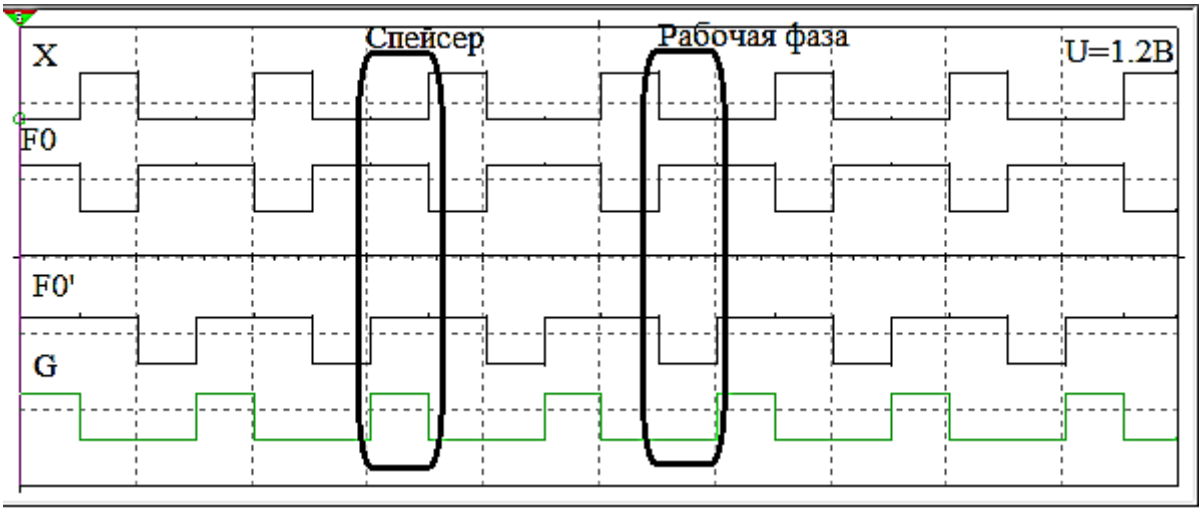



Рисунок П6.5 – Результат моделирования элемента DC LUT-ST зависимость каналов F0, F0’ от входной переменной и выход с общего индикатора

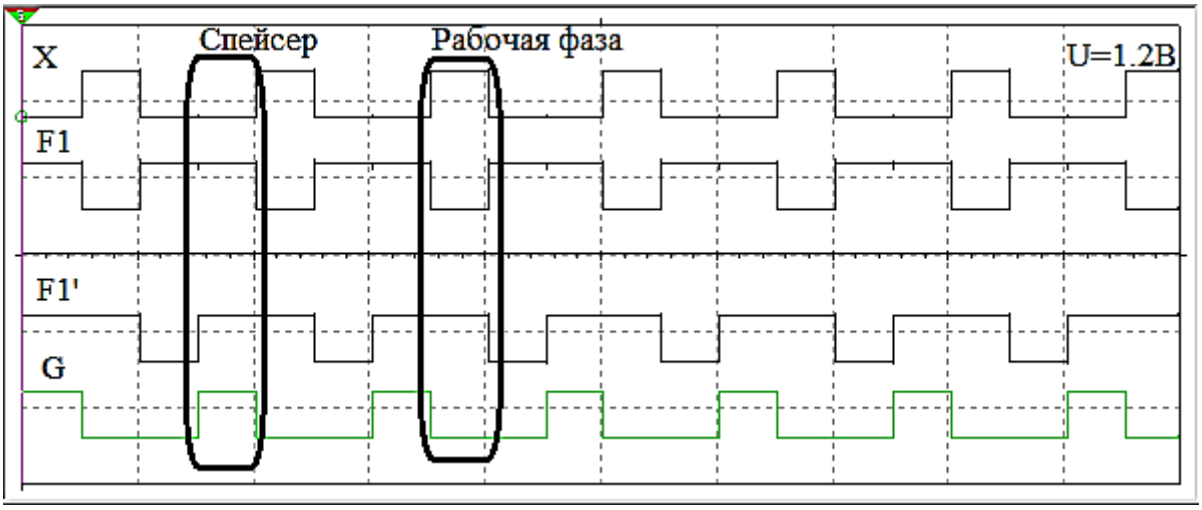



Рисунок П6.6 – Результат моделирования элемента DC LUT-ST зависимость каналов F1, F1’ от входной переменной и выход с общего индикатора

Окончание приложения №6

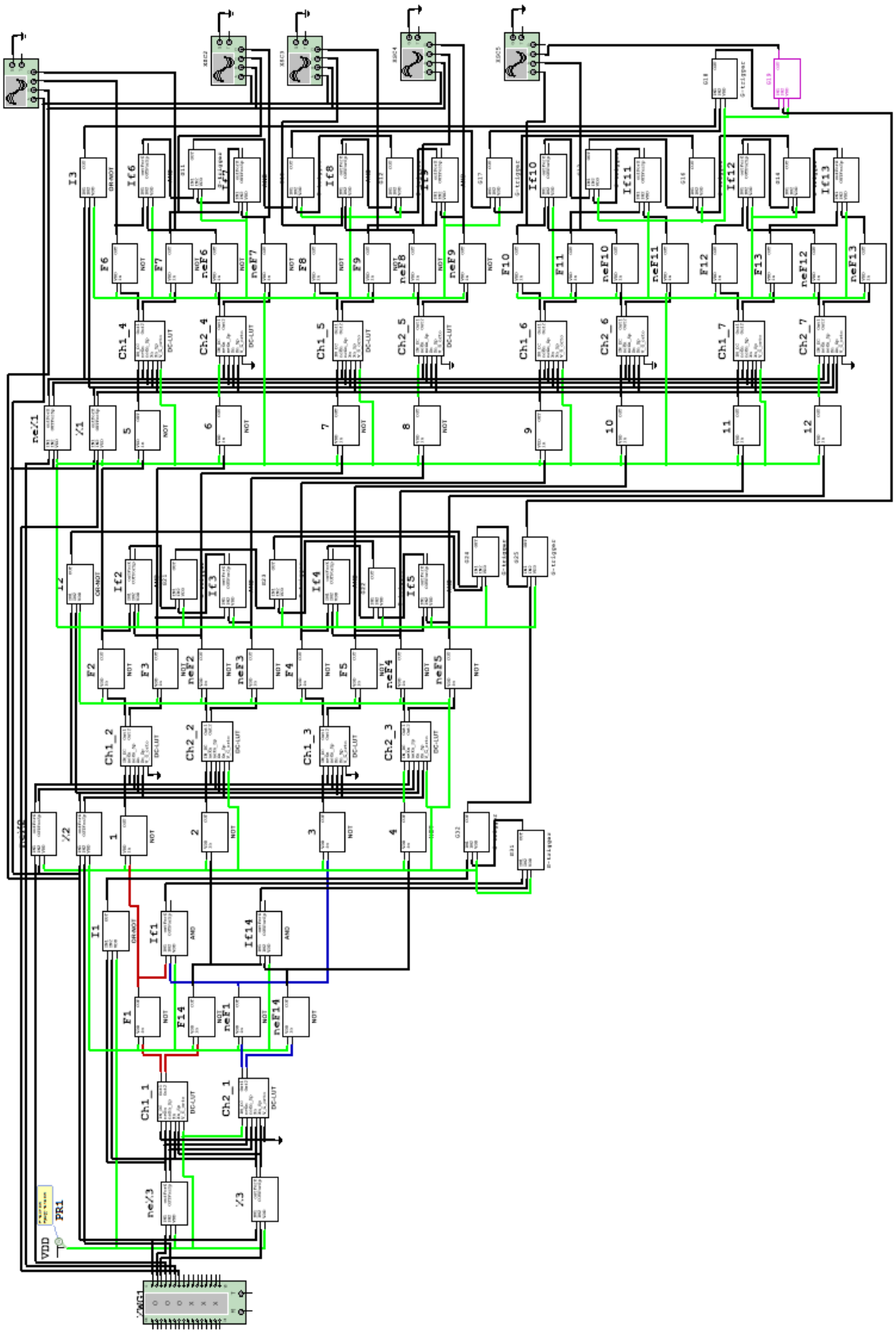


Рис. П6.7 – Схема электрическая принципиальная DC LUT-ST трех переменных