МИНИСТЕРСВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования
КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ ФИЗИКИ

ОТДЕЛЕНИЕ РАДИОФИЗИКИ И ИНФОРМАЦИОННЫХ СИСТЕМ

КАФЕДРА РАДИОЭЛЕКТРОНИКИ

И. А. Насыров

ЛАБОРАТОРНЫЙ ПРАКТИКУМ
ОСНОВЫ ПОСТРОЕНИЯ ЦИФРОВЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ

Часть 1. Функции алгебры–логики и синтез логических схем

Учебно–методическое пособие

Казань — 2012

УДК 631.396.6

Печатается по решению
редакционно—издательского совета
Отделения радиофизики и информационных систем
Института физики
Казанского (Приволжского) федерального университета

Рецензент
Кандидат физ.–мат. наук, доцент Г. В. Таюрская

И. А. Насыров
Лабораторный практикум. Основы построения цифровых логических устройств. Часть 1: Функции алгебры–логики и синтез логических схем. Учебно–методическое пособие. — Казань: Казанский университет, 2012. — 88 с.

Лабораторный практикум по основам построения цифровых логических устройств предназначен для самостоятельного освоения студентами радиофизических и технических специальностей.

В части 1 рассмотрены способы математического описания работы цифровых логических устройств, а также основы синтеза цифровых устройств с заданными техническими характеристиками.

Лабораторный практикум базируется на использовании образовательной платформы National Instruments ELVIS II$^+$ совместно с отладочной платой Digital Electronics FPGA Board. В процессе выполнения лабораторных работ студенты получат начальные навыки проектирования электронных устройств с использованием технологии LabVIEW.

Содержание

Введение

В учебно–методическом пособии представлен лабораторный практикум по основам построения цифровых логических устройств. При разработке настоящего лабораторного практикума реализован компетентностный подход, который будет способствовать формированию и развитию профессиональных навыков у обучающихся. Согласно пункта 7.7 Федерального государственного образовательного стандарта третьего поколения (ФГОС-3) по направлению подготовки «011800 — Радиофизика», данный лабораторный практикум отнесен к категории внеаудиторной самостоятельной работы студентов. В связи с этим в учебно–методическом пособии кроме, собственно, описания лабораторных работ дан теоретический материал в объеме, достаточном для выполнения расчетной и практической частей работы. Несмотря на это, при освоении теории будет очень полезно ознакомиться с дополнительной литературой [1, 2, 3].

Выполнение лабораторной работы осуществляется в четыре этапа. Первый этап посвящен ознакомлению с теоретической частью. На втором этапе необходимо выполнить расчетную часть работы. На третьем этапе осуществляется практическая реализация заданного цифрового логического устройства. На четвертом этапе в процессе обсуждения результатов выполнения работы с преподавателем выставляется оценка за выполнение лабораторной работы. Каждая лабораторная работа снабжена специальным **бланком отчета**, который необходимо распечатать и заполнять его по мере выполнения расчетных и практических заданий.

Учебно–методическое пособие состоит из трёх частей.

Часть 1 посвящена способам математического описания работы цифровых логических устройств, а также основам синтеза цифровых устройств с заданными техническими характеристиками.

В части 2 рассмотрены принципы построения и функционирования основных узлов комбинационных логических устройств (автоматов без памяти).

В части 3 рассмотрены особенности построения последовательностых логических устройств (автоматов с памятью).

В результате освоения лабораторного практикума по основам построения цифровых логических устройств обучающийся должен:

знать: основные принципы, законы построения и функционирования цифровых электронных систем, теоретические и экспериментальные методы оценки параметров электронных приборов;

уметь: пользоваться основными методами расчета радиотехнических и электронных систем;

владеть: навыками работы с современным экспериментальным оборудованием, методами обработки данных;

Практическая часть практикума базируется на использовании образовательной платформы National Instruments ELVIS II$^+$ совместно с отладочной платой Digital Electronics FPGA Board. В процессе выполнения лабораторных работ обучающиеся получат начальные навыки проектирования электронных устройств с использованием технологии LabVIEW.

Глава 1

ОБРАЗОВАТЕЛЬНАЯ ПЛАТФОРМА NI ELVIS II$^+$ И ОТЛАДОЧНАЯ ПЛАТА NI DE FPGA BOARD

1.1 Образовательная платформа ELVIS II$^+$

Образовательная платформа NI ELVIS II$^+$ (*National Instruments Educational Laboratory Virtual Instrumentation Suite II$^+$*) представляет собой аппаратно–программный комплекс, в состав которого входит комплект виртуальных измерительных приборов, а также информация, необходимая для работы с этими приборами.

Образовательная платформа для проектирования и создания прототипов NI ELVIS II$^+$ выполнена на базе среды графической разработки NI LabVIEW. Платформа NI ELVIS II$^+$ подключается к ПК посредством высокоскоростного USB-интерфейса. Программное обеспечение NI ELVISmx служит для управления функционированием аппаратных средств NI ELVIS II$^+$ с помощью спроектированных в LabVIEW лицевых панелей (Soft Front Panels – SFPs) следующих измерительных приборов:

- Генератора сигналов произвольной формы (Arbitrary Waveform Generator – ARB).
- Анализатора амплитудно- и фазочастотных характеристик (Bode Analyzer).
- Устройства чтения цифровых данных (Digital Reader).
- Устройства записи цифровых данных (Digital Writer).
- Цифрового мультиметра (Digital Multimeter – DMM).
- Анализатора спектра (Dynamic Signal Analyzer – DSA).
- Функционального генератора сигналов (Function Generator – FGEN).
- Анализатора импеданса (Impedance Analyzer).
- Осциллографа (Oscilloscope – Scope).
- Анализатора вольтамперной характеристики двухполюсников (Two-Wire Current Voltage Analyzer).
- Анализатора вольтамперной характеристики четырехполюсников (Three-Wire Current Voltage Analyzer).

- Регулируемых источников питания (Variable Power Supplies).

Кроме того, в комплект включены экспресс–функции (Express VIs) LabVIEW и наборы функций (Steps) SignalExpress для программирования NI ELVIS II в этих средах. Внешний вид NI ELVIS II$^+$ с поключенным оценочным модулем DE FPGA Board показан на рисунке 1.1.

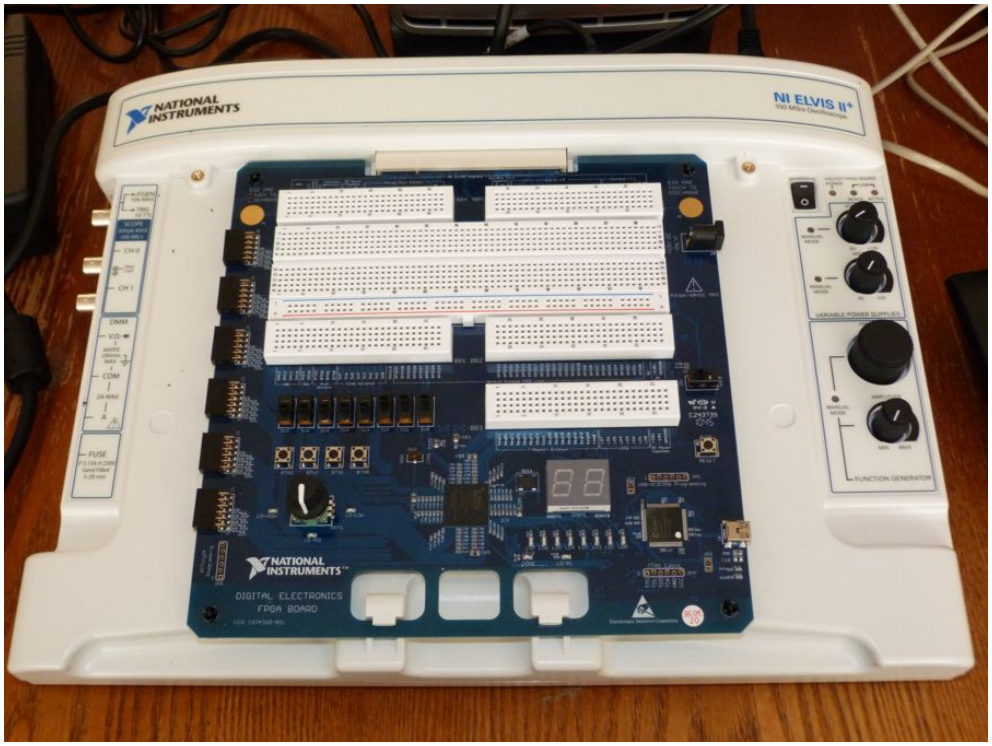


Рис. 1.1. Внешний вид станции ELVIS II$^+$ с отладочной платой DE FPGA Board

NI ELVIS II$^+$ в данной конфигурации эффективен для организации занятий по основам построения цифровых логических устройств. Комплект NI ELVIS II$^+$ предоставляет широкие возможности для измерений и испытаний, необходимых в ходе этих занятий, обеспечивает сохранение получаемых данных.

1.2 Отладочная плата Digital Electronics FPGA Board

FPGA — Field programmable gate array (дословно переводится как «массив логических вентилей, программируемых в условиях эксплуатации»), это «сверхбольшая» интегральная схема (СБИС) с массивом логических вентилей, не соединенных между собой функциональными связями в процессе производства. Другими словами, FPGA — это программируемая логическая интегральная схема (ПЛИС).

Отладочная плата NI Digital Electronics FPGA Board [4] — это инструмент разработки электронных схем, построенный на основе микросхемы ПЛИС XC3S500E Xilinx Spartan-3E. Помимо ПЛИС отладочная плата содержит: движковые переключатели; светодиоды; два семисегментных индикатора; нажимные кнопки; нажимную поворотную кнопку для выбора частоты внешнего синхросигнала и светодиоды для отображения выбранного режима работы; разъёмы Diglent Pmod для подключения внешних устройств; интерфейс USB, предназначенный для программирования микросхемы ПЛИС; инструментарий для разработки электронных устройств.

Внешний вид отладочной платы NI Digital Electronics FPGA Board приведен на рис. 1.2.

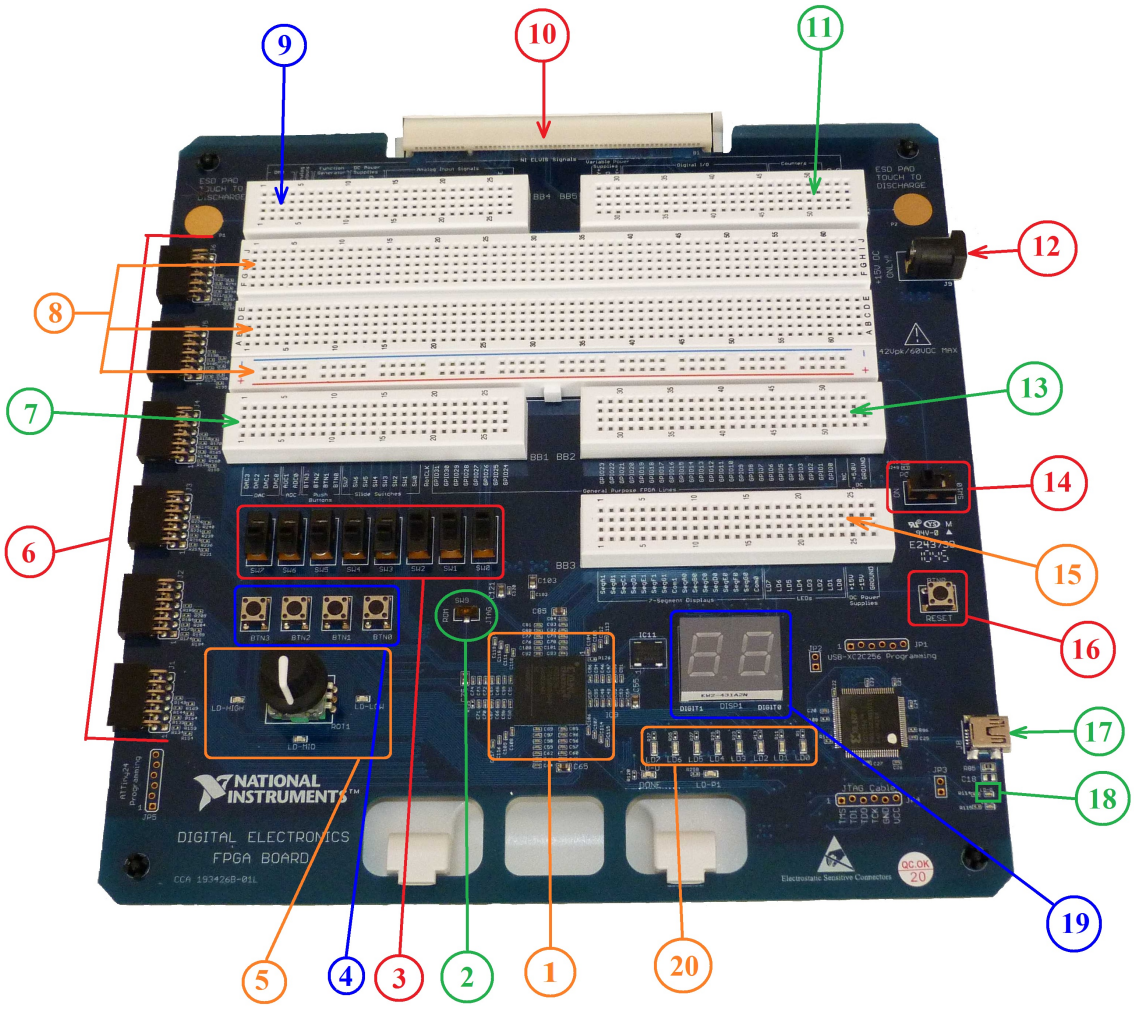


Рис. 1.2. Отладочная плата NI Digital Electronics FPGA Board

1. ПЛИС XC3S500E Xilinx Spartan-3E
2. Переключатель SW9
3. Движковые переключатели (SW0÷SW7)
4. Кнопки (BTN0÷BTN3)
5. Вращающийся нажимной переключатель
6. Разъёмы Digilent Pmod
7. Зона макетирования: сигнальный разъем BB1
8. Разъёмы общего назначения
9. Зона макетирования: сигнальный разъем BB4
10. Разъём для подключения к NI ELVIS II$^+$
11. Зона макетирования: сигнальный разъем BB5
12. Разъём подключения источника питания
13. Зона макетирования: сигнальный разъем BB2
14. Выключатель питания
15. Зона макетирования: сигнальный разъем BB3
16. Кнопка сброса (Reset)
17. Разъём USB
18. Светодиод LD–G
19. Семисегментные индикаторы
20. Светодиоды (LD0÷LD7)

Отладочную плату NI Digital Electronics FPGA Board можно подключить в двух различных режимах:

- **Автономный режим (Stand-alone Mode)**[1]. В этом режиме отладочная плата работает в качестве автономного или независимого устройства. Для подключения в автономном режиме необходим независимый источник постоянного тока (15 В, 650 мА), который подключается к разъему 12 на рис. 1.2. Разъёмы ВВ4 и ВВ5 (9, 11 на рис. 1.2) в этом режиме могут использоваться для решения задач макетирования.

- **Режим NI ELVIS** используется для подключения отладочной платы NI Digital Electronics FPGA Board к образовательной платформе NI ELVIS. В этом режиме отладочная плата NI Digital Electronics FPGA Board используется в качестве дополнительной макетной платы для NI ELVIS II$^+$. Поключение оценочной платы рассмотрено в разделе 1.3.

Как правило, проектирование устройств на базе ПЛИС выполняется с помощью специализированных программных инструментов разработки и языков программирования, например, *VHDL* или *Verilog*, освоение которых требует значительных усилий. Модуль LabVIEW FPGA дополняет базовый пакет LabVIEW возможностью реализации целевых систем с ПЛИС на платформе реконфигурируемого ввода/вывода National Instruments Reconfigurable I/O.

1.3 Подключение отладочной платы DE FPGA Board в режиме NI ELVIS

В режиме NI ELVIS отладочная плата NI Digital Electronics FPGA Board может быть использована как дополнительная макетная плата к рабочей станции NI ELVIS II$^+$.

1.3.1 Подключение электропитания

Стенд NI ELVIS II$^+$ имеет два последовательно подключенных выключателя. Первый, основной, расположен на задней панели, как это показано на левой панели рис. 1.3. При помощи этого выключателя подается общее питание на стенд.

Второй выключатель расположен в верхнем правом углу стенда, как показано на правой панели рис. 1.3. С помощью этого выключателя подается питание непосредственно на оценочный модуль DE FPGA Board.

Рядом со вторым выключателем расположены световые индикаторы статуса:

- **Индикатор POWER** -–– сигнализирует о том, что подано питание на оценочный модуль DE FPGA Board. Здесь следует заметить, что на самой плате имеется собственный выключатель питания (14 на рис. 1.2), который используется в автономном режиме подключения, в режиме NI ELVIS данный выключатель всегда должен быть установлен в положение «ON».

- **Индикаторы USB.** Их состояния приведены в таблице 1.1.

  – **READY** –– показывает, что оборудование NI ELVIS II$^+$ сконфигурировано должным образом и готово к соединению с компьютером.

  – **ACTIVE** –– показывает активность USB соединения с компьютером.

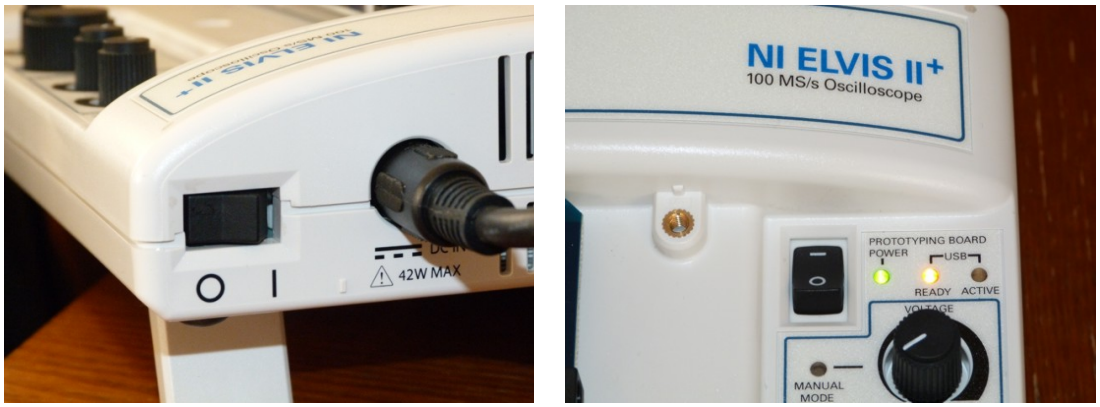


Рис. 1.3. Включение питания на платформе NI ELVIS II$^+$

Таблица 1.1. Состояния индикаторов USB рабочей станции

Индикатор ACTIVE	Индикатор READY	Описание
Выключен		Главный источник питания выключен.
Желтый	Выключен	Нет соединения с компьютером. Убедитесь, что USB–кабель подключен.
Выключен	Зеленый	Станция подключена по протоколу USB 2.0.
Выключен	Желтый	Станция подключена к высокоскоростному USB–порту.
Зеленый	Зеленый или желтый	Выполняется соединение.

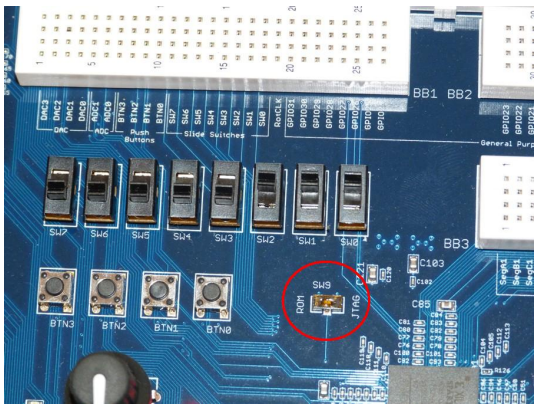
Внимание! Перед установкой или извлечением оценочной платы из рабочей станции убедитесь, что питание от платы отключено.

1.3.2 Выбор способа загрузки ПЛИС

На отладочной плате DE FPGA Board установлена ПЛИС Xilinx XC3S500E-4FTG256C, имеющая электронно–стираемое программируемое постоянное запоминающее устройство (ЭСППЗУ)[2] емкостью 4 Мбит для хранения микропрограммы конфигурации ПЛИС. Этот способ загрузки ПЛИС устанавливается *по умолчанию*. Альтернативным способом является загрузка ПЛИС через USB–JTAG программатор.

Выбор режима загрузки ПЛИС осуществляется при помощи движкового переключателя SW9 (2 на рис. 1.2).

[1] Данный режим в настоящем учебно–методическом пособии не рассматривается.
[2] Более употребимая аббревиатура EEPROM (electronically erasable programmable read-only memory).



Внимание! В лабораторном практикуме способ загрузки с EEPROM использоваться НЕ БУДЕТ. Поэтому движковый переключатель SW9 ВСЕГДА должен быть установлен в положение JTAG.

В случае, когда движковый переключатель установлен в положение JTAG, микросхема ожидает загрузки конфигурации через интерфейс USB–JTAG. В микросхему загружается конфигурация по умолчанию, которая заменяется затем конфигурацией, загружаемой через интерфейс USB–JTAG. Загруженная конфигурация сохраняется до выключения питания, сброса микросхемы ПЛИС (кнопка 16 на рис. 1.2) или загрузки новой конфигурации.

Для связи компьютера с отладочной платой и загрузки микропрограммы конфигурации в ПЛИС используется USB–кабель, который подключается к специальному входу на DE FPGA Board (17 на рис.1.2), как показано на рис.1.4.

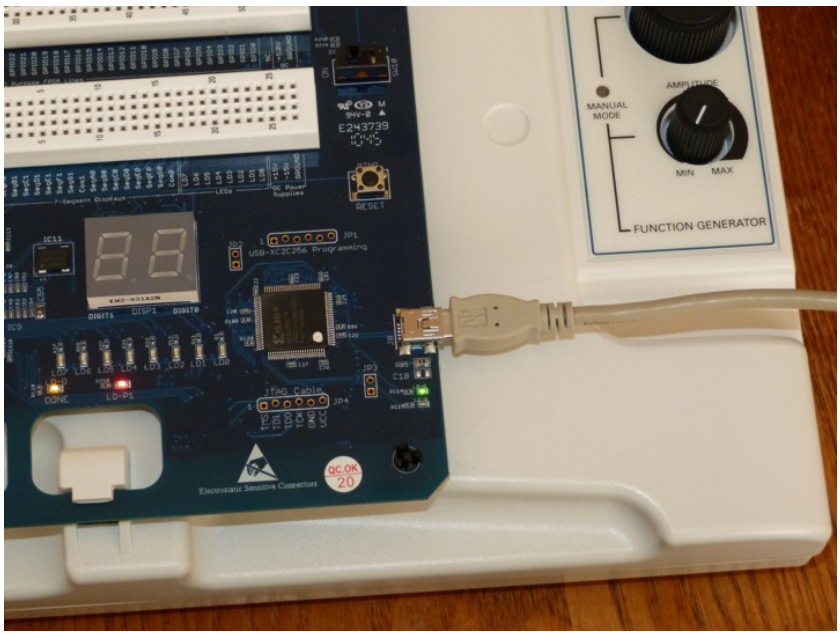


Рис. 1.4. Подключение JTAG–программатора. Горящий зеленым светодиод LD–G сигнализирует о том, что связь с компьютером установлена

1.4 Зона макетирования

Отладочная плата NI Digital Electronics FPGA Board содержит две зоны макетирования:

- Сигнальная зона макетирования (7, 9, 11, 13, 15 на рис. 1.2).
- Зона макетирования общего назначения (8 на рис. 1.2).

1.4.1 Сигнальная зона макетирования

ВВ1 (7 на рис. 1.2) –– зона макетирования для подключения к ЦАП, АЦП, кнопкам, движковым переключателям, внешнему синхросигналу и линиям общего назначения ПЛИС.

ВВ2 (13 на рис. 1.2) — зона макетирования для подключения к линиям общего назначения ПЛИС и источникам питания.

ВВ3 (15 на рис. 1.2) — зона макетирования для управления семисегментным индикатором на два знакоместа, светодиодами, подключения к источникам питания.

ВВ4 (9 на рис. 1.2) — зона макетирования для подключения к сигналам NI ELVIS, включая аналоговые входные сигналы, аналоговые выходные сигналы, функциональные генераторы, источники питания, цифровые входные/выходные сигналы.

ВВ5 (11 на рис. 1.2) –– зона макетирования для подключения к сигналам NI ELVIS, включая регулируемые источники питания, цифровые входные/выходные сигналы, выходные сигналы счётчиков, общие точки сигналов.

1.4.2 Зона макетирования общего назначения

Зона макетирования общего назначения состоит из двух разъёмов (8 на рис. 1.2). Эти разъёмы не подключены к каким-либо электронным компонентам платы.

При работе отладочной платы NI Digital Electronics FPGA Board в автономном режиме (Stand–alone Mode) разъёмы ВВ4 и ВВ5 сигнальной зоны макетирования могут быть использованы в качестве разъёмов общего назначения.

1.5 Аппаратная реализация периферийных устройств

В данном разделе рассмотрена аппаратная реализация периферийных устройств, необходимых для выполнения лабораторных работ.

1.5.1 Движковые переключатели

Отладочная плата NI Digital Electronics FPGA Board имеет восемь движковых переключателей, обозначенных SW0÷SW7 (3 на рис. 1.2). На рис. 1.5 показана схема включения движковых переключателей в отладочной плате.

Длительность дребезга контактов составляет 2 нс, в отладочной плате не имеется схем защиты от дребезга контактов. Выходное сопротивление движковых переключателей –– 2 кОм. При нахождении движкового переключателя в верхнем положении (положение «ON») переключатель подключает соответствующую линию к линии с напряжением 3.3 В («лог.1»). При нахождении движкового переключателя в нижнем положении (положение «OFF») переключатель подключает соответствующую линию к общей точке (земле) –– «лог.0».

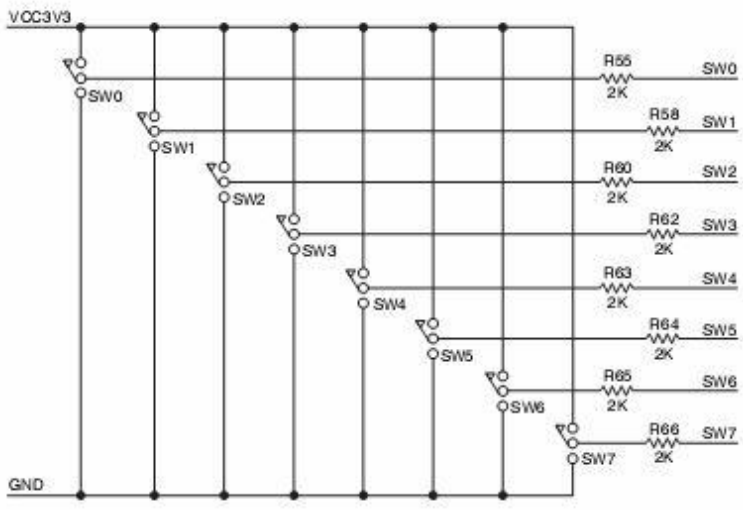


Рис. 1.5. Схема включения движковых переключателей [4]

Линии SW0÷SW7, на которых выставлено напряжение «лог.1» или «лог.0» с помощью движковых переключателей, выведены на разъём BB1 зоны макетирования. Также движковые переключатели напрямую подключены к соответствующим выводам ПЛИС.

1.5.2 Кнопки

Отладочная плата NI Digital Electronics FPGA Board имеет четыре кнопки BTN0÷BTN3 с моментальным установлением контакта (4 на рис. 1.2). На рис. 1.6 показана схема включения кнопок в отладочной плате.

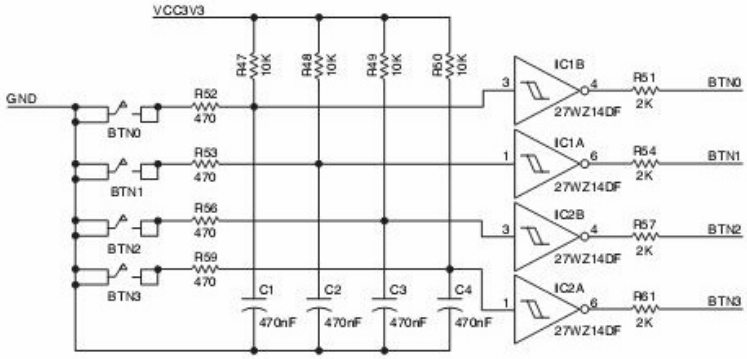


Рис. 1.6. Схема включения кнопок [4]

Нажатие на кнопку соединяет общую точку схемы со входом логического инвертора, поэтому на выходе логического инвертора, включённого в линию, соответствующую той или иной кнопке, появляется напряжение уровня «лог.1». Когда кнопка не нажата, на вход каждого из инверторов подаётся напряжение уровня «лог.1» с источника питания, что вызывает появление на выходе логического инвертора напряжения уровня «лог.0». Схема защиты от дребезга контактов состоит из резистора и конденсатора, включенных в линию каждой кнопки.

Линии BTN0÷BTN3, на которых выставлено напряжение «лог.1» или «лог.0» с помощью соответствующих кнопок, выведены на разъём BB1 зоны макетирования. Также кнопки напрямую подключены к соответствующим выводам ПЛИС.

1.5.3 Светодиоды

Отладочная плата NI Digital Electronics FPGA Board имеет восемь дискретных светодиодов LD0÷LD7 с планарными выводами (20 на рис. 1.2). На рис. 1.7 показана схема включения светодиодов в отладочной плате.

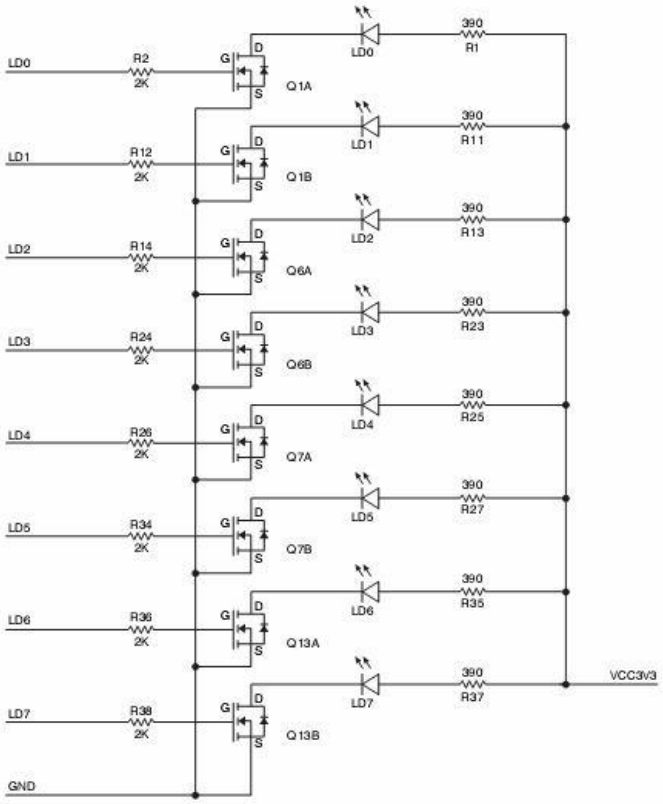


Рис. 1.7. Схема включения светодиодов [4]

К катоду каждого светодиода подключен токозадающий резистор номиналом 390 Ом, к катоду каждого светодиода подключен КМОП-драйвер (см. рис. 1.7).

Линии управления каждым из светодиодов выведены на разъём ВВ3 зоны макетирования. Для того, чтобы вызвать свечение требуемого светодиода, необходимо подать сигнал «лог.1» (3.3 В или 5 В) на линию управления выбранного светодиода.

1.5.4 Двунаправленные линии общего назначения

Отладочная плата NI Digital Electronics FPGA Board имеет 32 двунаправленные линии общего назначения GPIO0÷GPIO31. На рис. 1.8 показана схемотехника этих линий.

Каждая двунаправленная линия общего назначения соединена с ПЛИС через токозадающий резистор номиналом 200 Ом. Двунаправленные линии общего назначения выведены на разъёмы ВВ2 и ВВ3 зоны макетирования.

Двунаправленные линии общего назначения могут быть индивидуально сконфигурированы программным способом на ввод или вывод сигналов. Вводами/выводами сигналов для описываемых линий являются КМОП устройства, поэтому уровнем «лог.1» будет напряжение +3.3 В, допустима подача сигналов с уровнем «лог.1» +5 В.

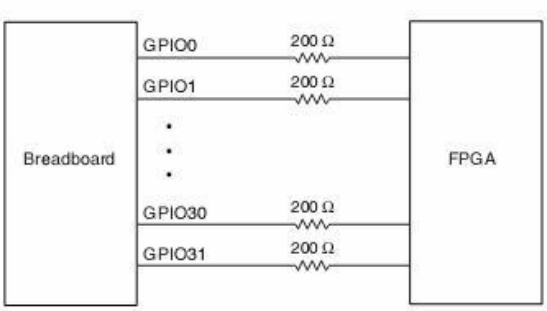


Рис. 1.8. Схемотехника двунаправленных линий общего назначения [4]

1.5.5 Семисегментный индикатор с двумя знакоместами

Отладочная плата NI Digital Electronics FPGA Board имеет семисегментный индикатор DISP 1 с двумя знакоместами, включенный по схеме с общим катодом. На рис. 1.2 показано расположение индикатора на плате (19). На рис. 1.9 показана схема включения индикатора в отладочной плате.

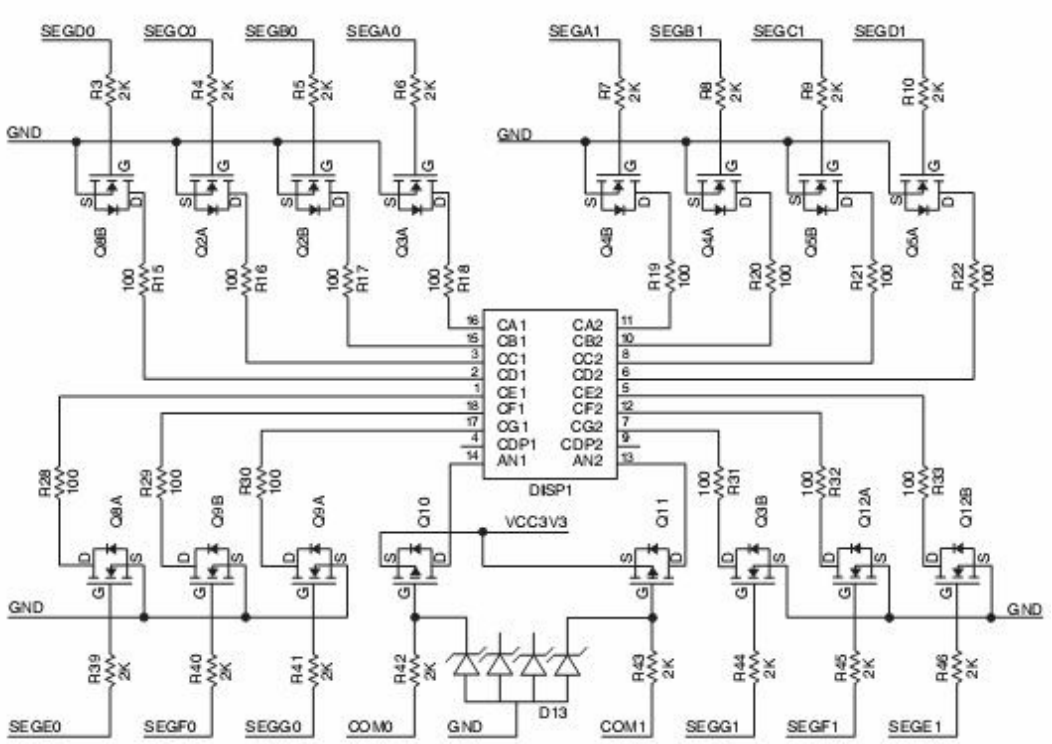


Рис. 1.9. Схема включения семисегментного индикатора с двумя знакоместами [4]

Каждая цифра индикатора состоит из семи сегментов, в каждый из которых встроен светодиод. На рис. 1.9 светодиоды сегментов, составляющие нулевую цифру индикатора обозначены как SEGx0, первую цифру индикатора –– SEGx1.

Схема включения индикатора позволяет вызвать свечение каждого светодиода независимо, что позволяет расширить количество символов, которые можно отобразить с помощью индикатора. Для того, чтобы вызвать свечение отдельного светодиода, на соответствующую линию управления нужно подать сигнал «лог.1» (3.3 В или 5 В). Линии управления COM0 и COM1 предназначены для выдачи сигнала разрешения/запрета

работы соответствующей цифры индикатора, что позволяет использовать индикатор в мультиплексном режиме. Линии управления светодиодами сегментов и линии управления COM0 и COM1 подсоединены к ПЛИС.

1.5.6 Вращающаяся нажимная кнопка и светодиоды

Отладочная плата NI Digital Electronics FPGA Board имеет вращающуюся нажимную кнопку ROT 1 (5 на рис. 1.2), которая служит для выбора частотного диапазона синхросигнала.

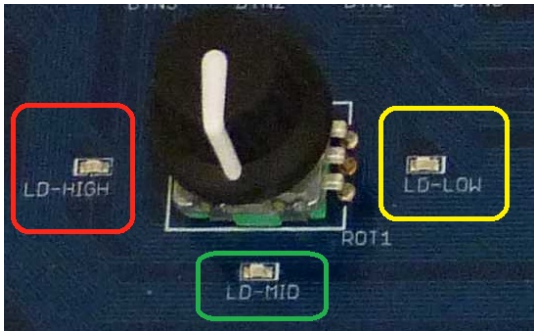


Рис. 1.10. Вращающаяся нажимная кнопка ROT 1

Нажатие на вращающуюся нажимную кнопку выбирает частотный диапазон, указываемый соответствующим светодиодом. На на рис. 1.10 показаны вращающаяся нажимная кнопка и светодиоды. Свечение соответствующего светодиода указывает на выбор одного из следующих частотных диапазонов:

- **LD-LOW** –– при выборе этого диапазона генератор синхросигналов генерирует сигнал в диапазоне 1 Гц ÷ 100 Гц.
- **LD-MID** –– при выборе этого диапазона тактовый генератор генерирует сигнал в диапазоне 100 Гц ÷ 100 кГц.
- **LD-HIGH** –– при выборе этого диапазона генератор синхросигналов генерирует сигнал в диапазоне 100 кГц ÷ 5 МГц.

Конкретное значение частоты генератора в пределах выбранного диапазона задаётся вращением кнопки.

Вывод генератора синхросигналов соединен с линией RotClk, выведённой на разъём BB1 зоны макетирования. Выход генератора синхросигналов не подсоединён к ПЛИС.

Глава 2

ОСНОВЫ АЛГЕБРЫ ЛОГИКИ И ВЫПОЛНЕНИЕ ЛОГИЧЕСКИХ ОПЕРАЦИЙ

2.1 Логические константы и переменные. Операции Булевой алгебры

Для описания алгоритмов работы цифровых устройств необходим соответствующий математический аппарат. Такой аппарат для решения задач формальной логики в середине XIX века разработал ирландский математик Джорж Буль. По его имени математический аппарат и получил название *булевой алгебры*, или *алгебры логики*.

Булева алгебра — это математическая система, оперирующая двумя понятиями: «событие истинно» и «событие ложно». Естественно ассоциировать эти понятия с цифрами, используемыми в двоичной системе счисления. Далее будем их называть соответственно логическими единицей (*лог.*1) и нулем (*лог.*0).

Все возможные логические функции k переменных можно образовать с помощью трех основных операций:

- **Конъю́нкция** (от лат. *conjunctio* союз, связь) — логическая операция, по своему применению максимально приближённая к союзу «и». Синонимы: логическое «И», **логическое умножение**, или просто «И».
- **Дизъю́нкция** (лат. *disjunctio* — разобщение) — логическая операция, по своему применению максимально приближённая к союзу «или» в смысле «или то, или это, или оба сразу». Синонимы: логическое «ИЛИ», включающее «ИЛИ», **логическое сложение**, иногда просто «ИЛИ».
- **Отрица́ние** в логике — унарная операция над суждениями, результатом которой является суждение, «противоположное» исходному. Синоним: логическое «НЕ», **инверсия**.

Приведем таблицы истинности для трех основных логических операций.

A	B	ИЛИ	И
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

A	НЕ
0	1
1	0

Практически все булевы функции для 1, 2 и 3-х переменных сложились исторически и имеют уникальные имена. Символы, участвующие в обозначениях элементарных функций, называются логическими связками (операциями), или функциональными символами. В литературе и различных языках программирования функциональные символы имеют различные условные обозначения. В таблице 2.1 приведены условные обозначения и исторические названия элементарных булевых функций.

Таблица 2.1. Обозначения и исторические названия некоторых булевых функций

Обозначение	Название
0	тождественный ноль, тождественная ложь, тождественное «НЕТ»
$x \downarrow y$, x ИЛИ-НЕ y, ИЛИ-НЕ(x,y), x NOR y, NOR(x,y)	НЕ-2ИЛИ, 2ИЛИ-НЕ, антидизъюнкция, функция Да́ггера, функция Ве́бба, стрелка Пи́рса
$x < y$, x LT y, LT(x,y), $\overline{x \leftarrow y}$	меньше, инверсия обратной импликации
$\overline{x}$, НЕ1(x,y), NOT1(x,y), x', $\neg x$,$\sim x$	отрицание (негация, инверсия) первого операнда
$x > y$, x GT y, GT(x,y), $\overline{x \rightarrow y}$	больше, инверсия прямой импликации
$\overline{y}$, НЕ2(x,y), NOT2(x,y), y', $\neg y$	отрицание (негация, инверсия) второго операнда
$x \oplus y$, $x +_2 y$, $x \neq y$, $x >< y$, $x <> y$, x XOR y, XOR(x,y)	сложение по модулю 2, не равно, измена, исключающее «или»
$x \mid y$, x NAND y, NAND(x,y), x И-НЕ y, И-НЕ(x,y)	НЕ-2И, 2И-НЕ, антиконъюнкция, пунктир Чулкова, штрих Ше́ффера
$x\&y$, $x \cdot y$, xy, $x \wedge y$, $x \times y$, x AND y, AND(x,y), x И y, И(x,y), $min(x,y)$	2И, конъюнкция
$x \equiv y$, $x = y$, x EQV y, EQV(x,y), $x \leftrightarrow y$	равенство, эквивалентность
y, ДА2(x,y), YES2(x,y)	второй операнд
$x \rightarrow y$, $x \leq y$, $x \supset y$, x LE y, LE(x,y)	меньше или равно, прямая импликация (от первого аргумента ко второму)
x, ДА1(x,y), YES1(x,y)	первый операнд
окончание на следующей странице	

начало на предыдущей странице	
Обозначение	Название
$x \leftarrow y$, $x \geq y$, $x \subset y$, x GE y, GE(x, y)	больше или равно, обратная импликация (от второго аргумента к первому)
$x \vee y$, $x + y$, x ИЛИ y, ИЛИ(x, y), x OR y, OR(x, y), $max(x, y)$	2ИЛИ, дизъюнкция
1	тождественная единица, тождественная истина, тождественное «ДА», тавтология

2.2 Основные аксиомы и законы алгебры–логики

Для логических операций (И, ИЛИ, НЕ), рассмотренных в предыдущем параграфе, справедлив ряд аксиом (тождеств) и законов, основные из которых даны в таблице 2.2. Для обозначения эквивалентности логических выражений используется знак равенства «=».

Таблица 2.2. Основные аксиомы и законы алгебры–логики

Аксиомы (тождества)	$0 \times A = 0$ $1 + A = 1$
	$1 \times A = A$ $0 + A = A$
	$A \times A = A$ $A + A = A$
	$\overline{\overline{A}} = A$
Законы коммутативности	$A + B = B + A$ $A \times B = B \times A$
Законы ассоциативности	$A + B + C = A + (B + C)$ $A \times B \times C = A \times (B \times C)$
Законы дистрибутивности	$A \times (B + C) = (A \times B) + (A \times C)$ $A + (B \times C) = (A + B) \times (A + C)$
Законы дуальности (теоремы Де–Моргана)	$\overline{A + B} = \overline{A} \times \overline{B}$ $\overline{A \times B} = \overline{A} + \overline{B}$
Законы поглощения	$A + A \times B = A$ $A \times (A + B) = A$

В общем случае логические выражения являются функциями логических переменных A, B, C, ..., каждая из которых может принимать значения 0 или 1. Если имеются k логических переменных, то они образуют 2^k возможных логических наборов из 0 и 1. При $k = 1$: $A = 0$ или $A = 1$; При $k = 2$: $AB = 00$, 01, 10, 11 и т. д. Для каждого набора переменных логическая функция F может принимать значения 0 или 1. Поэтому для k переменных можно образовать $l_k = 2^{2^k}$ различных логических функций. Таким образом, при увеличении k число l растет чрезвычайно быстро: при $k = 2$ получим $l_k = 16$; при $k = 3$ получим $l_3 = 256$; при $k = 4$ получим $l_4 = 65536$ и т. д.

Следует отметить, что алгебраические выражения тождеств и законов в таблице 2.2 заданы парами, и взаимной заменой операций И, ИЛИ и символов 0 и 1 из одного

выражения получается другое. Используя данные тождества и законы, можно получать новые логические выражения, а также доказывать справедливость тех или иных законов на основании других.

Например, с помощью второго закона дистрибутивности и тождества получаем соотношение:

$$A + \overline{A} \cdot B = (A + \overline{A}) \cdot (A + B) = A + B$$

Используя первый закон дистрибутивности, тождество, аксиому и закон коммутативности, получаем доказательство справедливости второго закона поглощения:

$$A \cdot (A + B) = A \cdot A + A \cdot B = A + A \cdot B = A \cdot (1 + B) = A.$$

Применение данных тождеств и законов позволяет производить упрощение логических функций, т. е. находить для них выражения, имеющие наиболее простую форму.

Используя законы ассоциативности, любую логическую функцию многих переменных ($k > 2$) можно представить в виде комбинации функции двух переменных.

Полный набор $2^{2^k} = 16$ логических функций двух переменных дан в таблице 2.3. Каждая функция обозначает одну из 16 возможных логических операций над двумя переменными A, B.

Таблица 2.3. Полный набор логических функций для двух переменных

A	0	0	1	1	Условное обозначение
B	0	1	0	1	и алгебраическое выражение
F_0	0	0	0	0	$F_0 = 0$
F_1	0	0	0	1	$F_1 = A \times B$
F_2	0	0	1	0	$F_2 = A \times \overline{B}$
F_3	0	0	1	1	$F_3 = A$
F_4	0	1	0	0	$F_4 = \overline{A} \times B$
F_5	0	1	0	1	$F_5 = B$
F_6	0	1	1	0	$F_6 = A \oplus B$
F_7	0	1	1	1	$F_7 = A + B$
F_8	1	0	0	0	$F_8 = A \downarrow B$
F_9	1	0	0	1	$F_9 = \overline{A \oplus B}$
F_{10}	1	0	1	0	$F_{10} = \overline{B}$
F_{11}	1	0	1	1	$F_{11} = B \to A$
F_{12}	1	1	0	0	$F_{12} = \overline{A}$
F_{13}	1	1	0	1	$F_{13} = A \to B$
F_{14}	1	1	1	0	$F_{14} = A \mid B = \overline{A \times B}$
F_{15}	1	1	1	1	$F_{15} = 1$

2.3 Способы записи функций алгебры логики

Рассмотрим некоторое логическое устройство, на входе которого присутствует некоторый n–разрядный двоичный код $x_{n-1}, \ldots, x_1, x_0$, на выходе соответственно m–разрядный двоичный код $z_{m-1}, \ldots, z_1, z_0$, (рис. 2.1). Для того, чтобы описать поведение этой схемы, необходимо определить зависимость каждой из m выходных переменных z_i от входного двоичного кода $x_{n-1}, \ldots, x_1, x_0$.

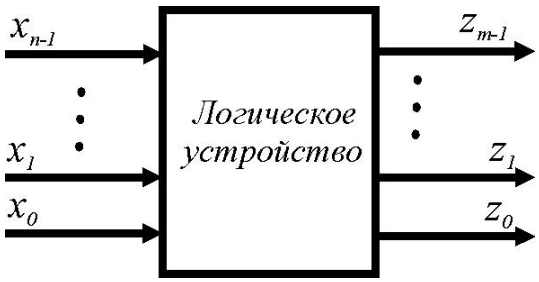


Рис. 2.1. Обобщенная схема логического устройства

Зависимость выходных переменных z_i, выраженная через совокупность входных переменных $x_{n-1}, \ldots, x_1, x_0$ с помощью операций алгебры–логики, носит название функции алгебры логики (ФАЛ). Иногда данную зависимость также называют переключательной функцией. Задать ФАЛ — это значит определить значения z_i для всех возможных комбинаций переменных $x_{n-1}, \ldots, x_1, x_0$. Очевидно, что для n-разрядного двоичного кода $x_{n-1}, \ldots, x_1, x_0$ существует $2n$ различных значений $z_i x_{n-1}, \ldots, x_1, x_0$.

Функция называется *полностью определенной*, если заданы 2^n ее значений. Если часть значений функции не задана, то она называется *частично определенной* или *недоопределенной*.

Иногда известно, что по условиям работы устройства появление некоторых входных кодов невозможно, и поэтому значения ФАЛ на этих кодах не задаются. При этом возникают так называемые факультативные или необязательные значения функции, которые могут задаваться произвольными значениями. Входные коды, для которых ФАЛ имеет факультативные значения, называются *запрещенными*.

Устройства, поведение которых описывается при помощи ФАЛ, называют *логическими*.

Для описания ФАЛ могут быть использованы различные способы. Основными из них являются: описание функции в словесной форме, в виде таблиц истинности, алгебраических выражений, последовательностей десятичных чисел и т. д.

2.3.1 Словесное описание ФАЛ

Данный вид описания наиболее часто применяется для первоначального, исходного описания поведения логического устройства. Проиллюстрируем словесное описание ФАЛ на примере.

Пример 2.1 *Логическая функция трех переменных равна единице, если хотя бы две входные переменные равны единице.*

2.3.2 Описание ФАЛ в виде таблицы истинности

Таблица, содержащая все возможные комбинации входных переменных $x_{n-1}, \ldots, x_1, x_0$ и соответствующие им значения выходных переменных z_i, называется *таблицей ис-*

тинности, или *комбинационной таблицей*. В общем случае таблица истинности содержит 2^n строк и $m+n$ столбцов. Проиллюстрируем построение таблицы истинности на примере.

Пример 2.2 *Составить таблицу истинности для ФАЛ из примера 2.1.*

Решение. Количество входных переменных $n=3$, т. о. строк будет — $2^3=8$. Количество выходных переменных $m=1$, т. е. количество столбцов — $m+n=1+3=4$. Составим таблицу истинности (см. таблицу 2.4). ■

Таблица 2.4. Таблица истинности для ФАЛ трех переменных

x_2	x_1	x_0	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

2.3.3 Описание ФАЛ в виде алгебраического выражения

При описании ФАЛ алгебраическим выражением используются две стандартные формы ее представления.

1. *Дизъюнктивная нормальная форма (ДНФ).* ДНФ называется логическая сумма элементарных логических произведений, в каждое из которых аргумент или его инверсия входят один раз. Получена ДНФ может быть из таблицы истинности с использованием следующего алгоритма:

   - для каждого набора переменных, на котором ФАЛ равна единице, записываются элементарные логические произведения входных переменных, причем переменные, равные нулю, записываются с инверсией. Полученные произведения называют *конституентами единицы*, или *минтермами* (m);
   - логически суммируются все конституенты единицы (минтермы).

   Пример 2.3 *Записать ДНФ для ФАЛ, заданной в примере 2.2.*

   Решение. Составим таблицу конституент единицы (минтермов) для ФАЛ, заданной в примере 2.2.

   Согласно приведенному выше алгоритму, используя минтермы из таблицы 2.5 и основные аксиомы (тождества) алгебры-логики (табл. 2.2), получим:

$$z(x_2,x_1,x_0)=z_0m_0+z_1m_1+z_2m_2+z_3m_3+z_4m_4+z_5m_5+z_6m_6+z_7m_7=$$
$$=0\cdot(\bar{x}_2\bar{x}_1\bar{x}_0)+0\cdot(\bar{x}_2\bar{x}_1x_0)+0\cdot(\bar{x}_2x_1\bar{x}_0)+1\cdot(\bar{x}_2x_1x_0)+0\cdot(x_2\bar{x}_1\bar{x}_0)+$$
$$+1\cdot(x_2\bar{x}_1x_0)+1\cdot(x_2x_1\bar{x}_0)+1\cdot(x_2x_1x_0)=$$
$$=\bar{x}_2x_1x_0+x_2\bar{x}_1x_0+x_2x_1\bar{x}_0+x_2x_1x_0$$

   Дизъюнктивную нормальную форму, полученную суммированием конституент единицы (минтермов), называют *совершенной дизъюнктивной нормальной формой* (СДНФ). ■

Таблица 2.5. Минтермы ФАЛ $z(x_2, x_1, x_0)$

x_2	x_1	x_0	Минтермы(m)	Значение функции
0	0	0	$m_0 = \overline{x}_2\overline{x}_1\overline{x}_0$	$z_0 = 0$
0	0	1	$m_1 = \overline{x}_2\overline{x}_1 x_0$	$z_1 = 0$
0	1	0	$m_2 = \overline{x}_2 x_1\overline{x}_0$	$z_2 = 0$
0	1	1	$m_3 = \overline{x}_2 x_1 x_0$	$z_3 = 1$
1	0	0	$m_4 = x_2\overline{x}_1\overline{x}_0$	$z_4 = 0$
1	0	1	$m_5 = x_2\overline{x}_1 x_0$	$z_5 = 1$
1	1	0	$m_6 = x_2 x_1\overline{x}_0$	$z_6 = 1$
1	1	1	$m_7 = x_2 x_1 x_0$	$z_7 = 1$

2. *Конъюнктивная нормальная форма (КНФ)*. КНФ называется логическое произведение элементарных логических сумм, в каждую из которых аргумент или его инверсия входят один раз. Получена КНФ может быть из таблицы истинности с использованием следующего алгоритма:

   - для каждого набора переменных, на котором ФАЛ равна нулю, записывают элементарные логические суммы входных переменных, причем переменные, значения которых равны единице, записывают с инверсией. Полученные суммы называют *конституентой нуля*, или *макстермами*;
   - логически перемножают все конституенты нуля (макстермы).

Пример 2.4 *Записать КНФ для ФАЛ, заданной в примере 2.2.*

Решение. Составим таблицу конституент нуля (макстермов) для ФАЛ, заданной в примере 2.2.

Таблица 2.6. Макстермы ФАЛ $z(x_2, x_1, x_0)$

x_2	x_1	x_0	Макстермы (M)	Значение функции
0	0	0	$M_0 = x_2 + x_1 + x_0$	$z_0 = 0$
0	0	1	$M_1 = x_2 + x_1 + \overline{x}_0$	$z_1 = 0$
0	1	0	$M_2 = x_2 + \overline{x}_1 + x_0$	$z_2 = 0$
0	1	1	$M_3 = x_2 + \overline{x}_1 + \overline{x}_0$	$z_3 = 1$
1	0	0	$M_4 = \overline{x}_2 + x_1 + x_0$	$z_4 = 0$
1	0	1	$M_5 = \overline{x}_2 + x_1 + \overline{x}_0$	$z_5 = 1$
1	1	0	$M_6 = \overline{x}_2 + \overline{x}_1 + x_0$	$z_6 = 1$
1	1	1	$M_7 = \overline{x}_2 + \overline{x}_1 + \overline{x}_0$	$z_7 = 1$

Согласно приведенному выше алгоритму, используя макстермы из таблицы 2.6 и основные аксиомы (тождества) алгебры-логики (табл. 2.2), получим:

$$z\,(x_2, x_1, x_0) = (z_0 + M_0)\cdot(z_1 + M_1)\cdot(z_2 + M_2)\cdot(z_3 + M_3)\cdot(z_4 + M_4)\times$$
$$\times\,(z_5 + M_5)\cdot(z_6 + M_6)\cdot(z_7 + M_7) =$$
$$= (x_2 + x_1 + x_0)\cdot(x_2 + x_1 + \bar{x}_0)\cdot(x_2 + \bar{x}_1 + x_0)\cdot(\bar{x}_2 + x_1 + x_0)$$

Конъюнктивную нормальную форму, полученную суммированием конституент нуля (макстермов), также называют *совершенной конъюнктивной нормальной формой* (СКНФ). ■

Рассмотренные методики позволяют получить математическую форму записи для самой функции. Иногда удобнее применять не саму ФАЛ, а ее инверсию. В этом случае при использовании вышеописанных методик для записи СДНФ необходимо выбирать нулевые, а для записи СКНФ - единичные значения функции.

Пример 2.5 *Для ФАЛ из примера 2.2 записать СДНФ и СКНФ инверсной функцией.*

Решение. Воспользовавшись таблицей 2.4, запишем
СДНФ: $\bar{z}\,(x_2, x_1, x_0) = \bar{x}_2\bar{x}_1\bar{x}_0 + \bar{x}_2\bar{x}_1x_0 + \bar{x}_2x_1\bar{x}_0 + x_2\bar{x}_1\bar{x}_0$.
СКНФ: $\bar{z}\,(x_2, x_1, x_0) = (x_2 + \bar{x}_1 + \bar{x}_0) \cdot (\bar{x}_2 + x_1 + \bar{x}_0) \cdot (\bar{x}_2 + \bar{x}_1 + x_0) \cdot (\bar{x}_2 + \bar{x}_1 + \bar{x}_0)$ ■

2.3.4 Описание ФАЛ в виде последовательности десятичных чисел

Иногда для сокращения записи ФАЛ представляют в виде последовательности десятичных чисел. При этом последовательно записывают десятичные эквиваленты двоичных кодов соответствующих конституент единицы и нуля (минтермов и макстермов).

Пример 2.6 *Записать в виде последовательности десятичных чисел ФАЛ из примеров 2.3 и 2.4*

Решение. В СДНФ из примера 2.3 первая конституента единицы (минтерм — $\overline{x}_2x_1x_0$) соответствует двоичному коду 011 (табл. 2.5). Десятичный эквивалент этого кода равен 3. Аналогично записываются все остальные конституенты:

$$z\,(x_2x_1x_0) = \sum (3, 5, 6, 7).$$

В СКНФ из примера 2.4 первая конституента нуля (макстерм — $x_2 + x_1 + x_0$) соответствует двоичному коду 000 (табл. 2.6). Десятичный эквивалент этого кода равен 0. Аналогично записывают все остальные конституенты:

$$z\,(x_2x_1x_0) = \prod (0, 1, 2, 4).$$

■

2.3.5 Кубические комплексы

В последнее время широкое распространение получило так называемое кубическое представление ФАЛ. Такое представление использует ограниченное число символов и поэтому применяется при автоматизации процессов логического проектирования цифровых интегральных схем (ИС).

Основой кубической формы является представление каждого набора входных переменных в качестве n–мерного вектора. Вершины этих векторов геометрически могут быть представлены как вершины n-мерного куба. Отмечая точками вершины векторов, для которых ФАЛ равна единице, получаем геометрическое представление функции куба.

Пример 2.7 *Задана ФАЛ* $z\,(x_2, x_1, x_0) = \sum (3, 4, 5, 6, 7)$. *Дать геометрическое представление в виде куба.*

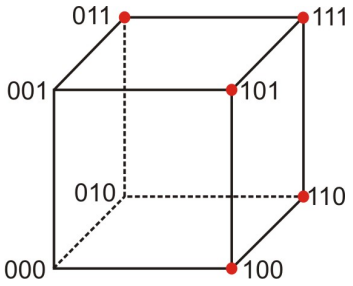


Рис. 2.2. Геометрическое представление ФАЛ

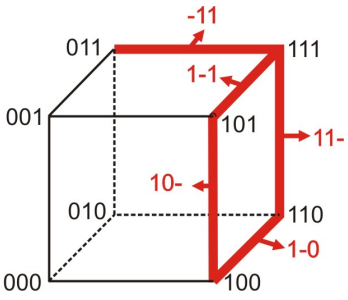


Рис. 2.3. Единичный кубический комплекс ФАЛ (см. пример 2.8)

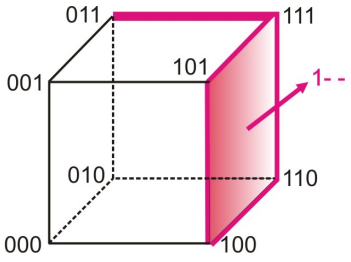


Рис. 2.4. Двоичный куб для ФАЛ (см. пример 2.8)

Решение. Графическое решение задачи проиллюстрировано на рисунке 2.2. ■

Очевидно, что наборы переменных, расположенные на концах ребер куба, отличаются только одной переменной. Такие наборы (коды) принято называть соседними.

Каждую функцию куба, в которой функция принимает единичное значение, называют *нулевым кубом* (**0–кубом**). Записывается 0–куб последовательностью образовавших его входных переменных, т.е. кодом, соответствующим конституенте единицы. Множество нулевых кубов образуют нулевой кубический комплекс K_0 ФАЛ.

Если два нулевых куба комплекса K_0 отличаются только по одной координате (переменной), т.е. два набора переменных, для которых ФАЛ равна единице, являются соседними, то они образуют *единичный куб* (**1–куб**). Геометрически это соответствует ребру исходного n–мерного куба (рис 2.3), 1–куб записывается последовательностью общих элементов образовавших его 0–кубов с прочерком несовпадающих элементов. Множество единичных кубов образует единичный кубический комплекс K_1.

Аналогично, если два единичных куба комплекса K_1 отличаются только по одной координате (переменной), то эти единичные кубы образуют *двоичный куб* (**2-куб**). Геометрически это соответствует грани исходного n-мерного куба (рис. 2.4). 2-куб также записывается последовательностью общих элементов образовавших его 1-кубов с прочерком несовпадающих элементов, а множество двоичных кубов образуют двоичный кубический комплекс K_2. И так далее.

Пример 2.8 *Для ФАЛ из примера 2.7 записать кубические комплексы.*

Решение. Нулевой кубический комплекс содержит пять членов по числу конституент единицы ФАЛ. $K_0 = (011, 100, 101, 110, 111)$.

Сравнивая записанные 0-кубы, можно увидеть, что 1–й и 5–й кубы отличаются только первым членом. Поэтому они образуют 1–куб вида $\breve{}11$. Аналогично, 2-ой и 3-й 0–кубы образуют 1–куб вида 10– и т.д. Единичный кубический комплекс заданной ФАЛ будет иметь вид: $K_1 = (-11, 10-, 11-, 1-1)$.

Аналогично может быть получен и двоичный кубический комплекс, состоящий из одного 2–куба: $K_2 = (1--)$. ■

Из сказанного следует, что размерность куба (**его ранг**) определяется числом несовпадающих координат, т. е. числом прочерков в его записи.

Объединение кубических комплексов $K_0, K_1, \ldots, K_m$ для ФАЛ n-переменных образует ее кубический комплекс: $K(z) = \bigcup (K_0, K_1, \ldots K_m)$.

Глава 3

СИНТЕЗ ЛОГИЧЕСКИХ СХЕМ

Пользуясь ФАЛ, мы до сих пор ничего не говорили о структуре логического устройства, представляя его аналогично устройству на рис. 2.1 в виде некоторого «черного ящика». Однако ФАЛ определят внутреннюю структуру логического устройства. Если мы располагаем элементарными узлами, реализующими основные логические операции, заданные постулатами в п. 2.1, то с их помощью можно построить логическую схему, выполняющую заданный алгоритм преобразования исходных логических переменных. В общем случае характер реальных логических переменных не имеет значения. Он может быть произвольным.

3.1 Стандарты на условные графические обозначения элементов цифровых схем

Инженеру–электронику постоянно приходится иметь дело с технической документацией на различные устройства, которая, как правило, доступна на INTERNET сайтах фирм–разработчиков и написана на языке оригинала (в основном английском). Кроме того, в современной переводной литературе (учебниках, статьях и т. п.) никто не «перерисовывает» принципиальные схемы электронных устройств в соответствии с российским стандартом, и они приводятся в оригинальном виде. Поэтому знакомство с различными стандартами на обозначения необходимо для понимания принципов работы электронных схем и их компонентов.

В настоящее время в мире существуют несколько общепринятых стандартов условных графических обозначений для элементов, реализующих основные логические операции. Цифровая техника последние 50 лет бурно развивается, номенклатура и функциональная нагруженность интегральных схем (ИС) постоянно изменяются, что влечет за собой периодическую модификикацию стандаров на условные графические обозначения в цифровых логических схемах.

В технической литературе используются несколько стандартов на условные обозначения элементов — российский (**ГОСТ 2.743–82**, заменён на **ГОСТ 2.743–91**); европейский (**DIN 40700** поддерживается в немецко- и франкоязычной литературе, в настоящее время заменен на **DIN EN 60617**); американский (**milspec 806B** поддерживается в англоязычной и японской литературе). Кроме этого, в русскоязычной технической литературе до появления ГОСТ 2.743–82 активно использовался стандарт **МЭК 117-15A**, созданный Международной электротехнической комиссией (International Electrotechnical Comission, IEC) в которую СССР, а затем и Россия входят с 1922 г. В настоящее время действующим стандартом МЭК является стандарт **IEC 60617**.

3.1.1 Обозначение выводов логических элементов

Необходимо сказать, что многие обозначения, в частности обозначения выводов, несущих логическую информацию, одинаковы во всех вышеперечисленных стандартах. Приведем обозначения выводов согласно российского стандарта ГОСТ 2.743–91 [5].

Выводы, несущие логическую информацию, подразделяют на статические и динамические, а также на прямые и инверсные.

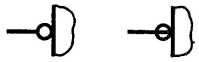
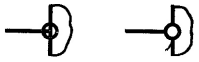
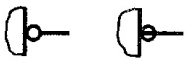
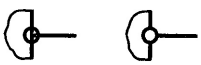
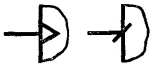
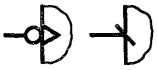
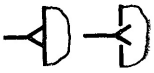
На прямом статическом выводе двоичная переменная имеет значение «1», если сигнал на этом выводе в активном состоянии находится в состоянии «*лог.*1» в принятом логическом соглашении.

На инверсном статическом выводе двоичная переменная имеет значение «1», если сигнал на этом выводе в активном состоянии находится в состоянии «*лог.*0» в принятом логическом соглашении.

На прямом динамическом выводе двоичная переменная имеет значение «1», если сигнал на этом выводе изменяется из состояния «*лог.*0» в состояние «*лог.*1» в принятом логическом соглашении.

На инверсном динамическом выводе двоичная переменная имеет значение «1», если сигнал на этом выводе изменяется из состояния «*лог.*1» в состояние «*лог.*0» в принятом логическом соглашении.

Таблица 3.1. Обозначение выводов элементов

№ п/п	Наименование	Обозначение	
		Форма 1	Форма 2
1.	Прямой статический вход		
2.	Прямой статический выход		
3.	Инверсный статический вход		
4.	Инверсный статический выход		
5.	Прямой динамический вход		
6.	Инверсный динамический вход		

Форма 1 является предпочтительной.

3.1.2 Базовые логические элементы

В соответствии с перечнем основных логических операций, перечисленных в разделе 2.1, различают три базовых логических элемента (ЛЭ): И, ИЛИ, НЕ. Приведем их условные графические обозначения, принятые в различных стандартах. Кроме этого, настоящий лабораторный практикум предполагает активную работу с системой графического программирования LabVIEW. Условные графические обозначения элементов, выполняющих логические операции, принятые в LabVIEW, в основном соответствуют стандарту **milspec 806B**, однако имеют некоторые особенности. Поэтому приведем их также[1].

[1] В скобках указано международное название логического элемента.

Таблица 3.2. Обозначение условное графическое логического элемента **И** (AND)

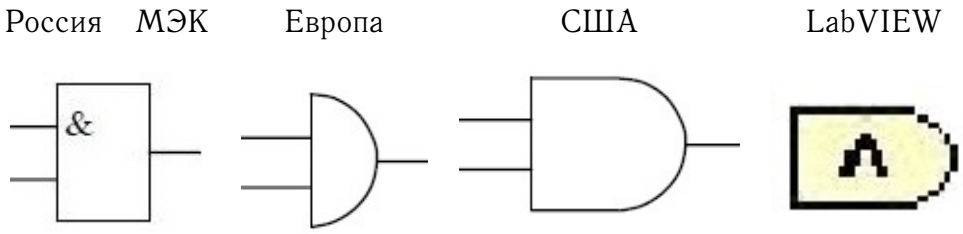


Таблица 3.3. Обозначение условное графическое логического элемента **ИЛИ** (OR)

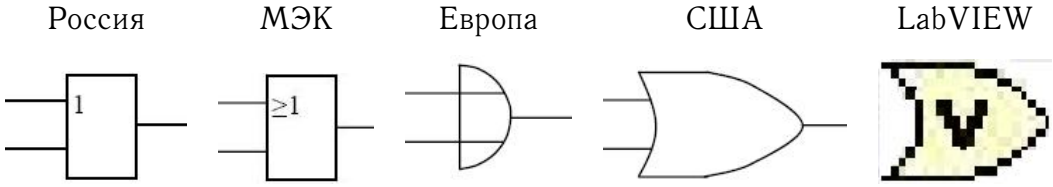
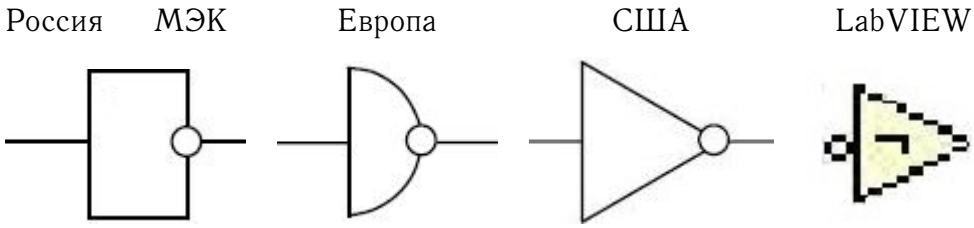


Таблица 3.4. Обозначение условное графическое логического элемента **НЕ** (NOT)

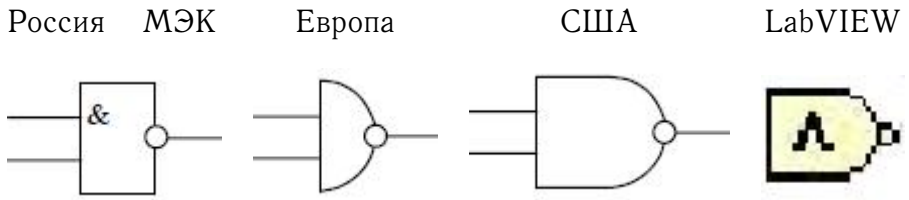


Наиболее распространенные логические элементы

Наряду с простейшими распространены и более сложные логические элементы, сочетающие в себе несколько простейших операций. Такими являются логические элементы И-НЕ, ИЛИ-НЕ, ЭКВИВАЛЕНТНОСТЬ, ИСКЛЮЧАЮЩЕЕ ИЛИ и т. п. Приведем условные графические обозначения некоторых из них.

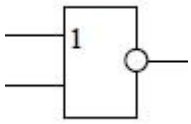
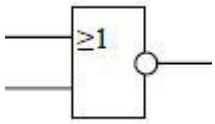
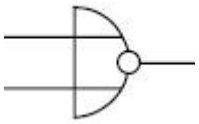
Штрих Шеффера. Элемент И-НЕ реализует функцию «штрих Шеффера» двух переменных $f = \overline{x \times y} = x|y$ (функция F_{14} в табл. 2.3). Условное обозначение логического элемента И-НЕ в любом стандарте объединяет в себе обозначение элемента И и кружок, являющийся признаком элемента НЕ.

Таблица 3.5. Логическая операция **И-НЕ**, *Штрих Шеффера* (NOT AND)



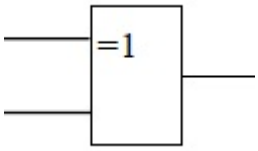
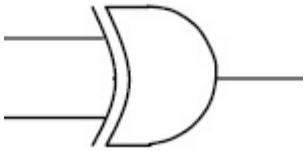
Стрелка Пирса. Элемент ИЛИ-НЕ реализует функцию «стрелка Пирса»: $f = \overline{x + y} = x \downarrow y$ (функция F_8 в табл. 2.3). Условные обозначения объединяют в себе обозначение элемента ИЛИ и кружок — символ операции отрицания (НЕ).

Таблица 3.6. Логическая операция **ИЛИ-НЕ**, *Стрелка Пирса* (NOT OR)

Россия	МЭК	Европа	США	LabVIEW
				

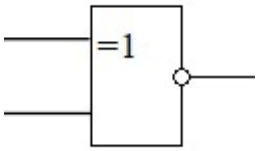
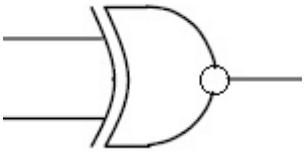
Исключающее ИЛИ, неравнозначность, сложение по модулю два означает «единица и только одна единица»: $f = x \cdot \overline{y} + \overline{x} \cdot y = x \oplus y$ (функция F_6 в табл. 2.3).

Таблица 3.7. Логическая операция **Исключающее ИЛИ** (eXclusive OR, XOR)

Россия	МЭК	США	LabVIEW
		

Исключающее ИЛИ-НЕ, равнозначность: $f = x \cdot y + \overline{x} \cdot \overline{y} = \overline{x \oplus y}$ (функция F_9 в табл. 2.3).

Таблица 3.8. Логическая операция **Исключающее ИЛИ-НЕ** (NOT eXclusive OR)

Россия	МЭК	США	LabVIEW
		

3.2 Синтез логического устройства

Для построения логической схемы необходимо логические элементы (ЛЭ), предназначенные для выполнения логических операций, указанных в ФАЛ, располагать от входа в порядке, определенном булевым выражением.

Пример 3.1 *Построить структурную схему логического устройства по ФАЛ из примера 2.1, т. е. определенную ФАЛ вида:*
$z\left(x_2, x_1, x_0\right) = \bar{x}_2 x_1 x_0 + x_2 \bar{x}_1 x_0 + x_2 x_1 \bar{x}_0 + x_2 x_1 x_0.$

Решение. Для реализации заданной ФАЛ в виде структурной логической схемы нам понадобятся три ЛЭ, реализующих операцию НЕ, т. к. исходная ФАЛ формируется тремя переменными (x_2, x_1, x_0), которые входят в нее как в прямом, так и в инверсном виде. Операция дизъюнкции должна быть выполнена четыре раза над тремя переменными, таким образом, для ее реализации нам понадобятся четыре ЛЭ, реализующих

операцию 3И. Последней выполняется операция конъюнкции над четырьмя выражениями, для реализации которой потребуется ЛЭ, реализующий операцию 4ИЛИ. Пример структурной логической схемы, реализующей заданную ФАЛ, приведен на рис. 3.1.

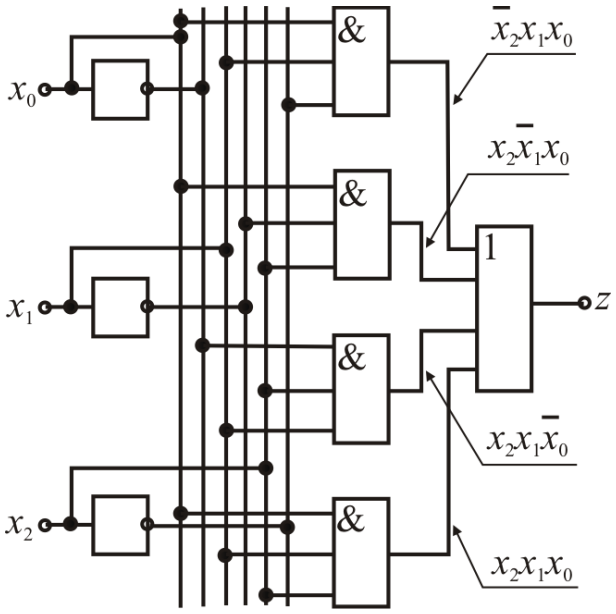


Рис. 3.1. Структурная схема логического устройства, реализующая ФАЛ вида $z\,(x_2, x_1, x_0) = \bar{x}_2x_1x_0 + x_2\bar{x}_1x_0 + x_2x_1\bar{x}_0 + x_2x_1x_0$

■

3.3 Переход от логической схемы к логической функции

Можно решить обратную задачу, т. е. по схеме логического устройсва перейти к логической функции. Обратная задача решается в несколько этапов:

- заданная схема разбивается по ярусам;
- начиная с последнего, выходы каждого элемента обозначаются проиндексированными функциями в зависимости от яруса, к которому относится элемент;
- записываются выходные функции каждого элемента в виде формул в соответствии с выбранными обозначениями логических операций;
- производится подстановка одних выходных функций через другие, используя входные переменные;
- записывается получившаяся булева функция через входные переменные;

Пример 3.2 *По заданной логической схеме (рис. 3.2) составить булеву функцию.*

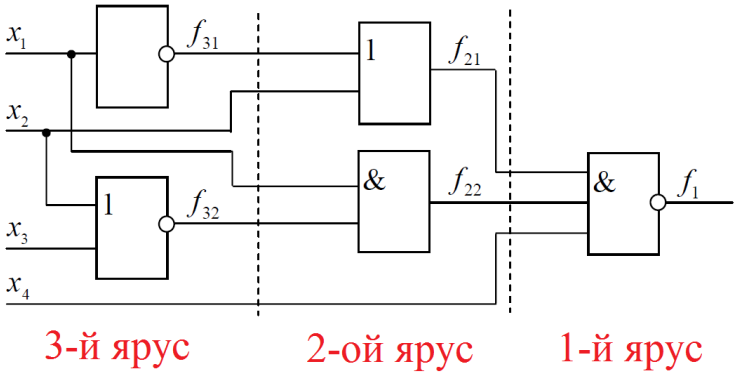


Рис. 3.2. Пример логической схемы устройства

Решение. Согласно приведённому выше алгоритму разобьём схему на ярусы, пронумеруем получившиеся ярусы, произведём индексирование выходных функций для каждого элемента (рис. 3.2). Запишем все функции, начиная с 1-го яруса:

1-й ярус : $f_1 = \overline{f_{21} \cdot f_{22} \cdot x_4}$.

2-ой ярус :

$$f_{21} = f_{31} + x_2; \quad f_{22} = f_{32} \cdot x_1.$$

3-й ярус :

$$f_{31} = \bar{x_1}; \quad f_{32} = \overline{x_2 + x_3}.$$

Запишем все функции, подставляя входные переменные x_1, x_2, x_3 и x_4:

$$f_{21} = \overline{x_1} + x_2; \quad f_{22} = x_1 \cdot (\overline{x_2 + x_3}).$$

Окончательно получим:

$$f_1 = f(x_4, x_3, x_2, x_1) = \overline{x_1 \cdot (\overline{x_1} + x_2) \cdot (\overline{x_2 + x_3}) \cdot x_4}.$$

■

Лабораторная работа №1. Знакомство с образовательной платформой NI ELVIS II$^+$ и системой графического программирования LabVIEW

Цель работы:

Целью лабораторной работы является знакомство с образовательной платформой *National Instruments* (NI) ELVIS II+. Знакомство со средой графического проектирования NI LabVIEW. Приобретение базовых знаний и навыков иерархического (модульного) проектирования цифровых схем, обучение созданию базовых элементов (*sub–VI*) и компоновке низкоуровневых функциональных блоков в графической среде программирования NI LabVIEW. Результаты проектирования необходимо проверить на оценочном модуле DE FPGA Board.

Задание на проектирование:

Создать в системе графического проектирования схему, представленную на рис. 3.1 и реализующую ФАЛ вида: $z = \bar{x}_2 x_1 x_0 + x_2 \bar{x}_1 x_0 + x_2 x_1 \bar{x}_0 + x_2 x_1 x_0$.

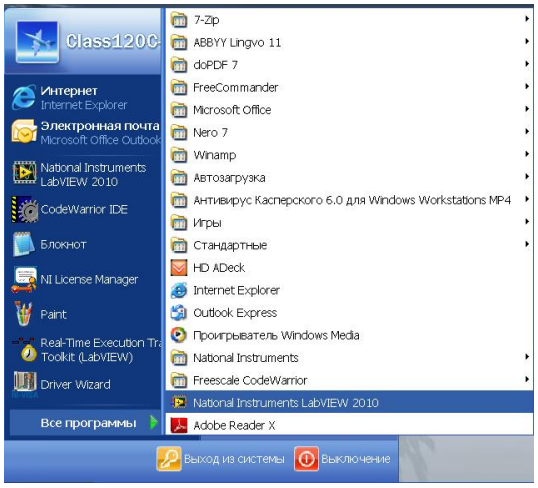
Выполнение задания:

Схема должна быть иерархической; используя функции самого низкого уровня, необходимо создать т. н. *sub–VI* — блоки, которые затем послужат функциональными блоками более высокого уровня.

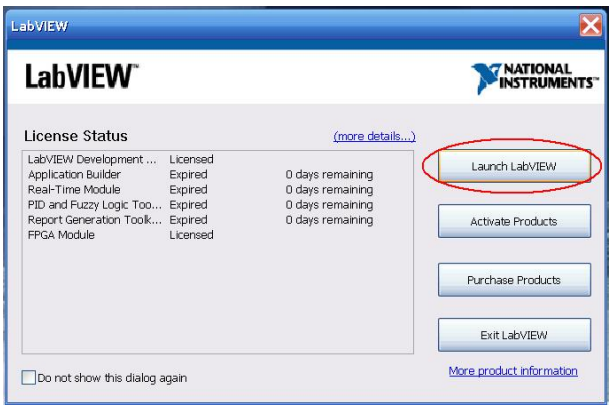
Для реализации логической схемы, представленной на рис. 3.1, понадобятся логические элементы, реализующие: операцию НЕ (**NOT**); операцию И (**AND**) над тремя входными переменными — 3И; операцию ИЛИ (**OR**) над четырьмя входными выражениями — 4ИЛИ. Блоки, выполняющие указанные логические операции, необходимо реализовать в виде функциональных *sub–VI* — блоков в среде графического проектирования LabVIEW. Затем при помощи указанных функциональных блоков собрать логическую схему (рис. 3.1). Проверить работоспособность схемы в оценочном модуле DE FPGA Board.

ЛР1.1. Запуск LabVIEW и создание нового проекта

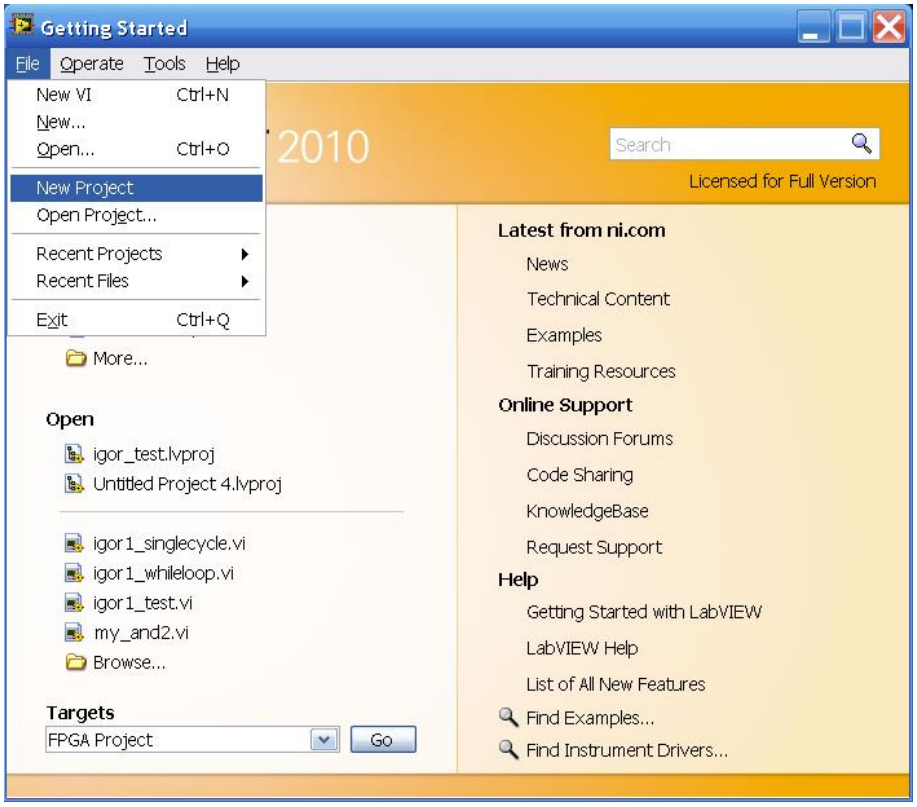
После того, как NI ELVIS II+ с установленной на ней платой подключена к компьютеру, все кабели скоммутированны, питание включено, можно запустить пакет LabVIEV на персональном компьютере.



Появится окошко, в котором необходимо выбрать **Launch LabVIEW**. Если при следующем запуске программы вы хотите проходить этот этап в автоматическом режиме, установите галочку *Do not show this dialog again*.



В окне «Getting Started» из меню **File** выбираем пункт меню **New Project**.



В «Project Explorer» нажимаем *правую* кнопку мышки на иконке **My Computer**, и выбираем **New** → **Targets and Device**, как это показано на левой панели рис. 3.3.

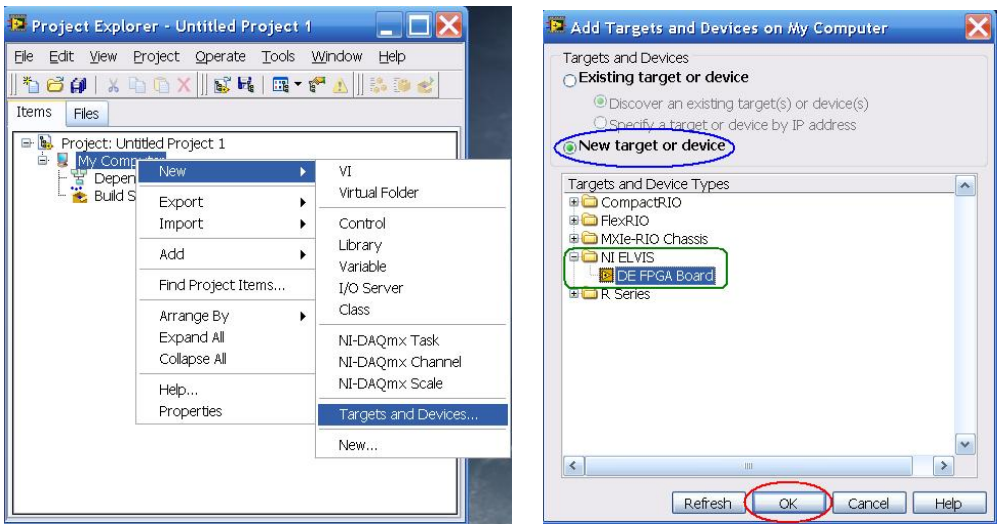
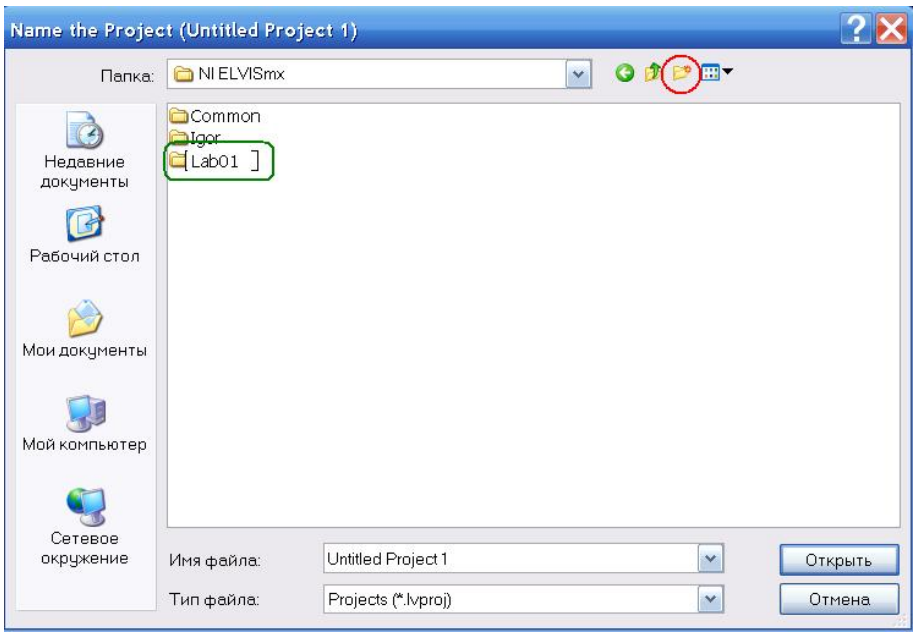


Рис. 3.3. Подключение нового оборудования к проекту

В окне диалога «Add Targets and Devices on My Computer» **обязательно необходимо** установить флажок в позицию *New Target or Device*. После этого в папке **NI ELVIS** выбрать **DE FPGA Board** (правая панель на рис. 3.3) и нажать **OK**.

Сохраним проект (нажимаем комбинацию клавиш «*Ctrl*+*S*»), в диалоговом режиме создадим новую папку со своим именем (например «IVANOV»), в этой папке создадим папку «Lab01», в которую сохраним новый проект под именем `Lab01.lvproj`.



ЛР1.2. Создание функциональных блоков

Для выполнения лабораторной работы нам понадобятся логические элементы **НЕ**, **3И**, **4ИЛИ**. В LabVIEW имеются примитивы, реализующие логические операции *НЕ*, *2И*, *2ИЛИ*. Таким образом, в первую очередь необходимо создать функциональные блоки, реализующие необходимые нам логические операции. Такие функциональные блоки в LabVIEW называются элементарными виртуальными приборами — *Sub–VI*.

В диалоговом окне «Project Explorer» щелкните *правой* кнопкой мышки по пункту **FPGA Target (Board1, DE FPGA Board)** и выберите **New** → **VI** (см. рис. 3.4).

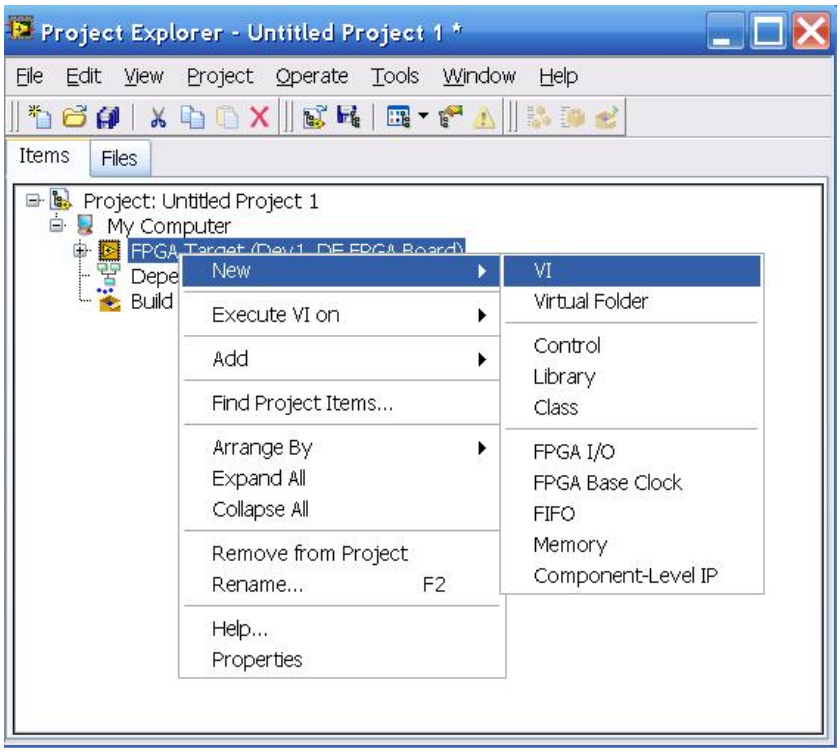


Рис. 3.4. Выбор — создание нового функционального блока

Появится окно «Front Panel», нажмите комбинацию клавиш *Ctrl-E* — откроется окно «Block Diagramm» (рис. 3.5).

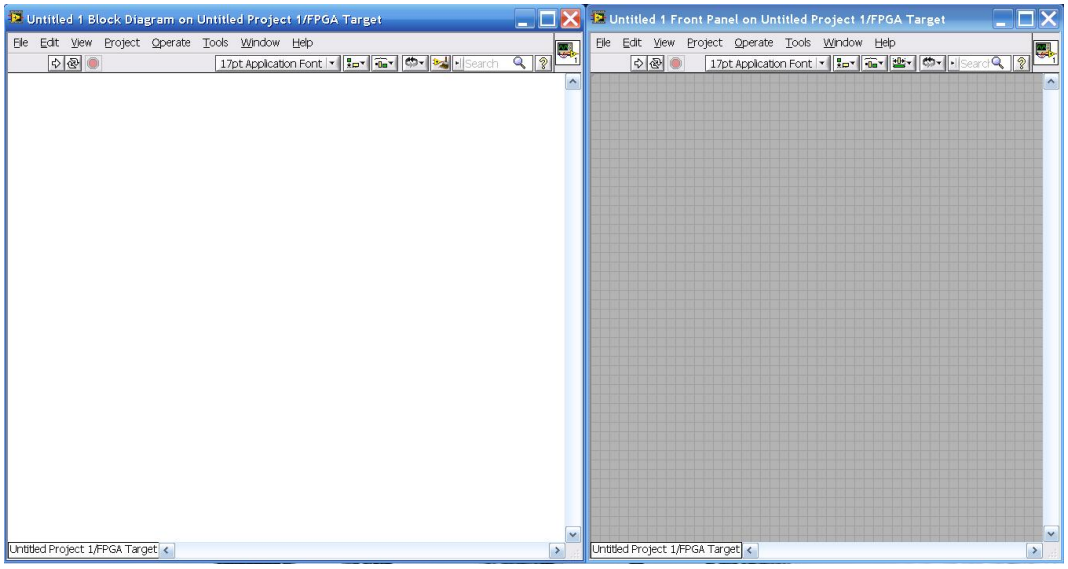


Рис. 3.5. Окна для создания нового функционального блока

Щелкните *правой* кнопкой мыши где–нибудь на белом поле (в рабочей области). В палитре функций откройте субпалитру **Boolean** (рис. 3.6). Вы можете также найти ее щелчком по кнопке *Search*. Выберите двухвходовый элемент **And** и поместите его на рабочую область блок–диаграммы.

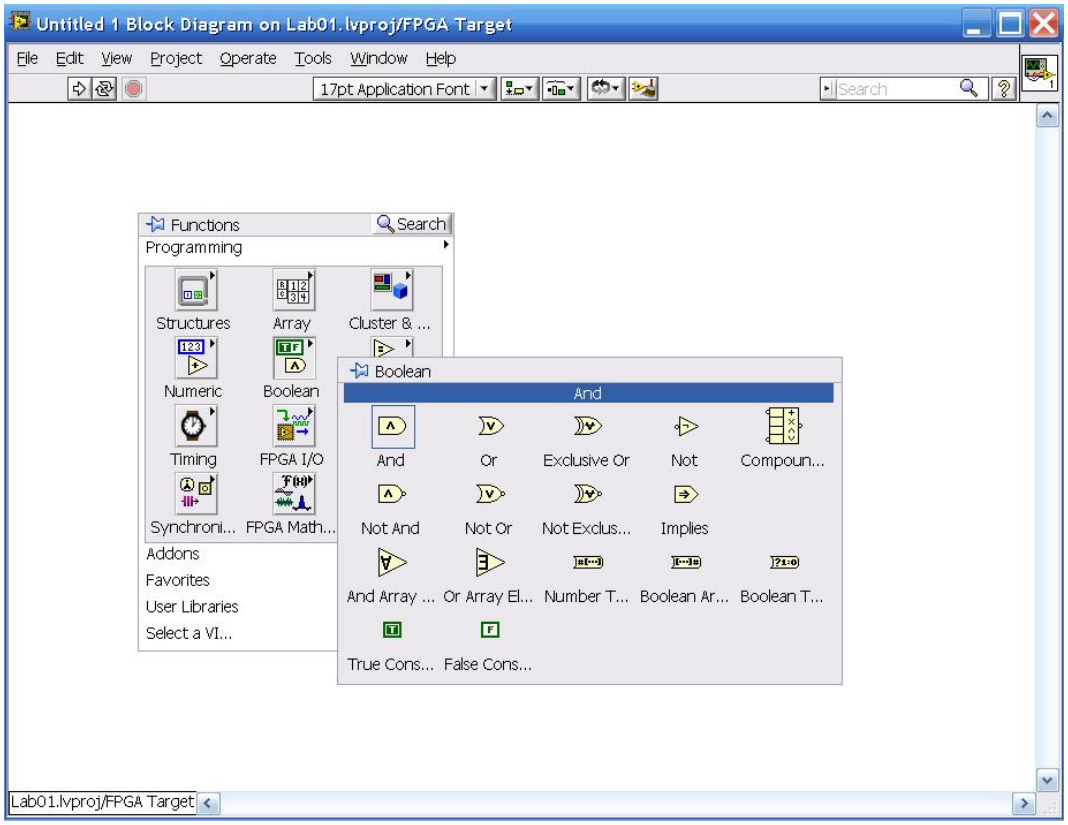
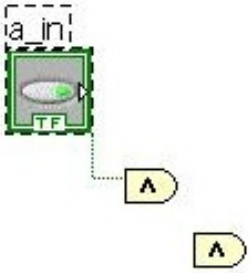
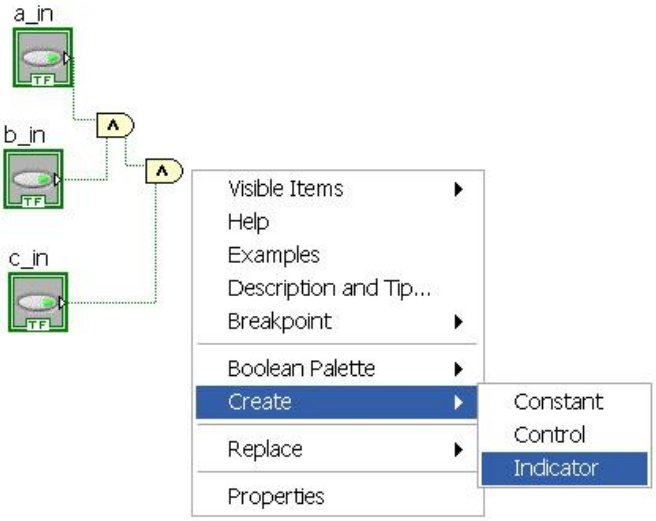


Рис. 3.6. Субпалитра функций Boolean

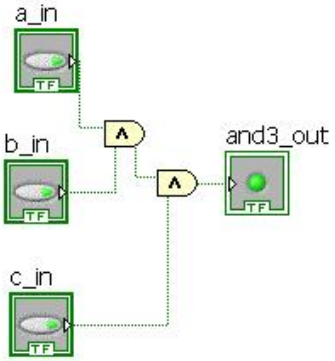
Щелкните *левой* кнопкой мышки по элементу и при помощи комбинаций квавиш *Ctrl-C* и *Ctrl-V* размножьте элемент **And** (так, чтобы получилось два элемента). Переместите курсор на левый угол первого элемента **And** и обратите внимание, что курсор принял вид инструмента соединения. Щелкните правой кнопкой мыши и выберите **Create** → **Control**. Создастся элемент управления с именем x (по умолчанию). Измените имя на `a_in`.



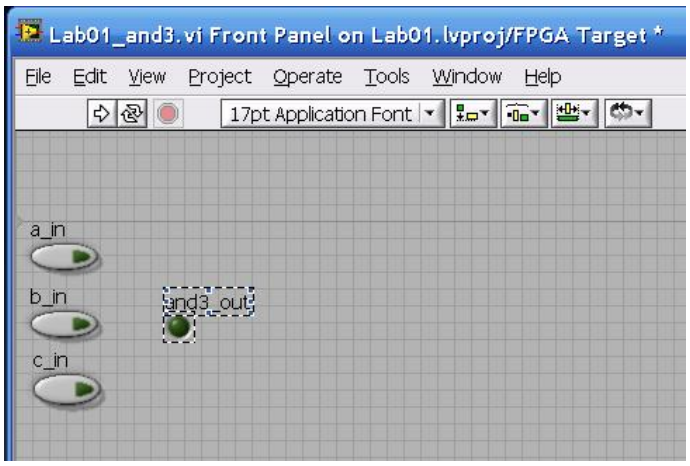
Аналогично создайте второй элемент управления (для второго входа первого элемента **And**) и присвойте ему имя `b_in`. Соедините выход первого элемента **And** с первым входом второго элемента **And**. Создайте третий элемент управления (для второго входа второго элемента **And**) и присвойте ему имя `c_in`. Создайте индикатор для выхода второго элемента **And** и присвойте ему имя `and3_out`.



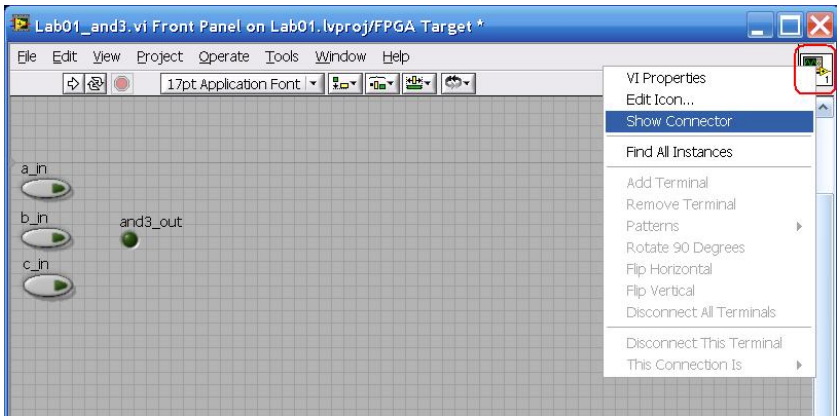
Перегруппируйте элементы управления и индикатор так, чтобы блок–диаграмма стала похожей на показанную ниже.



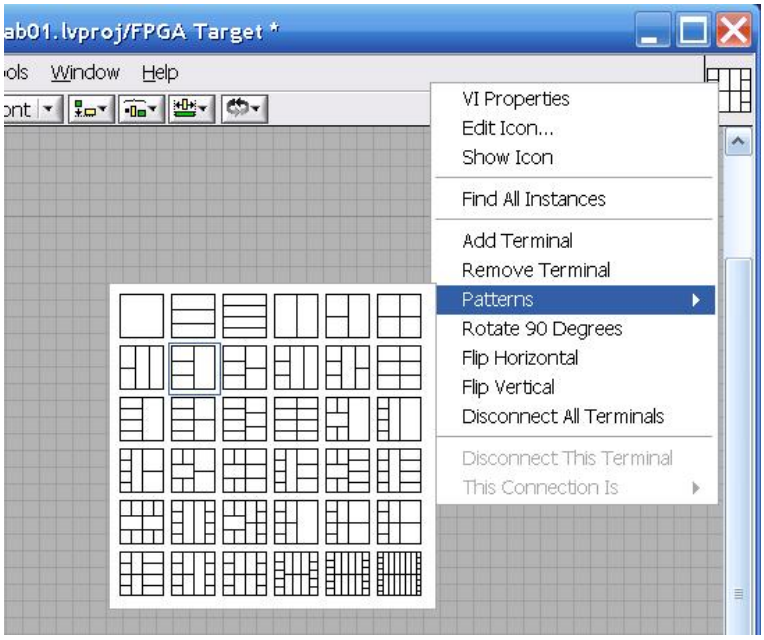
Перейдите в окно «Front Panel» и при помощи мышки перегруппируйте элементы управления функционального блока, как это показано на рисунке.



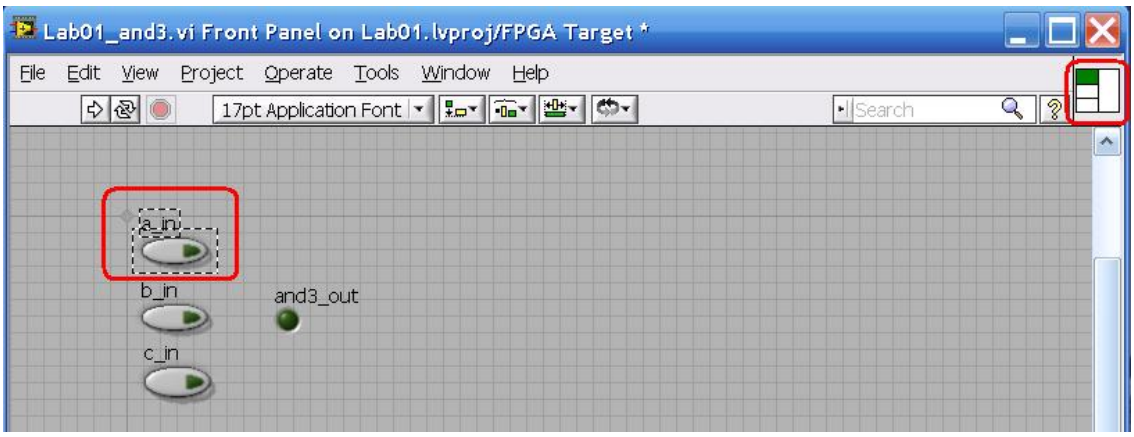
Щелкните *правой* кнопкой мышки по иконке, расположенной в правом верхнем углу, и выберите **Show Connector**.



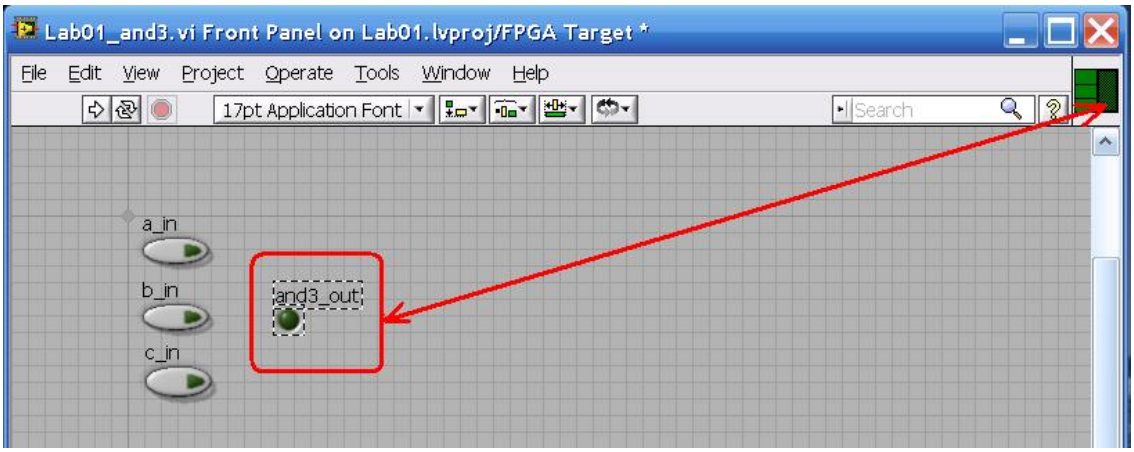
Еще раз щелкните правой кнопкой мыши по иконке, выберите пункт **Patterns** (шаблоны), а затем –– подходящий шаблон с достаточным количеством блоков для входов и выходов вашего VI.



Щелкните по входу в левом верхнем углу окна коннектора, а затем щелкните по кнопке `a_in`;

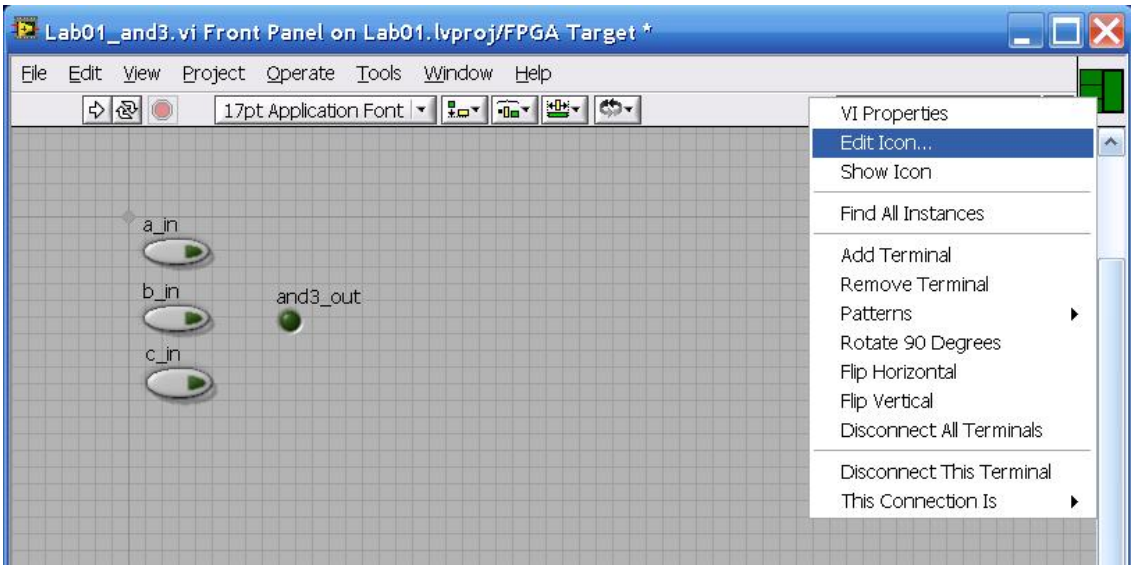


щелкните по входу в центре окна коннектора, а затем щелкните по кнопке `b_in`; щелкните по входу в левом нижнем углу окна коннектора, а затем по кнопке `c_in`; щелкните по крайнему правому блоку коннектора, а затем по индикатору выхода `and3_out`.

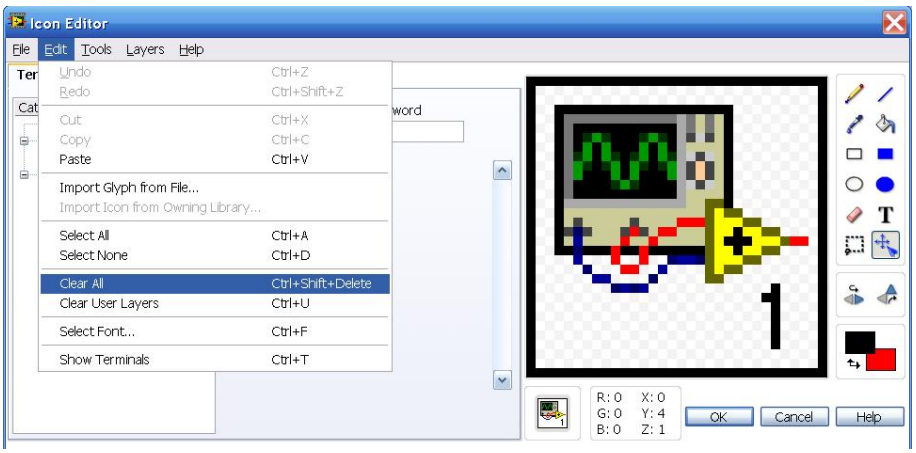


Важно: эти операции необходимо выполнять именно в таком порядке (вначале –– коннектор, затем элемент управления или индикатор). Если проделаеть это в другом порядке, то подключения получатся другими.

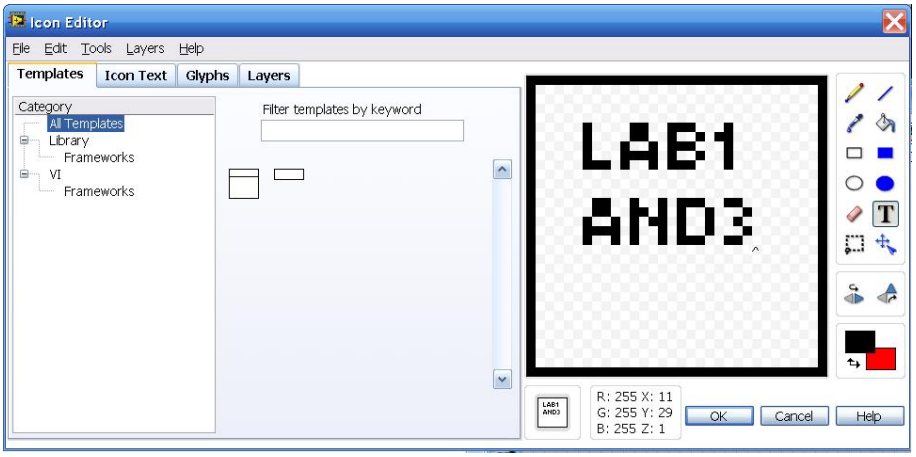
Щелкните правой кнопкой мыши по окну коннектора и выберите **Edit Icon**.



В окне редактора иконок выберите **Edit→Clear All** для очистки рисунка.

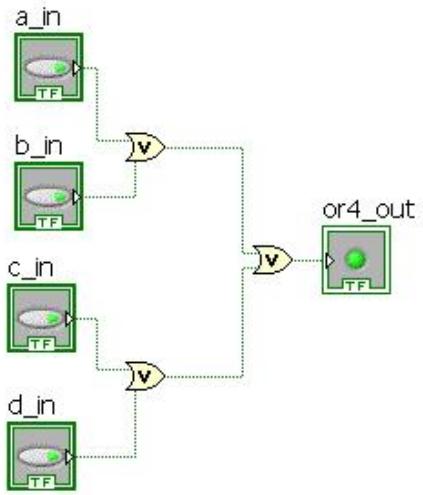


В палитре инструментов выберите инструмент ввода текста (**T**). В очищенном окошке введите **LAB1** (нажмите на клавишу *Enter*), а затем **AND3**. В палитре инструментов выберите инструмент рисования прямоугольника и нарисуйте в окошке прямоугольник.



Щелкните по кнопке **OK** и сохраните функциональный блок под именем — `Lab01_and3.vi`

Аналогично создайте 4-х входовой элемент ИЛИ — (**4ИЛИ**). Блок–диаграмма должна выглядеть так, как показано на рисунке ниже.



Порядок соединения коннекторов и элементов управления приведен на рис. 3.7. Сохраните функциональный блок под именем `Lab01_or4.vi`.

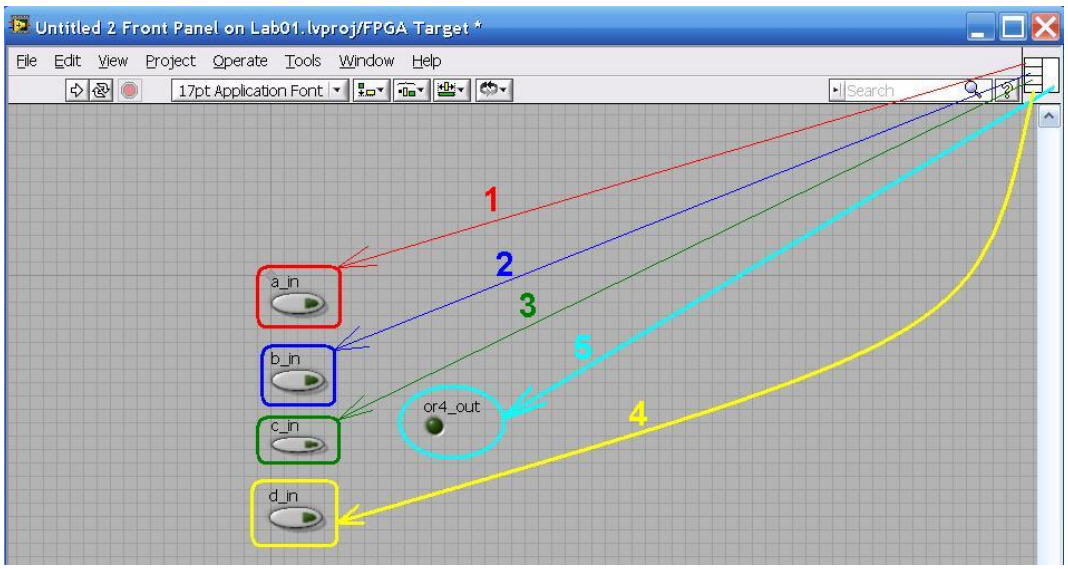


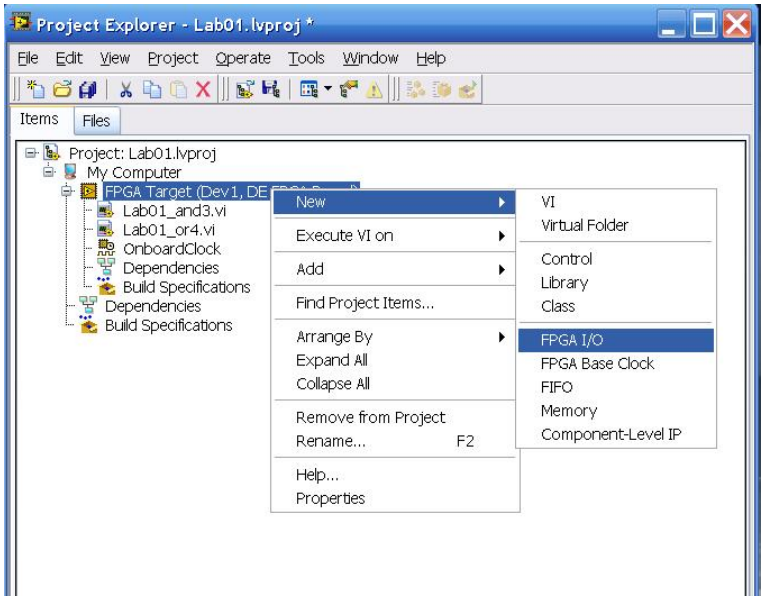
Рис. 3.7. Порядок соединения элементов управления в логическом элементе **4ИЛИ**

Контрольное задание 1. Составьте таблицы истинности для разработанных функциональных блоков, реализующих логические функции **3И** и **4ИЛИ**. Занесите получившиеся таблицы истинности в отчет по лабораторной работе №1.

ЛР1.3. Проектирование схемы, реализующую заданную ФАЛ

В предыдущем разделе мы научились создавать простые функциональные блоки (*sub–VI* в терминологии LabVIEW). На основе этих простых блоков создадим более сложную схему, реализующую заданную ФАЛ ($z = \bar{x}_2 x_1 x_0 + x_2 \bar{x}_1 x_0 + x_2 x_1 \bar{x}_0 + x_2 x_1 x_0$), структурная схема которой представлена на рис. 3.1.

Проектирование схемы начнем с того, что зададим каналы ввода–вывода. Для этого в окне «Project Explorer» щелкнем *правой* кнопкой мышки по пункту меню **FPGA Target (Board1, DE FPGA Board)** и выберем пункт меню **New→FPGA I/O**, как показано ниже.



Доступные в оценочном модуле DE FPGA Board каналы ввода–вывода отобразятся в левой части окна выбора (рис. 3.8). В секции «Available Resources» раскроем папку

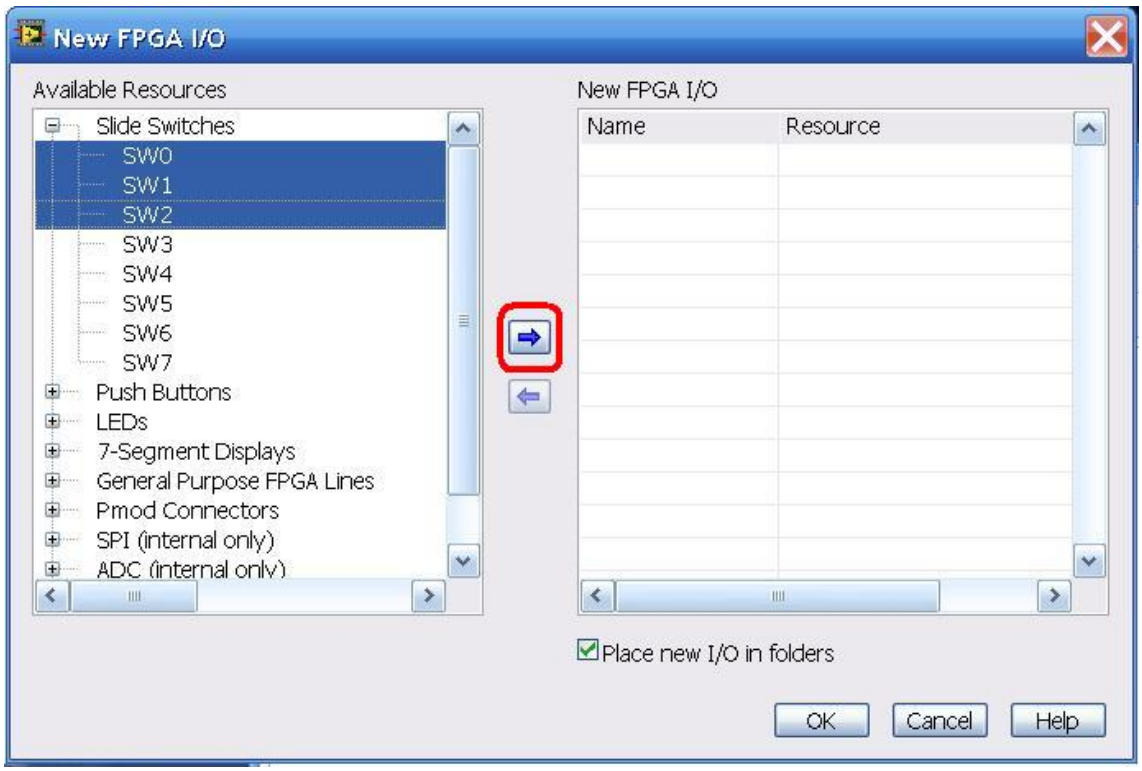


Рис. 3.8. Доступные в DE FPGA Board каналы ввода–вывода

Slde Swithes и, пользуясь комбинацией клавиш *Ctrl+Shift*, выберем переключатели **SW0**, **SW1** и **SW2**. Нажатием кнопки **Add** (обведено красным овалом на рис. 3.8) добавим выбранные элементы в проект. Аналогично, раскроем папку «LEDs», выберем светодиод **LED0** и добавим его в проект.

Данные виртуальные каналы ввода–вывода соответствуют реальным переключателям и светодиодам, расположенным на оценочной плате. В частности, задействованные в схеме переключатели и светодиод показаны на рис. 3.9 (обведены желтыми овалами).

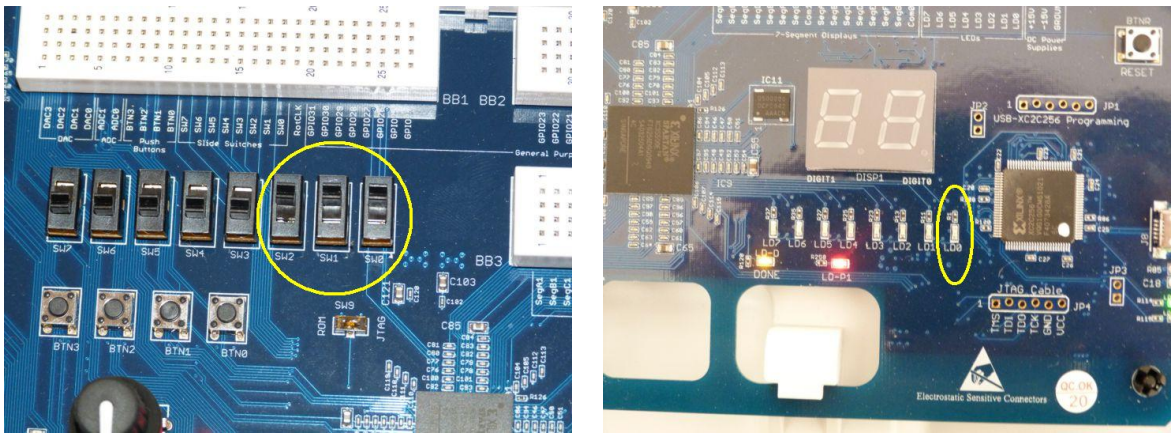


Рис. 3.9. Реальные каналы ввода–вывода на оценочной плате

После выполненных манипуляций в «Project Explorer», кроме уже присутствующих там функциональных блоков `Lab01_and3.vi` и `Lab01_or4.vi`, появятся добавленные нами переключатели и светодиод в соответствующих папках (рис. 3.10).

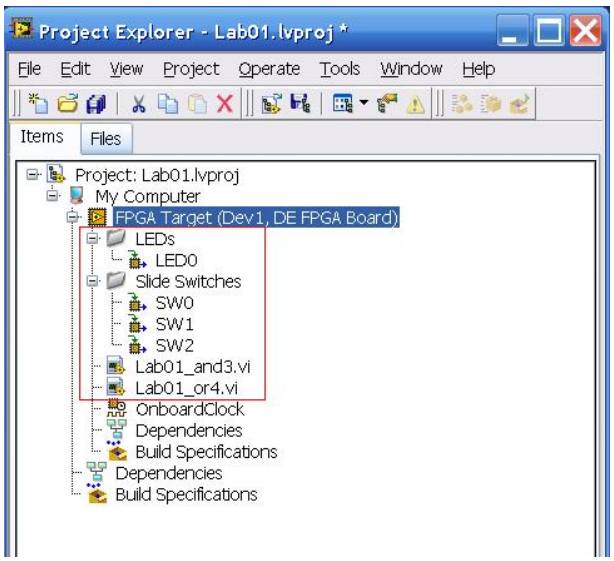


Рис. 3.10. Требуемый набор функциональных блоков

В «Project Explorer» нажмем на правую кнопку мышки на **FPGA Target (Board1, DE FPGA Board)** и при помощи меню **New→VI** создадим новый функциональный блок. При помощи коммбинации клавиш *Ctrl+E* откроем окно «Block Diagramm». Мышкой перетащим туда из «Project Explorer» переключатели (**SW0**, **SW1**, **SW2**), светодиод (**LED0**) и созданные нами функциональные блоки логических элементов (**3И**, **4ИЛИ**). Так как для реализации структурной схемы необходимы 4 логических элемента **3И**, то при помощи мышки и комбинаций клавиш *Ctrl+C* и *Ctrl+V* размножим этот элемент. Далее нажатием *правой* кнопки мышки в рабочей области окна «Block Diagramm» откроем палитру функций, в которой из субпалитры **Boolean** выберем логический элемент **Not — НЕ** (см. рис. 3.11).

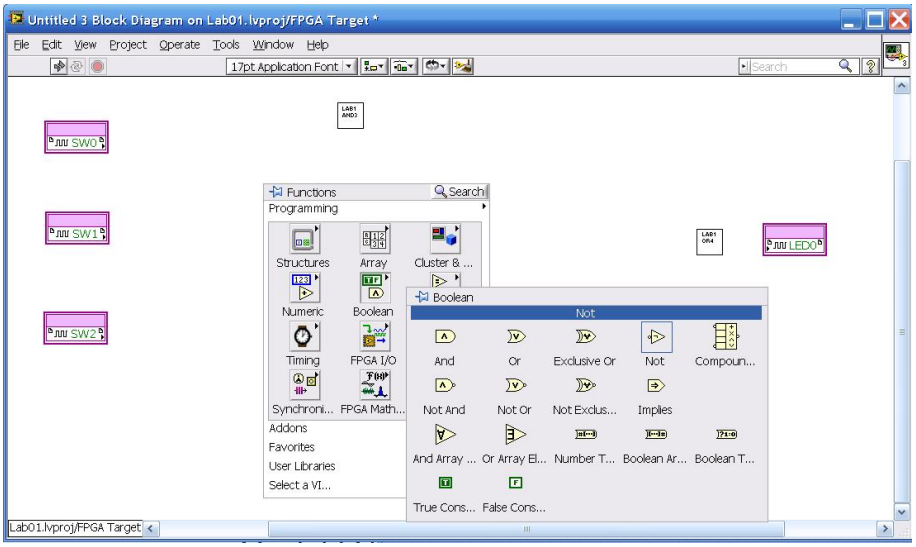


Рис. 3.11. Выбор элемента **НЕ**

Для реализации схемы, представленной на рис. 3.1, потребуются три таких элемента.

Затем при помощи мышки выстраиваем все функциональные блоки и соединяем их проводниками согласно заданной схемы. В результате получится блок–диаграмма,

похожая на представленную на рис. 3.12.

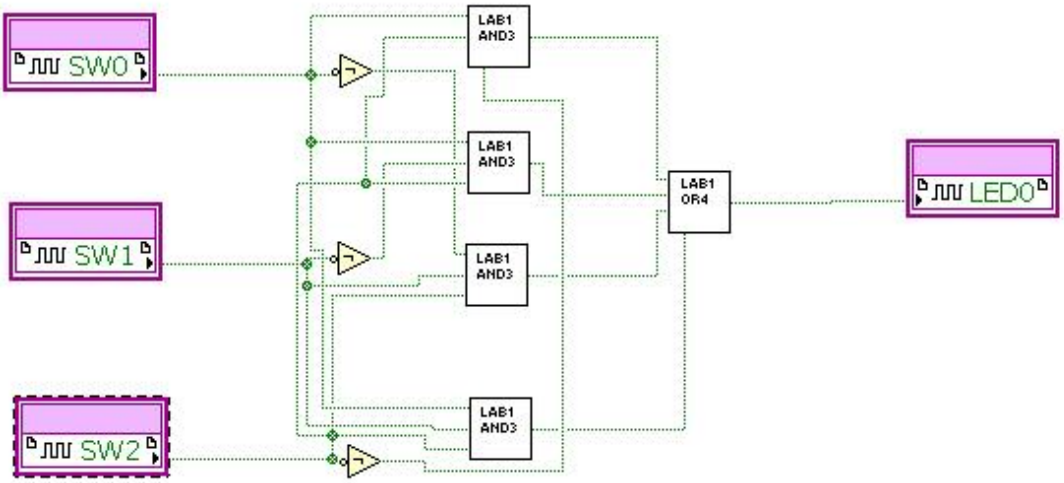
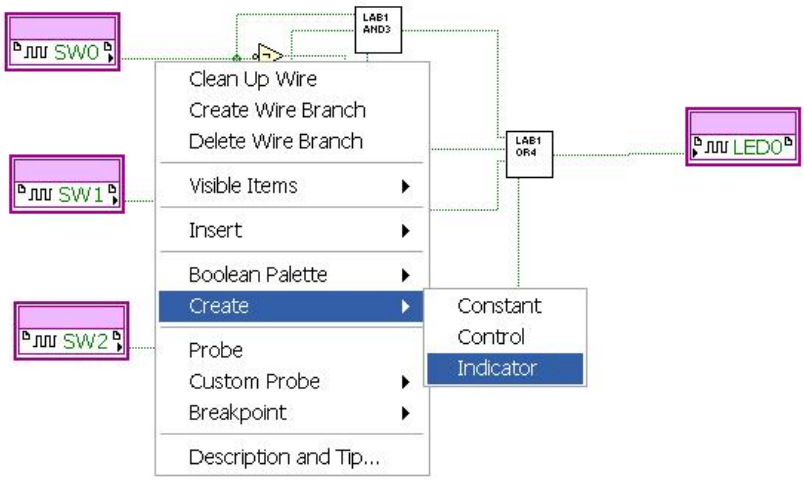


Рис. 3.12. Блок–диаграмма, реализующая ФАЛ

Необходимо отметить, что блок-диаграмма, представленная на рис. 3.12, не соответствует схеме, представленной на рис. 3.1 и, следовательно, описывается ФАЛ отличной от полученной в примере 2.3 СДНФ.

Контрольное задание 2. Используя описанную в 3.3 методику, запишите ФАЛ, соответствующую блок–диаграмме, представленной на рис. 3.12. Занесите полученную ФАЛ в отчет по лабораторной работе №1.

Добавим в блок–диаграмму индикаторы для переключателей и выхода, для этого щелкните *правой* кнопкой мышки по соответствующему проводнику и в контекстном меню выберите **Create→Indicator**.



Переименуйте индикаторы согласно названиям входов (x_0, x_1, x_2), выходной индикатор назовем **OUT**.

Переключитесь на «Front Panel» (комбинация клавиш *Ctrl+E*), на ней будут три индикатора входов и индикатор выхода. Можно при помощи мышки изменить их компоновку и внешний вид. Один из вариантов показан на рис. 3.13.

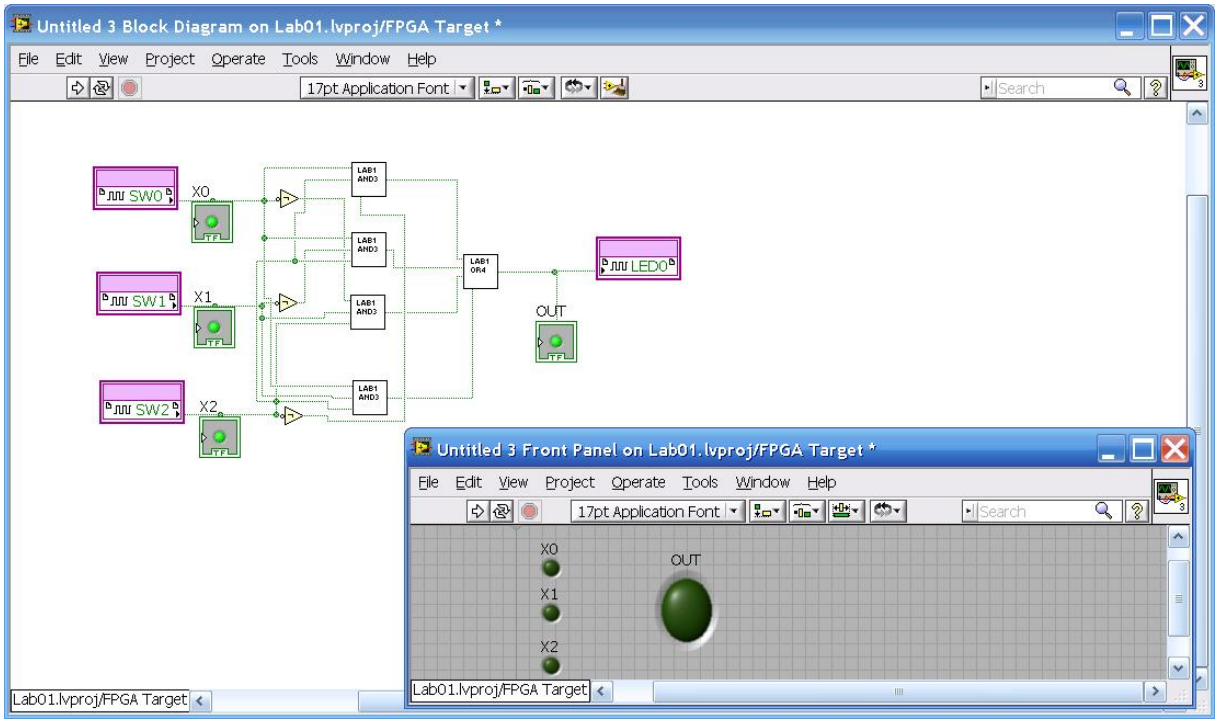


Рис. 3.13. Блок–диаграмма и лицевая панель

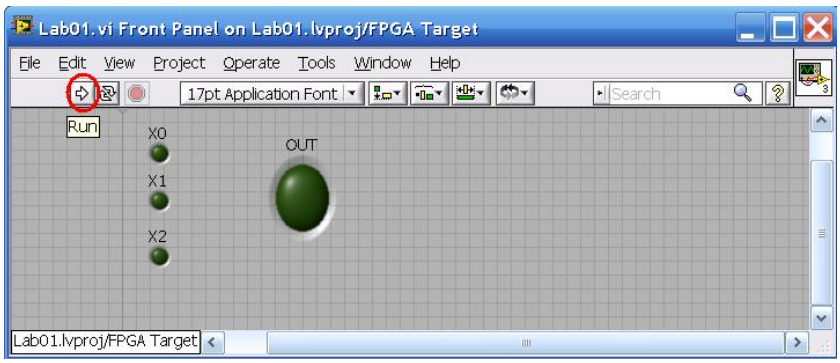
Сохраните проект в папку *Lab01* под тем же именем — **Lab01**.

ЛР1.4. Проверка результатов проектирования при помощи DE FPGA Board

После того, как схема собрана, проверим её работоспособность при помощи оценочного модуля DE FPGA Board.

Убедимся, что питание на ELVIS II$^+$ подано (на рис. 1.3 показано как должны гореть индикаторы), кабель JTAG–программатора подключен к оценочной платформе DE FPGA Board (рис. 1.4).

На лицевой панели («Front Panel») нажимаем кнопку **Run**



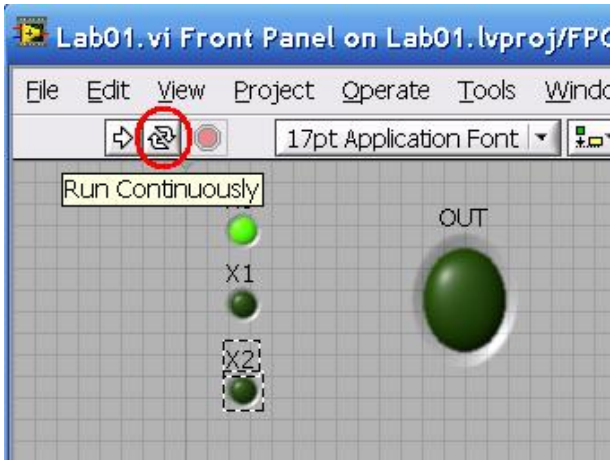
Начнется процесс компиляции проекта, это может занять определенное время. Сначала будет сгенерирован промежуточный файл. Затем запустится компилятор, который

создаст двоичный код, необходимый для «прошивки» ПЛИС. По окончании процесса компиляции код будет автоматически загружен в FPGA Board, она сконфигурируется, и в зависимости от положения переключателей выходной диод будет либо гореть, либо нет.

Если изменять состояния переключателей, то светодиод не будет реагировать, так как программа однократно выполнилась и остановилась.

Для того, чтобы посмотреть работу программы при различных положениях переключателей, необходимо сначала изменить их положение (см. рис. 3.9), а затем нажать кнопку **Run**. Двоичный код конфигурации FPGA загрузится через USB–JTAG кабель. В зависимости от установок входных переменных (x_0, x_1, x_2), определямых состоянием переключателей **SW0**, **SW1** и **SW2** на оценочной плате, светодиод **LED0** (и индикатор **OUT** на лицевой панели) будет либо гореть либо нет.

Если воспользоваться кнопкой запуска **Run Continuosly** (Запустить в непрерывном режиме),



то можно, изменяя состояния переключателей, наблюдать изменения состояния светодиода **LED0** на оценочной платформе DE FPGA Board и индикатора **OUT** на лицевой панели («Front Panel») в непрерывном режиме, как показано на рис. 3.14.

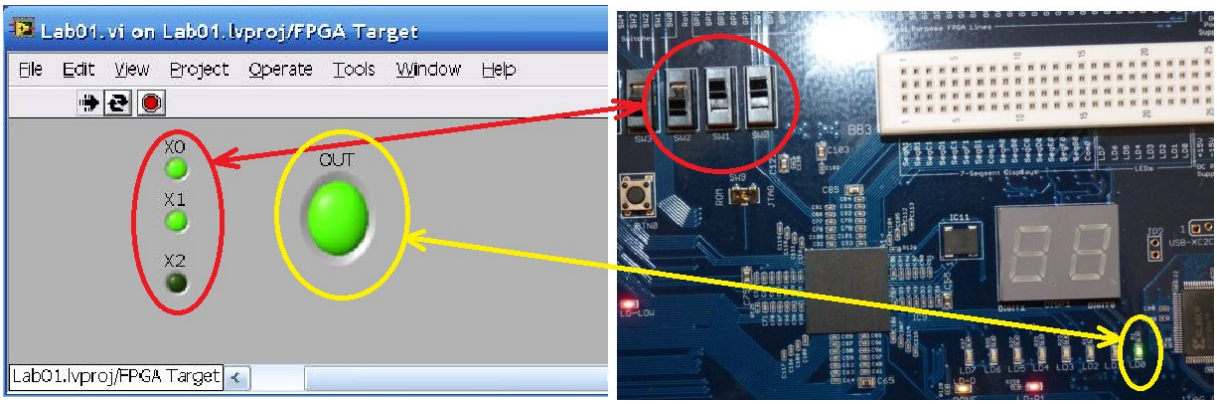


Рис. 3.14. Проверка результатов проектирования с помощью оценочного модуля

Остановить работу оценочного модуля в непрерывном режиме можно при помощи кнопки **Abort Execution** на лицевой панели.

Контрольное задание 3. Запустите собранную схему в непрерывном режиме. Используя разные комбинации включения переключателей **SW0÷SW2**, проверьте правильность работы схемы. Результаты проверки занесите в таблицу истинности.

Сравните получившиеся результаты с данными, приведенными в таблице истинности, описывающей данную схему (табл. 2.4). Результаты сравнения занесите в отчет по лабораторной работе №1.

Выводы по лабораторной работе №1

В этой работе:

- познакомились с образовательной платформой NI ELVIS II$^+$ и ситемой графического программирования LabVIEW;
- научились создавать элементарные функциональные блоки *sub-VI* на основе примитивов;
- научились проектировать иерархические схемы, реализующие заданные функции алгебры логики, с использованием созданных функциональных блоков;
- ознакомились с работой отладочной платы DE FPGA Board;
- приобрели навыки проверки результатов проектирования с использованием отладочной платы DE FPGA Board.

Отчет по лабораторной работе №1

Контрольное задание 1. Составьте таблицы истинности для логических элементов, реализующих операции **3И** и **4ИЛИ**:

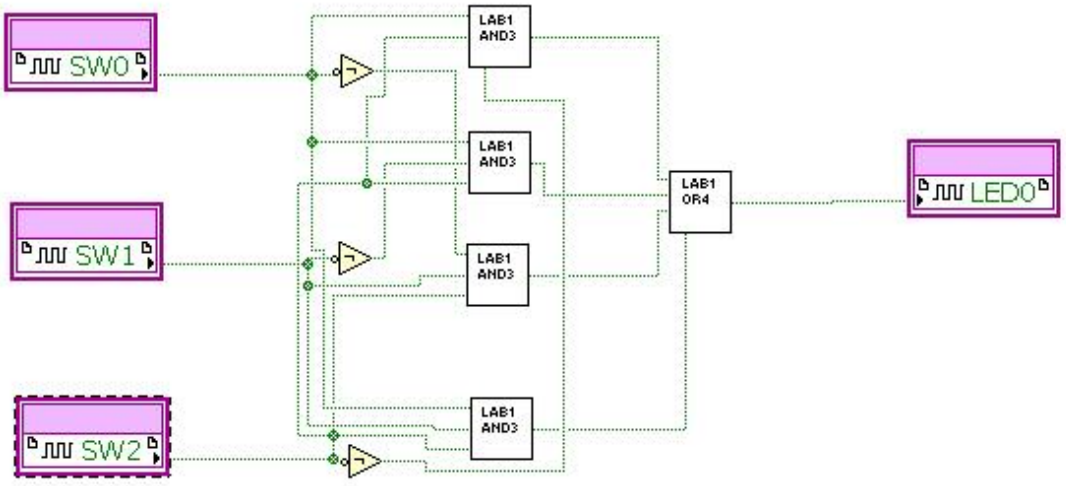
3И

x_2	x_1	x_0	z

4ИЛИ

x_3	x_2	x_1	x_0	z

Контрольное задание 2. Запишите ФАЛ, соответствующую блок–диаграмме, приведенной на рисунке



Контрольное задание 3. Используя разные комбинации включения переключателей **SW0÷SW2**, заполните таблицу истинности (слева):

x_2 (SW2)	x_1 (SW1)	x_0 (SW0)	z (LED0)
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

x_2	x_1	x_0	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Сравните получившийся результат с таблицей истинности, приведенной справа (*отметьте галочкой соответствующий результат*).

Таблицы совпадают — ☐ Таблицы не совпадают — ☐

Работу выполнил студент гр._____ _____ /_____/

«____» _____ 201__ г.

Оценка ____(балла(ов))

Работу принял _____ _____ /_____/

«____» _____ 201__ г.

Глава 4

МИНИМИЗАЦИЯ ФУНКЦИЙ АЛГЕБРЫ–ЛОГИКИ

В предыдущей главе было показано, что логическую схему, реализующую заданный алгоритм преобразования сигналов, можно синтезировать непосредственно по выражению, представленному в виде СДНФ или СКНФ. Однако полученная при этом схема, как правило, не оптимальна с точки зрения ее практической реализации. Поэтому исходные ФАЛ обычно минимизируют.

Целью минимизации логической функции является уменьшение стоимости ее технической реализации. Следует отметить, что сам критерий, в соответствии с которым выполняется минимизация ФАЛ, далеко не однозначен и зависит как от типа решаемой задачи, так и уровня развития технологии. Так, в те времена, когда цифровые устройства строились на дискретных элементах, минимизация числа этих элементов и числа построенных на их основе элементарных логических узлов однозначно определяла и уменьшение стоимости технической реализации. С появлением больших и сверхбольших интегральных схем (БИС и СБИС), стоимость которых определяется в основном площадью схемы на кристалле и мало зависит от числа входящих в нее транзисторов и других элементов, критерии минимизации ФАЛ претерпели существенные изменения. На первое место при проектировании самих ИС выдвигается требование регулярности их внутренней структуры и минимизация числа внешних соединений даже за счет увеличения числа элементов и внутренних соединений. Эти требования диктуются требованиями повышения надежности электронных средств.

Однако при проектировании аппаратуры с применением БИС и СБИС требование уменьшения числа корпусов ИС и количества их соединений между собой по-прежнему остается весьма важным.

Требование уменьшения числа элементарных ЛЭ, входящих в разрабатываемое устройство, в настоящее время также не потеряло своей актуальности. Объясняется это все более широким использованием при проектировании электронных средств программируемых логических СБИС широкого применения и полузаказных СБИС на основе базовых матричных кристаллов. Эти СБИС и БИС, как правило, содержат отдельные нескоммутированные между собой элементарные ЛЭ, например *Стрелка Пирса* или *Штрих Шеффера* (см. таблицу 2.3), или просто наборы транзисторов, резисторов и диодов, которые могут быть соединены между собой в соответствии с заданным алгоритмом обработки логических сигналов. Поскольку число элементов в одной СБИС задано из технологических соображений, то минимизация ФАЛ по критерию уменьшения числа используемых элементов позволяет на одном кристалле решать более сложные задачи логической обработки сигналов, т.е. в конечном счете уменьшать число требуемых ИС и связей между ними. Это снижает стоимость и повышает надежность

электронной аппаратуры.

Рассмотрим ряд методов, позволяющих провести минимизацию ФАЛ по критерию уменьшения числа элементарных ЛЭ.

4.1 Минимизация функций алгебры-логики при помощи кубических представлений

Наиболее просто и наглядно задача минимизации ФАЛ решается с использованием ее кубических представлений (см. раздел 2.3.5). Ранее было показано, что любая логическая функция n-переменных характеризуется своим кубическим комплексом $K(z)$, образованным кубическими комплексами $K_0, K_1, \ldots, K_{n-1}$. Из кубического комплекса $K(z)$ всегда можно выделить множество кубов $\Pi(z)$ таких, что каждый член комплекса K_0, т.е. вершина куба, будет включен по крайней мере в один куб из множества $\Pi(z)$. Множество кубов $\Pi(z)$ называется покрытием комплекса $K(z)$, или покрытием логической функции. Очевидно, что для любой ФАЛ существует несколько ее покрытий. В свою очередь, каждому покрытию $\Pi(z)$, так же как и самому комплексу, соответствует своя дизъюнктивная нормальная форма, получаемая логическим суммированием логических произведений, соответствующих выделенным кубам ФАЛ.

Сложность полученной таким образом ДНФ принято характеризовать понятием *«цена покрытия»* ($Ц_П$), которая равна сумме цен всех кубов, составляющих данное покрытие $\Pi(z)$: $Ц_П = \sum Ц_К$. В свою очередь, цена одного r-куба ФАЛ n-переменных определяется как разность полного числа входных переменных и ранга соответствующего куба, т.е. равна числу переменных в соответствующей дизъюнкции: $Ц_К = n - r$. Так, для ФАЛ трех переменных цена 0-куба равна трем, а 2-куба - единице.

В соответствии со сказанным, задача минимизации ФАЛ сводится к поиску покрытия $\Pi(z)$ кубического комплекса $K(z)$, имеющего минимальную цену.

Покрытие $\Pi(z)$ комплекса $K(z)$, имеющее минимальную цену, называется *покрытием Квайна*, а соответствующая этому покрытию ДНФ — называется *минимальной дизъюктивной нормальной формой* (МДНФ).

Пример 4.1 *Минимизировать ФАЛ, заданную в виде словесного описания в примере 2.1. Составить структурную схему логического устройства.*

Решение. Как было показано в примере 2.3, СДНФ для данной ФАЛ запишется в виде $z = \bar{x}_2 x_1 x_0 + x_2 \bar{x}_1 x_0 + x_2 x_1 \bar{x}_0 + x_2 x_1 x_0$.

Представим ФАЛ в виде трехмерного куба (рис. 4.1). Запишем кубический комплекс $K(z) = (011, 101, 110, 111, 11-, 1-1, -11)$. Нулевой кубический комплекс включает все вершины куба (зелёные точки на рис. 4.1):

$$\Pi_1(z) = K_0 = (011, 101, 110, 111).$$

Найдем цену покрытия: $Ц_{П_1} = \sum Ц_{К_0} = (3-0) + (3-0) + (3-0) + (3-0) = 12$.

Все вершины куба включаются так же в единичный кубический комплекс K_1 (ребра куба, выделенные красным на рис. 4.1), поэтому и он образует покрытие ФАЛ:

$$\Pi_2(z) = K_1 = (11-, 1-1, -11).$$

Найдем цену покрытия: $Ц_{П_2} = \sum Ц_{К_1} = (2-1) + (2-1) + (2-1) = 3$.

Перебирая сочетания кубов различных рангов, можно получить и другие покрытия ФАЛ, но здесь очевидно, что $\Pi_2(z)$ для данной ФАЛ будет иметь минимальную цену,

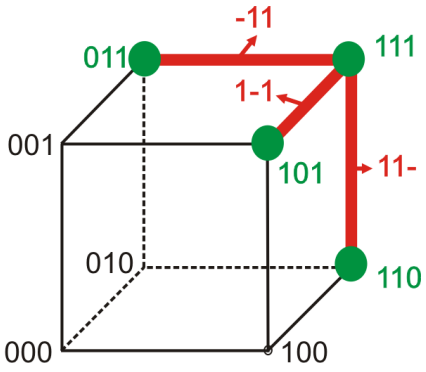


Рис. 4.1. Геометрическое представление кубического комплекса для ФАЛ

т. е. является покрытием Квайна. Соответствующая этому покрытию МДНФ запишется в виде:

$$z_2(x_2, x_1, x_0) = x_2x_1 + x_2x_0 + x_1x_0. \tag{4.1}$$

Анализируя получившееся выражение для ФАЛ, можно сказать, что для реализации этой функции в виде структурной схемы потребуются 3 логических элемента **2И** и один элемент **ЗИЛИ** (рис.4.2) в отличие от структурной схемы, представленной на рис 3.1, не потребуется ЛЭ реализующих операцию НЕ. ■

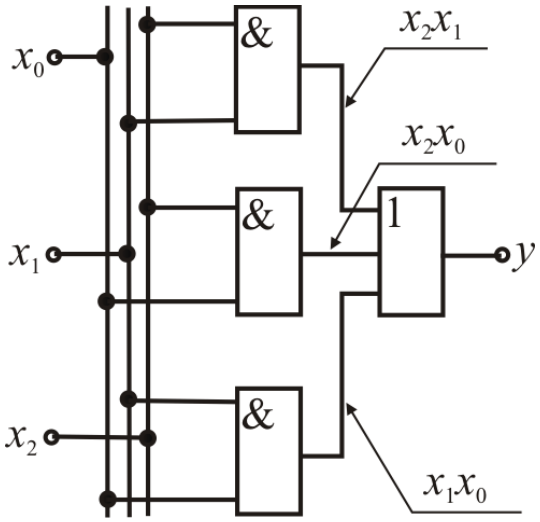


Рис. 4.2. Структурная схема логического устройства, реализующая МДНФ ФАЛ

Таким образом, полученная в результате минимизации функция и ее структурная схема проще. Техническая реализация такой схемы будет дешевле и надежней.

4.2 Минимизация функций алгебры–логики с использованием карт Вейча

Данный метод базируется на табличном представлении ФАЛ. Он широко используется при ручной, без применения ЭВМ, минимизации ФАЛ, число переменных в которой обычно не превышает пяти.

Карта Вейча — это прямоугольная таблица, число клеток в которой для ФАЛ n-переменных равно 2^n, каждой из клеток поставлен в соответствие некоторый набор

входных переменных, причем рядом расположенным клеткам соответствуют соседние наборы входных переменных (кодов), а в самих клетках записаны значения функций, определенные для этих кодов.

Рассмотрим построение карт Вейча для функций двух, трех и четырех переменных.

Карта Вейча функции двух переменных приведена в таблице 4.1. Она содержит четыре клетки и является плоской фигурой. Для удобства использования по краям карты указаны значения входных переменных, которые для соответствующих строк и столбцов остаются постоянными. Набор переменных для заданной клетки таблицы определяется как совокупность аргументов, постоянных для строк и столбцов, на пересечении которых она расположена.

Таблица 4.1. Карта Вейча функции двух переменных

	x_1	$\bar{x}_1$
x_0	$f(x_1, x_0)$	$f(\bar{x}_1, x_0)$
$\bar{x}_0$	$f(x_1, \bar{x}_0)$	$f(\bar{x}_1, \bar{x}_0)$

Карта Вейча функции трех переменных приведена в таблице 4.2. Она содержит восемь клеток. Наборы входных переменных, соответствующие крайним левому и правому столбцам, являются соседними. Поэтому данную карту удобно представить как поверхность цилиндра и она, в отличие от карты двух переменных, является объемной фигурой.

Таблица 4.2. Карта Вейча функции трёх переменных

	x_1	x_1	$\bar{x}_1$	$\bar{x}_1$
x_0	$f(\bar{x}_2, x_1, x_0)$	$f(x_2, x_1, x_0)$	$f(x_2, \bar{x}_1, x_0)$	$f(\bar{x}_2, \bar{x}_1, x_0)$
$\bar{x}_0$	$f(\bar{x}_2, x_1, \bar{x}_0)$	$f(x_2, x_1, \bar{x}_0)$	$f(x_2, \bar{x}_1, \bar{x}_0)$	$f(\bar{x}_2, \bar{x}_1, \bar{x}_0)$
	$\bar{x}_2$	x_2	x_2	$\bar{x}_2$

Карта Вейча функции четырех переменных приведена в таблице 4.3. Она содержит 16 клеток. Очевидно, что наборы входных переменных, соответствующие крайним левому и правому столбцам, как и в карте для трех переменных, являются соседними. Кроме этого соседние коды содержатся в нижней и верхней строках карты. Поэтому данная карта тоже является объемной фигурой и может быть представлена как поверхность тора.

Таблица 4.3. Карта Вейча функции четырёх переменных

	x_1	x_1	$\bar{x}_1$	$\bar{x}_1$	
x_0	$f(\bar{x}_3, \bar{x}_2, x_1, x_0)$	$f(x_3, \bar{x}_2, x_1, x_0)$	$f(x_3, \bar{x}_2, \bar{x}_1, x_0)$	$f(\bar{x}_3, \bar{x}_2, \bar{x}_1, x_0)$	$\bar{x}_2$
x_0	$f(\bar{x}_3, x_2, x_1, x_0)$	$f(x_3, x_2, x_1, x_0)$	$f(x_3, x_2, \bar{x}_1, x_0)$	$f(\bar{x}_3, x_2, \bar{x}_1, x_0)$	x_2
$\bar{x}_0$	$f(\bar{x}_3, x_2, x_1, \bar{x}_0)$	$f(x_3, x_2, x_1, \bar{x}_0)$	$f(x_3, x_2, \bar{x}_1, \bar{x}_0)$	$f(\bar{x}_3, x_2, \bar{x}_1, \bar{x}_0)$	x_2
$\bar{x}_0$	$f(\bar{x}_3, \bar{x}_2, x_1, \bar{x}_0)$	$f(x_3, \bar{x}_2, x_1, \bar{x}_0)$	$f(x_3, \bar{x}_2, \bar{x}_1, \bar{x}_0)$	$f(\bar{x}_3, \bar{x}_2, \bar{x}_1, \bar{x}_0)$	$\bar{x}_2$
	$\bar{x}_3$	x_3	x_3	$\bar{x}_3$	

Еще более сложную форму имеет карта Вейча функции пяти переменных. Ее можно представить как две карты Вейча функции четырех переменных, расположенные одна над другой и отличающиеся лишь значением одной переменной. Геометрически это можно представить как два тора, один из которых расположен в другом. Соседним кодам тут дополнительно соответствуют клетки, расположенные на разных торах

одна под другой. Ввиду сложности работы с такими картами данный способ редко используется при минимизации ФАЛ пяти переменных.

При минимизации ФАЛ используют либо ее нулевые, либо единичные значения. В обоих случаях получают равносильные выражения, которые, однако, могут отличаться по числу членов (т. е. цене) и выполняемым логическим операциям.

Алгоритм минимизации ФАЛ сводится к следующему.

1. На карте Вейча ФАЛ n–переменных выделяют прямоугольные области, объединяющие выбранные значения функции (лог. 0 или лог. 1). Каждая область должна содержать 2^k клеток, где k – целое число. Выделенные области могут пересекаться, т. е. одна или несколько клеток могут включаться в различные области.

2. Каждой из выделенных областей соответствует k-куб исходной ФАЛ, который представляется самостоятельным логическим произведением переменных, значения которых в рамках выделенной области остаются постоянными. Каждое произведение содержит $n-k$ переменных и носит название **импликанты**.

3. Из полученного множества выбирают минимальное число максимально больших областей, включающих все выбранные значения ФАЛ.

4. Логически суммируют импликанты, соответствующие выбранным областям. Полученная сумма образует МДНФ, т.е. является покрытием ФАЛ минимальной стоимости (покрытием Квайна).

При объединении клеток с единичными значениями ФАЛ получают МДНФ самой функции, а при объединении клеток с нулевыми значениями ФАЛ — МДНФ функции, инверсной заданной. Последнее легко объясняется при помощи тождеств, данных в таблице 2.2.

Пример 4.2 *Минимизировать ФАЛ, заданную в виде словесного описания в примере 2.1 при помощи карты Вейча. Получить МДНФ для самой функции и для функции, инверсной заданной. Сравнить цены покрытия.*

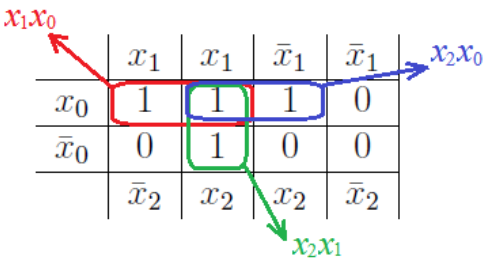
Решение. В примере 2.3 была получена СДНФ для данной ФАЛ $\to$
$z = \bar{x}_2x_1x_0 + x_2\bar{x}_1x_0 + x_2x_1\bar{x}_0 + x_2x_1x_0$. Составим карту Вейча для заданной функции

	x_1	x_1	$\bar{x}_1$	$\bar{x}_1$
x_0	$f(\bar{x}_2, x_1, x_0) = 1$	$f(x_2, x_1, x_0) = 1$	$f(x_2, \bar{x}_1, x_0) = 1$	$f(\bar{x}_2, \bar{x}_1, x_0) = 0$
$\bar{x}_0$	$f(\bar{x}_2, x_1, \bar{x}_0) = 0$	$f(x_2, x_1, \bar{x}_0) = 1$	$f(x_2, \bar{x}_1, \bar{x}_0) = 0$	$f(\bar{x}_2, \bar{x}_1, \bar{x}_0) = 0$
	$\bar{x}_2$	x_2	x_2	$\bar{x}_2$

Для удобства дальнейшей работы перепишем карту Вейча в виде:

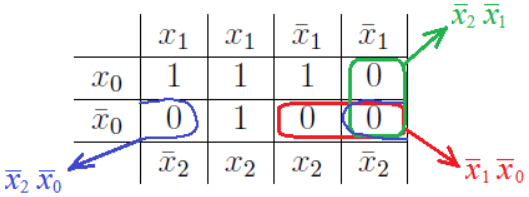
	x_1	x_1	$\bar{x}_1$	$\bar{x}_1$
x_0	1	1	1	0
$\bar{x}_0$	0	1	0	0
	$\bar{x}_2$	x_2	x_2	$\bar{x}_2$

Получим МДНФ самой функции, т. е. проведем минимизацию по тем значениям функции, на которых она равна лог. 1. Согласно приведенному выше алгоритму, на данной карте Вейча можем выделить три единичных куба.



Первый — x_1x_0; второй — x_2x_1; третий — x_2x_0. Таким образом, МДНФ самой функции будет: $z = x_2x_1 + x_2x_0 + x_1x_0$, т. е. получили тот же результат, что и в примере 2.7.

Получим МДНФ для функции, инверсной заданной. При выделении кубов учтем, что наборы входных переменных, соответствующие крайним левому и правому столбцам, являются соседними.



В этом случае так же, как и в предыдущем, можем выделить только три единичных куба: первый — $\bar{x}_1\bar{x}_0$; второй — $\bar{x}_2\bar{x}_1$; третий — $\bar{x}_2\bar{x}_0$. Таким образом, МДНФ обратной функции будет:

$$\bar{z} = \bar{x}_2\bar{x}_1 + \bar{x}_2\bar{x}_0 + \bar{x}_2\bar{x}_0. \tag{4.2}$$

Нетрудно убедиться, что цена покрытия МДНФ обратной функции так же, как и самой функции, равна 3, т. е. они равнозначны.

Как было показано в главе 2 дизъюкктивная и конъюнктивная нормальные формы являются равнозначными. Покажем это на данном примере.

Для доказательства инвертируем обе части равенства 4.2:

$$\overline{\bar{z}} = \overline{\bar{x}_2\bar{x}_1 + \bar{x}_2\bar{x}_0 + \bar{x}_1\bar{x}_0}.$$

Используя законы ассоциативности, основные тождества и теоремы Де–Моргана, запишем:

$$\overline{\bar{z}} = \overline{\bar{x}_2\bar{x}_1} \cdot \overline{\bar{x}_2\bar{x}_0 + \bar{x}_1\bar{x}_0} = \overline{\bar{x}_2\bar{x}_1} \cdot \overline{\bar{x}_2\bar{x}_0} \cdot \overline{\bar{x}_1\bar{x}_0} = (\overline{\bar{x}}_2 + \overline{\bar{x}}_1) \cdot (\overline{\bar{x}}_2 + \overline{\bar{x}}_0) \cdot (\overline{\bar{x}}_1 + \overline{\bar{x}}_0).$$

Окончательно получим МКНФ:

$$z = (x_2 + x_1) \cdot (x_2 + x_0) \cdot (x_1 + x_0). \tag{4.3}$$

Нетрудно убедиться, что аналогичное выражение мы могли бы получить, просто произведя логическое умножение Макстермов логической функции. ■

Лабораторная работа №2.
Реализация минимальной дизъюктивной и минимальной конъюнктивной нормальных форм в виде логического устройства

Цель работы:

Целью лабораторной работы является создать в LabVIEW логические схемы, реализующие МДНФ (4.1) и МКНФ (4.3) фукции алгебры–логики, полученные в разделе 4.1. Результаты проектирования проверить на оценочном модуле DE FPGA и системе NI ELVIS II^{+}.

Выполнение задания:

Синтезированные схемы должны быть иерархическими; используя функции низкого уровня, необходимо создать т. н. *sub-VI* — блоки, которые затем послужат функциональными блоками более высокого уровня.

ЛР2.1. Синтез логического устройства, описанного МДНФ

Схема реализующая МДНФ ФАЛ (4.1), представлена на рис. 4.2. Для её синтеза понадобятся логические элементы реализующие: операцию И над двумя входными переменными (**2AND**); операцию ИЛИ над тремя входными выражениями (**3OR**).

Порядок выполнения упражнения:

- запустите LabVIEW на персональном компьютере;
- в своей дирректории создайте новую папку `Lab02`;
- используя методики, освоенные в процессе выполнения лабораторной работы №1, создайте требуемые функциональные *sub-VI* — блоки в среде графического проектирования;
- используя указанные функциональные блоки, соберите логическую схему, представленную на рис. 4.2. Входные переменные x_2, x_1, x_0 должны задаваться при помощи движковых переключателей **SW2**, **SW1**, **SW0** соответственно. Состояние выходного сигнала (z) должно отображаться при помощи светодиода **LED0**;
- сохраните новый проект под именем `Lab02_1`;
- запустите проект в непрерывном режиме. Используя разные комбинации включения переключателей **SW0**÷**SW2**, проверьте правильность работы схемы. Результаты проверки занесите в таблицу;
- сравните получившиеся результаты с данными, полученными в ходе выполнения лабораторной работы №1;
- результаты сравнения занесите в отчет о выполнении лабораторной работы №2;

ЛР2.2. Синтез логического устройства, описанного МКНФ

Порядок выполнения упражнения:

- проаналируйте МКНФ ФАЛ (4.3). Самостоятельно синтезируйте логическую схему, реализующую данное выражение. Для обозначения логических операций используйте условные графические обозначения согласно ГОСТ 2.743-91;
- занесите схему в отчет по лабораторной работе №2;
- создайте в LabVIEW новый проект и сохраните его под именем `Lab02_2` в папку `Lab02`;
- создайте требуемые функциональные *sub-VI* — блоки в среде графического проектирования;
- используя созданные функциональные блоки, соберите логическую схему, реализующую МКНФ ФАЛ (4.3). Входные переменные x_2, x_1, x_0 должны задаваться при помощи движковых переключателей **SW2**, **SW1**, **SW0** соответственно. Состояние выходного сигнала (z) должно отображаться при помощи светодиода **LED0**;
- сохраните и откомпилируйте проект;
- запустите собранную схему в непрерывном режиме (т. е. с использованием кнопки **Run Continuosly**). Используя разные комбинации включения переключателей **SW0**÷**SW2**, проверьте правильность работы схемы. Результаты проверки занесите в таблицу;
- сравните получившиеся результаты с результатами, полученными в ходе выполнения пункта ЛР2.1;
- результаты сравнения занесите в отчет о выполнении лабораторной работы №2.

Выводы по лабораторной работе №2

В этой работе:

- познакомились с принципами и некоторыми методами минимизации ФАЛ;
- научились самостоятельно синтезировать логические схемы, реализующие ФАЛ;
- убедились в эквивалентности представления ФАЛ в виде дизъюнктивной нормальной формы и коньюнктивной нормальной формы.

Отчет по лабораторной работе №2

Контрольное задание 1. Используя разные комбинации включения переключателей **SW0÷SW2**, проверьте правильность работы схемы, собранной по МДНФ ФАЛ. Результаты проверки занесите в таблицу:

x_2 (SW2)	x_1 (SW1)	x_0 (SW0)	z (LED0)
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Сравните эту таблицу с таблицей истинности, полученной в лабораторной работе №1 (*отметьте галочкой соответствующий результат*).

Таблицы совпадают — ☐ Таблицы не совпадают — ☐

Контрольное задание 2. Нарисуйте логическую схему, реализующую МКНФ ФАЛ (4.3). Для обозначения логических операций используйте условные графические обозначения согласно ГОСТ 2.743-91.

Контрольное задание 3. Используя разные комбинации включения переключателей **SW0÷SW2**, проверьте правильность работы схемы, собранной по МКНФ ФАЛ. Результаты проверки занесите в таблицу:

x_2 (SW2)	x_1 (SW1)	x_0 (SW0)	z (LED0)
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Сравните получившиеся результаты с результатами, полученными в ходе выполнения упражнения ЛР2.1 (*отметьте галочкой соответствующий результат*).

Таблицы совпадают — ☐ Таблицы не совпадают — ☐

Объясните результат (*в письменной форме*):

_____

_____

Работу выполнил студент гр._____ _____ /_____/

«____» _____ 201__ г.

Оценка ____(балла(ов))

Работу принял _____ _____ /_____/

«____» _____ 201__ г.

4.3 Минимизация системы функций алгебры логики

В общем случае на выходе логического устройства формируется m–разрядный двоичный код (см. рис. 2.1), таким образом, его поведение описывается системой, состоящей из M ФАЛ. Минимизация структуры такого устройства может быть выполнена с использованием вышеприведенных методов при раздельной минимизации M структур, на выходе каждой из которых формируется только один выходной сигнал. Однако с точки зрения всего устройства такая структура, как правило, не будет оптимальной.

С точки зрения минимизации всей структуры необходимо, чтобы цепь формирования каждого выходного сигнала была выполнена не минимальным, а некоторым оптимальным способом, обеспечивающим, в конечном счете, минимальность общей структуры устройства. Минимизация в этом случае обеспечивается за счет использования общих цепей формирования сигнала для получения нескольких выходных функций. Последнее достигается выделением на картах Вейча различных выходных функций одинаковых областей.

Пример 4.3 *Минимизировать структуру устройства, алгоритм работы которого задан следующей таблицей истинности:*

Таблица 4.4. Таблица истинности устройства с тремя логическими выходами

Вход			Выход		
x_2	x_1	x_0	z_2	z_1	z_0
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	1	1	1
1	0	1	0	0	1
1	1	0	1	1	1
1	1	1	0	1	0

Решение. Нарисуем карты Вейча для каждой ФАЛ, входящей в заданную систему. Минимизируем данную систему ФАЛ по каждому выходу отдельно по известному алгоритму, как это показано на рис. 4.3.

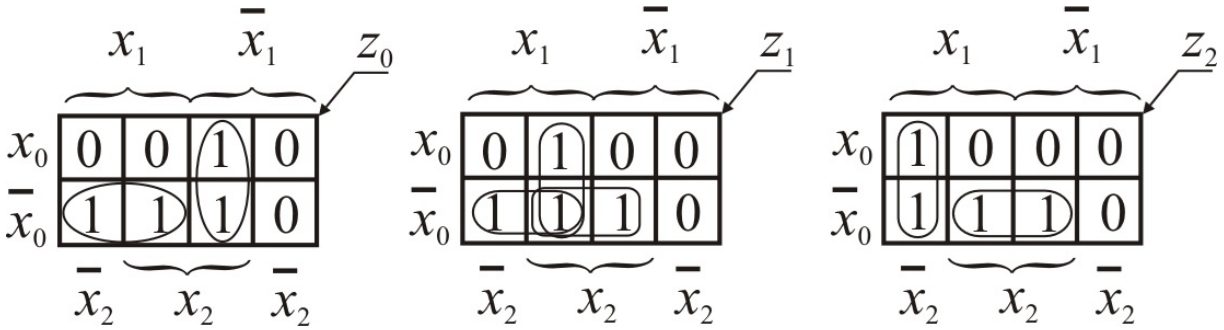


Рис. 4.3. Карты Вейча для системы ФАЛ, пример 4.3

Используя приведенные на рис. 4.3 карты Вейча, для заданной таблице истинности можно записать следующую систему минимальных ФАЛ:

$$\begin{aligned} z_0 &= x_1\bar{x}_0 + x_2\bar{x}_1, \\ z_1 &= x_1\bar{x}_0 + x_2\bar{x}_0 + x_2x_1, \\ z_2 &= \bar{x}_2x_1 + x_2\bar{x}_0. \end{aligned} \tag{4.4}$$

Техническая реализация данной системы потребует семь элементов **2И**, два элемента **2ИЛИ** и один элемент **3ИЛИ**, т.е. всего десять элементов (вариант решения задачи «в лоб»).

Внимательно рассмотрев систему ФАЛ (4.4) и карты Вейча (рис. 4.3), можно найти еще два варианта решения поставленной задачи.

Первый вариант.

Нетрудно заметить, что полученные выражения (4.4) содержат общие члены $x_1 \cdot \overline{x_0}$ и $x_2 \cdot \overline{x_0}$. Поэтому техническую реализацию устройства можно упростить. При использовании общих для нескольких элементов выходов для реализации потребуется: пять элементов **2И**, два элемента **2ИЛИ** и один элемент **3ИЛИ**, т. е. всего восемь элементов.

Второй вариант.

Анализ приведенных на рис. 4.3 карт Вейча показывает, что на входных кодах 010, 100 и 110 все три функции принимают единичное значение. Поэтому можно записать:

$$\begin{aligned} z_0 &= x_1\bar{x}_0 + x_2\bar{x}_0 + x_2\bar{x}_1 = \bar{x}_0\,(x_2 + x_1) + x_2\bar{x}_1, \\ z_1 &= x_1\bar{x}_0 + x_2\bar{x}_0 + x_2x_1 = \bar{x}_0\,(x_2 + x_1) + x_2x_1, \\ z_2 &= x_1\bar{x}_0 + x_2\bar{x}_0 + \bar{x}_2x_1 = \bar{x}_0\,(x_2 + x_1) + \bar{x}_2x_1. \end{aligned} \tag{4.5}$$

Реализация этой схемы потребует четыре элемента 2И и четыре элемента 2ИЛИ, т.е. всего также восемь элементов. Однако из схемы исключен трехвходной элемент, что, в конечном счете, приводит к упрощению ее технической реализации.

■

Таким образом, выделение при минимизации системы ФАЛ общих областей на картах Вейча позволяет получить наиболее простую ее техническую реализацию. При этом следует иметь ввиду, что общие области могут выделяться не на всех картах, а лишь на части из них. Как правило, это приводит к упрощению технической реализации.

Лабораторная работа №3. Синтез оптимальной схемы комбинацонного логического устройства

Цель работы:

Целью лабораторной работы является синтезировать в LabVIEW логические схемы, реализующие системы фукций алгебры–логики, полученные в разделе 4.3. Результаты проектирования проверить на оценочном модуле DE FPGA Bord и системе NI ELVIS II$^+$.

Задание на проектирование:

1. Синтезировать схему логического устройства, описанную системой уравнений 4.4.
2. Упростить полученную на предыдущем этапе схему с учетом того, что в систему ФАЛ 4.4 входит ряд общих членов.
3. Синтезировать схему логического устройства, описанную системой уравнений 4.5.

В проектируемой схеме *три входа* и *три выхода*. Входные логические переменные x_2, x_1, x_0 должны задаваться при помощи движковых переключателей **SW2**, **SW1**, **SW0** соответственно. Состояние выходных логических переменных z_2, z_1, z_0 должно отображаться при помощи светодиодов **LED2**, **LED1**, **LED0** соответственно.

Выполнение заданий:

В лабораторных работах №1 и №2 был создан ряд простейших функциональных узлов (логических элементов), которые могут быть использованы при выполнении настоящей лабораторной работы. Для этого в своей директории создайте папку с именем `Lab03` и скопируйте в неё из папок `Lab01` и `Lab02` *Sub-VI* — блоки двух– и трехвходовых элементов (И и ИЛИ), созаданных в ходе выполнения предыдущих лабораторных работ.

ЛР3.1. Синтез логического устройства, описанного тремя функциями алгебры–логики

Порядок выполнения упражнения:

- проаналируйте систему ФАЛ (4.4). Самостоятельно синтезируйте логическую схему, реализующую данную систему логических функций. Для обозначения логических операций используйте условные графические обозначения согласно ГОСТ 2.743-91;
- занесите схему в отчет по лабораторной работе №3;
- включите питание образовательной платформы NI ELVIS II$^+$ с установленной на ней оценочной платой DE FPGA Board. Запустите на ПК пакет LabVIEW;
- создайте в LabVIEW новый проект и сохраните его под именем `Lab03_1` в папку `Lab03`;
- используя готовые функциональные *sub-VI* — блоки, соберите логическую схему, реализующую систему ФАЛ (4.4);
- сохраните и откомпилируйте проект;
- запустите собранную схему в непрерывном режиме (т. е. с использованием кнопки **Run Continuosly**). Используя разные комбинации включения переключателей **SW0÷SW2**, проверьте правильность работы схемы. Результаты проверки занесите в таблицу.
- сравните полученный результат с таблицей 4.4;
- результаты сравнения занесите в отчет о выполнении лабораторной работы №3.

ЛР3.2. Синтез упрощенной схемы логического устройства

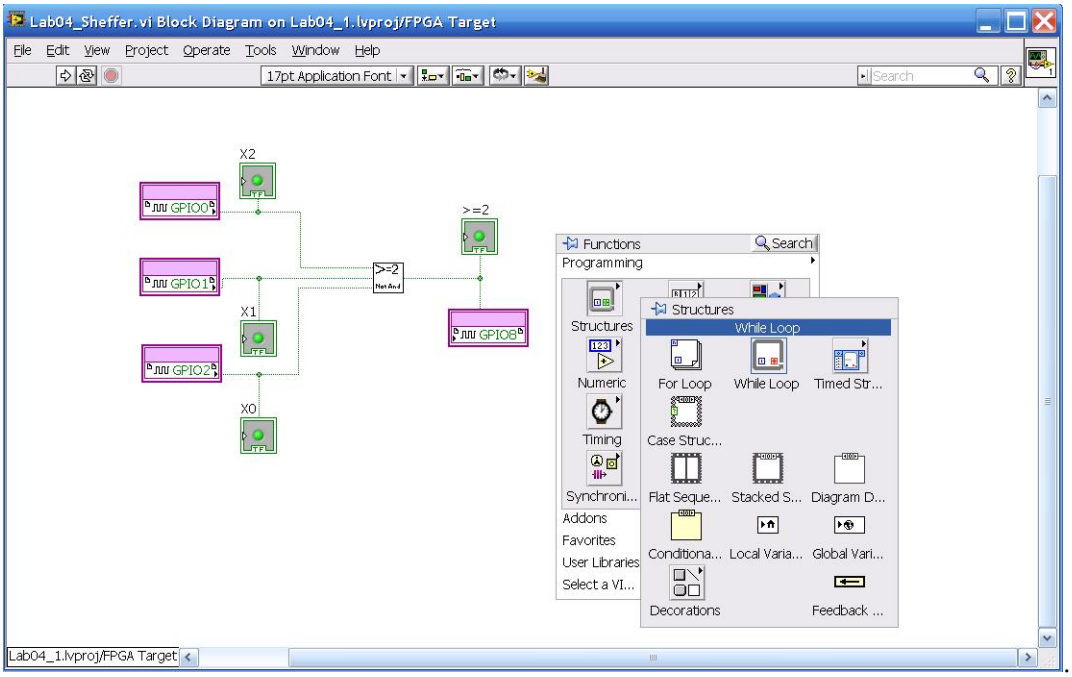
До настоящего упражнения мы запускали синтезированные схемы при помощи кнопки **Run Continuosly**. Данная конфигурация подразумевает, что собранные схемы тактируются от внешнего источника синхронизации, т. е. компьютера. Если в такой конфигурации отключить USB–кабель (см. рис. 1.4), то программа и логическое устройство работать перестанут. Проверим это:

- запустить созданную в п. ЛР3.1 схему логического устройства в непрерывном режиме (**Run Continuosly**);
- выдернуть USB–кабель из разъема 17 (рис. 1.2);
- результат работы устройства занести в отчет по лабораторной работе №3;
- вставьте USB–кабель обратно в разъем 17.

Однако реальные цифровые логические устройства должны работать без помощи персонального компьютера. Для этого на «борту» DE FPGA Board имееется собственный задающий тактовый генератор. Для того, чтобы программа запускалась в непрерывном режиме (т. е. по нажатии кнопки **Run**), необходимо задействовать внутренний генератор тактовых импульсов ПЛИС. Выполним эту операцию в настоящем упражнении.

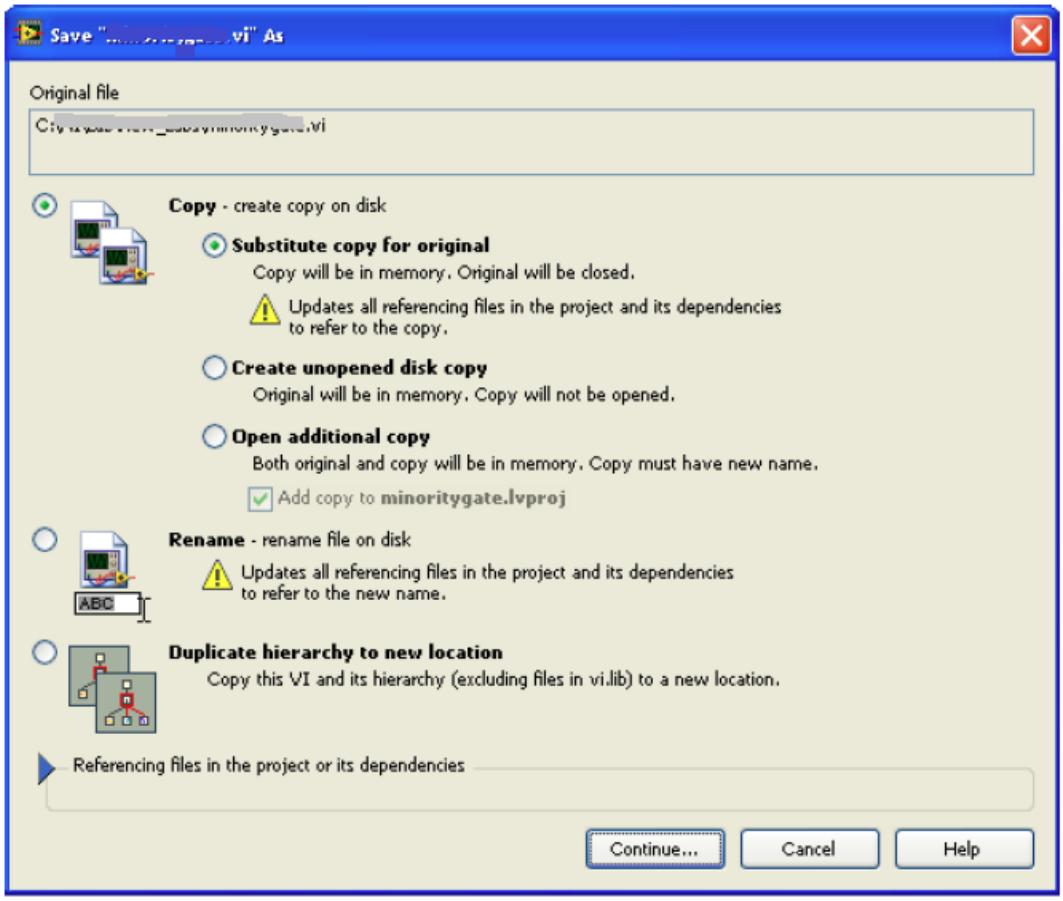
Порядок выполнения упражнения:

- упростите полученную в п. ЛР3.1 схему с учетом того, что все уравнения системы ФАЛ 4.4 содержат общие члены $x_1 \cdot \overline{x_0}$ и $x_2 \cdot \overline{x_0}$;
- синтезируйте логическую схему, реализующую данную систему логических функций. Для обозначения логических операций используйте условные графические обозначения согласно ГОСТ 2.743-91;
- занесите схему в отчет по лабораторной работе №3.
- переконфигурируйте схему, созданную в проекте `Lab03_1` согласно новой схемы;
- щелкните правой кнопкой мышки где-нибудь в рабочей области окна «Block Diagramm» и выберите из палитры **Structures** структуру **While loop**.



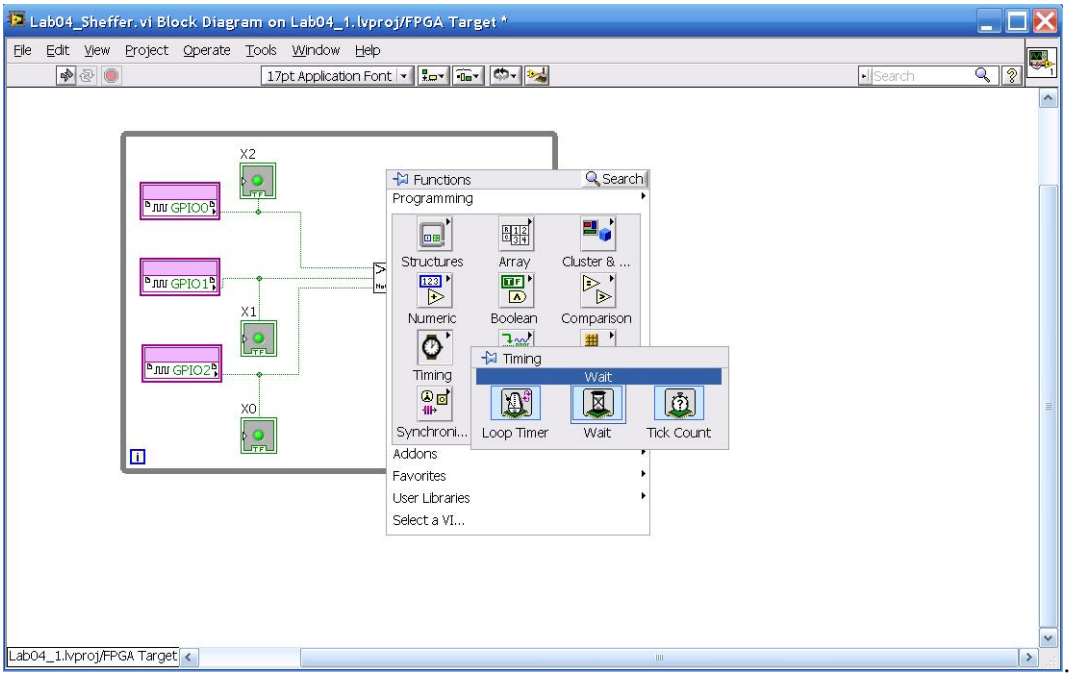
Нарисуйте прямоугольник, заключающий в себе созданную блок–диаграмму;

- сохраните новый *VI* командой меню **File→Save As** под именем `Lab03_2`. Появится окно:

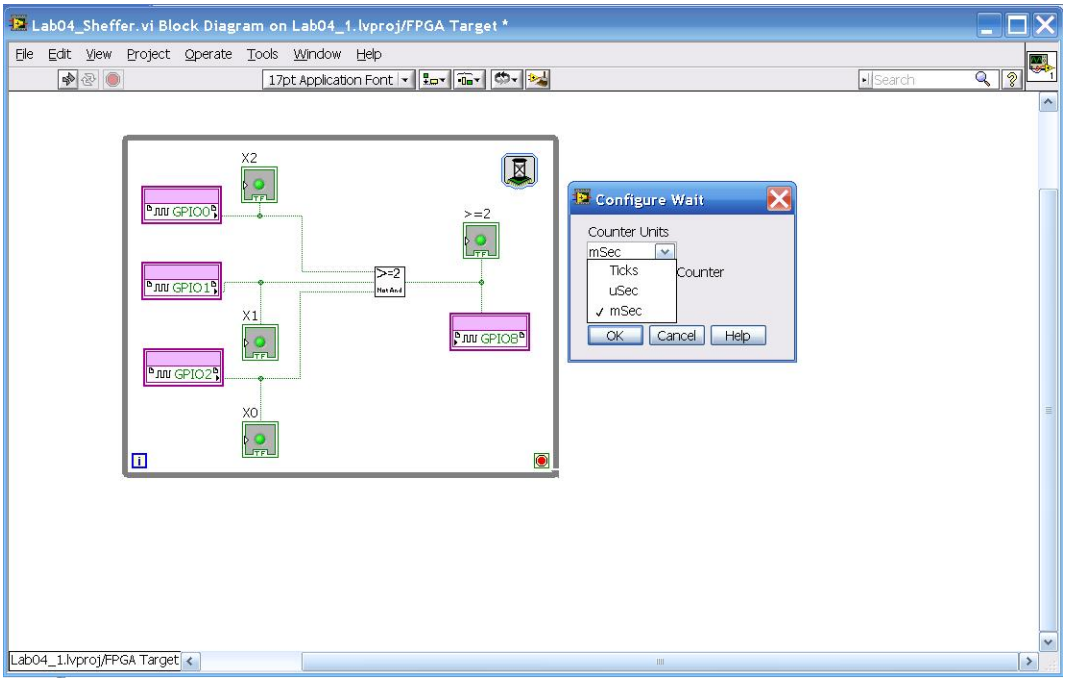


Щелкните по кнопке **Continue. . .**;

- щелкните правой кнопкой мыши в окне **Block–Diagram** и добавьте функцию **Wait** из палитры **Programming→Timing**.

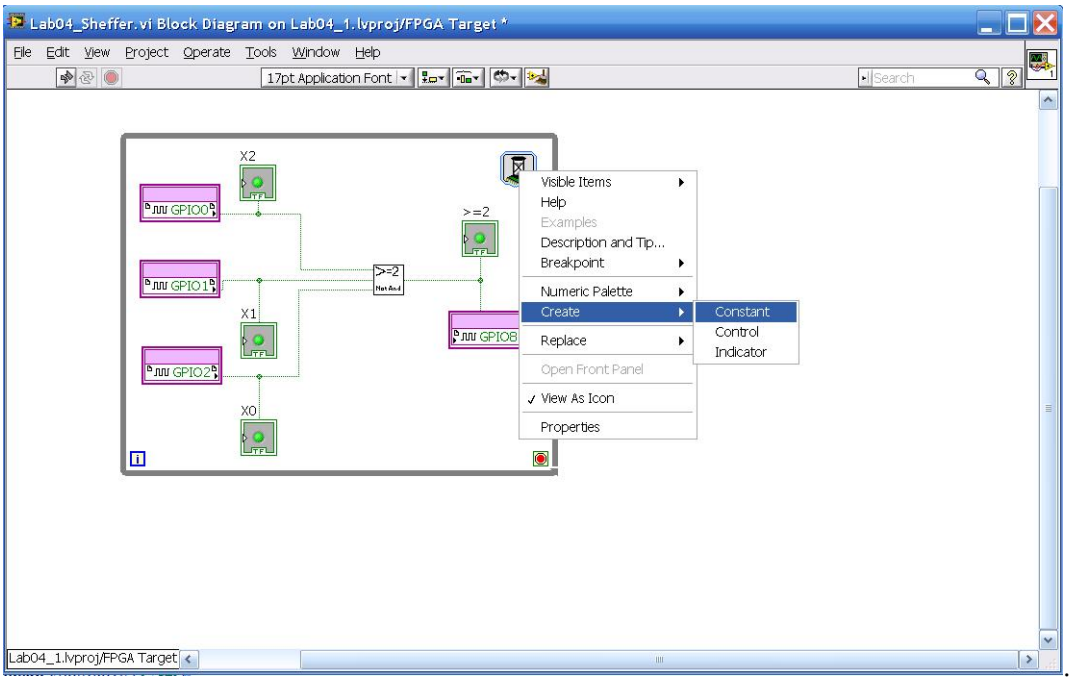


Выберите единицу счета *mSec*



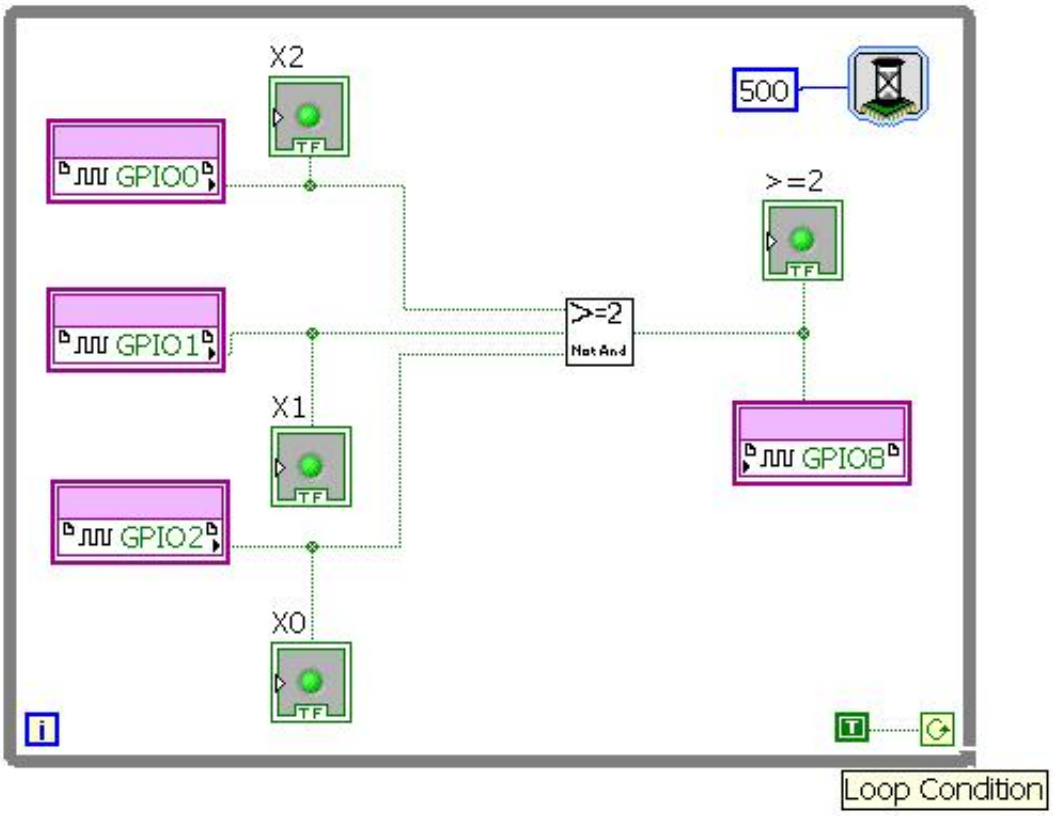
и нажмите кнопку **OK**;

- щелкните правой кнопкой мыши по пиктограмме **Wait** и выберите **Create→Constant**

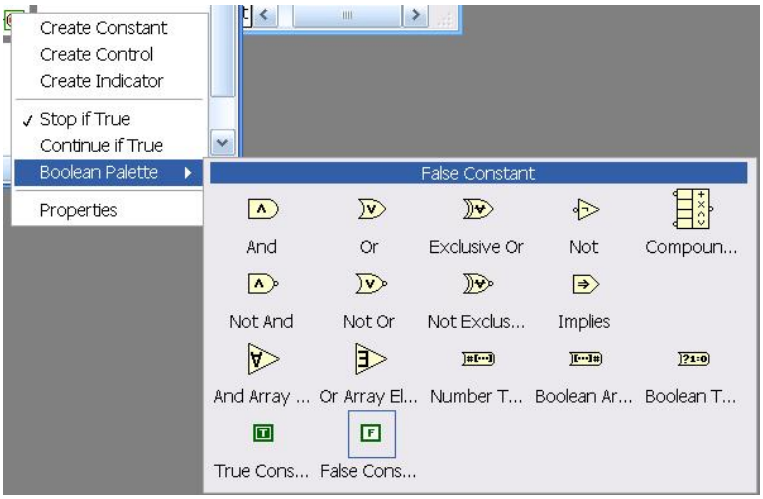


Измените значение константы с 0 на 500;

- щелкните правой кнопкой мыши по терминалу **Loop Condition**

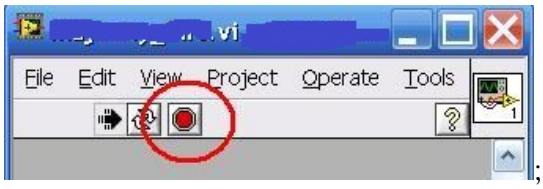


и выберите из палитры **Boolean** константу,



например, **False Constant** (F). В этом случае терминал **Loop Condition** должен быть красного цвета, если выбрать **True Constant** (T), то терминал **Loop Condition** должен быть белого цвета. Переключение цветов (т. е. состояние терминала **Loop Condition**) осуществляется при помощи мышки;

- запустите схему при помощи кнопки **Run**. Работающую в непрерывном режиме схему можно отключить при помощи кнопки **Stop**



;

- используя разные комбинации включения переключателей **SW0÷SW2**, проверьте правильность работы схемы. Результаты проверки занесите в таблицу;
- сравните полученный результат с таблицей 4.4;
- результаты сравнения занесите в отчет о выполнении лабораторной работы №3;
- удалите USB–кабель из разъема 17 (рис. 1.2). На экране компьютера появится сообщение об ошибке. Виртуальный прибор (*VI*), созданный в системе LabVIEW, прекратит свою работу, однако прошивка ПЛИС на DE FPGA Board продолжит свою работу. Убедитесь в этом, задавая различные входные коды при помощи движковых переключателей **SW0÷SW2**;
- результат работы прибора занесите в отчет по лабораторной работе №3;
- закройте проект `Lab03_2`;
- вставьте USB–кабель обратно в разъем 17.

ЛР3.3. Синтез оптимальной схемы логического устройства

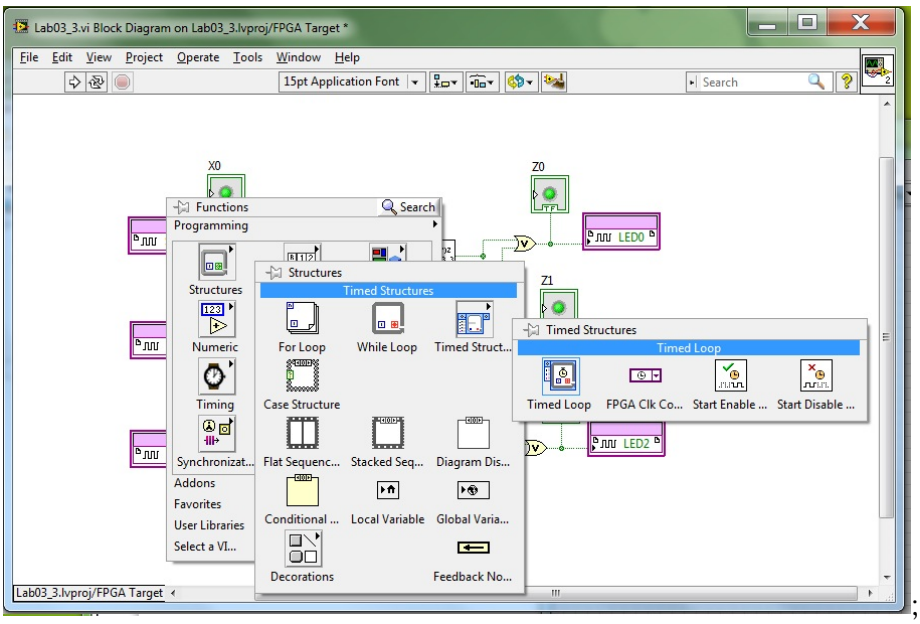
Спроектированная в п. ЛР3.2 схема работала как тактируемая, т. к. код логики был включен в структуру **While loop**. Это увеличивает задержку, поскольку за один такт синхронизации операции выполняются только на одном логическом уровне. Кроме этого, необходимы еще два такта синхронизации на осуществление задержки **While loop**. Если необходимо, чтобы созданная схема или ее часть работала как комбинационное устройство[1], необходимо заключить код в структуру **Single-Cycle Timed Loop**.

В этом упражнении рассмотрим создание комбинационной схемы логического устройства, реализующего систему ФАЛ (4.5).

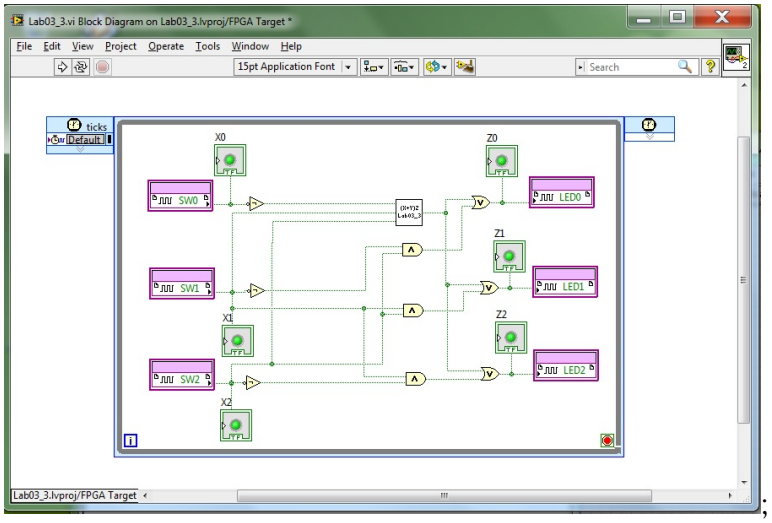
Порядок выполнения упражнения:

- проаналируйте систему ФАЛ (4.5). Самостоятельно синтезируйте оптимальную схему логического устройства. Для обозначения логических операций используйте условные графические обозначения согласно ГОСТ 2.743-91;
- занесите схему в отчет по лабораторной работе №3;
- создайте в LabVIEW новый проект и сохраните его под именем `Lab03_3`;
- соберите в LabVIEW схему согласно синтезированной оптимальной схемы и сохраните проект;
- создайте структуру **Single-Cycle Timed Loop**. Для этого щелкните правой кнопкой мышки в окне «Block Diagramm», в открывшемся окне «Functions» нажмите на кнопку **Structures**, в появившемся меню выберите пункт **Timed Structures**, где щелкните по пункту **Timed Loop**

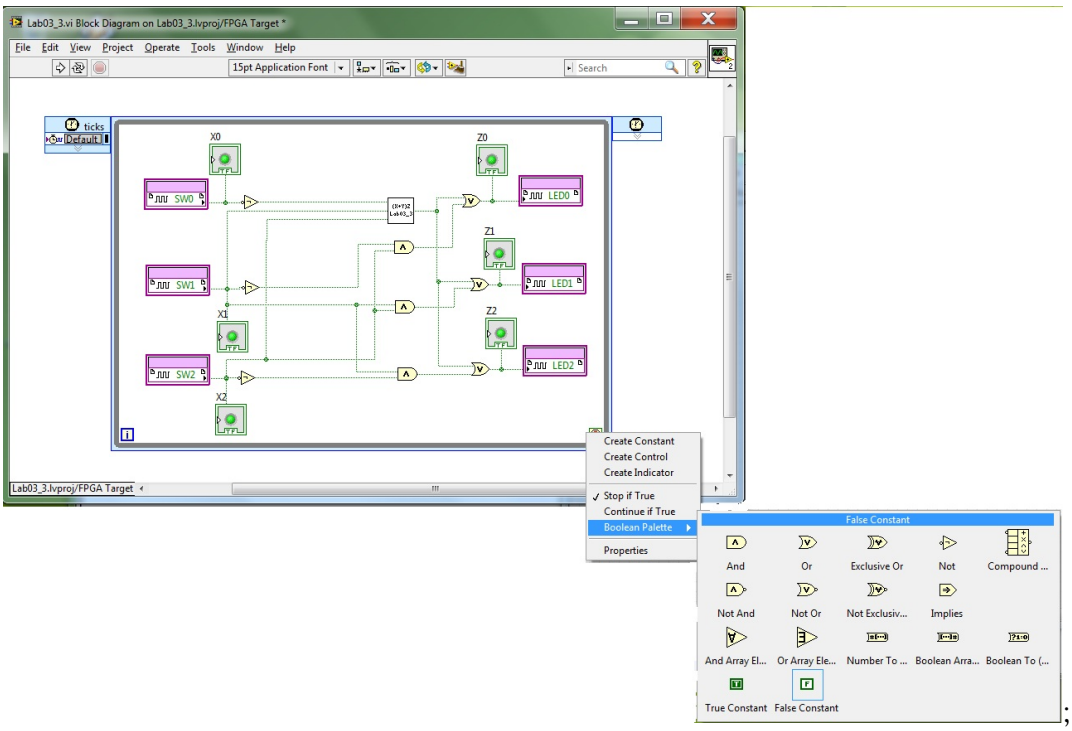
[1] Разработка типовых схем комбинационных логических устройств будет рассмотрена во второй части лабораторного практикума.



- появится инструмент для рисования временного цикла. С помощью этого инструмента заключите схему логического устройства в **Single-Cycle Timed Loop** структуру. Блок–диаграмма должна выглядеть так:



- добавьте константу для управления циклом, как это было описано в п. ЛР3.2



;

- сохраните проект и нажмите кнопку **Run**;
- используя разные комбинации включения переключателей **SW0÷SW2**, проверьте правильность работы схемы. Результаты проверки занесите в таблицу;
- сравните полученный результат с таблицей 4.4. Убедитесь, что результаты совпадают с полученными в пп. ЛР3.1 и ЛР3.2;
- результаты сравнения занесите в отчет о выполнении лабораторной работы №3;
- удалите USB–кабель из разъема 17 (рис. 1.2). На экране компьютера появится сообщение об ошибке. Виртуальный прибор (*VI*), созданный в системе LabVIEW, прекратит свою работу, однако прошивка ПЛИС на DE FPGA Board продолжит работу. Убедитесь в этом, задавая различные входные коды при помощи движковых переключателей **SW0÷SW2**;
- результат работы прибора занесите в отчет по лабораторной работе №3;
- закройте проект `Lab03_3`;
- вставьте USB–кабель в разъем 17.

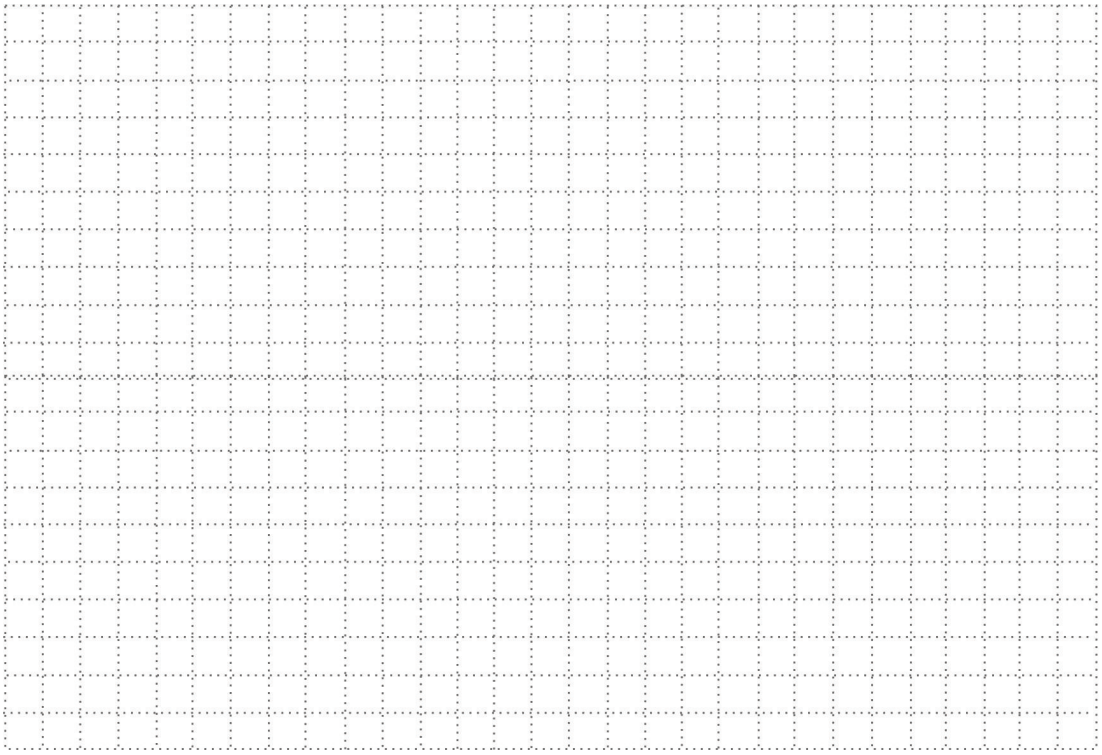
Выводы по лабораторной работе №3

В этой работе:

- познакомились с принципами оптимизации структуры логических устройств, описываемых системой ФАЛ;
- на практике убедились в идентичности результатов, получаемых при различных подходах, применяемых при минимизации системы логических уравнений;
- научились создавать в системе LabVIEW тактируемые и комбинационные схемы логических устройств.

Отчет по лабораторной работе №3

Контрольное задание 1. Нарисуйте логическую схему, реализующую систему ФАЛ (4.4). Для обозначения логических операций используйте условные графические обозначения согласно ГОСТ 2.743-91.



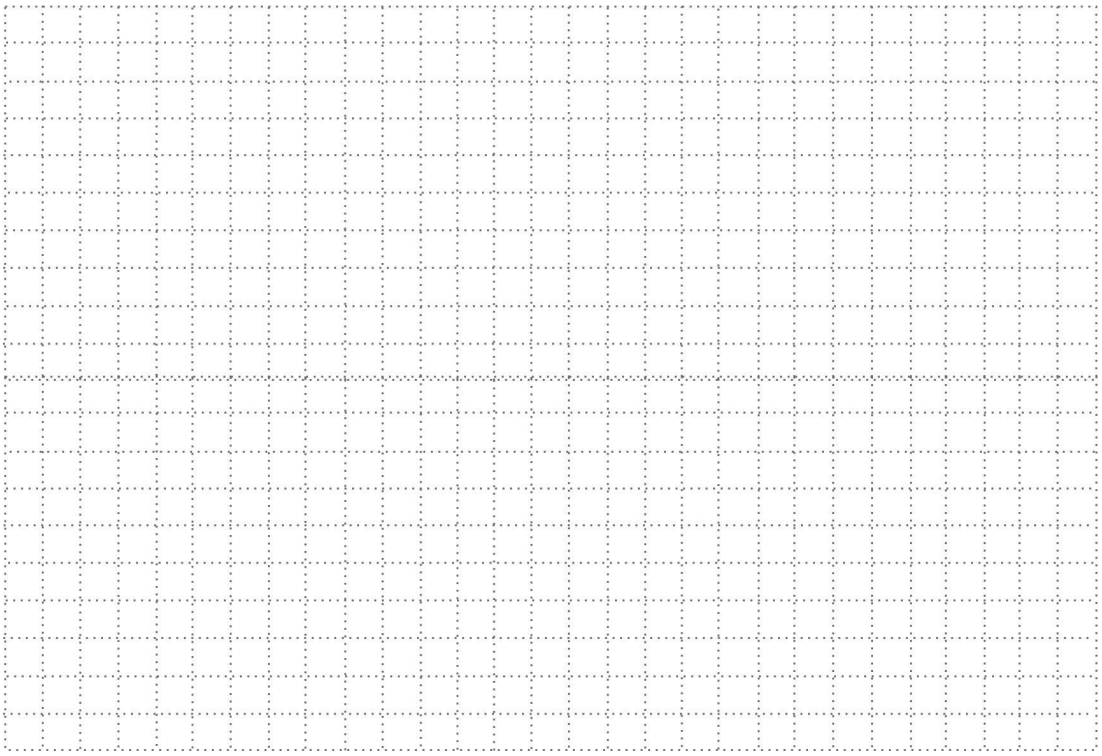
Используя разные комбинации включения переключателей **SW0÷SW2**, проверьте правильность работы схемы. Результаты проверки занесите в таблицу:

Вход			Выход		
x_2 (SW2)	x_1 (SW1)	x_0 (SW0)	z_2 (LED2)	z_1 (LED1)	z_0 (LED0)
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Сравните полученный результат с таблицей 4.4 (*отметьте галочкой соответствующий результат*).
Таблицы совпадают — ☐ Таблицы не совпадают — ☐

Удалите USB–кабель из разъема 17 (см. рис. 1.2).
Работа виртуального прибора (*Sub-VI*–блока) продолжается?
Да — ☐ **Нет** — ☐
Работа DE FPGA Board продолжается по заданному алгоритму?
Да — ☐ **Нет** — ☐

Контрольное задание 2. Нарисуйте логическую схему с учетом того, что все уравнения системы ФАЛ 4.4 содержат общие членые $x_1 \cdot \overline{x_0}$ и $x_2 \cdot \overline{x_0}$. Для обозначения логических операций используйте условные графические обозначения согласно ГОСТ 2.743-91.



Используя разные комбинации включения переключателей **SW0÷SW2**, проверьте правильность работы схемы. Результаты проверки занесите в таблицу:

Вход			Выход		
x_2 (SW2)	x_1 (SW1)	x_0 (SW0)	z_2 (LED2)	z_1 (LED1)	z_0 (LED0)
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Сравните полученный результат с таблицей 4.4 (*отметьте галочкой соответствующий результат*).
Таблицы совпадают — ☐ Таблицы не совпадают — ☐

Удалите USB–кабель из разъема 17 (см. рис. 1.2).
Работа виртуального прибора (*Sub-VI*–блока) продолжается?
Да — ☐ **Нет** — ☐
Работа DE FPGA Board продолжается по заданному алгоритму?
Да — ☐ **Нет** — ☐

Контрольное задание 3. Нарисуйте оптимальную схему логического устройства, реализующую систему ФАЛ (4.5). Для обозначения логических операций используйте условные графические обозначения согласно ГОСТ 2.743-91.

Используя разные комбинации включения переключателей **SW0÷SW2**, проверьте правильность работы схемы. Результаты проверки занесите в таблицу:

Вход			Выход		
x_2 (SW2)	x_1 (SW1)	x_0 (SW0)	z_2 (LED2)	z_1 (LED1)	z_0 (LED0)
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Сравните полученный результат с таблицей 4.4 (*отметьте галочкой соответствующий результат*).
Таблицы совпадают — ☐ Таблицы не совпадают — ☐

Удалите USB–кабель из разъема 17 (см. рис. 1.2).
Работа виртуального прибора (*Sub-VI*–блока) продолжается?
Да — ☐ **Нет** — ☐
Работа DE FPGA Board продолжается по заданному алгоритму?
Да — ☐ **Нет** — ☐

Объясните получившиеся результаты, сделайте собственные выводы (*в письменной форме*):

Работу выполнил студент гр._____ _____ /_____/

«____» _____ 201__ г.

Оценка ____(балла(ов))

Работу принял _____ _____ /_____/

«____» _____ 201__ г.

Глава 5

ФУНКЦИОНАЛЬНО ПОЛНАЯ СИСТЕМА ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ

5.1 Принцип двойственности

При сравнении таблиц истинности для операций И и ИЛИ (см. раздел 2.1) легко заметить, что если в условиях, определяющих операцию И, значения всех переменных и самой функции заменить инверсией, а знак логического умножения — знаком логического сложения, получим постулаты, определяющие операцию ИЛИ, и наоборот (см. табл. 2.2, теоремы Де-Моргана):

$$\begin{array}{llll} \text{если} & x_1 \cdot x_0 = z, & \text{то} & \bar{x}_1 + \bar{x}_0 = \bar{z}; \\ \text{если} & x_1 + x_0 = z, & \text{то} & \bar{x}_1 \cdot \bar{x}_0 = \bar{z}. \end{array} \tag{5.1}$$

Это свойство взаимного преобразования постулатов операций логического сложения и умножения носит название *принципа двойственности*.

Важным практическим следствием принципа двойственности является тот факт, что при записи логических выражений и, следовательно, построении логических схем, можно обойтись только двумя типами операций, например операциями И и НЕ или операциями ИЛИ и НЕ.

5.1.1 Функционально полная система логических элементов

Функционально полной системой называется совокупность ЛЭ, позволяющая реализовывать логическую схему произвольной сложности. Таким образом, системы двух элементов **И** и **НЕ**, а также **ИЛИ** и **НЕ** наравне с системой из трех элементов (**И, ИЛИ, НЕ**) являются функционально полными. На практике широкое применение получили ЛЭ, совмещающие функции элементов указанных выше функционально полных систем. Это ЛЭ, реализующие операции *штрих Шеффера* (F_{14} в табл. 2.3, операция И-НЕ) и стрелка Пирса (F_8 в табл. 2.3, операция ИЛИ-НЕ). По определению, каждый из этих ЛЭ также образует функционально полную систему. Условные графические обозначения для операции И-НЕ приведены в табл.3.5, а для операции ИЛИ-НЕ в табл. 3.6.

В качестве примера рассмотрим реализацию логических операций И, ИЛИ и НЕ с использованием только элемента ИЛИ-НЕ.

Если ЛЭ 2ИЛИ-НЕ включен по схеме, показанной на рис. 5.1, то при подаче на его вход логической переменной A на его выходе получим логическое выражение вида

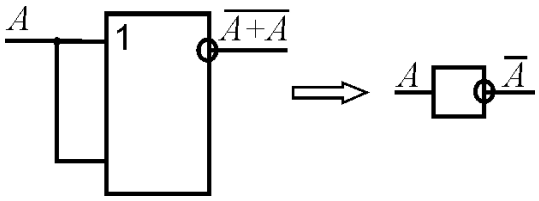


Рис. 5.1. Реализация ЛЭ НЕ на элементе ИЛИ-НЕ

$\overline{A+A}$, но согласно основным аксиомам алгебры–логики (см. табл. 2.2) можем записать $\overline{A+A}=\overline{A}$. Таким образом, мы получили элемент, реализующий операцию логического отрицания (НЕ).

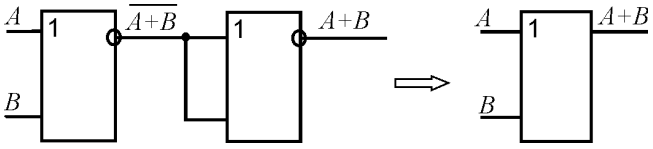


Рис. 5.2. Реализация ЛЭ ИЛИ на элементах ИЛИ-НЕ

Если на входы ЛЭ ИЛИ-НЕ поданы логические переменные A и B, тогда на его выходе получим выражение $\overline{A+B}$. Для реализации операции конъюнкции получившееся выражение необходимо инвертировать, что можно реализовать, применив к нему операцию отрицания (второй элемент ИЛИ-НЕ в структурной логической схеме на рис. 5.2). Таким образом, мы получили элемент, реализующий операцию конъюнкции (ИЛИ). Если на входы структурной схемы, изображенной на рис. 5.3, подать логиче-

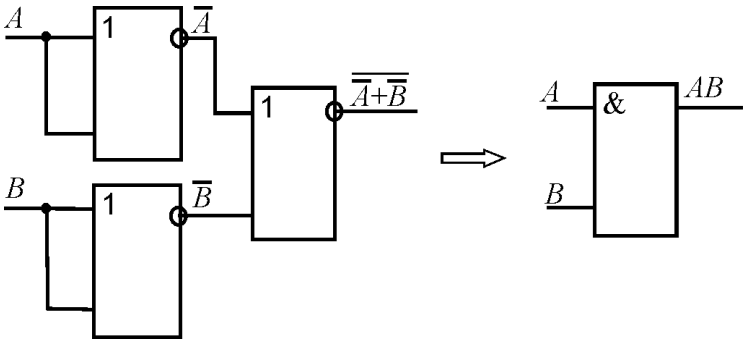


Рис. 5.3. Реализация ЛЭ И на элементах ИЛИ-НЕ

ские переменные A и B, то на выходе мы будем иметь выражение $\overline{\overline{A}+\overline{B}}$. Применяя теорему Де-Моргана к этому выражению, получим $\overline{\overline{A}+\overline{B}}=\overline{\overline{A}}\cdot\overline{\overline{B}}$, далее, последовательно применяя тождества из табл.2.2 к A и B, запишем $\overline{\overline{A}}\cdot\overline{\overline{B}}=A\cdot B$, т.е. данная структурная схема (рис. 5.3) реализует операцию дизъюнкции (И).

На основе аналогичных рассуждений можно показать выполнение основных логических операций с использованием только элемента И-НЕ.

5.2 Синтез логических схем в заданном базисе логических элементов

При построении логических схем обычно не пользуются функционально полной системой ЛЭ, реализующих все три основные логические операции: И, ИЛИ и НЕ. На практике с целью сокращения номенклатуры элементов пользуются функционально полной системой элементов, включающей только два элемента, выполняющие операции И-НЕ и ИЛИ-НЕ, или даже только один из этих элементов. Причем число входов этих элементов, как правило, задано. Поэтому вопросы синтеза логических устройств в заданном базисе ЛЭ имеют большое практическое значение.

На основании примеров, рассмотренных в параграфе 5.1.1, любую ФАЛ можно записать в требуемом базисе ЛЭ. При этом используются два технических приема: двойное инвертирование исходного выражения или его части и применение теорем Де–Моргана.

Если требуется привести ФАЛ к базису ЛЭ И-НЕ, то указанными приемами функция преобразуется к виду, содержащему только операции логического умножения и инверсии. Далее она переписывается через условные обозначения операции И-НЕ. Аналогично поступают при преобразовании ФАЛ к базису ЛЭ ИЛИ-НЕ. В этом случае в выражении оставляют только операции логического сложения и инверсии. Проиллюстрируем сказанное примерами.

Пример 5.1 *Синтезировать схему логического устройства, реализующую МДНФ ФАЛ (4.1), полученную в примере 4.2, в базисе двухвходовых логических элементов 2И-НЕ (Штрих Шеффера).*

Решение. В первую очередь приведем МДНФ ФАЛ (4.1) к базису логических элементов *Штрих Шеффера*. Воспользуемся описанными выше техническими приемами. Перепишем выражение

$$z = x_2 \cdot x_1 + x_2 \cdot x_0 + x_1 \cdot x_0$$

в виде

$$z = x_2 \cdot x_1 + [x_2 \cdot x_0 + x_1 \cdot x_0].$$

Дважды инвертируем выражение в скобках

$$z = x_2 \cdot x_1 + \overline{\overline{[x_2 \cdot x_0 + x_1 \cdot x_0]}}.$$

Воспользуемся теоремами Де–Моргана и перепишем выражение в виде

$$z = x_2 \cdot x_1 + [\overline{\overline{x_2 \cdot x_0} \cdot \overline{x_1 \cdot x_0}}].$$

Операция вида $\overline{A \cdot B} = A|B$ — называется Штрих Шеффера, т. е. можем переписать ФАЛ следующим образом:

$$z = x_2 \cdot x_1 + [(x_2|x_0)|(x_1|x_0)].$$

Аналогично, дважды инвертируем все выражение:

$$z = \overline{\overline{x_2 \cdot x_1 + [(x_2|x_0)|(x_1|x_0)]}}.$$

Применяем теоремы Де–Моргана:

$$z = \overline{\overline{x_2 \cdot x_1} \cdot \overline{[(x_2|x_0)|(x_1|x_0)]}}.$$

И окончательно получаем:

$$z = (x_2|x_1)|[\overline{(x_2|x_0)|(x_1|x_0)}]. \tag{5.2}$$

Синтезируем схему логического устройства. Всего в выражение (5.2) входят пять элементов, реализующих логическую операцию 2И-НЕ. Однако нам необходимо выполнить еще и операцию инверсии для выражения, заключённого в квадратные скобки. Для этого потребуется еще один логический элемент 2И-НЕ, при помощи которого выполним операцию НЕ. Таким образом, всего потребуется шесть таких элементов.

Синтезированная схема логического устройства показана на рис. 5.4.

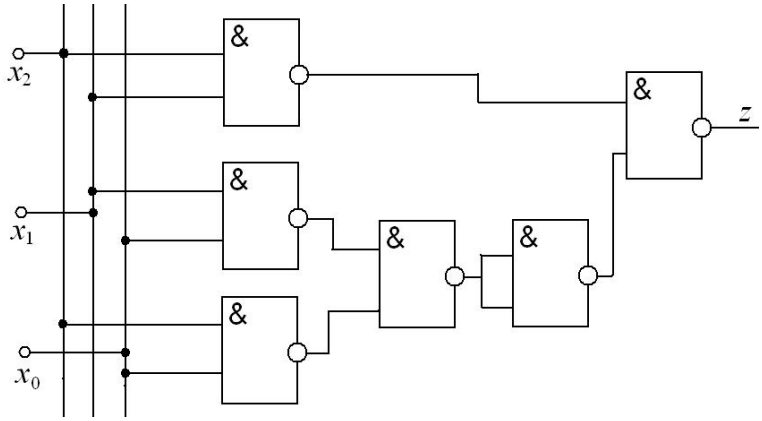


Рис. 5.4. Реализация ФАЛ в базисе логических элементов Штрих Шеффера

■

Пример 5.2 *Синтезировать схему логического устройства, реализующую МКНФ ФАЛ (4.3), полученную в примере 4.2, в базисе двухвходовых логических элементов 2ИЛИ-НЕ (Стрелка Пирса).*

Решение.

Приведём полученную в примере 4.2 МКНФ (4.3) к логической операции *Стрелка Пирса*. Воспользуемся законом ассоциативности и дважды инвертируем выражение в квадратных скобках:

$$z = (x_2 + x_1) \cdot [\overline{\overline{(x_2 + x_0) \cdot (x_1 + x_0)}}].$$

Используя теремы Де–Моргана, перепишем это выражение в виде:

$$z = (x_2 + x_1) \cdot [\overline{(\overline{x_2 + x_0}) + (\overline{x_1 + x_0})}].$$

Операция вида $\overline{A + B} = A \downarrow B$ — называется Стрелка Пирса, перепишем ФАЛ следующим образом:

$$z = (x_2 + x_1) \cdot [(x_2 \downarrow x_0) \downarrow (x_1 \downarrow x_0)].$$

По аналогии с примером 5.1 дважды инвертируем полученное выражение:

$$z = \overline{\overline{(x_2 + x_1) \cdot [(x_2 \downarrow x_0) \downarrow (x_1 \downarrow x_0)]}} = \overline{\overline{(x_2 + x_1)} + [\overline{(x_2 \downarrow x_0) \downarrow (x_1 \downarrow x_0)}]}.$$

Окончательно получим:

$$z = (x_2 \downarrow x_1) \downarrow [\overline{(x_2 \downarrow x_0) \downarrow (x_1 \downarrow x_0)}]. \tag{5.3}$$

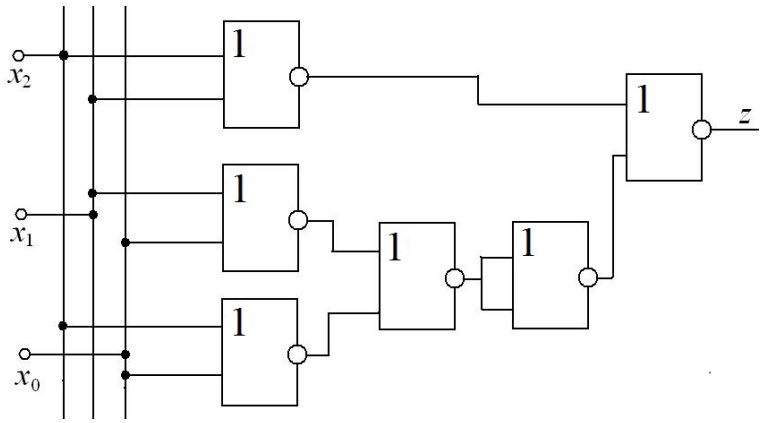


Рис. 5.5. Реализация ФАЛ в базисе логических элементов itСтрелка Пирса

Синтезируем схему логического устройства. Всего в выражение (5.3) входят пять элементов, реализующих логическую операцию 2ИЛИ-НЕ. Кроме этого потребуется еще один элемент 2ИЛИ-НЕ для реализации операции инверсии над частью выражения, заключённого в квадратных скобках, т. е. всего необходимо шесть элементов.

Синтезированная схема логического устройства показана на рис. 5.5.

■

Лабораторная работа №4. Создание логической схемы в заданном базисе логических элементов

Цель работы: Целью лабораторной работы является создать в LabVIEW логическую схему, реализующую фукцию алгебры–логики в заданном базисе логических элементов. Результаты проектирования проверить на оценочном модуле DE FPGA и при помощи виртуального тестера.

Задание на проектирование:

1. Создать в системе графического проектирования схему, представленную на рис. 5.4, реализующую ФАЛ (5.2). Проверить работоспособность схемы при помощи оценочного модуля DE FPGA Board.
2. Создать в системе графического проектирования схему, представленную на рис. 5.5, реализующую ФАЛ (5.3). Проверить работоспособность схемы при помощи виртуального тестера.

Входные переменные x_2, x_1, x_0 должны задаваться при помощи движковых переключателей **SW2**, **SW1**, **SW0** соответственно. Состояние выходного сигнала (z) должно отображаться при помощи светодиода **LED0**.

Выполнение задания: Включите персональный компьютер и подайте питание на образовательную платформу NI ELVIS II$^+$ с установленной на ней платой DE FPGA Board. Создайте в своей рабочей директории папку с именем `Lab04`.

При выполнении заданий воспользуйтесь логическими элементами **Not And** и **Not Or**, входящими в палитру Boolean.

ЛР4.1. Синтез логической схемы в базисе логических элементов 2И–НЕ

Порядок выполнения упражнения:

- запустите LabVIEW и создайте в нем новый проект, сохраните проект под именем `Lab04_1` в папке `Lab04`;
- используя логические элементы *Шртих Шеффера* (**Not And**), соберите схему логического устройства, представленного на рис. 5.4;
- по освоенной в ходе выполнения лабораторной работы №3 методике реализуйте данную схему как комбинационную (т. е. заключите её в структуру **Single-Cycle Timed Loop**);
- запустите проект нажатием кнопки **Run**. Используя разные комбинации включения переключателей **SW0÷SW2**, проверьте правильность работы схемы. Результаты проверки занесите в таблицу.
- сравните получившиеся результаты с данными, полученными в ходе выполнения лабораторных работ №1 и №2;
- таблицу и результаты сравнения занесите в отчет о выполнении лабораторной работы №4;
- закройте проект `Lab04_1`.

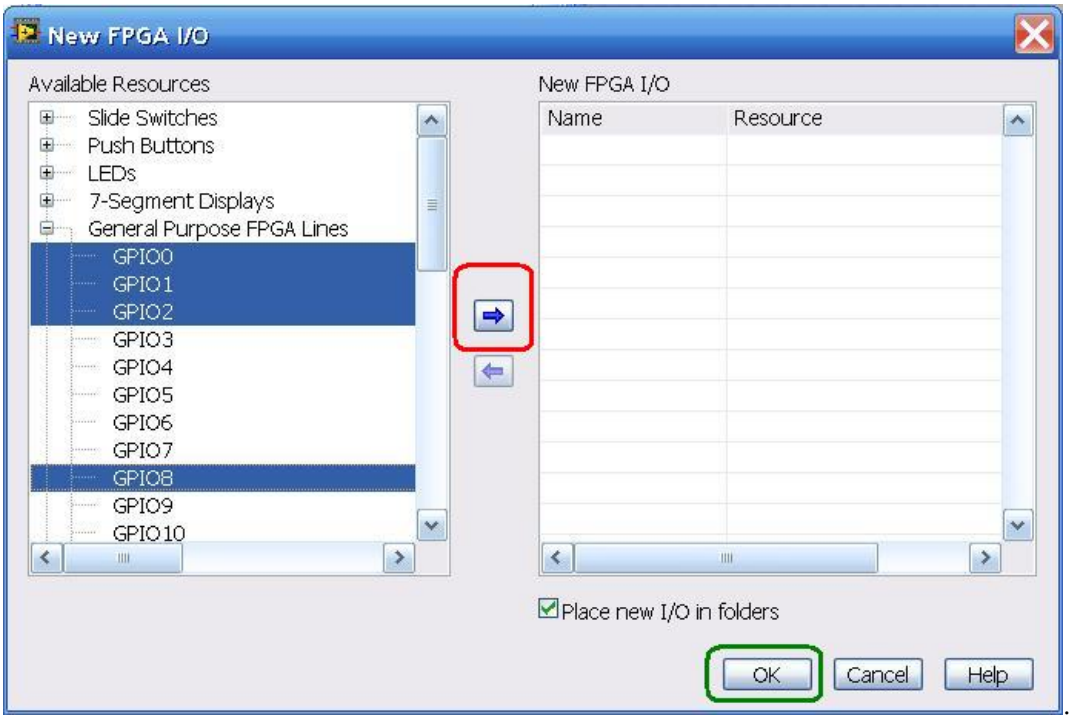
ЛР4.2. Синтез логической схемы в базисе логических элементов 2ИЛИ–НЕ

Работоспособность схемы, исследуемой в этом упражнении, необходимо будет проверить при помощи виртуального тестера. Виртуальный тестер — это виртуальный прибор (*Sub–VI* — блок), предназначенный для тестирования правильности работы цифровых логических схем.

На рабочем столе персонального компьютера находится папка *Виртуальный Тестер*, откройте её и скопируйте все находящиеся в ней файлы в папку `Lab04`.

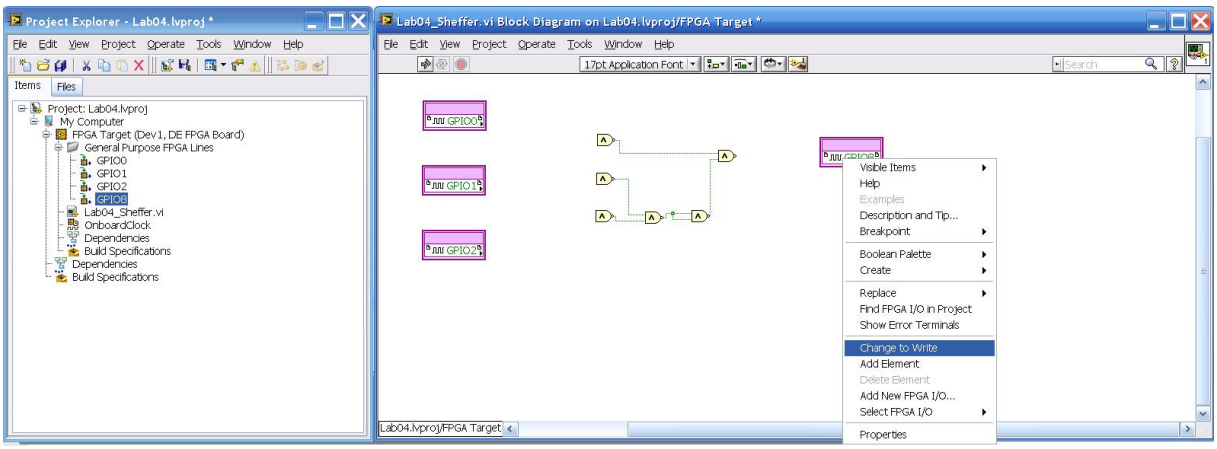
Порядок выполнения упражнения:

- создайте в LabVIEW новый проект и сохраните его под именем `Lab04_2`;
- в окне «Block Diagramm», используя логические элементы *Стрелка Пирса* (**Not Or**), соберите схему логического устройства, представленного на рис. 5.5;
- в «Project Explorer» нажмите правую кнопку мышки на **FPGA Target** и выберите **New→FPGA I/O**;
- в открывшемся окне щелкните по **General Purpose FPGA Lines**, выберите **GPIO0**, **GPIO1**, **GPIO2** и **GPIO8**



Добавьте их в проект и нажмите **OK**. Линии **GPIO2**, **GPIO1** и **GPIO0** будут использоваться для передачи сигналов соответствующих значений логических переменных x_2, x_1, x_0, а линия **GPIO8** для записи состояния выходной переменной z.

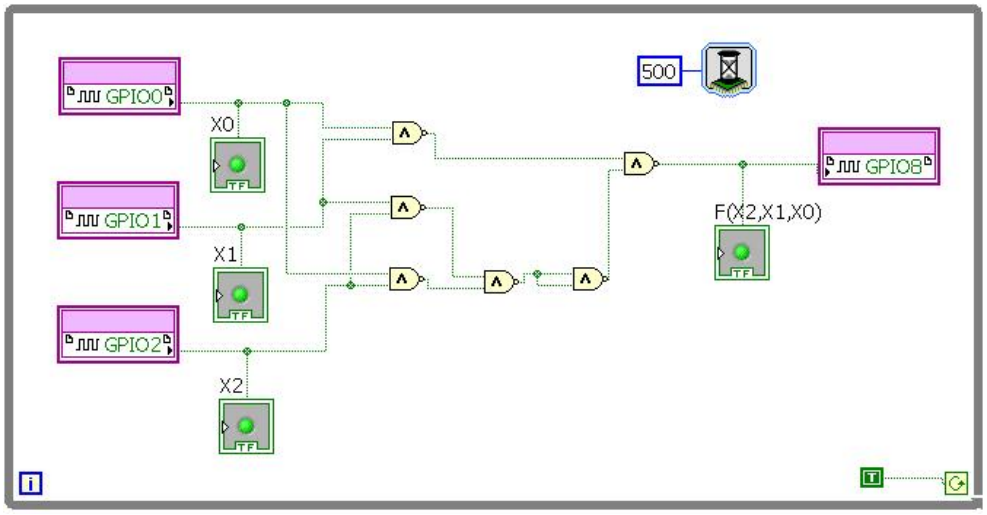
- в «Project Explorer» выберите линии ввода/вывода (**GPIO...**) и перетащите их мышкой в «Block Diagramm». По умолчанию все **GPIO...** линии настроены на чтение информации из FPGA Board. Линию **GPIO8** необходимо перенастроить на запись. Щелкните правой кнопкой мышки по **GPIO8**, и в контекстном меню выберите **Change to Write**;



- проведите оставшиеся соединительные линии;

- используя методику, освоенную при выполнении упражнения ЛР3.2, преобразуйте схему в тактируемую от внутреннего генератора ПЛИС. Для константы **Wait** установите значение 500 *mSec*. Сохраните проект. Собранная схема должна выглядеть примерно так[1]:

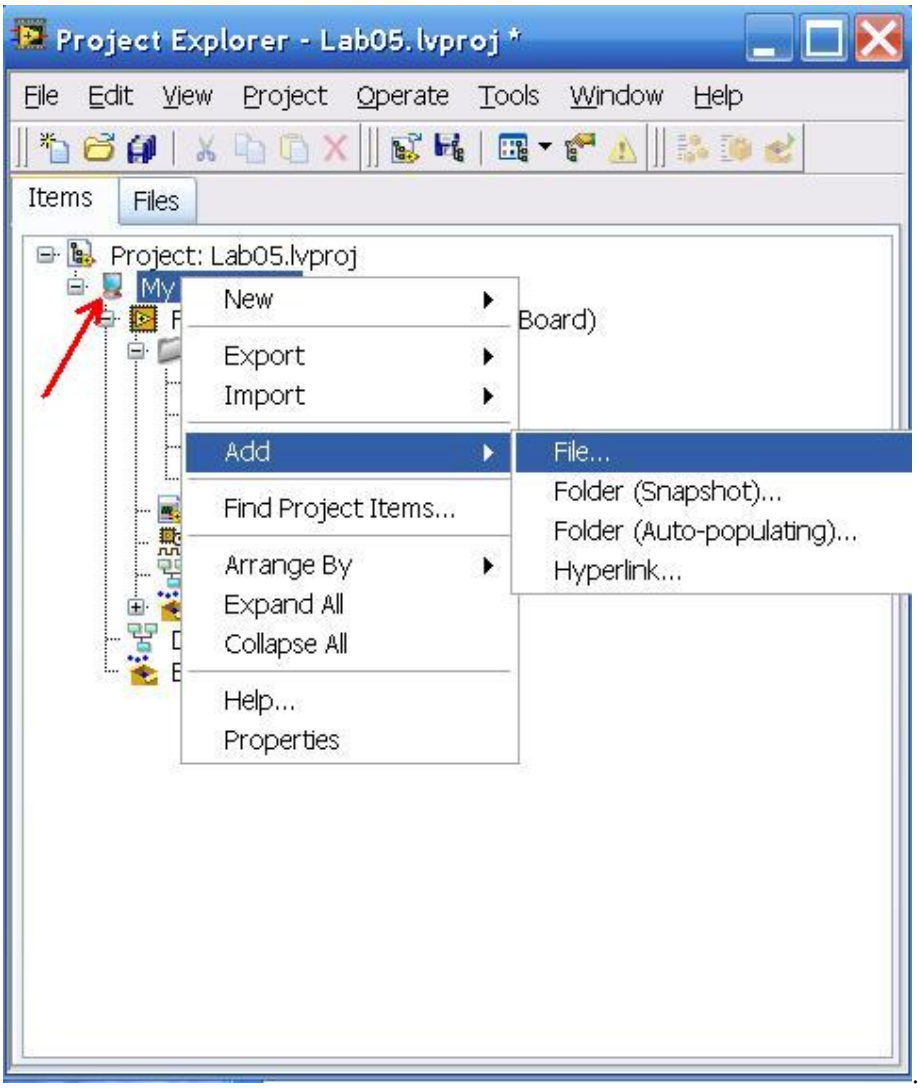
[1] Обратите внимание, на каких логических элементах собрана приведенная на рисунке схема. Отметьте этот факт в отчете по лабораторной работе №4.



;

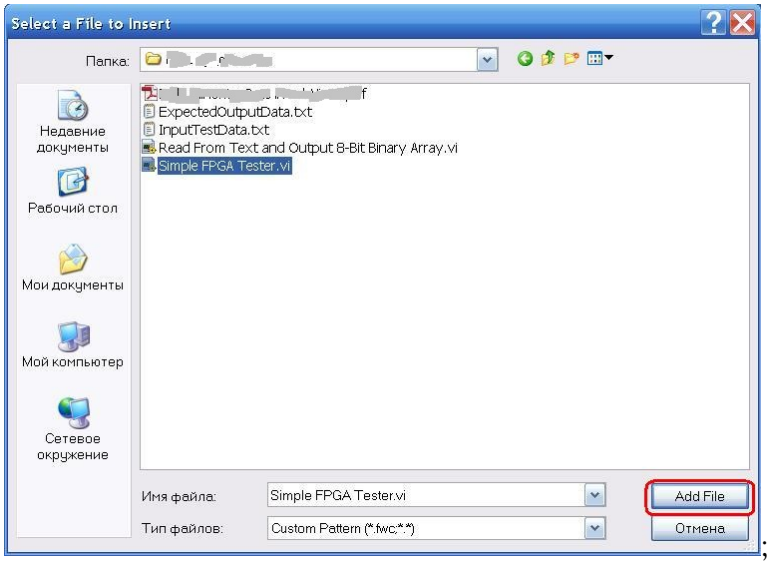
- нажмите кнопку **Run**, запустится процесс компиляции проекта. После того, как процесс компиляции завершится и двоичный код загрузится в ПЛИС, закройте окно компилятора, нажав на кнопку **OK**. FPGA Board готова для работы с виртуальным тестером, а программа `Lab04_2` выполняется;

- настроим виртуальный тестер для работы со схемой. В окне «Project Explorer» нажмите правую кнопку мышки на иконке **My Computer** (показано красной стрелкой), в контекстном меню выберите **Add→File. . .**



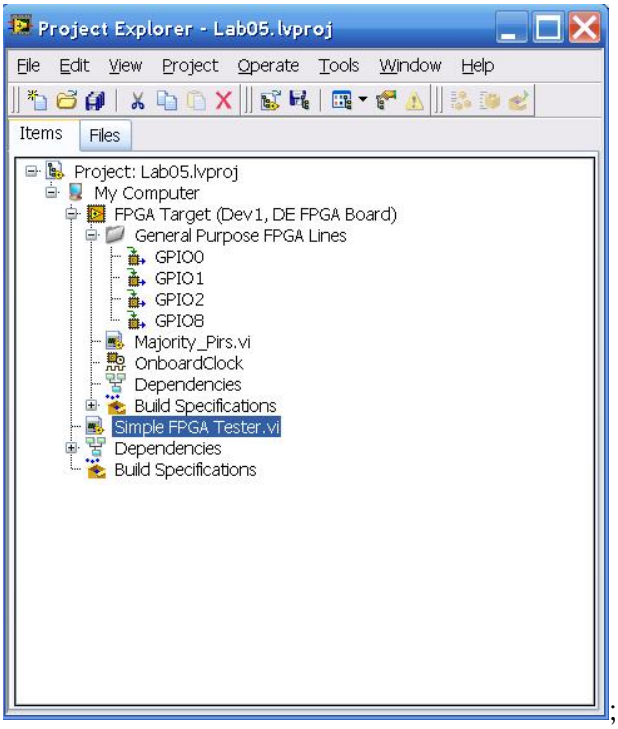
.

В появившемся окне выберите виртуальный прибор *Simple FPGA Tester* и нажмите кнопку **Add File**



;

- окно «Project Explorer» должно выглядеть следующим образом:



;

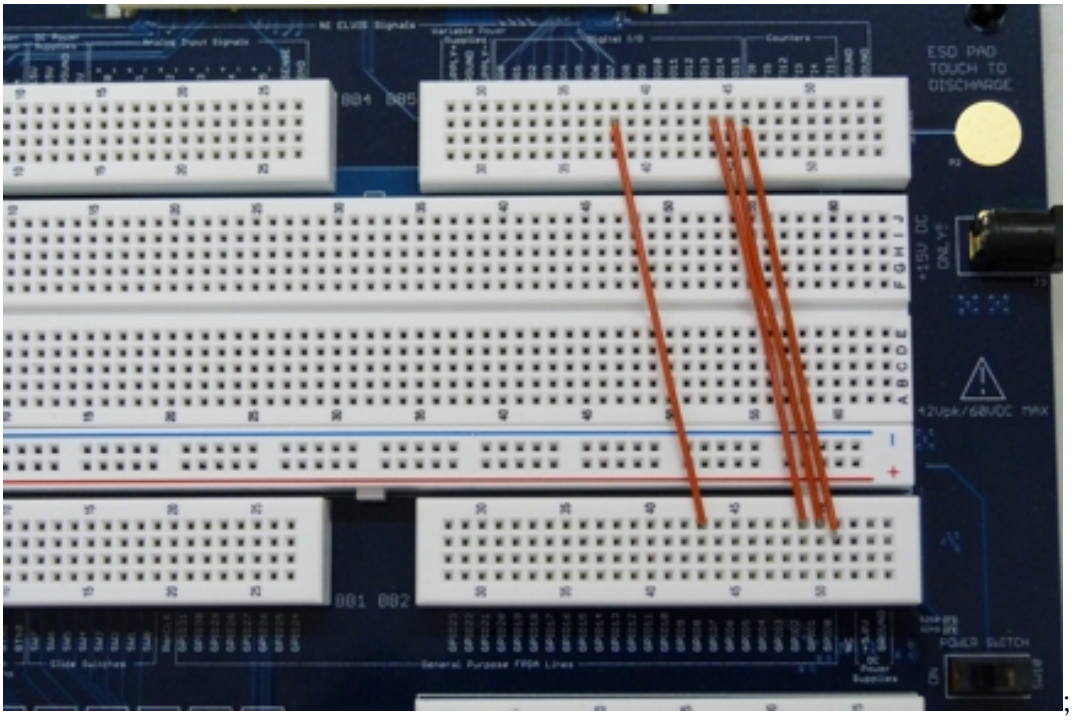
- возьмите у инженера (под расписку) четыре соединительных провода для макетной платы;

- физически соедините контакты разъемов **BB2** (13 на рис. 1.2) и **BB5** (11 на рис. 1.2) согласно следующей таблицы

Разъём BB2	Разъём BB5
GPIO0	DIO0
GPIO1	DIO1
GPIO2	DIO2
GPIO8	DIO8

.

Зона макетирования должна выглядеть, как показано на рисунке:

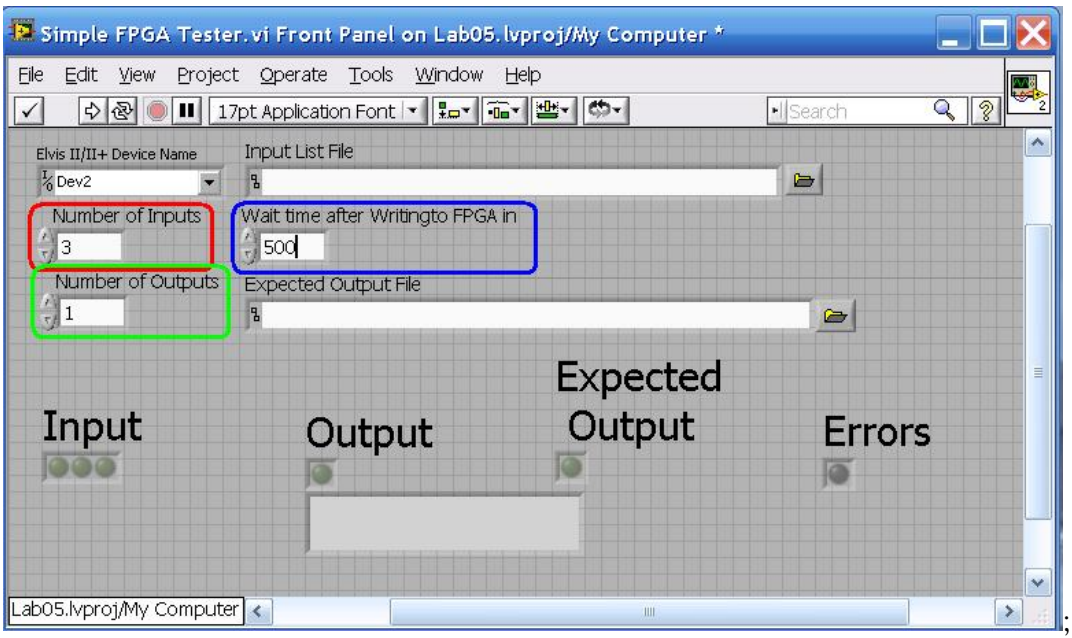


;

- продемонстрируйте инженеру собранную схему. Если схема собрана правильно, он должен расписаться в отчете о выполнении лабораторной работы №4. Дальнейшее выполнение лабораторной работы без разрешения инженера ЗАПРЕЩЕНО;
- нажмите на стрелку в поле Elvis II/II^{+} Device Name и выберите устройство, соответствующее Elvis II

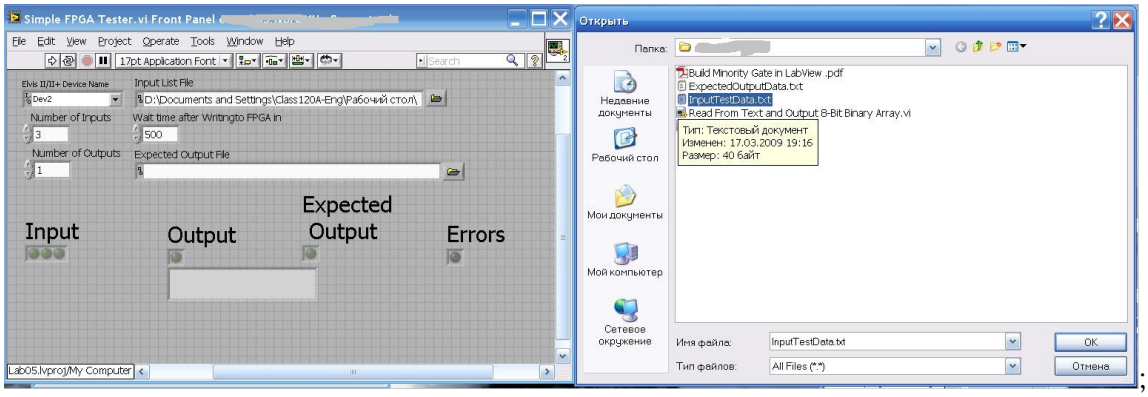
;

- измените **Number of Inputs** на **3** и **Number of Outputs** на **1**. Измените время **Wait**. . . на **500 ms**



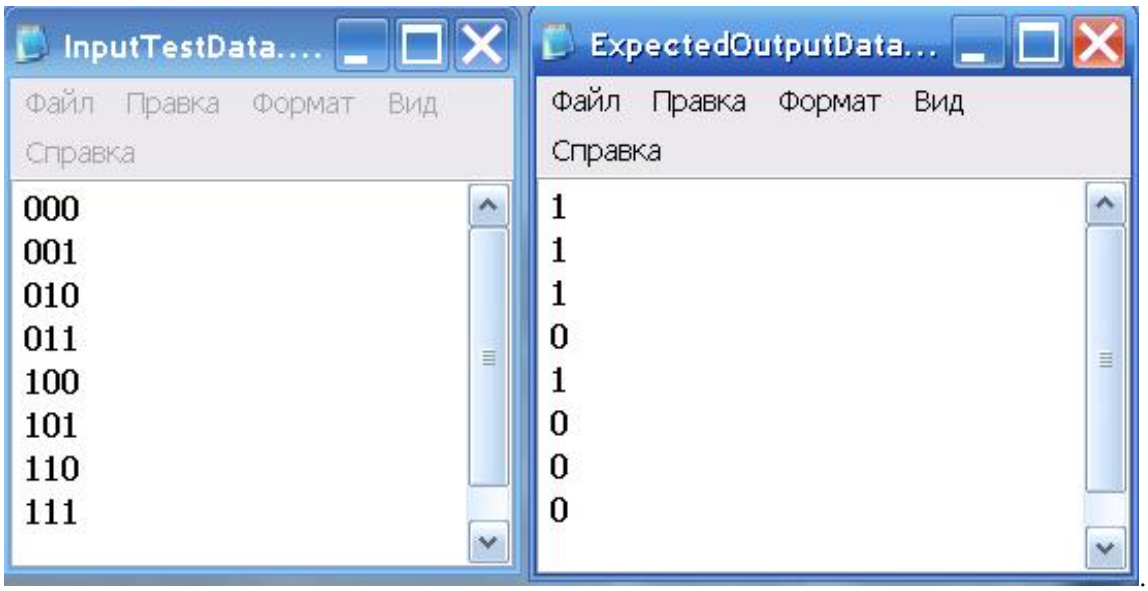
;

- нажмите кнопку **Browse** ( ) на **Input List File** и добавьте файл *InputTestData.txt* из папки, в которой находится проект;



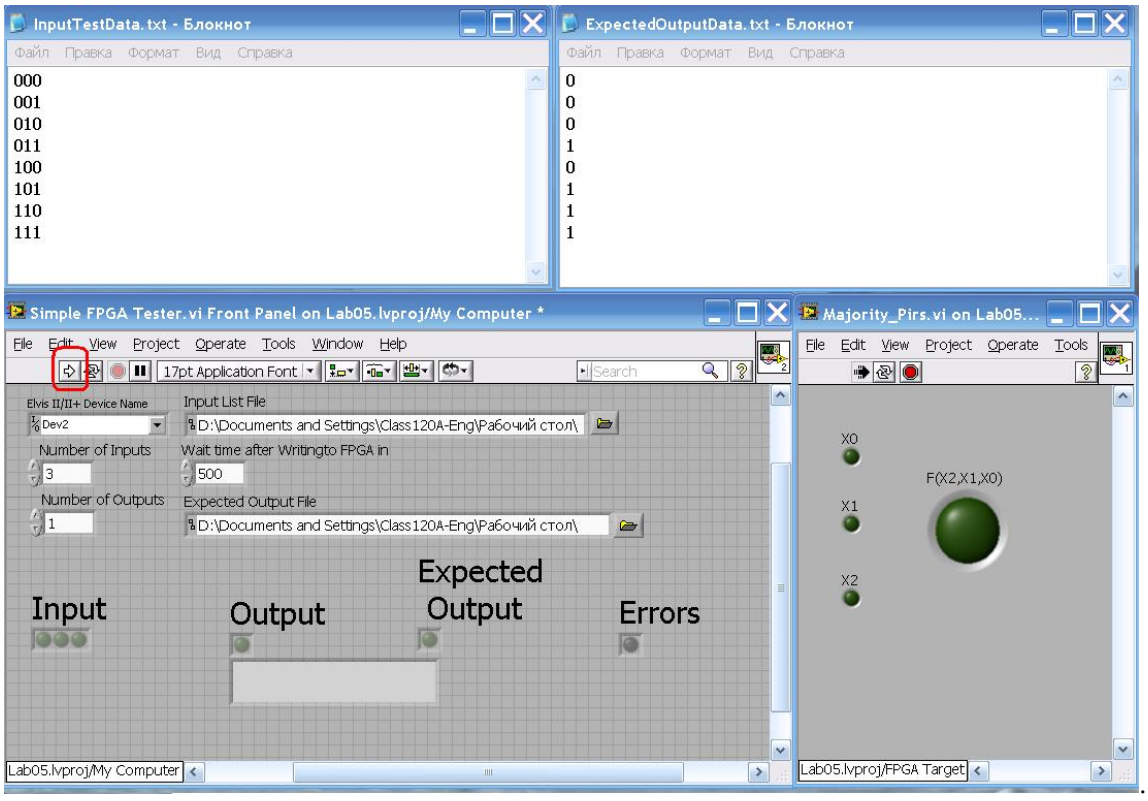
;

- таким же образом добавьте в проект файл *ExpectedTestOutput.txt*. Файлы *InputTestData.txt* и *ExpectedTestOutput.txt* являются обычными текстовыми файлами. Используя **Проводник** Windows или любой другой файловый менеджер, откройте их в **Блокноте**. В файле *InputTestData.txt* хранятся входные коды (x_2, x_1 и x_0). Файл *ExpectedTestOutput.txt* содержит ожидаемые состояния выходного кода (z)



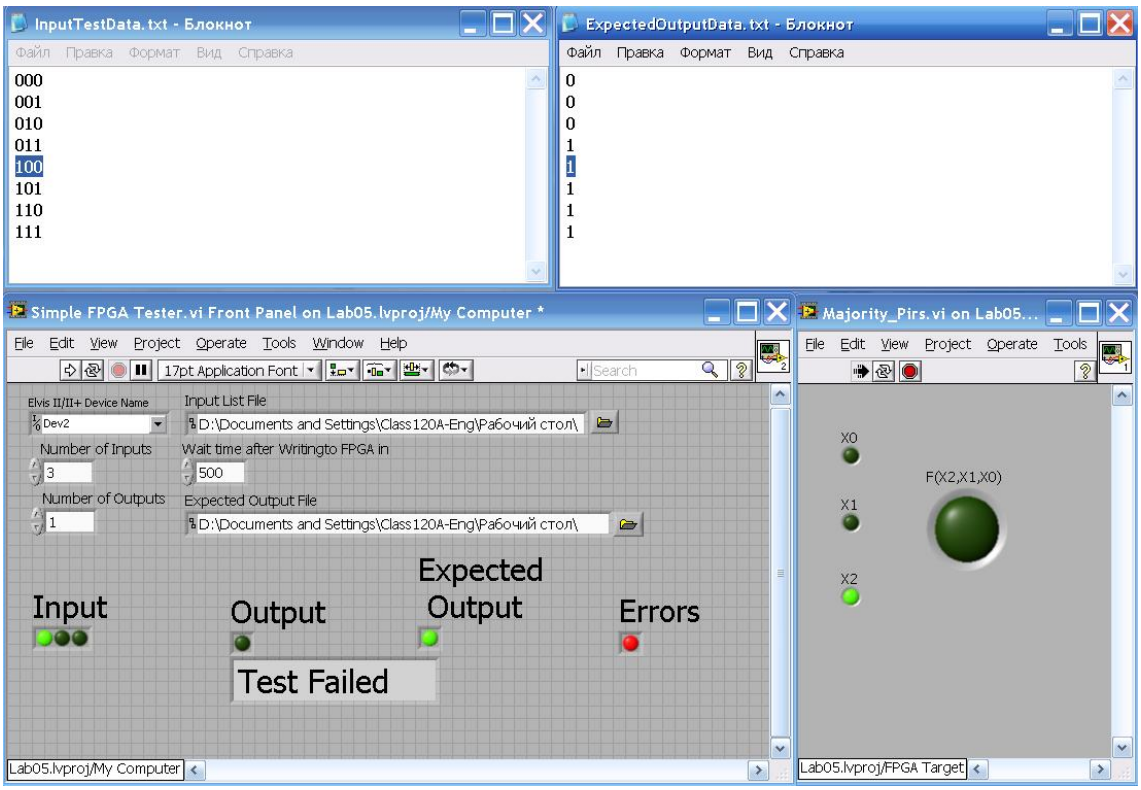
Отредактируйте эти файлы таким образом, чтобы входные и выходные коды соответствовали таблице истинности 2.4. Сохраните эти файлы.

- убедитесь, что проект `Lab04_2` продолжает выполняться. Нажмите на кнопку **Run** на лицевой панели виртуального тестера.



Тест запустится на исполнение, результаты отобразятся на лицевой панели, если данные в файлы внесены корретно, то в виртуальном тестере появится надпись `Test Passed` (тест прошёл);

- остановите работу виртуального тестера (кнопка **Stop** в окне «Simpe FPGA Tester»);
- измените в файле ожидаемых значений (*ExpectedTestOutput.txt*) какое-либо значение с 0 на 1, сохраните этот файл и опять нажмите кнопку **Run**;



При достижении ошибочного ожидаемого выходного кода выполнение программы прервётся, появится сообщение об ошибке `Test Failed`, а при помощи индикаторов (Input/Output — коды) будет показан входной код, состояние сигнала на котором не соответствует ожидаемому выходному значению;

- проделайте предыдущий пукт для трёх–четырёх различных выходных кодов. Результаты проверки занесите в отчет по лабораторной работе №4;
- остановите исполнение `Lab04_2.vi`, закройте проект, закройте LabVIEW, выключите питание оценочного модуля DE FPGA Board и NI ELVIS II$^+$.

Выводы по лабораторной работе №4

В этой работе:

- познакомились с принципами построения цифровых схем в заданном базисе логических элементов;
- научились проверять результаты проектирования при помощи виртуального тестера.

Отчет по лабораторной работе №4

Контрольное задание 1. Используя разные комбинации включения переключателей **SW0÷SW2**, проверьте правильность работы схемы, собранной на логических элементах *Штрих Шеффера*. Результаты проверки занесите в таблицу:

x_2 (SW2)	x_1 (SW1)	x_0 (SW0)	z (LED0)
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Сравните получившиеся результаты с данными, полученными в ходе выполнения лабораторных работ №1 и №2, сделайте выводы (*в письменной форме*):

Контрольное задание 2. Синтезируйте в LabVIEW схему на логических элементах *Стрелка Пирса*

Физически соедините контакты разъемов **BB2** (13 на рис. 1.2) и **BB5** (11 на рис. 1.2), продемонстрируйте инженеру собранную схему.

Схема собрана правильно,
дальнейшее выполнение лабораторной работы разрешено,

инженер по лаборатории _____ /_____/

Опишите работу виртуального тестера для различных кодов в файле *Ожидаемых значений*, сделайте выводы:

Работу выполнил студент гр._____ _____ /_____/

«____» _____ 201__ г.

Оценка ____(балла(ов))

Работу принял _____ _____ /_____/

«____» _____ 201__ г.

Литература

1. Фрике, К. Вводный курс цифровой электроники / К. Фрике. — М.: Техносфера, 2004. — 432 с.

2. Опадчий, Ю. Ф. Аналоговая и цифровая электроника. (Полный курс): Учебник для вузов / Ю. Ф. Опадчий, О. П. Глудкин, А. И. Гуров; Под ред. О. П. Глудкина. — М.: Горячая линия–Телеком, 2002. — 768 с.

3. Янсен, Й. Курс цифровой электроники: В 4-х т. Т.1. Основы цифровой электроники на ИС / Й. Янсен. — М.: «Мир», 1987. — Т. 1. — 334 с.

4. Отладочная плата NI Digital Electronics FPGA Board. Руководство по эксплуатации / National Instruments Corporation. — 2009. — 50 с.

5. ГОСУДАРСТВЕННЫЙ СТАНДАРТ СОЮЗА ССР. — ГОСТ 2.743-91. ЕДИНАЯ СИСТЕМА КОНСТРУКТОРСКОЙ ДОКУМЕНТАЦИИ. ОБОЗНАЧЕНИЯ УСЛОВНЫЕ ГРАФИЧЕСКИЕ В СХЕМАХ. ЭЛЕМЕНТЫ ЦИФРОВОЙ ТЕХНИКИ.