

Лабораторная работа № 1  
**ПЛИС: оптимизация комбинационных схем**

**Цель работы:** изучить метод минимизации логических функций на основе карт Карно; получить практические навыки проектирования и симуляции работы комбинационных схем с помощью Quartus II Web Edition.

## 1. Основы схемотехники логических функций

### Основы булевой алгебры

Любые цифровые устройства опираются на логические элементы, которые могут находиться только в двух состояниях –  $\{0, 1\}$ . Высокий электрический потенциал в цифровой схеме, как правило, соответствует логической «1», а низкий – логическому «0». Математику двухзначных чисел называют **алгеброй логики** или **булевой алгеброй**. В булевой алгебре **функции**, также как и их **аргументы**, могут принимать только два значения. Поэтому область определения логических функций всегда конечная. Количество наборов аргументов  $z$  для  $n$  параметров функции  $f(x_1, x_2, \dots, x_n)$  равно

$$z = 2^n,$$

а количество булевых функций, обозначающих логические операции над  $n$  переменными, определяется формулой

$$N_z = 2^z.$$

Основу алгебры логики составляют операции: **конъюнкция** (операция логического умножения - **И (AND)**; обозначения:  $\&, \bullet, \wedge, *$ ), **дизъюнкция** (операция логического сложения - **ИЛИ (OR)**; обозначения:  $\vee, +$ ) и **инверсия** (операция отрицания – **НЕ (NOT)**; обозначения:  $\neg, !$ ). Они образуют функционально полную систему – **базис**. Суперпозиция элементов базиса позволяет определить **любую произвольно заданную функцию двоичного аргумента**. Базисы могут быть **избыточными** и **минимальными**. Например, система функций: И, ИЛИ, НЕ (**базис Буля** или **булев базис**) – избыточный базис, так как при удалении из него некоторых функций (функции ИЛИ или функции И) система остаётся полной. Минимальными базисами являются системы: 1) И, НЕ; 2) ИЛИ, НЕ; 3) И-НЕ (**базис Шеффера**); 4) ИЛИ-НЕ (**базис Пирса**). При удалении из таких базисов любой операции они перестают быть полными системами функций. Одни и те же преобразования логических переменных можно задать в различных базисах. Выбор базиса зависит от простоты реализации той или иной функции с помощью элементарных логических операций. Чаще всего используются базисы Шеффера и Пирса, т. к. они содержат только одну операцию, и для реализации сложной схемы нужен только один тип логических элементов.

## Формулы логической функции

**Простая конъюнкция** – это конъюнкция некоторого конечного набора переменных, или их отрицаний, где каждая переменная встречается не более одного раза. **Дизъюнктивная нормальная форма (ДНФ)** – это дизъюнкция простых конъюнкций. **Совершенная дизъюнктивная нормальная форма (СДНФ)** – ДНФ относительно некоторого заданного конечного набора переменных, в каждую конъюнкцию которой входят все переменные данного набора. Например, функция  $f(a,b,c)=(\bar{a} \wedge b \wedge \bar{c}) \vee (\bar{a} \wedge \bar{b} \wedge c) \vee (a \wedge b)$  является ДНФ, но не является СДНФ, потому что в последней простой конъюнкции отсутствует переменная  $c$ .

**Простая дизъюнкция** – это дизъюнкция одной или нескольких переменных, или их отрицаний, где каждая переменная встречается не более одного раза. **Конъюнктивная нормальная форма (КНФ)** – это конъюнкция простых дизъюнкций. **Совершенная конъюнктивная нормальная форма (СКНФ)** – КНФ относительно некоторого заданного конечного набора переменных, в каждую дизъюнкцию которой входят все переменные данного набора. Например,  $f(a,b,c)=(a \vee b) \wedge (\bar{a} \vee \bar{b} \vee \bar{c})$  является КНФ, но не является СКНФ, потому что в первой простой конъюнкции содержатся только две из трех переменных функции.

В алгебре логики доказана **теорема разложения**, которая позволяет представить любую функцию  $n$  переменных в двух стандартных формах: СДНФ и СКНФ.

## Таблица истинности логической функции

**Таблица истинности** позволяет описать логическую функцию с помощью полного набора значений во всей области определения. Таблица состоит из  $n+1$  столбца и  $2^n$  строк, где  $n$  - число аргументов функции. В первых  $n$  столбцах записываются всевозможные значения аргументов, а в  $n+1$  столбце записываются значения функции. Таблица истинности связана с формулами логической функции в виде СДНФ и СКНФ следующими алгоритмами.

### **Алгоритм построения СДНФ:**

1. В таблице истинности найти все наборы аргументов, при которых логическая функция принимает значение 1.
2. Записать для каждого найденного набора простую конъюнкцию по правилу: отдельная переменная входит в конъюнкцию с отрицанием, если в наборе она принимает значение 0; в противном случае переменная входит в конъюнкцию без отрицания.
3. Объединить все простые конъюнкции с помощью дизъюнкции.

### **Алгоритм построения СКНФ:**

1. В таблице истинности найти все наборы аргументов, при которых логическая функция принимает значение 0.
2. Записать для каждого найденного набора простую дизъюнкцию по правилу: отдельная переменная входит в дизъюнкцию с отрицанием, если

в наборе она принимает значение 1; в противном случае переменная входит в дизъюнкцию без отрицания.

3. Объединить все простые дизъюнкции с помощью конъюнкции.

**Например,** построим СДНФ и СКНФ логической функции с заданной таблицей истинности:

$a$	$b$	$f(a,b)$
0	0	1
0	1	0
1	0	1
1	1	1

Рис. 1. Таблица истинности логической функции

**СДНФ:** 1). В первой, третьей и четвертой строках исходной таблицы логическая функция равна единице. 2). Запишем простые конъюнкции для этих строк:  $\bar{a} \wedge \bar{b}$ ,  $a \wedge \bar{b}$ ,  $a \wedge b$ . 3). Объединяя простые конъюнкции операциями дизъюнкции получим логическую функцию в виде СДНФ:  $f(a,b) = (\bar{a} \wedge \bar{b}) \vee (a \wedge \bar{b}) \vee (a \wedge b)$ . Для проверки построим таблицу истинности логической функции в виде СДНФ (рис. 2) и сопоставим ее с исходной таблицей (рис. 1):

$a$	$b$	$\bar{a} \wedge \bar{b}$	$a \wedge \bar{b}$	$a \wedge b$	$(\bar{a} \wedge \bar{b}) \vee (a \wedge \bar{b}) \vee (a \wedge b)$
0	0	1	0	0	1
0	1	0	0	0	0
1	0	0	1	0	1
1	1	0	0	1	1

Рис. 2. Таблица истинности логической функции в виде СДНФ

Сравнение показывает, что формула логической функции в виде СДНФ дает таблицу истинности идентичную исходной.

**СКНФ:** 1). Во второй строке исходной таблицы логическая функция равна нулю. 2). Запишем простую дизъюнкцию для этой строки:  $a \vee \bar{b}$ . 3). Других простых дизъюнкций нет, поэтому СКНФ логической функции имеет вид:  $f(a,b) = a \vee \bar{b}$ . Для проверки построим таблицу истинности логической функции в виде СКНФ (рис. 3) и сопоставим ее с исходной таблицей (рис. 1):

$a$	$b$	$a \vee \bar{b}$
0	0	1
0	1	0
1	0	1
1	1	1

Рис. 3. Таблица истинности логической функции в виде СКНФ

Сравнение показывает, что формула логической функции в виде СКНФ дает таблицу истинности идентичную исходной.

**Выводы из примера:** Совпадение исходной таблицы истинности с таблицами истинности логической функции в виде СДНФ и СКНФ показывает:

- возможность записи функции как минимум двумя разными формулами:
  1. СДНФ -  $f(a,b) = (\bar{a} \wedge \bar{b}) \vee (a \wedge \bar{b}) \vee (a \wedge b)$ ;
  2. СКНФ -  $f(a,b) = (a \vee b)$ .
- правильность работы алгоритмов построения совершенных нормальных форм логической функции по заданной таблице истинности;
- эквивалентность представления логической функции с помощью таблицы истинности и формул.

### Минимизация логической функции с помощью карты Карно

Совершенные нормальные формы (СНФ) дают однозначные представления булевой функции, но являются **избыточными**. Программная или аппаратная реализация СНФ не является оптимальной, т.к. не позволяет получить минимальный программный код или минимальное количество логических элементов цифровой схемы, выполняющих преобразования входных бинарных сигналов в соответствие с логической функцией. **Минимизацией** логических функций называют упрощение их записи с целью получения минимального числа символов (букв) в формуле.

Примечание. Обратите внимание, что минимизация направлена на сокращение количества букв, но не переменных. Например, в формуле булевой функции  $f(a,b) = (\bar{a} \wedge \bar{b}) \vee (a \wedge \bar{b}) \vee (a \wedge b)$  используется 2 переменные, но количество букв равно 6.

**Минимальная нормальная форма** (дизъюнктивная - **МДНФ** или конъюнктивная - **МКНФ**) логической функции должна содержать не больше букв, чем любая другая ее нормальная форма.

Минимизация логических выражений может осуществляться по правилам булевой алгебры различными методами: диаграммы Вейча, диаграммы Венна, табличный метод, метод Куайна, метод неопределенных коэффициентов, но наиболее простым и наглядным является графический способ минимизации с помощью **карт Карно**, опубликованный в 1953 г. Морисом Карно.

Карта Карно представляет собой таблицу истинности, отформатированную особым образом, пригодным для наглядной ручной минимизации. Результатом минимизации является либо МДНФ, либо МКНФ. В первом случае работа ведётся с ячейками карты, где находятся единицы, во втором – с ячейками, где находятся нули.

Карты Карно, как правило, применяются для минимизации булевых функций от 2, 3 и 4 переменных. Для функции двух переменных карта Карно – это квадрат размером 2x2 ячейки. Для функций трех переменных карта Карно – это прямоугольник 2x4 или 4x2 ячейки. Для функций четырёх переменных карта Карно – это квадрат 4x4 ячейки. Заголовки строк и столбцов карты Карно содержат коды состояний (наборы переменных) логической функции. Причем эти коды упорядочены так, чтобы смежные значения отличались только одной цифрой

(код Грея). Для кодирования состояний можно использовать любые сочетания переменных, и начинать кодирование с любого значения.

Исходной информацией для работы с картой Карно является таблица истинности минимизируемой функции. Карта Карно содержит  $2^n$  ячеек, каждая из которых ассоциируется с уникальным набором входных переменных  $X_1 \dots X_n$ . Таким образом, между таблицей истинности и картой Карно имеется взаимно однозначное соответствие, и карту Карно можно считать своеобразной формой представления таблицы истинности (рис. 4).

$X_1$	$X_2$	$X_3$	$X_4$	$F$
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

а)

$X_3 X_4$		00	01	11	10
$X_1 X_2$	00	1	0	0	1
	01	1	0	0	1
	11	0	1	1	0
	10	1	0	0	1

б)

		$x_3$			
		1	0	0	1
$x_1$	1	1	0	0	1
	0	1	0	0	1
	0	0	1	1	0
	1	1	0	0	1
		$x_4$			

в)

Рис. 4. Таблица истинности (а) и карта Карно (б, в) заданной логической функции

Прямоугольную область в карте Карно, которая состоит из  $2^k$  одинаковых значений (единиц или нулей в зависимости от того, какую форму нужно получить) будем называть **склейкой**, **группой** или **областью**. Распределение всех имеющихся в карте Карно нулей (единиц) по склейкам будем называть **покрытием**. С целью минимизации булевой функции необходимо построить такое покрытие карты Карно, чтобы количество склеек было минимальным, а размер каждой склейки максимально возможным. Для этого необходимо руководствоваться следующими правилами:

- Склею ячейки одной и той же карты Карно можно осуществлять как по единицам (рис. 5-а), так и по нулям (рис. 5-б). Первое необходимо для получения ДНФ, второе — для получения КНФ.

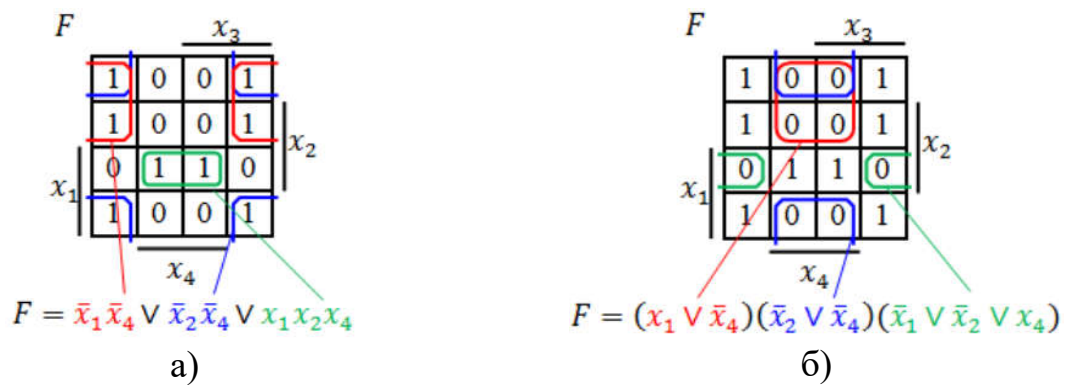


Рис. 5. Склейка ячеек карт Карно для получения ДНФ (а) и КНФ (б)

- Склеивать можно только прямоугольные области с числом единиц (нулей), являющимся целой степенью двойки (1, 2, 4, 8, 16, 32... ячейки) (рис. 6).

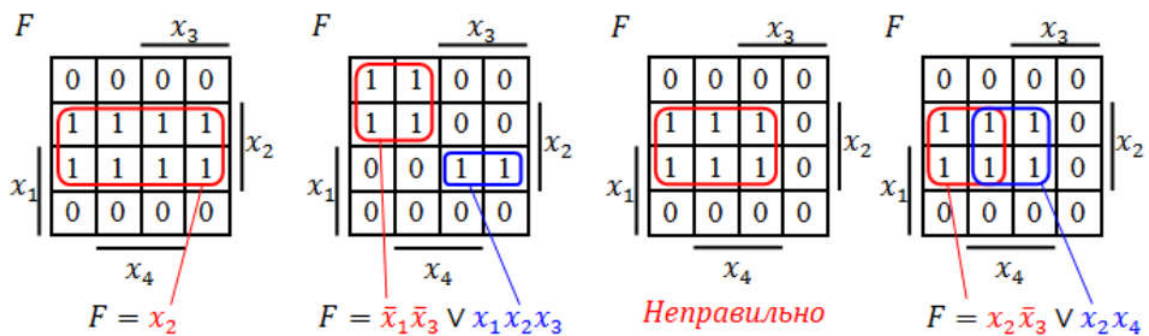


Рис. 6. Результаты склейки прямоугольных областей на карте Карно

- Рекомендуется выбирать максимально возможные области склейки. Если область склейки не является максимально возможной, это не будет ошибкой, однако ДНФ (КНФ) не получится минимальной (рис. 7).

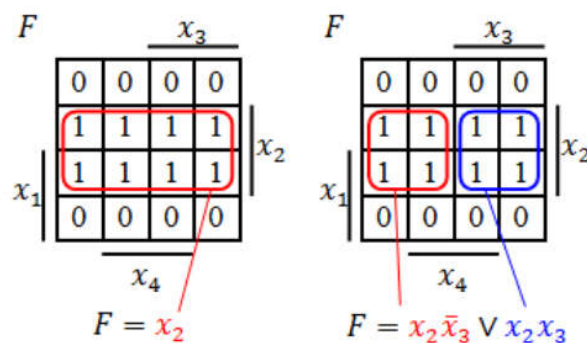


Рис. 7. Результаты выбора размера склеиваемых областей

- В некоторых ситуациях в раскладке образуется изолированная единица или ноль, которую невозможно включить в какую-либо область. В этом случае единица (ноль) склеивается «сама с собой». Нельзя оставлять одиночные единицы (нули), так как это приведёт к некорректной записи выражения для функции. Все единицы (нули) должны попасть в какую-либо область (рис. 8).

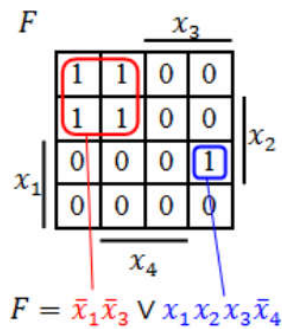


Рис. 8. Значимость отдельных значений на карте Карно

- Область, которая подвергается склейке, должна содержать одинаковые значения — только единицы или только нули (рис. 9).

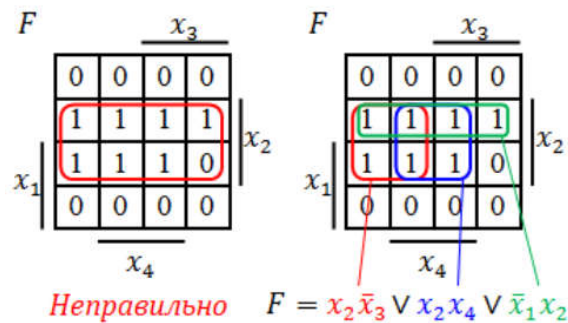


Рис. 9. Склеиваемые области должны содержать однородные значения

- Для карт Карно с числом переменных 3 и 4 применимо правило - крайние клетки каждой горизонтали и каждой вертикали граничат между собой и могут объединяться в прямоугольники (топологически карта Карно представляет собой тор). Следствием этого правила является смежность всех четырёх угловых ячеек карты Карно для 4 переменных (рис. 10).

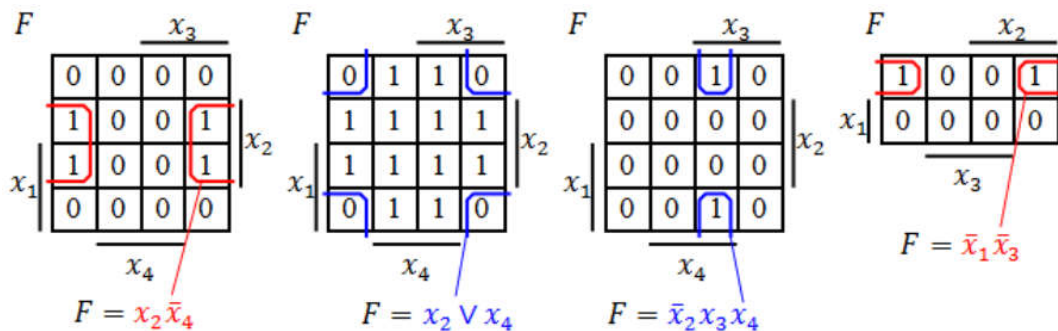


Рис. 10. Смежность ячеек на карте Карно

- Одна ячейка карты Карно может входить сразу в несколько областей. Это следует из очевидного свойства булевых функций: повторение уже существующего слагаемого (сомножителя) не влияет на функцию:  $A \vee A = A$ ;  $A \cdot A = A$  (рис. 11).



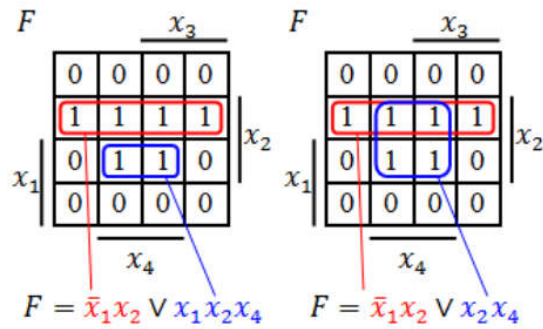


Рис. 11. Повторное вхождение ячеек в разные области на карте Карно

- Не следует без нужды включать клетку во все возможные склейки, это не является ошибкой, но усложняет формулу. С точки зрения минимальности ДНФ (КНФ) число склеек должно быть как можно меньше, а число ячеек в склейке должно быть максимально возможным (рис. 12).

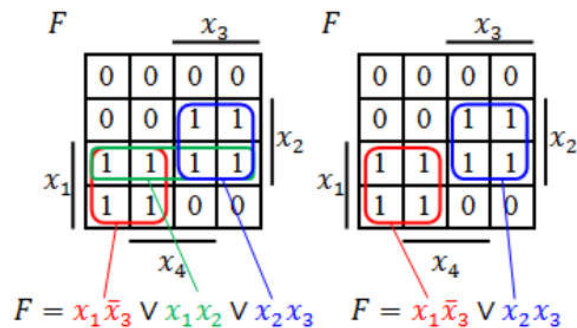


Рис. 12. Влияние числа областей на количество букв в формуле

- В отличие от СДНФ и СКНФ, ДНФ и КНФ не всегда единственны. Для некоторых функций существует несколько эквивалентных друг другу ДНФ (КНФ), которые соответствуют разным способам покрытия карты Карно прямоугольными областями. Очень часто две различные ДНФ (КНФ) имеют одинаковую сложность, что не позволяет сделать однозначный выбор минимальной формулы (рис. 13).

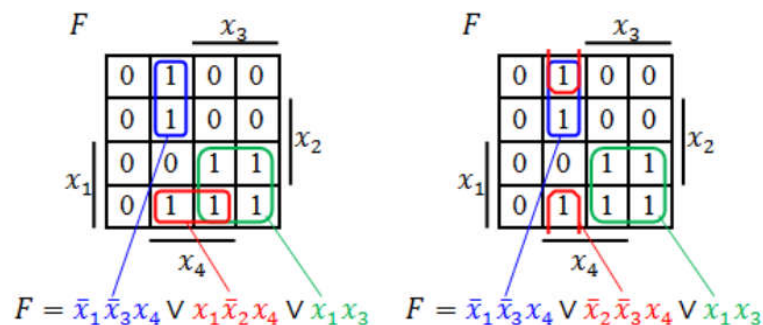


Рис. 13. Неопределенность минимальных формул

#### Реализация логической функции в базисах И-НЕ и ИЛИ-НЕ

На аппаратном уровне наиболее простую реализацию имеют базисы Шеффера (И-НЕ) и Пирса (ИЛИ-НЕ). Каждый из них включает всего одну элементарную функцию, для построения которой требуется минимум транзисторов. Для



изготовления элементов И-НЕ и ИЛИ-НЕ применяют технологии на основе биполярных и полевых транзисторов (рис. 14, 15).

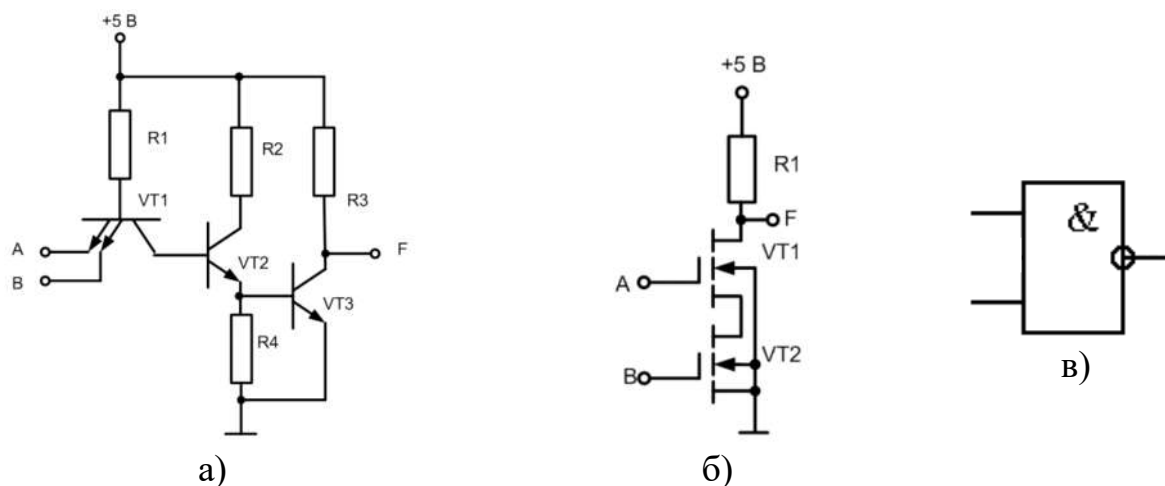


Рис. 14. Схема логического элемента И-НЕ в технологиях: ТТЛ (а), пМОП (б) и его обозначение (в)

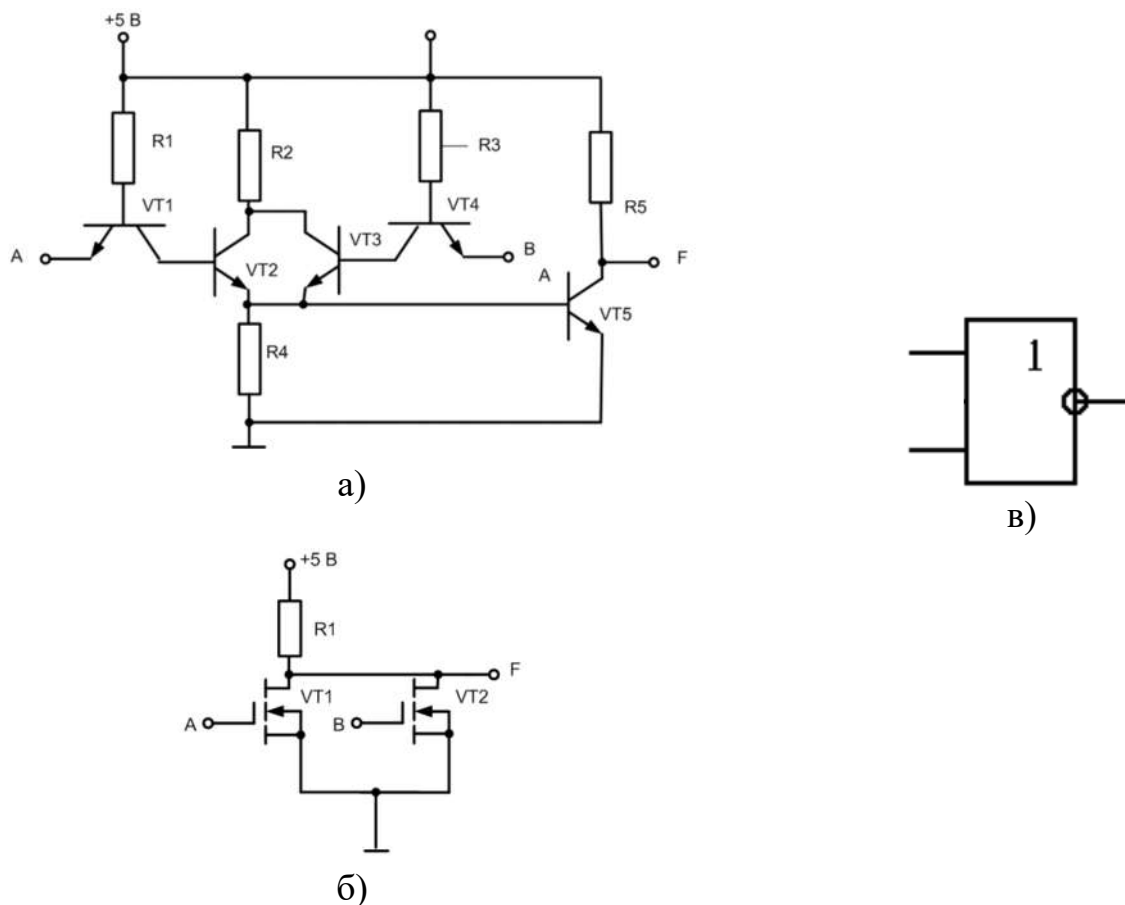


Рис. 15. Схема логического элемента ИЛИ-НЕ в технологиях: ТТЛ (а), пМОП (б) и его обозначение (в)

В базисах Шеффера и Пирса функцию НЕ можно получить объединением всех входов логического элемента, а конъюнкцию (И) и дизъюнкцию (ИЛИ) добавлением отрицания на выход элементов И-НЕ и ИЛИ-НЕ соответственно (рис. 16).

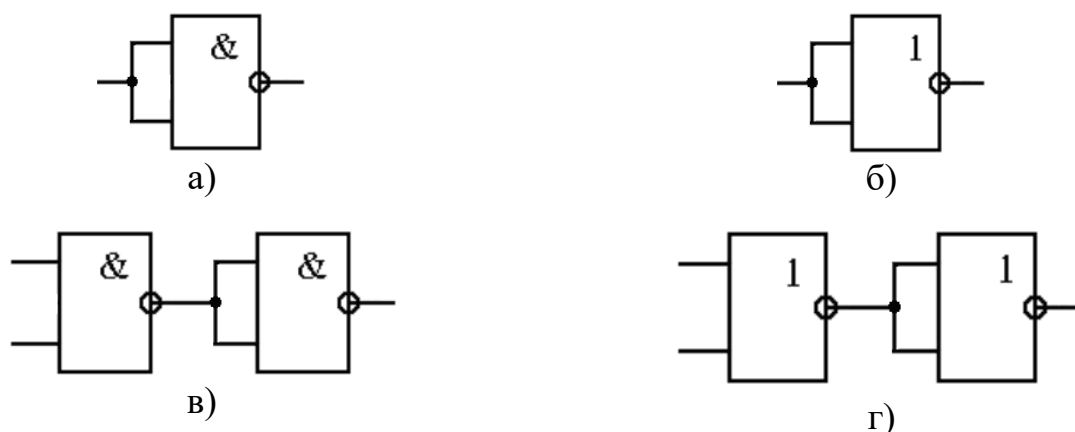


Рис. 16. Реализация логических функций НЕ (а, б), И (в), ИЛИ (г) в базисах Шеффера и Пирса

Для приведения логических формул к базисам Шеффера и Пирса используются законы двойного отрицания и де Моргана (см. Приложение 1).

## 2. Задание

1. Создайте папку «LabFPGA(<фамилия на латинице>-<номер группы>)-1». Например, студент Иванов из группы 11916 должен создать папку: LabFPGA(Ivanov-1191)-1.
2. Внутри папки «LabFPGA...» создайте папки: «Task1\_1», «Task1\_2» и «Task1\_3». В них далее следует размещать проекты среды Quartus Web Edition, соответственно предназначенные для решения: обучающей, самостоятельной и индивидуальных частей задания.

### 2.1. Обучающая часть

**Задача «Hello, Quartus»:** С помощью метода Карно получить МДНФ логической функции трех аргументов, представленной таблицей истинности на рисунке 17, построить ее ДНФ для базиса И-НЕ, в приложении Quartus реализовать принципиальную схему оптимизированной логической функции и исследовать работу электронной схемы, используя встроенный симулятор сигналов Quartus.

<i>a</i>	<i>b</i>	<i>c</i>	<i>f</i>
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Рис. 17. Таблица истинности заданной логической функции

### Оптимизация логической функции

1. На основе таблицы истинности логической функции строим **СДНФ**. Получим следующую формулу:

$$f(a,b,c) = (\bar{a} \cdot \bar{b} \cdot \bar{c}) \vee (\bar{a} \cdot \bar{b} \cdot c) \vee (\bar{a} \cdot b \cdot \bar{c}) \vee (a \cdot \bar{b} \cdot c)$$

2. Для минимизации формулы логической функции используем метод Карно. Выбор сочетания аргументов и кода начального состояния произволен для карты Карно, поэтому используем такой вид (рис. 18):

<b>a \ b·c</b>	<b>11</b>	<b>10</b>	<b>00</b>	<b>01</b>
<b>1</b>				
<b>0</b>				

Рис. 18. Выбор формы карты Карно и кодирования состояний

Важно лишь, чтобы код состояния в строке или столбце карты отличался от соседнего значения одним битом.

3. В соответствие с таблицей истинности заданной логической функции заполним карту Карно (рис. 19):

<b>a \ b·c</b>	<b>11</b>	<b>10</b>	<b>00</b>	<b>01</b>
<b>1</b>	0	0	0	1
<b>0</b>	0	1	1	1

Рис. 19. Заполнение карты Карно в соответствие с таблицей истинности

4. Для построения МДНФ используем ячейки карты Карно, в которых логическая функция равна 1. Объединим их в группы, число элементов в которых кратно  $2^n$  (рис. 20):

<b>a \ b·c</b>	<b>11</b>	<b>10</b>	<b>00</b>	<b>01</b>
<b>1</b>	0	0	0	1
<b>0</b>	0	1	1	1

Рис. 20. Выделение групп единичных значений логической функции

5. Каждой группе соответствует простая конъюнкция. Переменная равная нулю входит в простую конъюнкцию с отрицанием, а переменная равная единице входит в простую конъюнкцию в прямом виде (без отрицания). Если в группе какая-либо переменная принимает оба значения (ноль и

единицу), то она исключается из простой конъюнкции. Группам состояний, обозначенным на карте Карно овалами, соответствуют простые конъюнкции:  $(\bar{a} \cdot \bar{c})$ ,  $(\bar{b} \cdot c)$ .

6. **МДНФ** заданной логической функции получается дизъюнкцией простых конъюнкций, определенных в п. 5:

$$f(a,b,c) = (\bar{a} \cdot \bar{c}) \vee (\bar{b} \cdot c)$$

7. В базисе (И, ИЛИ, НЕ) заданная логическая функция реализуется схемой, представленной на рисунке 21.

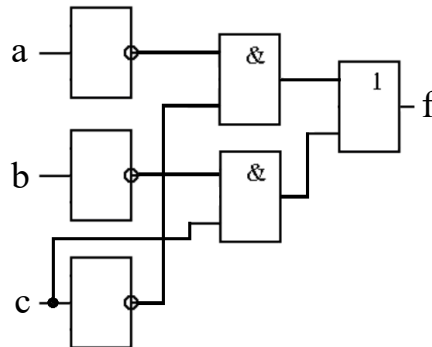


Рис. 21. Схема заданной логической функции на базисе (И, ИЛИ, НЕ)

8. Выразим формулу оптимизированной логической функции в базисе (И-НЕ). Для этого используем законы **двойного отрицания** и **де Моргана**:

$$\overline{\overline{x}} = x; \quad \overline{x \vee y} = \bar{x} \wedge \bar{y}.$$

В итоге получим:

$$f(a,b,c) = (\bar{a} \cdot \bar{c}) \vee (\bar{b} \cdot c) = \overline{\overline{(\bar{a} \cdot \bar{c}) \vee (\bar{b} \cdot c)}} = \overline{(\overline{\bar{a} \cdot \bar{c}}) \cdot (\overline{\bar{b} \cdot c})}.$$

Ее схема представлена на рисунке 22.

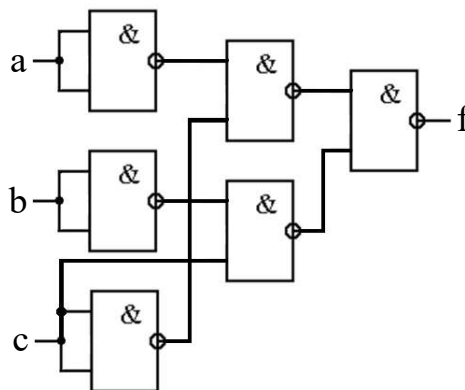


Рис. 22. Схема заданной логической функции на базисе (И-НЕ)

Сопоставление таблицы истинности логической функции, построенной на базисе (И-НЕ), и таблицы заданной функции (рис. 17) доказывает их эквивалентность. Для реализации логической функции в базисе (И-НЕ) требуется 6 логических элементов, как и при ее реализации на базисе (И, ИЛИ, НЕ). Однако в схеме на основе (И, ИЛИ, НЕ) использовано 3 типа элементов (рис. 21), а в схеме на основе (И-НЕ) применен только 1 тип

элементов (рис. 22). Такая унификация дает преимущество базисам Шеффера и Пирса при выборе элементарной базы цифровых микросхем.

### Создание проекта Quartus

9. Запустите приложение Quartus Web Edition (закройте стартовый диалог, щелкнув крестик в правом верхнем углу, и дождитесь проверки обновлений).
10. Создайте новый проект. Для этого щелкните меню «File – New...». В появившемся диалоговом окне выделите пункт «New Quartus II Project» и нажмите кнопку «Ok».
11. В появившемся диалоговом окне (рис. 23) укажите путь к проекту (папка «...LabFPGA(<фамилия на латинице>-<номер группы>)-1\Task1\_1\») и его название по такому шаблону: «<инициалы ФИО на латинице>1\_1». Например, проект Сидорова Ивана Петровича должен иметь название: **sip3\_1**. Нажмите «Finish».

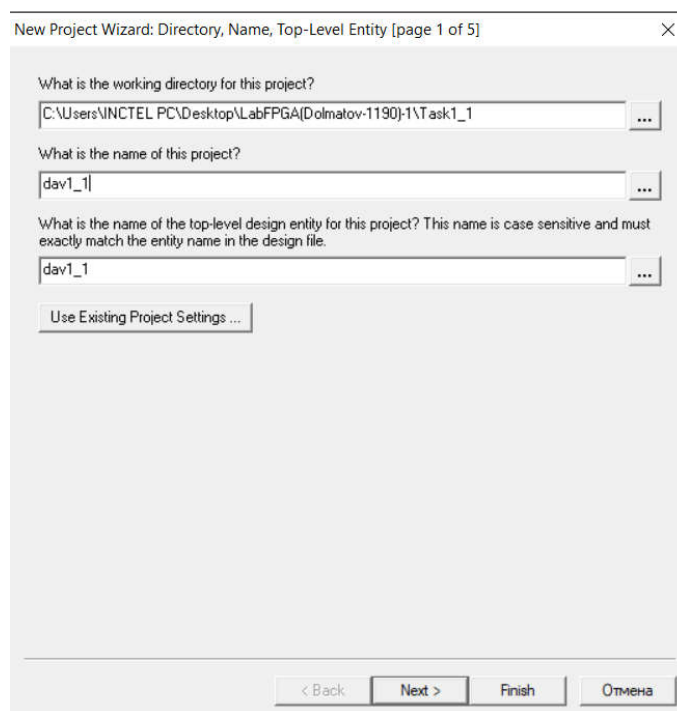


Рис. 23. Диалоговое окно создания проекта

### Построение схемы логической функции в Quartus

12. Выберите меню «File – New...». В появившемся диалоговом окне выделите пункт «Block Diagram/Schematic File» и нажмите «Ok». В приложении Quartus откроется окно для построения принципиальной схемы.
13. Щелкните правой кнопкой мыши в открывшемся окне и выберите во всплывающем меню пункт «Insert – Symbol...».
14. Диалоговое окно «Symbol» содержит панель «Libraries», в которой должен быть указан путь к папке с библиотеками. Раскройте библиотечную папку и выберите библиотеку примитивов (рис. 24).

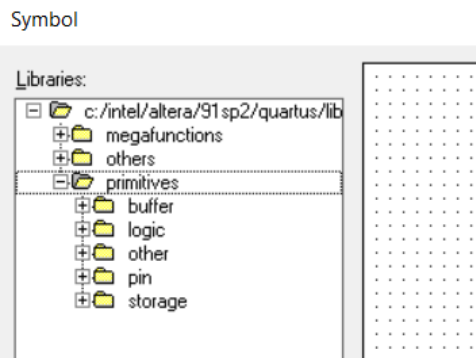


Рис. 24. Выбор библиотечной папки с примитивами

Затем раскройте папку «logic» и в ней найдите элемент «nand2». Выберите его и нажмите «Ok».

15. Используя сетку схемного окна разместите в нем 6 логических элементов И-НЕ (рис. 25).

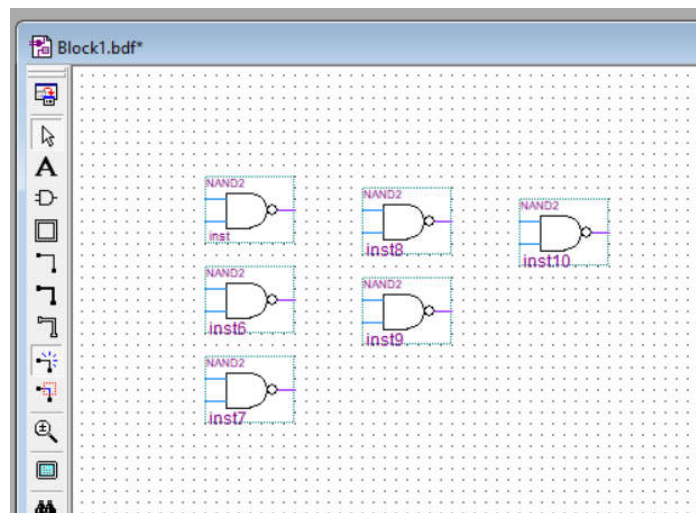



Рис. 25. Первый этап построения схемы логической функции

16. На панели инструментов схемного окна выберите «Orthogonal Node Tool»  и соедините элементы И-НЕ так, как показано на рисунке 26.

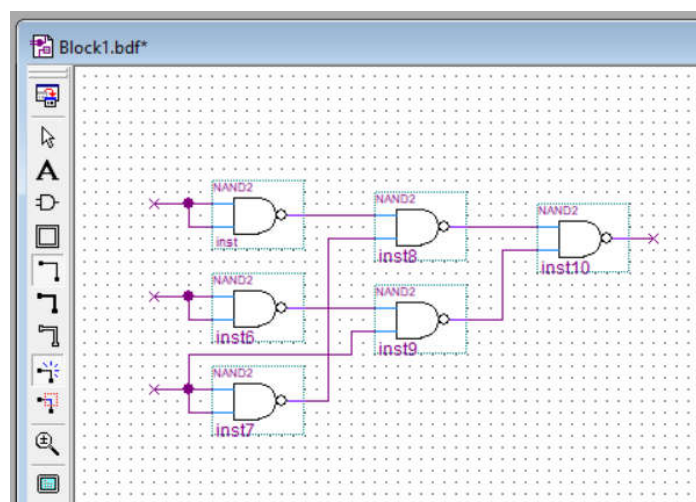


Рис. 26. Второй этап построения схемы логической функции



17. Для вставки в схему входных и выходных элементов раскройте диалоговое окно «Symbol». В папке примитивов разверните каталог «pin» и выберите элемент «Input». Добавьте в схему 3 входных элемента. В каталоге «pin» выберите элемент «Output» и добавьте его в схему.
18. Используя сетку схемного окна и инструмент «Orthogonal Node Tool» соедините входные и выходной элемент с элементами И-НЕ как показано на рисунке 27.

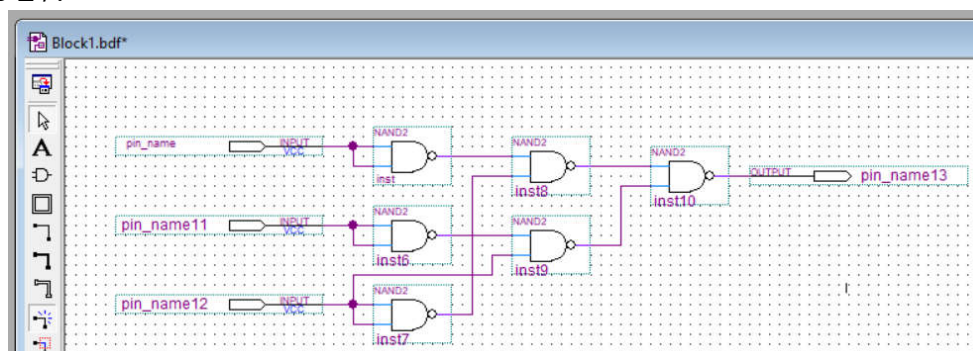


Рис. 27. Третий этап построения схемы логической функции

19. Дважды щелкните левой кнопкой мыши на элементе «Input» в схемном окне. В появившемся диалоге «Pin Properties» определите имя входного сигнала в соответствии с аргументом логической функции и задайте уровень сигнала по умолчанию: GND. Назначьте названия и значение по умолчанию для всех входных элементов «Input». Аналогично укажите имя выходного элемента «Output». В результате описанных действий должна получиться схема логической функции подобная рисунку 28.

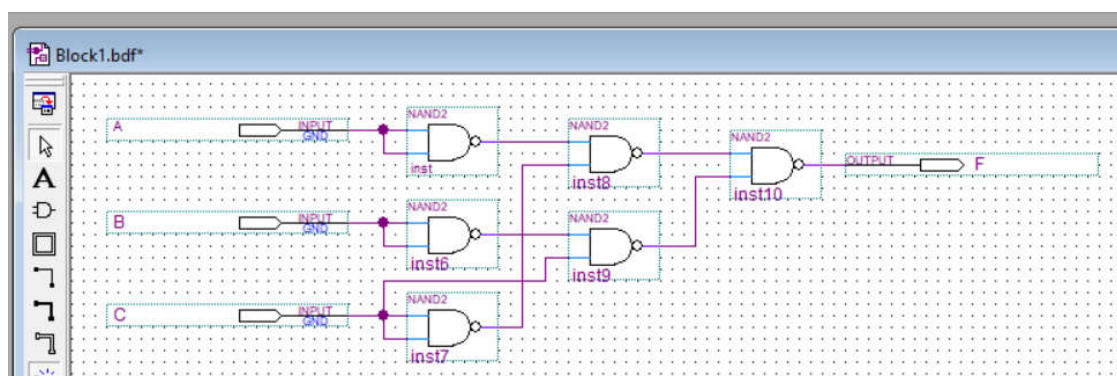


Рис. 28 Результат построения схемы логической функции

20. Сохраните файл со схемой логической функции. Для этого схемное окно должно быть активным в приложении Quartus. Выберите меню «File – Save As...». Укажите такое же имя как у файла проекта и нажмите «Save» («Сохранить»). Файл со схемой имеет расширение «.bdf».
21. После построения схемы в проекте Quartus ее следует сделать основным элементом в иерархии проекта. Для этого при активном схемном окне нужно выбрать меню «Project – Set as Top-Level Entity».
22. В завершение построения схемы логической функции нужно выполнить компиляцию проекта с помощью меню «Processing – Start Compilation». Если

схема построена верно, то компиляция завершится без ошибок сообщением: «Full Compilation was successful».

### Симуляция работы схемы логической функции

23. Проверить правильность работы логической функции в проекте Quartus можно с помощью встроенного симулятора. Настройка работы симулятора начинается с выбора меню «File – New...». В появившемся диалоговом окне следует выбрать пункт «Vector Waveform File» и нажать «Ok». Появится окно симулятора, основными элементами которого является панель инструментов, список сигнальных линий, участвующих в симуляции, и область с диаграммой сигналов.
24. Щелкните правой кнопкой мыши по пустому списку сигнальных линий и во всплывающем меню выберите «Insert Node or Bus...».
25. В диалоговом окне «Insert Node or Bus» нажмите кнопку «Node Finder...».
26. В диалоговом окне «Node Finder» настройте поля фильтра, как показано на рисунке 29.

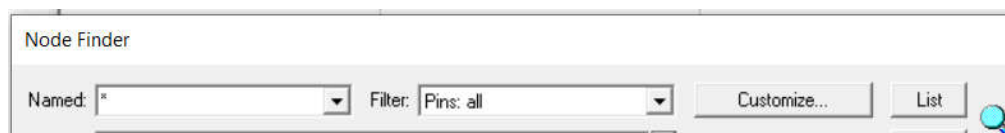



Рис. 29. Настройка параметров диалога «Node Finder»

Нажмите кнопку «List». В списке «Nodes Found» должны появиться названия выводов логической функции, построенной ранее в проекте Quartus. С помощью кнопок со стрелками перенесите все названия в список «Selected Nodes». Нажмите «Ok». Произойдет возврат в диалог «Node Finder». В нем также нажмите «Ok». Теперь в окне симулятора отображаются названия всех выводов схемы логической функции. В области временной диаграммы сигналов на всех входных линиях установлено значение по умолчанию, которое ранее было определено при построении схемы логической функции. Для выхода логической функции значение неопределенно. Его временная диаграмма будет построена в результате симуляции.

27. Настройте временную диаграмму входных сигналов логической функции так, чтобы она позволяла реализовать всю таблицу истинности заданной логической функции. На рисунке 17 видно, что с наибольшей частотой изменяется сигнал на входе «С», а наименьшую частоту имеет сигнал на входе «А». Так для реализации таблицы истинности заданной функции достаточно 1 периода изменения сигнала на входе «А», 2-х периодов на входе «В» и 4-х периодов на входе «С». Пусть период колебания сигнала на входе «А» составляет 80 нс. Чтобы на временной диаграмме задать регулярный сигнал с таким периодом выделите вход «А» в списке симулятора и щелкните по элементу «Overwrite Clock»  на панели инструментов. В диалоговом окне «Clock» установите следующие значения

параметров: «Start time» - 0 ps; «End time» - 80 ns; «Period» - 80 ns; «Offset» - 0 ns; «Duty cycle» - 50%. Аналогичным способом установите для входа «B» сигнал с периодом 40 ns и общей длительностью 80 ns (EndTime - StartTime). Для входа «C» период должен составлять 20 ns при общей длительности сигнала 80 ns. В результате должна получиться временная диаграмма входных сигналов, показанная на рисунке 30. Она позволит симулировать все возможные комбинации аргументов логической функции.

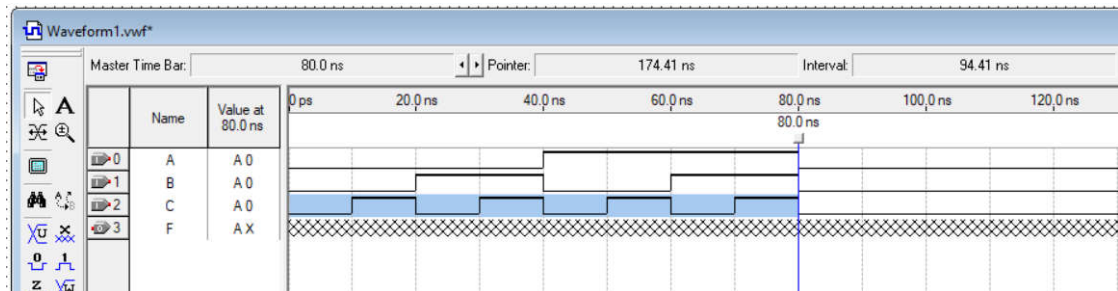


Рис. 30. Результат генерации входных сигналов для получения полного набора значений функции

28. Для установки конечного времени симуляции выберите меню «Edit – End Time...» и в поле «Time» задайте значение 80 ns.
29. Сохраните параметры симуляции, выбрав при активном окне симулятора меню «File – Save As...». В диалоговом окне укажите название файла с параметрами симулятора. Например, имя этого файла может совпадать с названием проекта. Файл симулятора в проекте имеет расширение «.vwf».
30. Задайте тип симуляции – «Functional». Для этого выберите меню «Assignments – Settings...». В диалоговом окне «Settings - ...» в списке «Category» выберите пункт «Simulator Settings». Затем в поле «Simulator mode» выберите режим «Functional», а в поле «Simulator input» укажите имя файла с параметрами симулятора, которое было задано в пункте 29. Нажмите «Ok».
31. Построенная схема будет использоваться для функциональной симуляции сигналов. Поэтому следует построить специальный **netlist** с помощью меню «Processing – Generate Functional Simulation Netlist».
32. Теперь можно запустить симуляцию с помощью меню «Processing – Start Simulation». В результате появится окно с отчетом симулятора «Simulation Report» и диалоговое окно, сообщающее об успешном завершении симуляции. Последнее можно закрыть, щелкнув «Ok».
33. При активном окне «Simulation Report» щелкните меню «Edit – Grid Size...» и в диалоговом окне «Grid Size» в поле «Period» установите значение 5 ns. Теперь для временной диаграммы сформируется сетка с шагом 5 ns, которая позволит мышью дискретно перемещать курсор в виде вертикальной синей линии и исследовать состояния входных и выходного сигналов построенной схемы (рис. 31).

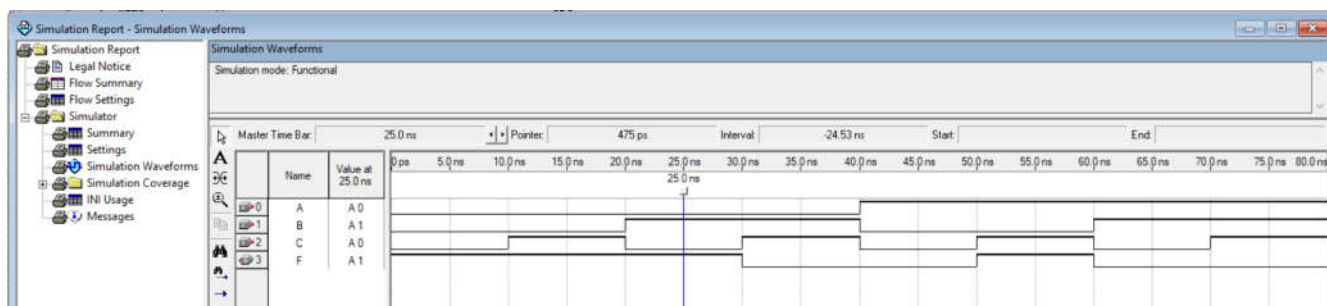


Рис. 31. Анализ функциональности схемы с помощью отчета симулятора

34. С помощью точек {5, 15, 25, 35, 45, 55, 65, 75} ns на временной диаграмме постройте таблицу сигналов схемы, реализованной в проекте Quartus, и сравните ее с таблицей истинности заданной логической функции.

35. Сохраните проект Quartus.

36. В соответствии с требованиями раздела 3 подготовьте отчет и сделайте представление работы над заданием преподавателю.

## 2.2. Самостоятельная работа

1. Для таблицы истинности, представленной на рисунке 17, запишите формулу логической функции в виде СКНФ.
2. С помощью карты Карно постройте МКНФ логической функции в булевом базисе.
3. Запишите формулу МКНФ в базисе Пирса.
4. В проекте Quartus (Task1\_2) постройте схему логической функции на основе МКНФ в базисе Пирса.
5. С помощью симулятора Quartus постройте таблицу истинности разработанной схемы логической функции.
6. Выполните сравнительный анализ исходной логической функции и ее реализации на основе логических элементов ИЛИ-НЕ. Приведите особенности реализации логической функции в ПЛИС (количество логических элементов, транзисторов, время срабатывания и др.)

## 2.3. Индивидуальное задание

1. В соответствии с вариантом индивидуального задания, приведенного в **Приложении 2**, запишите формулу логической функции в виде СДНФ и СКНФ.
2. С помощью карты Карно постройте МДНФ и МКНФ логической функции в булевом базисе.
3. Запишите формулы МНФ в базисе, указанном в варианте индивидуального задания.

4. В проекте Quartus (Task1\_3) постройте схемы логической функции на основе МДНФ и МКНФ в базисе, указанном в варианте индивидуального задания.
5. С помощью симулятора Quartus постройте таблицы истинности разработанных схем логической функции.
6. Выполните сравнительный анализ исходной логической функции и ее реализаций на основе элементов базиса, указанного в варианте индивидуального задания. Приведите особенности реализации логической функции в виде МДНФ и МКНФ в ПЛИС (количество логических элементов, транзисторов, время срабатывания и др.).

### 3. Подготовка отчета, представление и оценка работы

#### Структура отчета

1. Для каждой части задания требуется создать отдельный отчет. Файл отчета должен именоваться также как файл соответствующего проекта Quartus и иметь расширение «.doc». Отчет по части задания следует размещать в папке проекта (Task1\_1, Task1\_2 или Task1\_3).
2. Отчет должен включать следующие разделы и пункты:
  - В разделе «Дано» следует привести исходную таблицу истинности логической функции, тип нормальной формы оптимизированной функции (МДНФ или МКНФ); базис реализации формулы логической функции в электронной схеме (Шеффера или Пирса).
  - В раздел «Решение» нужно поместить:
    - a. формулу совершенной нормальной формы логической функции в соответствие с заданием (дизъюнктивную или конъюнктивную);
    - b. карту Карно с обозначенными группами покрытия;
    - c. формулу минимальной нормальной формы логической функции в соответствие с заданием (дизъюнктивную или конъюнктивную), построенную по карте Карно в базисе (И, ИЛИ, НЕ);
    - d. формулу минимальной нормальной формы в соответствие с заданием в базисе Шеффера или Пирса.
    - e. изображение окна приложения Quartus со схемой логической функции;
    - f. изображение окна «Simulation Report»;
    - g. результаты симуляции работы логической функции, представленные в виде таблицы со следующей структурой:

t, ns	Аргументы функции				Значение функции
	$x_1$	$x_2$	$x_3$	...	$f$
5	0	0	0	...	1
15	0	0	0	...	0
...	...	...	...	...	...

- В разделе «Заключение» следует сделать обоснованный вывод об особенностях реализации заданной функции в программируемых логических интегральных схемах (ПЛИС).

### Представление и защита работы

3. Каждая часть задания лабораторной работы может быть представлена отдельно. В ходе представления студент должен дать пояснения по выводу формул СНФ и МНФ в разных базисах в соответствии с заданием, продемонстрировать в приложении Quartus схему логической функции и навыки работы с симулятором сигналов.
4. Преподаватель оценивает знания по оптимизации логических функций на основе карт Карно и навыки разработки и симуляции работы схем в приложении Quartus. Студент должен быть готов ответить на дополнительные вопросы преподавателя, связанные с построением проекта Quartus, в рамках материалов лабораторной работы.
5. После получения оценки за представленные проекты лабораторной работы их вместе с отчетами следует загрузить на сайт Eluniver. Отчет по каждой части задания оценивается отдельно.
6. После представления обучающей и самостоятельной частей работы студент получает право ответить на 2 контрольных вопроса из обозначенного в разделе 4 списка.

### Структура оценки лабораторной работы

№	Вид оценки	Максимальный балл
1.	Проект «Обучающая часть»	10
2.	Проект «Самостоятельная работа»	15
3.	Проект «Индивидуальное задание»	25
4.	Отчет «Обучающая часть»	5
5.	Отчет «Самостоятельная работа»	5
6.	Отчет «Индивидуальное задание»	10
7.	Контрольный вопрос 1	15
8.	Контрольный вопрос 2	15
<b>Итого:</b>		<b>100</b>



#### 4. Контрольные вопросы

1. Какова размерность вектора значений логической функции 4-х аргументов?
2. Сколько можно записать различных логических функций 3-х аргументов?
3. Что понимают под булевой алгеброй?
4. Является ли в логической функции трех аргументов  $f(a,b,c)$  выражение  $a \cdot \bar{b}$  простой конъюнкцией? Поясните.
5. Что понимается под простой дизъюнкцией? Приведите пример.
6. Можно ли выражение  $a \cdot b \cdot \bar{c} \vee c \cdot \bar{a} \cdot b$  считать совершенной дизъюнктивной нормальной формой логической функции 4-х аргументов? Поясните.
7. Можно ли выражение  $a \cdot \bar{b} \vee \bar{a} \cdot \bar{b}$  считать совершенной конъюнктивной нормальной формой логической функции 2-х аргументов? Поясните.
8. Какое количество строк содержит таблица истинности логической функции 5 аргументов?
9. Какие строки таблицы истинности используются для построения СДНФ?
10. В каком виде входит переменная в простую дизъюнкцию, если она принимает значение 1? Приведите пример.
11. Являются ли СДНФ и СКНФ эквивалентными формулами записи логической функции?
12. Каким образом можно проверить эквивалентность двух логических формул?
13. Чем различаются переменная и буква логической функции? Поясните.
14. Что понимается по минимальной нормальной формой логической функции?
15. Какое количество групп на карте Карно соответствует формуле  $f = a \vee \bar{b} \cdot \bar{a}$ ?
16. Является ли формула  $f = a \cdot \bar{c} \vee a \cdot b \vee b \cdot c$  МНФ? Поясните.
17. Сколько букв будет иметь логическая формула четырех аргументов, если на карте Карно выделено две области размерностью  $2 \times 2$ ?
18. Каково минимальное количество полевых транзисторов требуется для построения логических элементов базиса (И, НЕ)?
19. Какие из известных базисов булевой алгебры являются избыточными?
20. Какие базисы алгебры логики имеют только одну элементарную функцию?
21. Как в базисе И-НЕ можно реализовать логическую функцию отрицания?
22. Как в приложении Quartus можно построить схему логической функции?
23. Каким образом в приложении Quartus осуществляется именованное выведение логической схемы?
24. Какое действие на проект Quartus оказывает выбор меню «Project – Set as Top-Level Entity»?
25. Каким образом с помощью симулятора можно построить таблицу истинности логической схемы, разработанной в Quartus?
26. Как в Quartus создается netlist? Каково его назначение?
27. Какие типы работы симулятора реализованы в Quartus?
28. Как в Quartus можно задать временной интервал симуляции?
29. Для чего в Quartus служит команда «Edit – Grid...»?
30. Как в Quartus можно построить входной сигнал с коэффициентом заполнения 30%?

## Приложение 1. Законы булевой алгебры

N п/п	Аналитическое выражение	Примечания
1	$X = 0$ , если $X \neq 1$ ; $X = 1$ , если $X \neq 0$	Определение двоичной (булевой) переменной
2	$0 \cdot 0 = 0$ ; $1 + 1 = 1$	Второму соотношению нет аналога в обычной арифметике
3	$1 \cdot 1 = 1$ ; $0 + 0 = 0$	
4	$1 \cdot 0 = 0 \cdot 1 = 0$ ; $0 + 1 = 1 + 0 = 1$	
5	$\bar{0} = 1$ $\bar{1} = 0$	Определение операции «инверсия», «отрицание» (НЕ)
6	$X + 0 = X$ ; $X \cdot 1 = X$	
7	$1 + X = 1$ ; $0 \cdot X = 0$	$1 + X + Y + \dots = 1$
8	$X + X = X$ ; $X \cdot X = X$	$nX = X$ ; $X^n = X$
9	$\overline{(\bar{X})} = X$ $\overline{(\bar{\bar{X}})} = \bar{\bar{X}} = X$	Двойная инверсия оставляет логическое выражение неизменным
10	$X + \bar{X} = 1$ $\bar{X} \cdot X = 0$	
11	$X \cdot Y + X \cdot Z = X \cdot (Y + Z)$ $(X + Y) \cdot (X + Z) = X + Y \cdot Z$	Распределительный закон Закон поглощения
12	$\overline{X + Y + Z + \dots} = \bar{X} \cdot \bar{Y} \cdot \bar{Z} \cdot \dots$ $\overline{X \cdot Y \cdot Z \cdot \dots} = \bar{X} + \bar{Y} + \bar{Z} \dots$	Теорема де Моргана
13	$f(x_1, x_2, \dots, x_n) = x_1 f(1, x_2, \dots, x_n) + \bar{x}_1 \cdot f(0, x_2, \dots, x_n)$ ; $f(x_1, x_2, \dots, x_n) = [x_1 + f(0, x_2, \dots, x_n)] \cdot [\bar{x}_1 + f(1, x_2, \dots, x_n)]$	Теорема разложения

## Приложение 2. Варианты индивидуальных заданий

Номер варианта	Вектор значений логической функции	Базис для построения схемы
1.	(1,1,1,0,0,0,0,1,1,0,1,0,0,1,1,1)	(И, НЕ)
2.	(1,0,0,0,0,1,1,1,1,0,0,0,0,0,1,1)	(ИЛИ-НЕ)
3.	(1,1,1,0,0,0,0,0,1,1,0,0,0,1,1,0)	(И-НЕ)
4.	(0,0,0,0,0,1,1,1,1,0,0,0,1,1,1,0)	(ИЛИ, НЕ)
5.	(0,0,0,1,1,1,0,0,0,1,1,0,1,1,0,0)	(И-НЕ)
6.	(1,0,0,0,0,1,1,0,0,0,1,1,1,1,0,1)	(И, НЕ)
7.	(1,1,0,0,0,0,0,0,1,1,0,0,0,0,1,1)	(ИЛИ-НЕ)
8.	(0,0,1,1,1,1,1,1,0,0,1,1,0,0,0,0)	(ИЛИ, НЕ)
9.	(1,1,0,0,1,0,0,0,0,1,1,1,0,0,1,1)	(И-НЕ)
10.	(1,1,1,1,1,1,0,0,0,1,1,1,1,1,0,0)	(ИЛИ-НЕ)
11.	(0,0,0,0,0,1,1,1,0,0,1,1,1,1,0,0)	(И, НЕ)
12.	(1,1,1,0,0,0,0,1,1,0,1,1,1,0,0,1)	(ИЛИ, НЕ)
13.	(0,0,0,0,1,0,0,0,1,1,1,0,0,1,1,0)	(И-НЕ)
14.	(1,0,0,1,1,1,1,0,0,0,0,1,1,1,0,1)	(И, НЕ)
15.	(1,1,1,1,1,1,0,0,1,1,0,0,1,1,0,0)	(ИЛИ-НЕ)
16.	(0,0,0,0,1,1,1,1,1,1,0,0,0,1,1,0)	(ИЛИ, НЕ)
17.	(0,1,1,1,1,1,0,0,0,0,1,1,1,0,0,0)	(И, НЕ)
18.	(1,1,1,1,1,0,0,0,1,1,0,0,1,0,0,0)	(ИЛИ-НЕ)
19.	(0,0,0,1,1,1,1,0,0,0,1,1,0,0,1,1)	(И-НЕ)
20.	(1,0,1,0,1,0,1,1,1,1,0,0,0,0,1,1)	(ИЛИ, НЕ)
21.	(1,1,0,0,0,0,0,0,0,1,1,1,1,0,1,1)	(ИЛИ-НЕ)
22.	(0,0,0,0,0,0,0,1,1,1,1,1,0,1,0,1)	(И, НЕ)
23.	(1,1,1,1,1,1,0,1,0,1,0,0,0,0,0,1)	(И-НЕ)
24.	(0,1,0,1,0,0,0,0,1,1,1,1,1,1,0,0)	(И, НЕ)
25.	(1,1,1,0,0,1,1,1,0,0,1,1,1,1,0,0)	(ИЛИ, НЕ)
26.	(1,1,0,0,0,1,0,0,0,0,0,0,0,1,1,1)	(ИЛИ-НЕ)
27.	(1,1,1,1,1,1,1,0,1,1,1,0,0,0,0,1)	(И, НЕ)
28.	(0,0,0,0,0,0,1,1,1,0,0,0,1,1,1,1)	(ИЛИ, НЕ)
29.	(1,1,0,0,0,0,0,0,1,1,1,1,0,0,0,1)	(ИЛИ-НЕ)
30.	(0,0,1,1,1,0,0,1,0,0,0,1,0,0,0,1)	(И-НЕ)