ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ДИСТАНЦИОННОГО ОБУЧЕНИЯ И ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

Кафедра «Вычислительные системы и информационная безопасность»

Сборник задач

по дисциплине

«Интерфейсы автоматизированных систем»

Авторы
Ганжур М.А,
Ганжур А.П

Ростов-на-Дону, 2018

Аннотация

Сборник задач предназначен для студентов очной, заочной форм обучения направлений 10.03.01 «Информационная безопасность», 09.03.02 «Информационные системы и технологии»

Авторы

ст. преподаватель кафедры «Вычислительные системы и информационная безопасность» Ганжур М.А,
ст. преподаватель кафедры «Вычислительные системы и информационная безопасность» Ганжур А.П.

Оглавление

ЛАБОРАТОРНАЯ РАБОТА 1

ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ И СХЕМЫ

ЦЕЛЬ РАБОТЫ

Ознакомление с основными характеристиками логических элементов и основами синтеза логических схем.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ И РАСЧЕТНЫЕ ФОРМУЛЫ

1. ОПРЕДЕЛЕНИЯ КОМБИНАЦИОННЫХ И ПОСЛЕДОВАТЕЛЬНОСТНЫХ УСТРОЙСТВ

Устройства, реализующие функции алгебры логики, называют *логическими* или *цифровыми* и классифицируют по различным отличительным признакам. Так, по характеру информации на входах и выходах логические устройства подразделяют на устройства последовательного, параллельного и смешанного действия, а по схемному решению и характеру связи между входными и выходными переменными с учётом их изменения по тактам работы – на комбинационные и последовательностные.

В *комбинационных* устройствах значения (0 или 1) сигналов на выходах в каждый конкретный момент времени полностью определяются значениями (комбинацией, набором) действующих в данный момент цифровых входных сигналов. В *последовательностных* же устройствах значения выходных сигналов в *n*-такте определяются не только значениями входных сигналов в этом такте, но и зависят от внутренних состояний устройств, которые произошли в результате воздействия входных сигналов в предшествующие такты.

Данная работа посвящена изучению простейших комбинационных логических устройств, реализующих логические функции сложения, умножения и отрицания.

2. ОСНОВНЫЕ ЭЛЕМЕНТЫ АЛГЕБРЫ ЛОГИКИ

Анализ комбинационных устройств удобно проводить с помощью алгебры логики, оперирующей только с двумя понятиями: истинным (логическая 1) и ложным (логический 0). В результате, функции, отображающие информацию, принимают в каждый момент времени только значения 0 или 1. Такие функции называют *логическими*, а сигналы (входные и выходные переменные) – *двоичными* (бинарными).

Схемные элементы, при помощи которых осуществляется преобразование поступающих на их входы двоичных сигналов и непосредственное выполнение предусмотренных логических операций, называют *логическими* устройствами.

В общем случае логическое устройство может иметь *n* вхо-

дов и m выходов. Рассматривая входные сигналы x_1, x_2, …, x_n в качестве аргументов, можно соответствующие выходные сигналы представлять в виде функции $y_i = f(x_0, x_1, x_2, \ldots, x_n)$ с помощью операций алгебры логики.

Функции алгебры логики (ФАЛ), иногда называемые *переключательными* функциями, обычно представляют в алгебраической форме (в виде математического выражения), например $y_i = (x_0 \wedge x_1) \vee (x_1 \wedge x_2)$, или в виде таблиц истинности (комбинационных таблиц).

Таблица истинности содержит всевозможные комбинации (наборы) бинарных значений входных переменных с соответствующими им бинарными значениями выходных переменных; каждому набору входных сигналов соответствует определенное значение выходного сигнала – значение логической функции y_i. Максимальное число возможных различных наборов (строк) зависит от числа входных переменных n и равно 2^n.

В булевой алгебре выделяют три основные функции: конъюнкция, дизъюнкция, отрицание. Остальные функции являются производными от приведенных выше.

Основные логические операции состоят из следующих элементарных преобразований двоичных сигналов:

• *логическое сложение* или *дизъюнкция*, обозначаемое символом "$\vee$" (или "+") и называемое также операцией ИЛИ. При этом число аргументов (слагаемых x) может быть любым. Эта операция для функции двух переменных x_1 и x_2 описывается в виде логической формулы

$$y = x_1 \vee x_2 = x_1 + x_2.$$

Это значит, что y истинно (равно 1), если истинно хотя бы одно из слагаемых x_1 или x_2. И только в случае, когда все слагаемые x равны 0, результат логического сложения y также равен 0.

Условное обозначение, таблица истинности и другие показатели этой логической функции приведены во втором столбце табл. 1;

• *логическое умножение* или *конъюнкция*, обозначаемое символом "$\wedge$" (или "$\cdot$") и называемое также операцией И. При этом число аргументов (сомножителей x) может быть любым. Эта операция для функции двух переменных x_1 и x_2 описывается в виде логической формулы

$$y = x_1 \wedge x_2 = x_1 \cdot x_2 = x_1 x_2.$$

Это значит, что y истинно (равно 1), если истинны сомножители x_1 и x_2. В случае, если хотя бы один из сомножителей ра-

вен 0, результат логического умножения *y* равен 0.

Условное обозначение, таблица истинности и другие показатели логической функции И приведены в третьем столбце табл. 1;

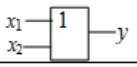
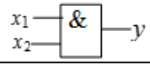
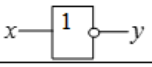
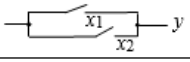
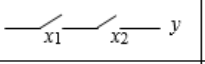
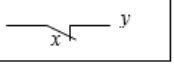
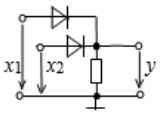
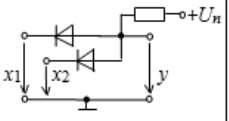
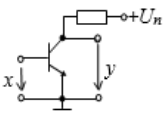
• *логическое отрицание* или *инверсия*, обозначаемое чёрточкой над переменной и называемое операцией НЕ. Эта операция записывается в виде

$$y = \bar{x}.$$

Это значит, что *y* истинно (равно 1), если *x* ложно (равно 0), и наоборот. Очевидно, что операция *y* выполняется над одной переменной *x* и её значение всегда противоположно этой переменной (см. четвертый столбец табл. 1).

Таблица 1

Формы отображения основных логических функций

Наименование функции →	Дизъюнкция	Конъюнкция	Инверсия
Символическая	∨ или +	∧ или ·	$\bar{x}$
Буквенная	ИЛИ	И	НЕ
Условная графическая			
Аналитическая	$y = x_1 \vee x_2 = x_1 + x_2$	$y = x_1 \wedge x_2 = x_1 x_2$	$y = \bar{x}$
Табличная (истинности)	x_1 | x_2 | y 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1	x_1 | x_2 | y 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1	x | y 0 | 1 1 | 0
Контактная			
Схемотехническая			

Основные логические операции ИЛИ, И и НЕ позволяют аналитически описать, а логические элементы ИЛИ (*дизъюнктор*), И (*конъюнктор*) и НЕ (*инвертор*) – реализовать комбинационное устройство любой степени сложности, т. е. операции $y = x_1 + x_2$, $y = x_1 x_2$ и $y = \bar{x}$ обладают функциональной полнотой и составляет функционально полный набор.

В качестве примера рассмотрим функцию неравнозначности

y двух переменных x_1 и x_2, принимающая значение 1 при $x_1 \neq x_2$ и значение 0 при $x_1 = x_2 = 0$ или при $x_1 = x_2 = 1$, т. е. $y = \bar{x}_1 x_2 + x_1 \bar{x}_2$.

Операцию неравнозначности чаще называют *суммированием по модулю* 2 и обозначают $y = x_1 \oplus x_2$.

Примеры контактной и простейшей схемной реализаций дизъюнктора, конъюнктора и инвертора приведены в предпоследней и последней строках табл. 1.

3. БАЗОВЫЕ ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ

Особое значение в цифровой электронике имеют универсальные (базовые) логические элементы, способные образовать функционально полный набор, с помощью которых можно реализовать синтез устройств любой сложности. При интегральной технологии удобство изготовления одного базового элемента имеет решающее значение. Поэтому базовые логические устройства составляют основу большинства цифровых ИМС.

К универсальным логическим операциям (устройствам) относят две разновидности базовых элементов:

- *функцию Пирса*, обозначаемую символически вертикальной стрелкой ↓ (стрелка Пирса) и отображающую операцию ИЛИ-НЕ. Для простейшей функции двух переменных x_1 и x_2 функция $y = 1$ тогда и только тогда, когда $x_1 = x_2 = 0$:

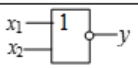
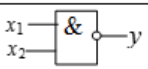
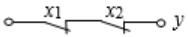
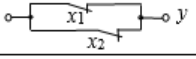
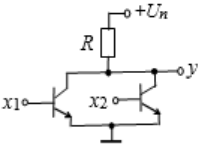
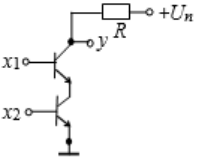
$$y = x_1 \downarrow x_2 = \overline{x_1 + x_2};$$

- *функцию Шеффера*, обозначаемую символически вертикальной черточкой | (штрих Шеффера) и отображающую операцию И-НЕ. Для простейшей функции двух переменных x_1 и x_2 функция $y = 0$ тогда и только тогда, когда $x_1 = x_2 = 1$:

$$y = x_1 \mid x_2 = \overline{x_1 x_2}.$$

Таблица 2

Формы отображения базовых логических функций

Наименование функции →	Функция Пирса	Функция Шеффера
Символическая	↓	|
Буквенная	ИЛИ-НЕ	И-НЕ
Условная графическая		
Аналитическая	$y=x_1 \downarrow x_2$	$y=x_1 \mid x_2$
Табличная (истинности)	x_1 x_2 y 0 0 1 0 1 0 1 0 0 1 1 0	x_1 x_2 y 0 0 1 0 1 1 1 0 1 1 1 0
Контактная		
Схемо-техническая		

При одних и тех же значениях аргументов обе функции отображают операцию инверсии. Важнейшие показатели функций Шеффера и Пирса представлены в табл. 2.

В последней строке табл. 2 приведены примеры построения двухвходовой схемы ИЛИ-НЕ, в которой к нагрузочному резистору *R* подключены коллекторы двух параллельно включенных биполярных транзисторов *p-n-p*-типа, эмиттеры которых заземлены, и схемы И-НЕ, в которой последовательно включены два биполярных транзистора *p-n-p*-типа (эмиттер нижнего транзистора подключен к земле) и нагрузочный резистор *R*.

4. ПРЕДСТАВЛЕНИЕ ЛОГИЧЕСКИХ ФУНКЦИЙ МАТЕМАТИЧЕСКИМИ ВЫРАЖЕНИЯМИ

Наиболее распространенным способом задания логических функций является табличная форма. Таблицы истинности позволяют полно и однозначно установить все существующие логические связи.

При табличном представлении логических функций их записывают в одной из канонических форм: совершенной дизъюнктивной нормальной форме (СДНФ) или совершенной конъюнктивной нормальной форме (СКНФ).

Математическое выражение логической функции в СДНФ получают из таблицы истинности следующим образом: для каждого набора аргументов, на котором функция равна 1, записывают элементарные произведения переменных, причем переменные, значения которых равны нулю, записывают с инверсией. Полученные произведения, называемые *конституентами единицы* или *минтермами*, суммируют.

Таблица 3

№	*a*	*b*	*c*	*y*
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Запишем логическую функцию *y* трех переменных *a*, *b* и *c*, представленной в виде табл. 3, в СДНФ:

$$y(a,b,c)=\bar{a}bc+a\bar{b}c+ab\bar{c}+abc.$$

Совершенной конъюнктивной нормальной формой называют логическое произведение элементарных сумм, в каждую из которых аргумент или его отрицание входят один раз.

При этом для каждого набора аргументов таблицы истинности, на котором функция *y* равна 0, составляют элементарную сумму, причем переменные, значение которых равно 1, записывают с отрицанием. Полученные суммы, называемые *конституентами нуля* или *макстермами*, объединяют операцией логического умножения.

Для функции (табл. 3) СКНФ

$$y(a,b,c)=(a+b+c)(a+b+\bar{c})(a+\bar{b}+c)(\bar{a}+b+c).$$

5. ПЕРЕХОД ОТ ЛОГИЧЕСКОЙ ФУНКЦИИ К ЛОГИЧЕСКОЙ СХЕМЕ

Для построения логической схемы необходимо логические элементы, предназначенные для выполнения логических операций, располагать, начиная от входа, в порядке, указанном в булевом выражении.

Построим структуру логического устройства, реализующего логическую функцию трех переменных

$$y=(a+b+c)(a+b+\bar{c})(\bar{a}+b+c)(\bar{a}+\bar{b}+c).$$

Слева располагаем входы *a*, *b* и *c* с ответвлениями на три инвертора, затем четыре элемента ИЛИ и, наконец, элемент И на выходе (рис. 1).

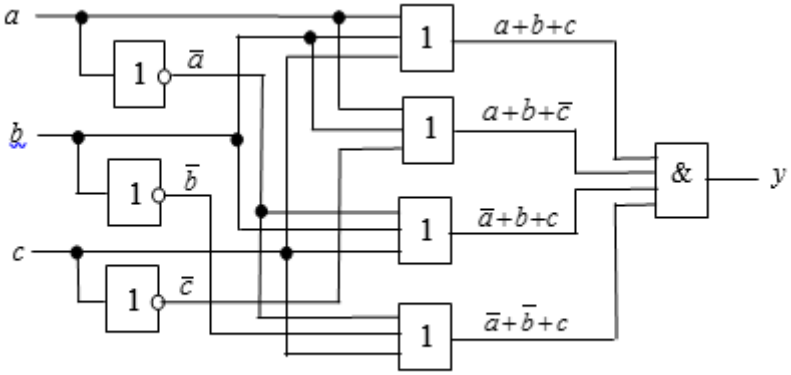


Рис. 1

Итак, любую логическую функцию можно реализовать непосредственно по выражениям, представленным в виде СДНФ или СКНФ. Однако, полученная таким образом схема, как правило, не оптимальна с точки зрения её практической реализации: она громоздка, содержит много логических элементов и возникают трудности в обеспечении её высокой надёжности.

Алгебра логики позволяет преобразовать формулы, описывающие сложные высказывания с целью их упрощения [10]. Это помогает в конечном итоге определить оптимальную структуру того или иного логического устройства, реализующего любую сложную функцию. Под оптимальной структурой принято понимать такое построение логического устройства, при котором число входящих в его состав элементов минимально.

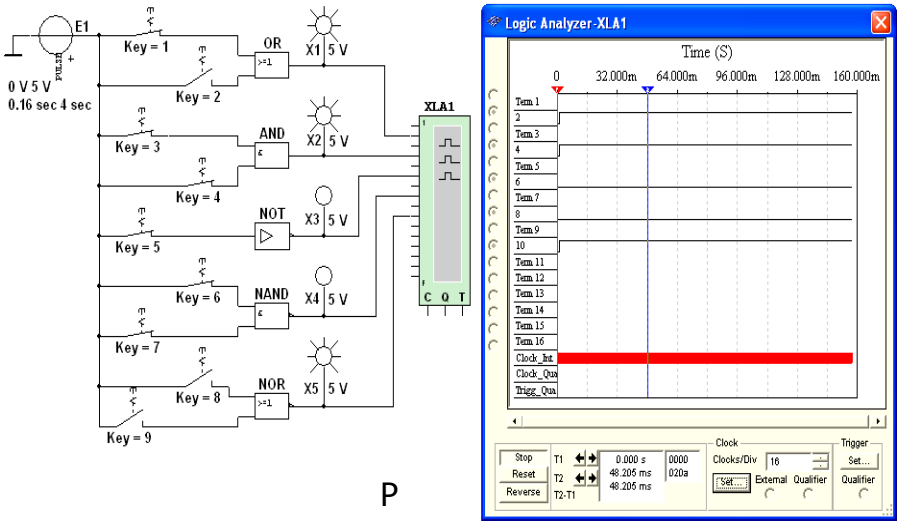
УЧЕБНЫЕ ЗАДАНИЯ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ИХ ВЫПОЛНЕНИЮ

Задание 1. **Запустить** лабораторный комплекс **Multisim**. **Собрать** схему в программе и **Сохранить** на страницу своего отчёта (схема (рис. 2)).

Схема (рис. 2) собрана на двоичных основных [**OR** (ИЛИ), **AND** (И) и **NOT** (НЕ)] и универсальных (базовых) [**NAND** (И-НЕ) и **XOR** (ИЛИ-НЕ)] логических элементах, расположенных в библиотеке **Misc Digital/TIL** с уровнем высокого напряжения 5 В. В схе-

му включены ключи **1**, **2**, …, **9**, пробники **X1**, **X2**, …, **X5** с пороговыми напряжениями 5 В, генератор прямоугольных сигналов **E1** с амплитудой E = 5 В, длительностью импульса $t_и$ = = 0,16 с и периодом T = 4 с, и логический анализатор **XLA1**.

Для удобства измерения сигналов выходы логических элементов подключены к входам 2, 4, 6, 8 и 10 анализатора **XLA1**. При моделировании происходит медленная развёртка временных диаграмм в окне анализатора. По достижению интервала времени, равном 70…80% ширины окна, следует посредством кнопки **Run/Stop** выключать процесс моделирования.



Р

Оперируя ключами **1**, **2**, …, **9**, **сформировать** все возможные комбинации аргументов x_1 и x_2 (00, 10, 01 и 11) на входе дизъюнктора (**OR**), конъюнктора (**AND**), штриха Шеффера (**NAND**) и стрелки Пирса (**NOR**) и **записать** значения выходных логических функций $y_к$ (0 или 1) в табл. 4.

Заметим, что если ключ замкнут, то на этот вход элемента будет подана логическая единица (положительный потенциал 5 В), а при разомкнутом ключе – логический ноль. Поскольку инвертор (**NOT**) имеет один вход, то для формирования двух значений входного сигнала (логической единицы или логического нуля) достаточно одного ключа **5**.

Значения функций исследуемых элементов можно контролировать с помощью пробников **X1**, **X2**, …, **X5**: если выходной сигнал элемента равен логической единице, то включенный на выходе этого элемента пробник светится. Так, при положении

ключей схемы (рис. 2) функции элементов **OR**, **AND** и **NOR** равны логической единице.

Таблица 4

Дизъюнктор [ИЛИ (**OR**)]			Конъюнктор [И (**AND**)]			Инвертор [НЕ (**NOT**)]		Штрих Шеффера [И-НЕ (**NAND**)]			Стрелка Пирса [ИЛИ-НЕ (**NOR**)]		
$x1$	$x2$	y	$x1$	$x2$	y	x	y	$x1$	$x2$	y	$x1$	$x2$	y
0	0		0	0		0		0	0		0	0	
0	1		0	1				0	1		0	1	
1	0		1	0		1		1	0		1	0	
1	1		1	1				1	1		1	1	

Задание 2. "**Перетащить**" из библиотеки **Misc Digital\TIL** на рабочее поле среды **Multisim** необходимые логические элементы и **собрать** схему для реализации заданной в табл. 5 логической функции *y* с тремя аргументами *a*, *b* и *c*. **Скопировать** собранную логическую схему на страницу отчёта.

Таблица 5

Вариант	Логическая функция
1, 6, 11, 16, 21, 26	$y=(\bar{a}b+\bar{c})(\bar{a}+\bar{b}+c)(a+b+c).$
2, 7, 12, 17, 22, 27	$y=(a+b+\bar{c})(\bar{a}+\bar{b}c)(a+\bar{b}+\bar{c}).$
3, 8, 13, 18, 23, 28	$y=(b+a\bar{c})(\bar{a}+bc)(a+\bar{b}+c).$
4, 9, 14, 19, 24, 29	$y=(\bar{a}\bar{b}+\bar{c})(a+\bar{b}+c)(ab+\bar{c}).$
5, 10, 15, 20, 25, 30	$y=(a+\bar{b}c)(\bar{a}+b+\bar{c})(ab+c).$

В качестве примера соберём схему для реализации логической функции

$$y=(ab+\bar{c})(\bar{a}+\bar{b}+c)(a+b+c).$$

Анализ функции показывает, что для построения логической схемы нам потребуются три инвертора, три дизъюнктора, причем один дизъюнктор с двумя, а два – с тремя входами, и два конъюнктора, причём один с двумя, а другой с тремя входами.

"Перетащим" на рабочее поле среды MS10 необходимые модели логических элементов из библиотеки **Misc Digital\TIL**, располагая их, начиная с входа, а именно:

– три инвертора NOT (**NOT1**, **NOT2** и **NOT3**) для получения инверсий $\bar{a}, \bar{b}$ и $\bar{c}$ аргументов *a*, *b* и *c*;

– конъюнктор **AND1** с двумя входами для реализации функции *ab*;

– три дизъюнктора: **OR2** для реализации функции $y_1 = a + b + c$, **OR3** для реализации функции $y_2 = \bar{a}+\bar{b}+c$ и **OR1,** реа-

лизующий функцию y_3 = = $ab + \bar{c}$, разместив их друг под другом (см. рис. 3).

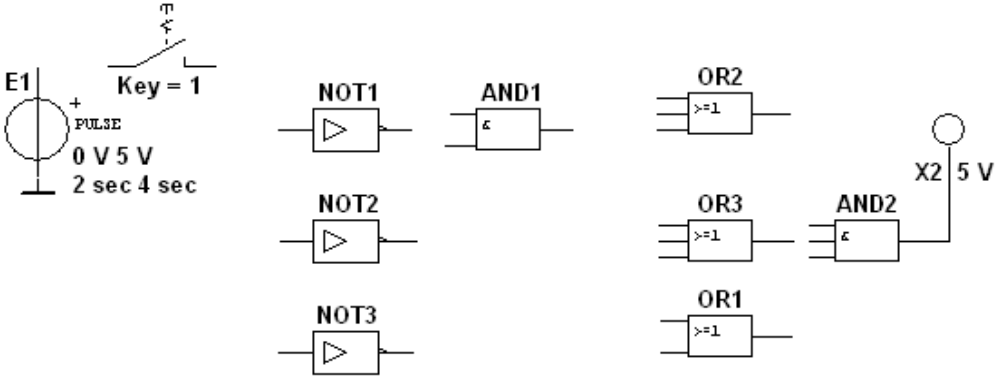
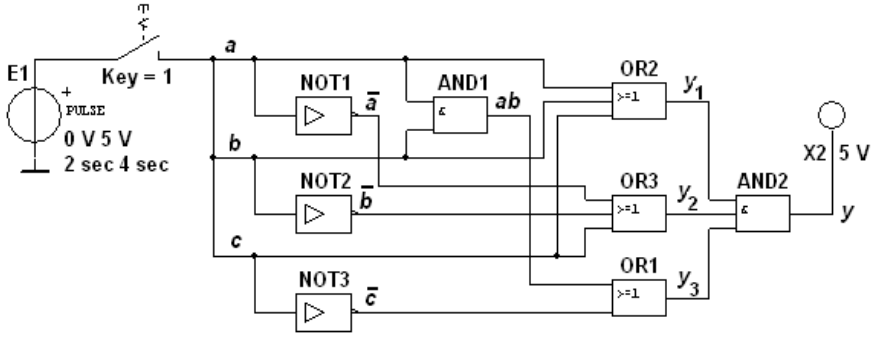


Рис. 3

Для выполнения функции логического умножения $y = y_1y_2y_3$ добавим в схему конъюнктор **AND2** с тремя входами, к выходу которого подключим логический пробник **X2** (уровень высокого напряжения 5 В) для сигнализации появления логической единицы на выходе схемы. "Перетащим" из соответствующих библиотек на рабочее поле источник прямоугольных сигналов **E1** и ключ **1**, расположив их на входе схемы.

Соединив "проводниками" входы и выходы элементов в соответствии с логическими выражениями составляющих заданной функции и записав в отчёте ожидаемые результаты выполнения операций на выходах элементов (рис. 4), приступим к моделированию.



Р

С этой целью вначале щелкнем мышью на кнопке **Run/Stop**, затем нажмём управляющую ключом клавишу с цифрой **1** клавиатуры. Если соеди- нения элементов выполнены

правильно, то пробник **X2** засветится. При выключении ключа **1** пробник гаснет и т. д. По окончании моделирования щёлкнем мышью на кнопке **Run/Stop**.

Примечания. **1**. Основным измерительным прибором для проверки цифровых электронных схем является логический пробник. После двойного щелчка мышью на его изображении в открывшемся окне нужно задать уровень высокого напряжения, например, 5 В (см. рис 4), при котором он светится. Если пробник не светится, то это обычно означает, что уровень проверяемого напряжения находится в промежутке между высоким и низким. Поиск неисправностей нужно начинать с проверки подачи сигналов высокого уровня генератором сигналов на входы элементов, затем проверить правильность выполнения ими логических функций в схеме и проконтролировать появление сигналов на выходах.

2. Таблицы истинности для рассмотренных библиотечных логических элементов можно вызвать нажатием клавиши помощи **F1** после выделения на схеме соответствующего элемента.

СОДЕРЖАНИЕ ОТЧЕТА

1. Наименование и цель работы.
2. Перечень приборов, использованных в экспериментах, с их краткими характеристиками.
3. Изображения электрической схемы для испытания логических элементов и собранной схемы для реализации заданной логической функции.
4. Таблицы истинности, отображающие работу исследуемых логических элементов.
5. Выводы по работе.

ТЕСТОВЫЕ ЗАДАНИЯ К РАБОТЕ

1. Укажите **признаки,** характеризующие основные логические элементы.

 На входах логических элементов аналоговые сигналы, а на выходах – цифровые

 Операции логического сложения, логического умножения и инверсия не составляют функционально полный набор

 Используя основные логические операции И, ИЛИ и НЕ, можно аналитически выразить любую сложную логическую функцию

 Минимальный логический базис составляют операции ИЛИ и НЕ или И и НЕ

 Входные и выходные сигналы логических элементов

могут принимать только два значения: логическую 1 и логический 0

 Операция логического сложения совпадает с операцией обычного сложения

2. Укажите **выражение** логической функции двух переменных x_1 и x_2, реализуемой элементом "Стрелка Пирса".

○ $y=\bar{x}_1x_2+x_1\bar{x}_2$ ○ $y=\overline{x_1x_2}$ ○ $y=\overline{x_1+x_2}$

○ $y=x_1\oplus x_2$ ○ $y=x_1+x_2$ ○ $y=x_1x_2$

3. Укажите **выражение** логической функции двух переменных x_1 и x_2, реализуемой элементом "Штрих Шеффера".

○ $y=\bar{x}_1x_2+x_1\bar{x}_2$ ○ $y=\overline{x_1x_2}$ ○ $y=x_1\oplus x_2$

○ $y=\overline{x_1+x_2}$ ○ $y=x_1+x_2$ ○ $y=x_1x_2$

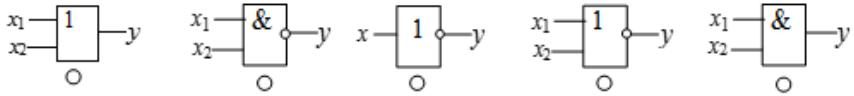
4. Укажите **выражение** логической функции трех переменных *а*, *б* и *с*, записанной в совершенной дизъюнктивной нормальной форме (СДНФ).

○ $y(a,b,c)=\bar{a}bc+a\bar{b}c+ab\bar{c}+abc$

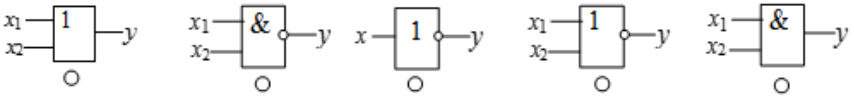
○ $y(a,b,c)=(a+b+c)(a+b+\bar{c})(a+\bar{b}+c)(\bar{a}+b+c)$

○ $y(a,b,c)=(\bar{a}b+c+a\bar{b}c)(ab\bar{c}+\overline{a\bar{b}}+\bar{c}a)$

5. Укажите **элемент** ИЛИ-НЕ.



6. Укажите **элемент** И.



7. Укажите значение **функции** $y=(ab+\bar{c})(\bar{a}+\bar{b})$, если $a=b=c=1$.

○ 1 ○ 0

ЛАБОРАТОРНАЯ РАБОТА 2

ПРЕОБРАЗОВАТЕЛИ КОДОВ

ЦЕЛЬ РАБОТЫ

Ознакомление с основными характеристиками и испытание интегральных преобразователей кодов (дешифратора, шифрато-

ра, демультиплексора и мультиплексора).

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ И РАСЧЕТНЫЕ ФОРМУЛЫ

Кодом называют систему символов для представления информации в форме, удобной для обработки, хранения и передачи. В цифровой технике для записи кодовых символов, или просто кода, используют две цифры: 0 и 1. *Преобразователи кодов* служат для перевода одной формы бинарного числа (кодовой комбинации) в другую, например, преобразование двоично-десятичного кода в семисегментный код индикатора. Входные и выходные коды преобразователей связаны между собой. Эту связь задают логическими функциями или в виде таблицы переключений. Рассмотрим наиболее распространённые в цифровой технике виды преобразователей кодов.

1. ДЕШИФРАТОР

Дешифратор (DC) или *декодер* – комбинационная схема с n входами и $m = 2^n$ выходами ($m > n$), преобразующая двоичный входной n-код (кодовое слово) в унитарный. На одном из m выходов дешифратора появляется логическая 1, а именно на том, номер которого соответствует поданному на вход двоичному коду.

На всех остальных выходах дешифратора выходные сигналы равны нулю. Дешифратор используют, когда нужно обращаться к различным цифровым устройствам по адресу, представленному двоичным кодом.

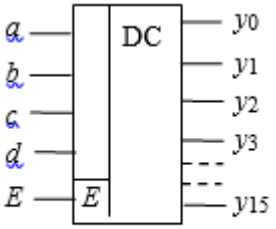


Рис. 1

Условное изображение дешифратора 4x16 (читаемого "четыре в шестнадцать") на схемах дано на рис.1. Дешифратор содержит число выходов, равное числу комбинаций входных переменных: от $y_0 = \bar{a}\bar{b}\bar{c}\bar{d}$ до $y_{15} = abcd$ при $n = 4$ и $m = 2^n = 16$.

Применяются также неполные дешифраторы с меньшим числом выходов (10 или 12 при четырех переменных на входе, тогда ряд комбинаций на входе не используется).

Каждый выход полного дешифратора реализует конъюнкцию входных переменных (код адреса) или их инверсий: при

наборе $\bar{a}\bar{b}\bar{c}\bar{d}$ (0000) y_0 = 1, при $\bar{a}bcd$ (0111) y_7 = 1, при *abcd* (1111) y_{15} = 1 и т. д.

Дешифраторы часто имеют *разрешающий* (управляющий, стробирующий) вход *E*. При *E* = 1 дешифратор функционирует как обычно, при *E* = 0 на всех выходах устанавливается 0 независимо от поступающего кода адреса. Дешифраторы широко используют во многих устройствах, в том числе в качестве преобразователей двоичного кода в десятичный.

2. ШИФРАТОР

Шифратор (CD) или *кодер* выполняет функцию, обратную функции дешифратора. Условное изображение шифратора 16x4 (16 в 4) на схемах показано на рис. 2, *а*. Классический шифратор имеет *n* входов и *m* выходов ($m < n$), и при подаче сигнала 1 на один из входов (и не более) на выходе кодера появляется двоичный код номера возбужденного выхода. Число входов и выходов такого шифратора связано соотношением $n = 2^m$.

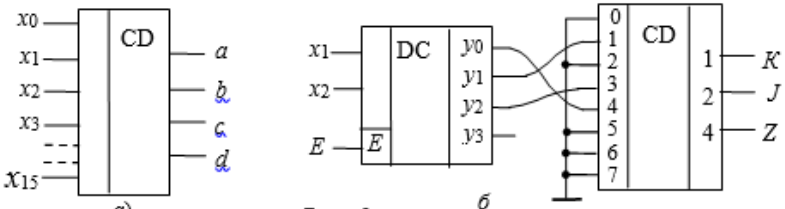


Рис. 2

Области использования шифраторов – отображение в виде двоичного кода номера нажатой кнопки или положения многопозиционного переключателя, а также номера устройства, подавшего сигнал на обслуживание в микропроцессорных системах. Шифраторы входят в состав микросхем контроллеров прерываний, например КР580ВН59.

Для решения многих конкретных задач необходимо синтезировать преобразователи различных кодов. В качестве примера на рис. 2, *б* представлена схема кодового преобразователя, состоящая из пары декодер DC – кодер CD, реализующая логику работы ($y = a + b\bar{c} + \bar{b}c$) некоторого трёхцветного светофора *K*, *J* и *Z*, управляемого двухразрядным двоичным кодом *X*. При этом вначале дешифруется каждая комбинация исходного кода, в результате чего на соответствующем выходе декодера появляется логическая 1. Затем этот логический сигнал, значение которого определено номером выхода декодера, подаётся на кодер и на его выходах устанавливается преобразованный код.

Число входов дешифратора DC равно двум (x_1 и x_2), число выходов – трём (числу выходов преобразователя) y_0, y_1 и y_2. Соединения дешифратора и шифратора выполнены в соответствии с заданной логической функцией y. Часть выходов декодера и входов кодера не используется.

Эффективно стыкуются друг с другом декодер и кодер, построенные на элементах И-НЕ: первый имеет инверсные выходы, а второй – инверсные входы. Если некоторым входным комбинациям соответствует одна и та же выходная, то соответствующие выходы декодера объединяют на элементе ИЛИ и выход последнего подают на нужный вход кодера.

Проектирование кодовой преобразовательной схемы на паре декодер-кодер оказывается в среднем более выгодным и по числу корпусов, и по быстродействию, чем при проектировании из готовых базовых логических микросхем И-НЕ и ИЛИ-НЕ. Однако потребляемая мощность в этом случае может оказаться больше, чем у схемы из отдельных элементов. Затраты времени инженера на логическое проектирование по схеме декодер-кодер неизмеримо меньше, чем затраты на проектирование преобразователя из россыпи.

3. МУЛЬТИПЛЕКСОР

Мультиплексор (MS) – это функциональный узел, осуществляющий подключение (*коммутацию*) одного из нескольких входов к выходу y. На выход такого устройства передаётся логический уровень того информационного разряда, номер которого в двоичном коде задан на адресных входах x_1 и x_2. Условное изображение мультиплексора на четыре входа и возможный вариант его структурной схемы показаны на рис. 3, *а* и *б*.

При $x_1 = 0$ и $x_2 = 0$, $y = a$; при $x_1 = 0$ и $x_2 = 1$, $y = b$; при $x_1 = 1$ и $x_2 = 0$, $y = c$ и при $x_1 = 1$ и $x_2 = 1$, $y = d$.

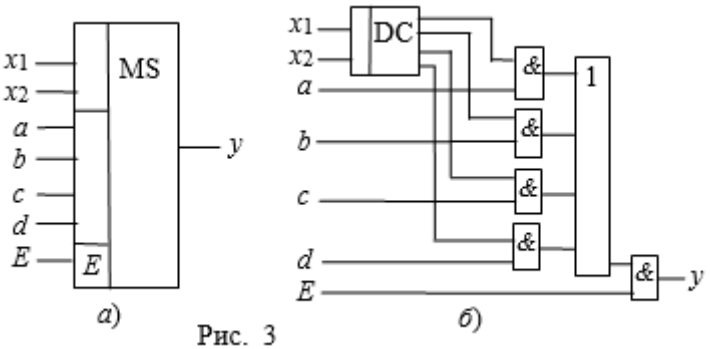


Рис. 3

Функционирование муль- типлексора описывается выра-

жением

$$y = a\bar{x}_1\bar{x}_2 + b\bar{x}_1 x_2 + cx_1\bar{x}_2 + dx_1 x_2.$$

Вход *E* – разрешающий: при *E* = 1 мультиплексор работает как обычно, при *E* = 0 выход узла находится в неактивном состоянии, мультиплексор заперт. Серийные узлы выпускаются с числом адресных входов *n* = 2, 3 и 4 при возможном числе 2^n коммутируемых входов. При необходимости коммутировать большее количество входов используют несколько мультиплексоров. Мультиплексоры находят широкое применение в устройствах отображения информации в различных устройствах управления.

Так как мультиплексор может пропустить на выход сигнал с любого информационного входа, адрес которого установлен на соответствующих адресных входах, то на основе мультиплексоров реализуют логические функции, подавая на информационные входы логические 1 или 0 в соответствии с таблицей переключений, а на адресные входы – аргументы функции.

4. ДЕМУЛЬТИПЛЕКСОР

Демультиплексор (DMS) выполняет функцию, обратную функции мультиплексора, т. е. производит коммутацию одного входного сигнала на 2^n выходов, где *n* – число адресных входов x_i. Он осуществляет преобразование информации из последовательной формы (последовательно-параллельной) в параллельную. Демультиплексор имеет один информационный вход *D* и несколько выходов, причем вход подключается к выходу y_i, имеющему заданный адрес.

В качестве примера на рис. 4, *а* дано условное графическое обозначение демультиплексора, имеющего четыре выхода, закон функционирования которого задан (табл. 1). Пользуясь табл. 1, запишем переключательные функции для выхода устройства:

$$y_0 = D\bar{x}_1\bar{x}_2;\ y_1 = D\bar{x}_1 x_2;\ y_2 = Dx_1\bar{x}_2;\ y_3 = Dx_1 x_2.$$

Функциональная схема демультиплексора, реализующая эти выражения, приведена на рис. 4, *б*.

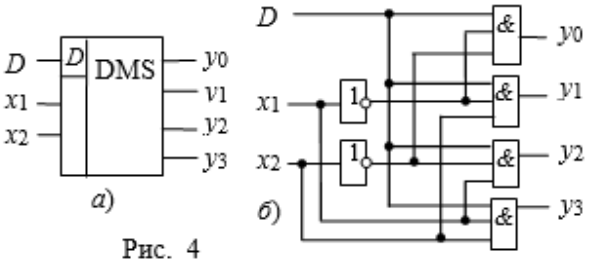


Рис. 4

Таблица 1

D	x1	x2	y3	y2	y1	y0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Если общее число выходов разрабатываемого устройства превышает имеющиеся в выпускаемых интегральных микросхемах, то используют параллельное подключение нескольких схем. На рис. 5, *а* показано демультиплексорное дерево, построенное на мультиплексорах с четырьмя выходами. Объединяя мультиплексор с демультиплексором, получают комбинационное устройство, в котором по заданным адресам один из входов подключается к одному из его выходов (рис. 5, *б*).

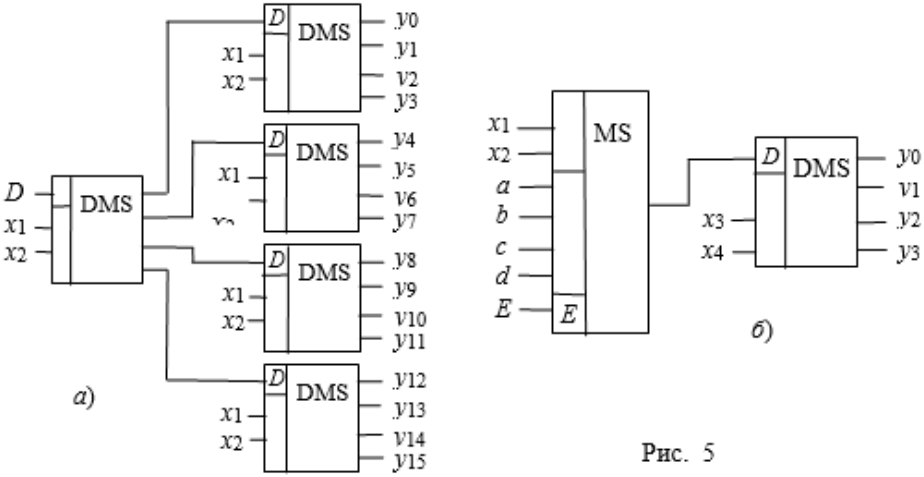


Рис. 5

УЧЕБНЫЕ ЗАДАНИЯ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ИХ ВЫПОЛНЕНИЮ

Задание 1. **Запустить** лабораторный комплекс **Multisim, собрать** на рабочем поле среды схему для испытания *дешифратора* **DC** (рис. 6) и **установить** в диалоговых окнах компонентов их параметры или режимы работы. **Скопировать** схему (рис. 6) на страницу отчёта.

Схема (рис. 6) содержит:

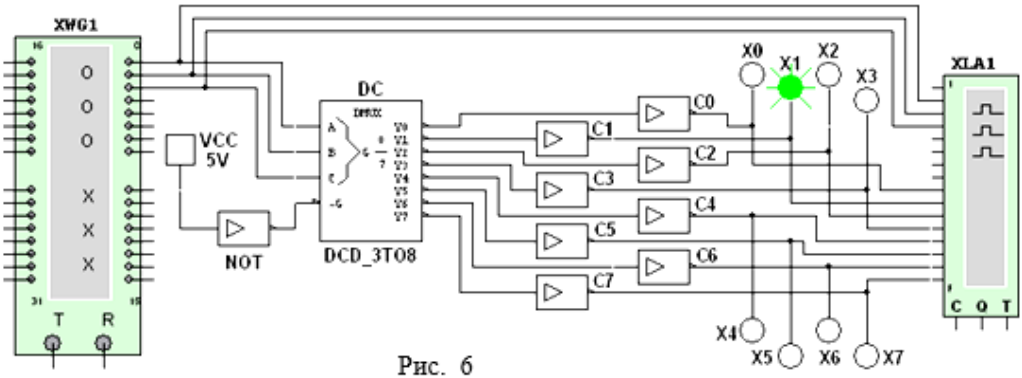


Рис. 6

• интегральный *дешифратор* **DC** (*decoder*) 3x8, имеющий 3 информационных входа **A**, **B** и **C** (для кода 4–2–1), 8 выходов (**Y0**, …, **Y7**) и преобразующий позиционный 3-разрядный двоичный код в *унитарный* "1 из 8": в выходной 8-разрядной кодовой комбинации только одна позиция занята единицей, а все остальные – нулевые (см. рис. 7, справа). В зависимости от входного двоичного кода, например 001, на выходе **DC** появляется сигнал 1 только на одной (второй, см, рис. 6) из 8-ми выходных линий, к которым подключены пробники **X0**, …, **X7**.

Данный тип шифратора относится к шифраторам с разным уровнем входных и выходных сигналов: активные входные уровни соответствуют уровню логической 1, а активные выходные сигналы – уровню логического 0. Для получения активных выходных уровней, равных 1, к выходам дешифратора подключено восемь инверторов **C0**, …, **C7**;

• логический генератор слова **XWG1** (f_T = 500 кГц) с записанными логическими словами в его ячейки памяти, которые эквивалентны десятичным числам от 0 до 7 (см. рис. 7, слева);

• логический анализатор **XLA1**, на экран которого выводятся временные диаграммы как трёх входных (**A**, **B, C**), так и восьми **(Y0**, **Y1**, …, **Y7**) выходных сигналов при пошаговом режиме **Step** генератора **XWG1**;

• источник **VCC**, напряжение 5 В с выхода которого подано на инвертор **NOT**. Логический 0 с инвертора подается на управляющий вход дешифратора **DC**: при $\overline{G} = 0$ дешифратор находится в активном состоянии.

Запустить программу моделирования дешифратора. Щёлкая мышью на кнопке **Step** генератора **XWG1**, последовательно **подавать** на вход дешифра- тора логические слова. **Убе-**

диться, что при подаче на вход дешифратора каждой новой двоичной кодовой комбинации засвечивается только один пробник, который "распознаёт" свой входной код.

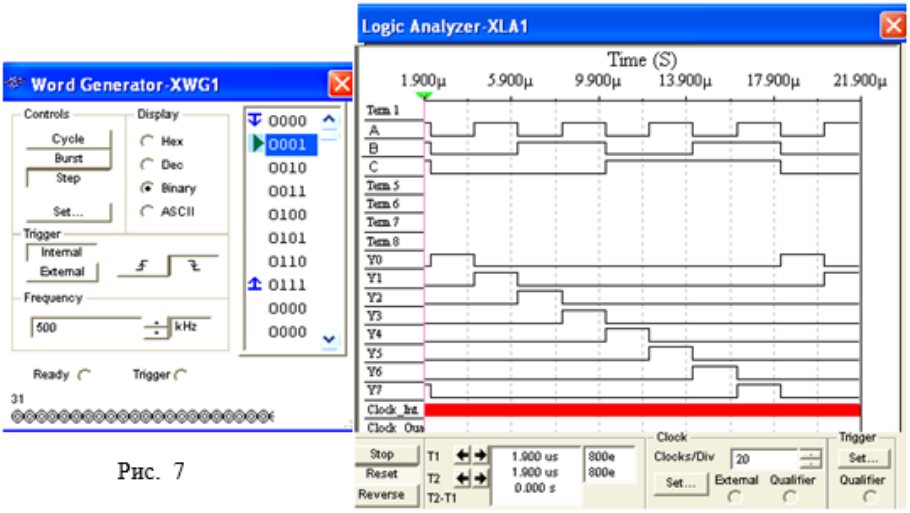


Рис. 7

Скопировать временные диаграммы входных и выходных сигналов дешифратора на страницу отчёта. По результатам моделирования **составить** и **заполнить** таблицу переключений (функций $Y_i = (A_i B_i C_i ; G_i))$ на выходах дешифратора **DC** 3x8.

Задание 2. **Собрать** на рабочем поле среды **Multisim** схему для испытания *шифратора* **CD** (рис. 8) и **установить** в диалоговых окнах компонентов их параметры или режимы работы.

Скопировать схему (рис. 8) на страницу отчёта.

Интегральный *шифратор* **CD** 8x3 (из 8 в 3) имеет 8 входов **D0**, **D1**, …, **D7**, подключенных к выходам **Y0**, **Y1**, …, **Y7** дешифратора **DC**, и три инверсных выхода **A0**, **A1**, **A2**, к которым через инверторы **C0**, **C1**, **C2** подключены логические пробники **X0**, **X1**, **X2** и семисегментный индикатор **Ind**. Содержимое ячеек памяти генератора слова **XWG1**: 000, 001, …, 111 (см. рис. 7, слева).

Запустить программу моделирования шифратора. Щёлкая мышью на кнопке **Step** генератора **XWG1**, последовательно **подавать** на вход дешифратора логические слова. **Убедиться**, что при подаче с выхода **DC** на вход шифратора **CD** 8-разрядной последовательности, в которой только одна позиция занята единицей, а остальные – нулями, на выходе шифратора формируются

3-разрядные двоичные коды **A0A1A2**, где **A0 = A**, **A1 = B** и **A2 = C**, соответствующие двоичным кодовым комбинациям на входе дешифратора **DC**.

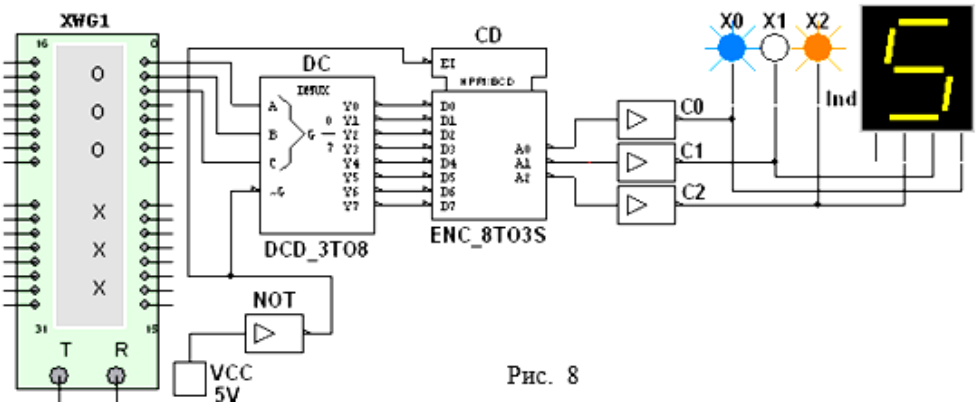


Рис. 8

По результатам моделирования (по засвечиванию логических пробников **X0**, **X1**, **X2** и показаниям индикатора **Ind**) **составить** и **заполнить** таблицу переключений на выходе шифратора **CD** 8x3.

Преобразовать схему дешифратора **DC** 3x8 и шифратора **CD** 8x3 (см. рис. 8) в схему **DC** 2x4 и шифратора **CD** 4x2, отсоединив провод **C**, подходящий к дешифратору, и провод **A2** с выхода шифратора, и **составить** таблицы переключений дешифратора 2x4 и шифратора 4x2.

Задание 3. **Собрать** на рабочем поле среды **Multisim** схему для испытания *демультиплексора* **DMS** (рис. 9) и **установить** в диалоговых окнах компонентов их параметры или режимы работы.

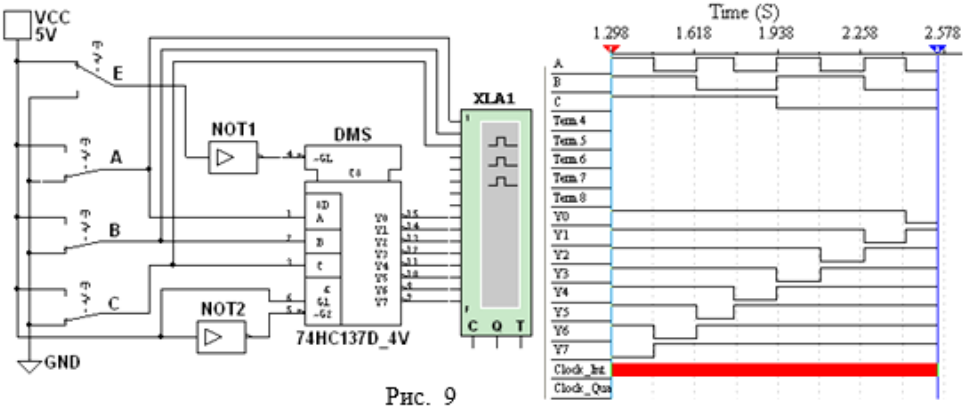


Рис. 9

Демультиплексор **DMS** 1x8 (из 1 в 8) (рис. 9) имеет один информационный вход (с активными высоким **G1** и низким **G2** уровнями), три адресных **A**, **B**, **C** входа, разрешающий **GL** вход с активным низким уровнем и восемь **Y0**, **Y1**, …, **Y7** инверсных выходов, соединённых с входами логического анализатора **XLA1**. На вход анализатора также подаются сигналы с адресных входов **A**, **B**, **C**. С помощью ключей **A**, **B** и **C** можно сформировать восемь трёхразрядных двоичных адресных слов. При последовательной подаче формируемых ключами адресных слов от 111 до 000 на экран анализатора **XLA1** при моделировании выводятся 8-разрядные кодовые последовательности с одним активным (низким) уровнем.

Для обеспечения медленного перемещения лучей на экране анализатора **XLA1 установить** частоту его таймера f_a = 500 Гц и число импульсов, приходящихся на одно деление, **Clocs/div** = 80.

Задать код ключей 111 и **щелкнуть** мышью на кнопке **Run/Stop**. Кривые адресных и выходных логических сигналов медленно разворачиваются во времени на экране анализатора.

Остановить (щелчком мыши на кнопке **Stop)** процесс моделирования при приближении лучей анализатора к линии разметки экрана.

Повторять перечисленные выше операции для спадающих счётных комбинаций адресных сигналов (с 110 до 000) до тех пор, пока не будет записан процесс моделирования при адресном слове 000 (см. рис. 9, справа).

Убедиться, что для каждой комбинации адресных сигналов демультиплексор формирует логический 0 на одном из восьми выходов, номер которого соответствует определенному кодовому слову на входе, т. е. демультиплексор подобен коммутатору, посредством которого поток цифровой информации разделяется на 8 выходных потоков.

Скопировать схему (рис. 9) и временные диаграммы входных и выходных сигналов на страницу отчёта.

Если адресные входы **A**, **B** и **C** принять в качестве информационных входов, а вход **G1** (**G2**) в качестве входа разрешения работы, то мультиплексор превратится в дешифратор.

Задание 4 (выполняется факультативно). **Собрать** на рабочем поле среды MS10 схему для испытания *демультиплексора* **DMS** 1x16 (из 1 в 16) (рис. 10) и **установить** в диалоговых окнах компонентов их параметры или режимы работы. **Скопировать** схему (рис. 10) в отчёт.

С целью автоматизации процесса моделирования к входу демультиплексора **DMS** подключен логический генератор **XWG1** с записанными в его ячейки памяти адресными кодами от 0000 до 1111, а для визуализации сигналов на выходах включены 16 логических пробников **X1**, **X2**, …, **X16** и логический анализатор **XLA2**.

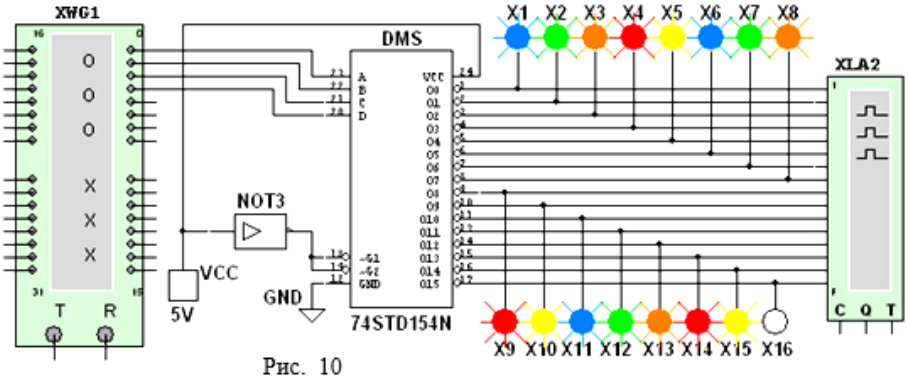


Рис. 10

Запустить программу моделирования демультиплексора **DMS** 1x16. Последовательно **подавать (**щелкая мышью на кнопке **Step** генератора **XWG1)** на вход демультиплексора логические слова, начиная с комбинации 0000 адресного сигнала и заканчивая комбинацией 1111, и **наблюдать** за изменениями выходных сигналов по показаниям индикаторов и в окне анализатора **XLA2**.

В исследуемой модели демультиплексора соответствующий активный выход имеет низкий логический уровень (рис. 30 11), поэтому пробник на этом выходе не светится. Так, при подаче последней кодовой комбинации 1111 на вход демультиплексора не светится пробник **X16**, так как активным является выход **15** (см. рис. 10).

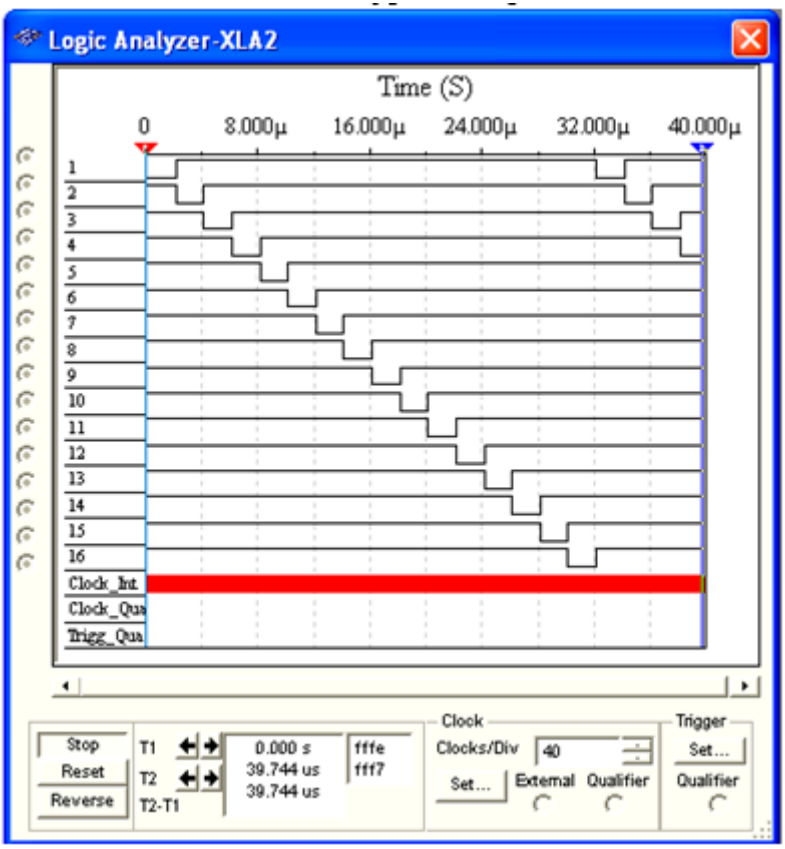
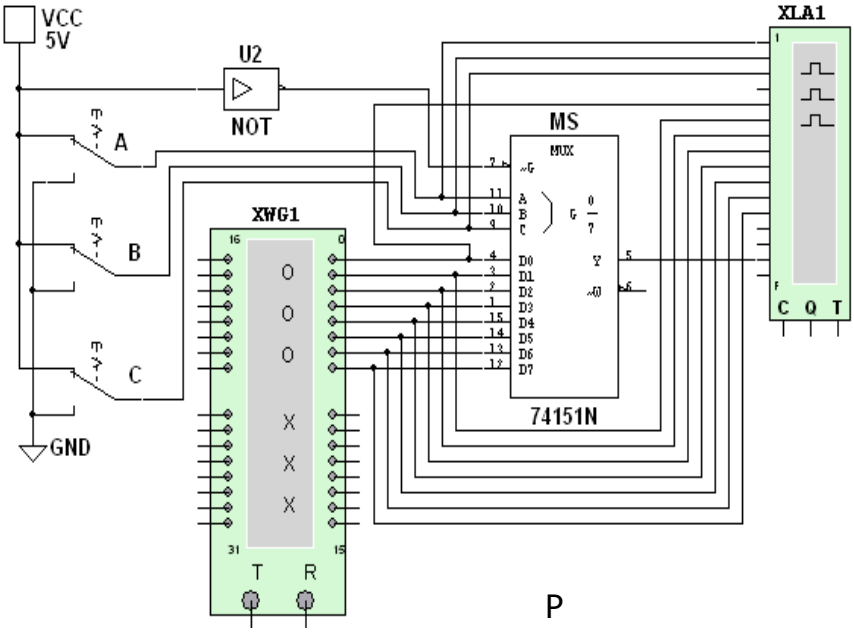


рис.11

Скопировать на страницу отчёта временные диаграммы выходных сигналов демультиплексора **DMS** 1x16.

Примечание. Демультиплексоры как таковые промышленностью не выпускаются, поскольку режим мультиплексирования может быть реализован как частный случай в других устройствах – в дешифраторах.

Задание 5. **Собрать** на рабочем поле среды MS10 схему (рис. 12) для испытания *мультиплексора* **MS** 8x1 (из 8 в 1) и **установить** в диалоговых окнах компонентов их параметры или режимы работы. **Скопировать** схему (рис. 12) на страницу от-



Р

чёта.

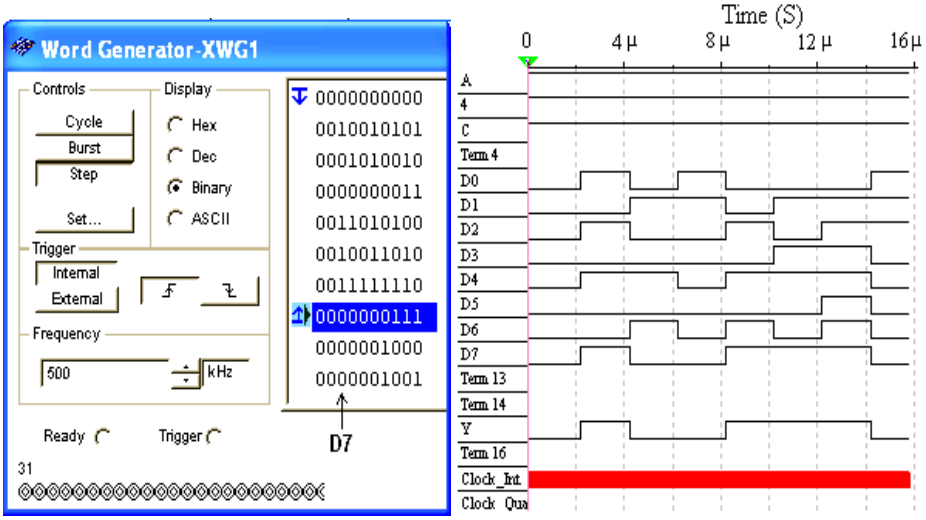
Мультиплексор **MS** с разрешающим входом **G** осуществляет передачу сигнала с каждого информационного входа **D0**, **D1**, …, **D7**, заданного 3-разрядным кодом **ABC** – адресом выбираемого входа, на единственный выход **Y**. Разрядность (3) управляющего сигнала определяет количество входов ($2^3 = 8$), с которых мультиплексор может принимать информацию. Если предположить, что к входам **D0**, **D1**, …, **D7** мультиплексора **MS** присоединено 8 источников цифровых сигналов – генераторов последовательных двоичных слов, то байты от любого из них можно передавать на выход **Y.**

Для иллюстрации работы мультиплексора **MS** запишем в ячейки памяти генератора **XWG1** произвольные 8-разрядные кодовые слова (рис. 13, слева), а с помощью ключей **A**, **B**, **C** сформируем управляющий сигнал 111. Последовательно щёлкая мышью на кнопке **Step** генератора **XWG1** и при **G** = 1, поступающие на вход **D7** мультиплексора байты (сигнал 01001110) с 8-го разряда (на рис. 13, слева 8-й разряд показан стрелкой) логических слов генератора **XWG1** пере- даются на выход **Y** и на вход

анализатора (см. рис. 13, справа).

Если ключ **A** установить в нижнее положение (сформировав, тем самым, адресный код 011), то с входа **D3** на выход **Y** мультиплексора будут поступать байты 4-го разряда логических слов, записанных в ячейки памяти генератора **XWG1**, и т. д.

Записать в первые восемь ячеек памяти генератора **XWG1** произвольные 8-разрядные кодовые слова, **задать** частоту $f_г$ = 500 кГц и режим **Step** его работы (см. рис. 13, слева).



Р

Задать частоту $f_а$ = 20 МГц таймера логического анализатора **XLA1** и количество импульсов таймера **Clock/div** = 20, приходящихся на одно деление.

Установить с помощью ключей **A**, **B** и **C** адресный код (самостоятельно или по указанию преподавателя), например 100_2 (4_{10}) и **запустить** программу моделирования мультиплексора. **Получить** и **скопировать** временные диаграммы входных сигналов **D0**, **D1**, …, **D7** и выходного сигнала **Y** мультиплексора на страницу отчёта.

Примечание. Таблицы переключений на выходах для рассмотренных библиотечных преобразователей кодов можно вызвать нажатием клавиши помощи **F1** после выделения на схеме соответствующего преобразователя.

СОДЕРЖАНИЕ ОТЧЁТА

1. Наименование и цель работы.
2. Перечень приборов, использованных в экспериментах, с

их краткими характеристиками.

3. Изображения электрических схем для испытания дешифратора, шифратора, демультиплексора и мультиплексора.

4. Копии временных диаграмм и таблицы переключений, отображающие работу исследуемых преобразователей кодов.

5. Выводы по работе.

ТЕСТОВЫЕ ЗАДАНИЯ К РАБОТЕ

1. Укажите **задачи**:

а) Для демультиплексирования данных и адресной логики в запоминающих устройствах, а также для преобразования двоично-десятичного кода в десятичный с целью управления индикаторными и печатающими устройствами;

б) Для преобразования десятичных чисел в двоичные или в двоично-десятичный код, например, в микрокалькуляторах, в которых нажатие десятичных клавишей вызывает генерацию соответствующих двоичных кодов;

в) Для хранения и преобразования многоразрядных двоичных чисел;

г) Для коммутации в заданном порядке сигналов, поступающих с нескольких входных шин на одну выходную;

д) Для распределения в требуемой последовательности по нескольким выходам сигналов с одного информационного входа, в частности, для передачи информации по одной линии от нескольких установленных на ней датчиков,

при решении которых используется:

1. Шифратор: ○ *а)* ○ *б)* ○ *в)* ○ *г)* ○ *д)*
2. Дешифратор: ○ *а)* ○ *б)* ○ *в)* ○ *г)* ○ *д)*
3. Мультиплексор: ○ *а)* ○ *б)* ○ *в)* ○ *г)* ○ *д)*
4. Демультиплексор: ○ *а)* ○ *б)* ○ *в)* ○ *г)* ○ *д)*

2. Укажите, с **какого разряда** бинарного слова генератора логического слова XWG будет передаваться информация на выход мультиплексора 8x3 при адресном коде 100 на его входе?

○ 1 ○ 3 ○ 5 ○ 7 ○ 9

3. Укажите число **выводов** дешифратора при трёх информационных входах.

○ 2 ○ 4 ○ 6 ○ 8 ○ 16

4. Укажите назначение **стробирующих** входов в преобразователях кодов.

○ Для синхронизации работы преобразователей
○ Для увеличения числа коммутируемых информационных входов, а также для блокирования работы преобразователей
○ Для увеличения числа адресных входов

5. Укажите, в каком **преобразователе** выбор входа по его номеру (адресу) осуществляется с помощью двоичного кода?

○ В шифраторе ○ В дешифраторе ○ В мультиплексоре ○ В демультиплексоре

6. Укажите **число выводов** у шифратора при четырёх информационных входах.

○ 16 ○ 8 ○ 4 ○ 2 ○ 1

7. Укажите, какой из приведенных преобразователей кодов выпускается промышленностью только в **составе других устройств**?

○ Шифратор ○ Дешифратор ○ Демультиплексор ○ Мультиплексор

ЛАБОРАТОРНАЯ РАБОТА 3

ЦИФРОАНАЛОГОВЫЙ ПРЕОБРАЗОВАТЕЛЬ

ЦЕЛЬ РАБОТЫ

Ознакомление с принципом работы и испытание интегрального цифроаналогового преобразователя.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ И РАСЧЁТНЫЕ ФОРМУЛЫ

1. СТРУКТУРА РЕЗИСТИВНЫХ МАТРИЦ ЦАП

При построении устройств, связывающих цифровое устройство с объектами, использующими информацию в непрерывно изменяющейся форме, требуется преобразование информации из аналоговой формы в цифровую и из цифровой в аналоговую. Устройство, осуществляющее автоматическое преобразование непрерывно изменяющихся во времени аналоговых значений фи-

зической величины (напряжения, тока) в эквивалентные значения числовых кодов, называют *аналого-цифровым преобразователем* (АЦП). Устройство, осуществляющее автоматическое преобразование входных значений, представленных числовыми кодами, в эквивалентные им значения какой-нибудь физической величины (напряжения, тока и др.), называют *цифроаналоговым преобразователем* (ЦАП).

Итак, цифроаналоговый преобразователь предназначен для прямого преобразования входного двоичного кода, например, $A_i(a_2a_1a_0)$ в аналоговый эквивалент. Выходная аналоговая величина, обычно напряжение $u_{вых}$, иногда нормированное $u_{вых.н} = u_{вых}/u_{вых.max}$, соответствует кодовой комбинации A_i, поступившей на вход, и воспроизводится для дискретных моментов времени (рис. 1, *а*). Сменяющиеся входные цифровые коды обуславливают сменяющееся ступенчатое напряжение на выходе (L – идеальная передаточная характеристика ЦАП).

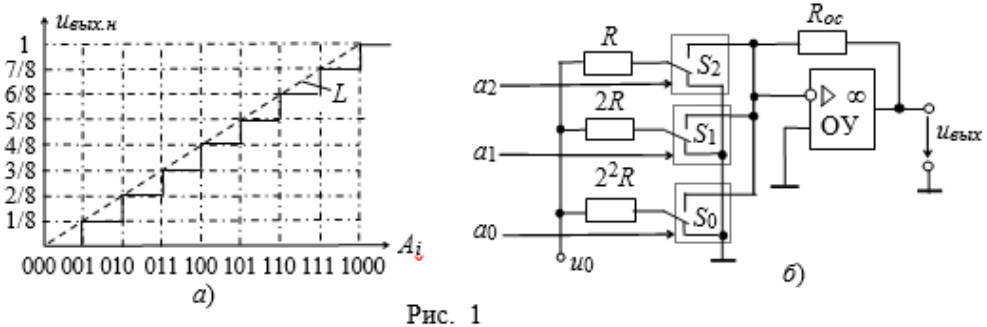


Рис. 1

Существует два широко распространенных способа цифроаналогового преобразования с использованием:

- резистивной матрицы с весовыми двоично-взвешенными сопротивлениями;
- резистивной матрицы с двумя номиналами сопротивлений, которую обычно называют матрицей R–$2R$.

ЦАП с весовыми *двоично-взвешенными сопротивлениями* (рис. 1, *б*) состоит: из n переключателей S_i (по одному на каждый разряд), управляемых двоичным кодом A_i; из матрицы двоично-взвешенных резисторов с сопротивлениями $2^{n-1}R$; источника опорного напряжения u_0 и выходного операционного усилителя ОУ, с помощью которого суммируются токи, протекающие через резисторы с двоично-взвешенными сопротивлениями, для получения аналогового выходного напряжения $u_{вых}$.

Каждый i-й разряд управляет переключателем S_i, который подключается к источнику опорного напряжения u_0, когда

a_i = 1, или к общей шине, когда a_i = 0. Сопротивления резисторов $2^{n-1}R$ (n – номер разряда входного кода), соединенных с ключами, таковы, что обеспечивают пропорциональность в них тока двоичному весу соответствующего разряда входного кода. Следовательно, ток на входе ОУ и выходное напряжение ЦАП:

$$i = \frac{a_{n-1}u_0}{R} + \frac{a_{n-2}u_0}{2R} + \ldots + \frac{a_1u_0}{2^{n-1}R} + \frac{a_0u_0}{2^nR}; \quad u_{вых} = -R_{oc}i = -u_0\frac{R_{oc}}{2^nR}\sum_{i=0}^{n-1}a_i2^i.$$

Напряжение на выходе ЦАП пропорционально "весу" присутствующего на входах кода, а максимальное значение имеет место, когда все разряды примут значение 1, т. е.

$$u_{max} = \left| u_0\frac{(2^n-1)R_{oc}}{2^nR} \right|,$$

и оно всегда меньше опорного напряжения на шаг квантования $u_oR_{oc}/(2^nR)$.

Номиналы сопротивлений резисторов в младшем и старшем разрядах отличаются в 2^{n-1} раз и должны быть выдержаны с высокой точностью. Например, для 12-разрядного ЦАП использование в старшем разряде резистора с сопротивлением 10 кОм потребует включения в младший разряд преобразователя резистора с сопротивлением порядка 20 МОм. Широкий набор номиналов резисторов и требования их высокой точности, в особенности при значительном числе разрядов n входного кода, создают трудности при реализации ЦАП посредством интегральной технологии.

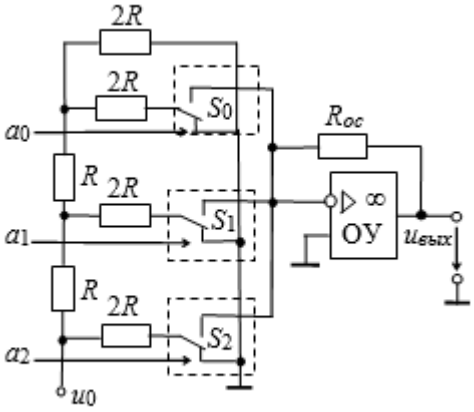


Рис. 2

Во второй схеме ЦАП с матрицей *R-2R* используют резисторы с двумя номиналами сопротивлений, причём резисторы с сопротивлением *R* включены в каждый разряд (см. рис. 2 при *n* = 3). Однако в этой схеме увеличиваются значения паразитных ёмкостей.

Принцип функционирования схемы основан на свойстве резистивного делителя *R*-2*R* сохранять постоянное сопротивление нагрузки для источника опорного напряжения при замыкании ключей. Вследствие этого на выводах резистора *R*, начиная со старшего *n* – 1 разряда, опорное напряжение последовательно делится пополам, как и входящий в каждый узел матрицы ток. При этом напряжение на выходе преобразователя с матрицей *R*–2*R*:

$$u_{вых} = -u_0 \frac{R_{ос}}{R}(a_{n-1}2^{-1} + a_{n-2}2^{-2} + \ldots + a_1 2^{-(n-1)} + a_0 2^{-n}) = -u_0 \frac{R_{ос}}{2^n R}\sum_{i=0}^{n-1} a_i 2^i.$$

Таким образом, выходное напряжение ЦАП пропорционально сумме напряжений со своими весами, обусловленными переключателями, подключенными к источнику опорного напряжения u_0.

Недостатком ЦАП с матрицей *R*–2*R* является сильное влияние на точность преобразования нестабильности сопротивлений переключателей в замкнутом состоянии, что снижает временную и температурную стабильность характеристик ЦАП. Этот недостаток в значительной степени удаётся устранить в схемах код-напряжение, выполненных на базе полупроводниковой технологии с использованием тонкоплёночных резисторов на кристалле и переключателей на КМДП-транзисторах, в которых нелинейность от ±0,8% до ±0,003% от опорного напряжения u_o, время установления тока от 5 мкс до десятых долей микросекунд и менее, часто выходной диапазон напряжения ±5 В. Опорное напряжение в схемах ЦАП может выбираться разной полярности или двуполярным.

2. ОСНОВНЫЕ ПАРАМЕТРЫ ЦАП

Основными параметрами ЦАП являются число разрядов *n* = 8, …, 24 и *абсолютная разрешающая способность* – среднее значение минимального изменения сигнала на выходе ЦАП, обусловленное увеличением или уменьшением его кода на единицу. Теоретически ЦАП, преобразующий *n*-разрядные двоичные коды, должен обеспечить 2^n различных значений выходного сигнала с разрешающей способностью $1/(2^n - 1)$. При числе разрядов *n* = 8

количество независимых квантов (ступеней) выходного напряжения ЦАП равно $2^8 - 1 = 255$, при $n = 12$, $2^{12} - 1 = 4095$ и т. д.

Абсолютное значение минимального кванта напряжения определяется как предельным принимаемым числом $2^n - 1$, так и максимальным выходным напряжением ЦАП, по-другому называемым напряжением шкалы или опорным напряжением u_o. Значение абсолютной разрешающей способности ЦАП, часто обозначаемое ЗМР (значение младшего разряда), при $n = 8$ и опорном напряжении $u_o = 5$ В

ЗМР = $u_o/(2^8 - 1) = 5/255 \approx 0{,}0196$ В = 19,6 мВ.

Отличие реального значения разрешающей способности от теоретического обусловлено погрешностями и шумами входящих в ЦАП узлов. Точность ЦАП определяется значением абсолютной погрешности δ_a и нелинейностью преобразователя $\delta_н$. *Абсолютная погрешность* δ_a характеризуется отклонением максимального значения выходного напряжения u_{max} от расчётного, соответствующего конечной точке характеристики идеального преобразователя, и измеряется обычно в единицах ЗМР.

Нелинейность преобразователя $\delta_н$ характеризует отклонение действительной характеристики от линейной (от прямой линии *L*, см. рис. 1, *а*), проведенной через центры ступенек или через нуль и точку максимального значения выходного сигнала.

Из динамических параметров наиболее важным является максимальная частота преобразования f_{max} (десятки и сотни килогерц) – наибольшая частота дискретизации, при которой параметры ЦАП соответствуют заданным значениям.

Работа ЦАП часто сопровождается специфическими переходными импульсами в выходном сигнале, возникающими из-за разности времени открывания и закрывания аналоговых переключателей в ЦАП. Особенно значительно выбросы проявляются, когда входной код 01…111 сменяется кодом 10…000, а переключатель старшего разряда ЦАП открывается позже, чем закрываются переключатели младших разрядов. Вследствие определённой идеализации при моделировании библиотечных ЦАП среды MS10 не всегда удаётся определить отмеченные выше параметры.

Библиотечные интегральные схемы ЦАП среды MULTISIM требуют для своей работы подключения только постоянного эталонного напряжения, заземления и входных сигналов.

УЧЕБНЫЕ ЗАДАНИЯ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ИХ ВЫПОЛНЕНИЮ

Задание 1. **Запустить** среду **Multisim, собрать** на рабочем поле среды схему для испытания интегрального *цифроанало-*

гового преобразователя (рис. 3, *а*) и **установить** в диалоговых окнах компонентов их параметры или режимы работы.

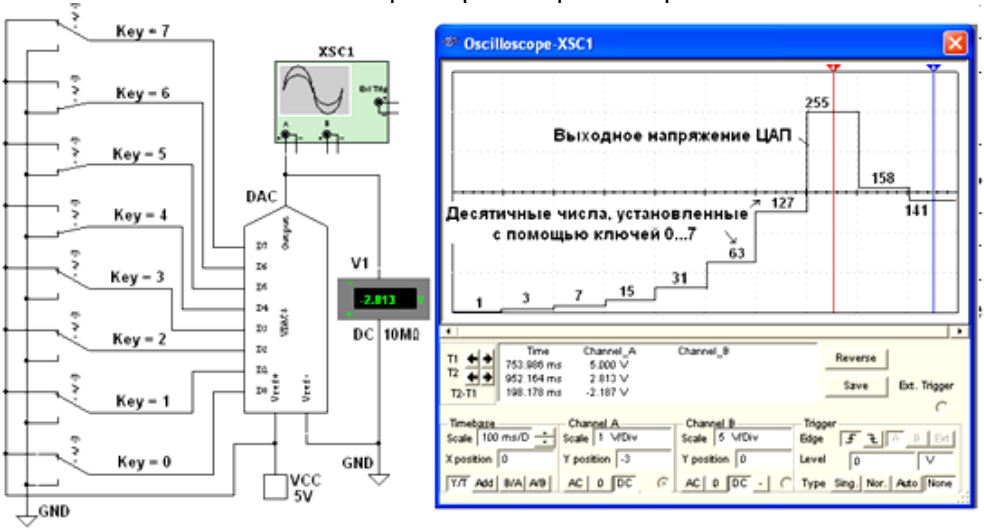


Рис. 3

Скопировать схему (рис. 3, *а*) на страницу отчёта.

В схеме (рис. 3, *а*) использован библиотечный (**Mixed**) 8-разрядный цифроаналоговый преобразователь **DAC**, на входы которого подаются сформированные с помощью переключателей **0**, …, **7** двоичные коды от 00000000 до 11111111_2 (FF_{16} или 255_{10}). Выходное напряжение ЦАП можно измерить с помощью вольтметра **V1** или осциллографа **XSC1**, воспользовавшись визирными линиями, расположенными на его экране.

Задание 2. **Получить** на экране осциллографа **XSC1** ступенчатое выходное напряжение ЦАП (рис. 3, *б*). Для этого нужно вначале **замкнуть** переключатель **0**, т. е. **подать** напряжение 5 В на вход **D0** ЦАП, и **запустить** программу моделирования. На выходе ЦАП формируется напряжение, равное ЗМР. Затем во время остановок моделирования **замыкать** поочерёдно переключатели **1**, **2**, …, **7**, подавая входные десятичные комбинации 3, 7, 15, 31, 63, 127, 255 на входы **D0**, …, **D7** ЦАП (рис. 3, *б*).

Повторить эксперимент, подавая на входы ЦАП сформированные с помощью переключателей шестнадцатеричные коды от 0 до FF (255_{10}) через шаг 10_{16} (16_{10}) и занося в табл. 1 показания вольтметра **V1** (значения выходного напряжения $u_{Вых}$ ЦАП) при напряжении источника **VCC** u_o = 5 В. **Найти** частичные и усреднённое значение ступени, частичные и усреднённое значение МЗР. **Построить** график $u_{Вых}(N)$, выбрав соответствующие мас-

штабы для напряжений и входных десятичных чисел *N*, откладываемых по осям координат.

Таблица 1

№ п/п	Входной десятичный код *N*	Выходное напряжение, $u_{вых}$, В	Напряжение ступени $u_{вых2} - u_{вых1}$, В	Значение младшего разряда МЗР = $(u_{вых2} - u_{вых1})/16$, В
1	0	0	0	–
2	15			
3	31			
4	47			
5	63			
6	79			
7	95			
8	111			
9	127			
10	143			
11	159			
12	175			
13	191			
14	207			
15	223			
16	239			
17	255			

Задание 3. **Собрать** на рабочем поле среды **Multisim** схему для испытания *цифроаналогового преобразователя* (рис. 4, *а*) и **установить** в диалоговых окнах компонентов их параметры или режимы работы. **Скопировать** схему (рис. 4, *а*) на страницу отчёта.

Провести моделирование ЦАП, запрограммировав генератор **XWG1** (частота генерации сигналов $f_г$ = 1 кГц) на возрастание и убывание шестнадцатеричных чисел от 0 до FF (255_{10}) при шаге 10_{16} (16_{10}).

Составить таблицу и **занести** в неё выходные напряжения ЦАП и величину ступеней, которые выводятся в нижнем окне осциллографа **XSC2**.

Измерение напряжений **проводить** с помощью визирных линий осциллографа, устанавливая их на двух соседних ступенях (см. рис. 4, *б*) при различных кодовых комбинациях на выходе генератора **XWG1** и напряжении u_0 = 5 В источника **VCC**.

Так, при входных десятичных числах 175 и 191 и напряжении u_0 = 5 В выходные напряжения ЦАП соответственно равны 3,437 В и 3,750 В, а напряжение ступени – 312,5 мВ. При этом ЗМР = 312,5/16 = 19,53 В. **Найти** и **сравнить** усреднённое значение ЗМР с расчётным значением.

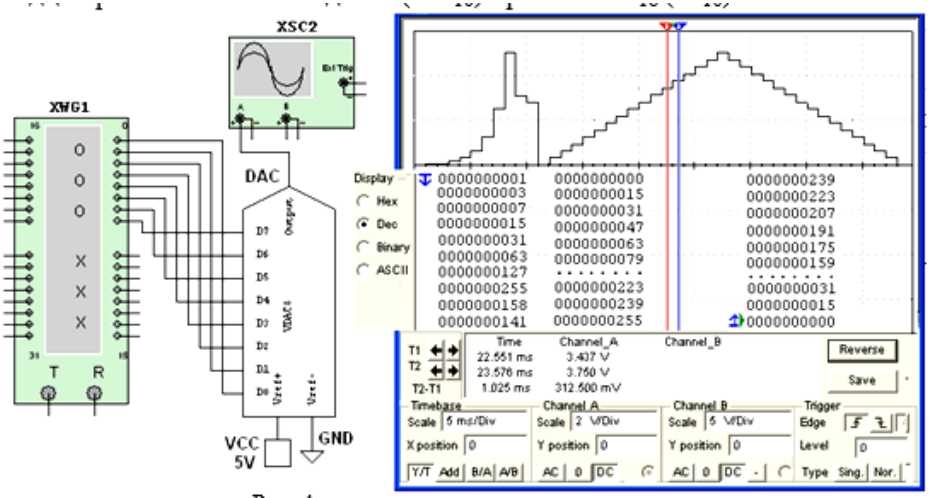


Рис. 4

Установить напряжение u_o = 10 В источника **VCC** и **повторить** моделирование ЦАП при опорном напряжении 10 В. **Построить** графики $u_{вых}(N)$ при u_o = 5 В и u_o = 10 В на одном рисунке, выбрав соответствующие масштабы для напряжений и входных десятичных чисел N, откладываемых по осям координат.

СОДЕРЖАНИЕ ОТЧЁТА

1. Наименование и цель работы.
2. Перечень приборов, использованных в экспериментах, с их краткими характеристиками.
3. Изображения электрических схем для испытания цифро-аналогового преобразователя.
4. Копии графиков выходного напряжения исследуемого ЦАП, отображающих его работу.
5. Графики $u_{вых}(N)$ при различных значениях опорного напряжения.
6. Выводы по работе.

ТЕСТОВЫЕ ЗАДАНИЯ К РАБОТЕ

1. Укажите **назначение** ЦАП.

○ Для преобразования информации в аналоговой форме в цифровые коды

○ Для преобразования цифрового кода N в пропорциональное аналоговое значение напряжения $u(N)$

○ Для деления числа или частоты повторения импульсов на заданный коэффициент K

○ Для преобразования информации из последовательной во времени формы представления в параллельную форму

2. Укажите, какая **структура резистивных матриц** ЦАП имеет преимущество при изготовлении преобразователя посредством интегральной технологии?

○ Матрица с весовыми резисторами

○ При изготовлении ЦАП с помощью интегральной технологии структура матриц не играет существенного значения, так как высокая точность и быстродействие систем код-напряжение зависят от типа переключателей (ключей) во входной разрядной цепи

○ Матрица *R*-2*R*

3. Определите понятие "**абсолютная разрешающая способность"** ЦАП.

○ Это возможное количество уровней аналогового сигнала, делённое на количество двоичных разрядов входного кода

○ Это наибольшее значение отклонения аналогового сигнала от расчётного.

○ Это максимальное отклонение ступенчато нарастающего выходного сигнала от прямой линии, соединяющей точки нуля и максимального выходного сигнала

○ Это среднее значение минимального изменения сигнала на выходе ЦАП, обусловленное увеличением или уменьшением его кода на единицу

4. Укажите, для чего выбирают опорное напряжение **двуполярным**?

○ Чтобы преобразовать двоичные коды в ток

○ Для обеспечения работы ЦАП, содержащего резистивную матрицу с весовыми резисторами, диодные ключи и систему управления ключами

○ Для увеличения диапазона $\pm u_{вых}$ выходного напряжения

○ Чтобы получать на выходе двуполярное напряжение $\pm u_{вых}$ при различных входных кодах

○ Чтобы максимальное выходное напряжение ЦАП не было меньше опорного напряжения u_o на величину ЗМР (ЗМР – значение младшего разряда)

5. Укажите **перспективы развития** ЦАП.

□ Повышение быстродействия ключей и уменьшение времени установки ОУ

□ Построение ЦАП без резистивной матрицы

- Применение стабилизированных источников опорного напряжения
- Уменьшение разрядности преобразователя код-напряжение (до 4…6)
- Улучшение качества резистивных матриц

ЛАБОРАТОРНАЯ РАБОТА 4

АНАЛОГО-ЦИФРОВОЙ ПРЕОБРАЗОВАТЕЛЬ

ЦЕЛЬ РАБОТЫ

Ознакомление с принципом работы и испытание интегрального 8-разрядного аналого-цифрового преобразователя.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ И РАСЧЁТНЫЕ ФОРМУЛЫ

1. СТРУКТУРНАЯ СХЕМА АЦП ПОСЛЕДОВАТЕЛЬНОГО ДЕЙСТВИЯ

Аналого-цифровой преобразователь (АЦП) – устройство, предназначенное для преобразования аналоговых величин в их цифровой эквивалент в различных системах исчисления. Входным сигналом АЦП в течение некоторого промежутка времени Δt является постоянное напряжение, равное отсчёту $u_{вх}(k\Delta t)$ входной аналоговой функции $u_{вх}$. За это время на выходе АЦП формируется цифровой (обычно двоичный) код

$$A_i(a_{n-1}a_{n-2}\ldots a_1 a_0),$$

соответствующий дискретному отсчёту напряжения $u_{вх}(k\Delta t)$. Количественная связь для любого момента времени определяется соотношением

$$A_i = u_{вх}(k\Delta t)/\Delta u \pm \delta_i,$$

где Δu – шаг квантования входного аналогового напряжения $u_{вх}$; δ_i – погрешность преобразования напряжения $u_{вх}(k\Delta t)$ на данном шаге.

Физический процесс аналого-цифрового преобразования состоит из дискретизации по времени аналогового сигнала, квантования по уровню и кодирования [8]. Процесс *дискретизации* аналогового сигнала длительностью $t_{вх}$ выполняется в соответствии с теоремой Котельникова, определяющей необходимый шаг дискретизации $\Delta t \le 1/(2f_m)$, где f_m – максимальная частота спектра входного сигнала, и число шагов $M = t_{вх}/\Delta t$.

Процесс *квантования по уровню* дискретизированной функции $u_{вх}(k\Delta t)$ заключается в отображении бесконечного множества её значений на некоторое множество конечных значений $u_д(k)$,

равное числу уровней квантования $N = u_{вх.max}/\Delta u$. Процесс квантования по уровню (округление каждого значения $u_{вх}(k\Delta t)$ до ближайшего уровня $u_Д(k)$) приводит к возникновению ошибки (шума) квантования, максимальное значение которой $\pm 1/2\Delta u$ определяется разрядностью используемого выходного кода. При увеличении разрядности выходного кода ошибка квантования может быть уменьшена до сколь угодно малой величины, но не может быть сведена к нулю выбором параметров устройства, так как она присуща данному алгоритму.

Процесс *кодирования* заключается в замене найденных квантованных $N + 1$ значений входного сигнала $u_Д(k)$ некоторыми цифровыми кодами.

На рис. 1, *а* приведена характеристика идеального АЦП в нормированных единицах входного напряжения $u_{вх.н} = u_{вх}/u_{вх.max}$. Кроме ошибки квантования, при оценке точности АЦП учитывают дополнительные погрешности: *инструментальную* (погрешность смещения нуля, вызывающей смещение пунктирной прямой L влево или вправо от начала координат, см. рис. 1, *а*) и *апертурную*, возникающую из-за несоответствия значения входного сигнала $u_Д(k)$ преобразованному цифровому коду A_i. Несоответствие возникает, если входной сигнал в течение интервала дискретизации Δt изменяется более чем на значение шага квантования Δu.

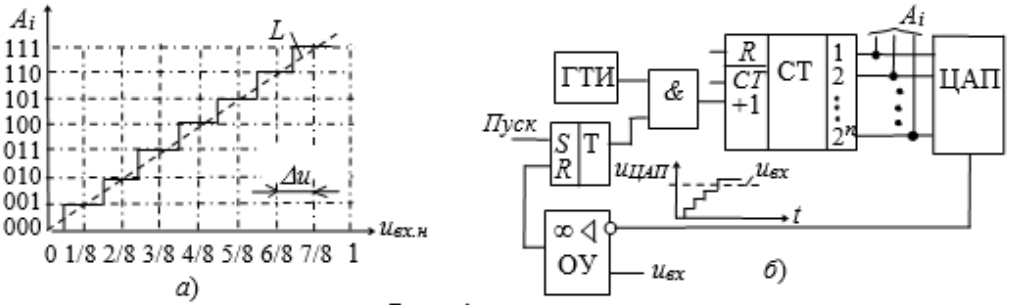


Рис. 1

2. ОСНОВНЫЕ ПАРАМЕТРЫ АЦП

К основным параметрам АЦП относят:

– число разрядов выходного кода $n = 8, \ldots, 16$, отображающего исходную аналоговую величину, которое может формироваться на выходе АЦП. При использовании двоичного кода $n = \log_2(N + 1)$, где $N + 1$ – максимальное число кодовых комбинаций (уровней квантования) на выходе АЦП;

– диапазон изменения входного напряжения $u_{вх.max}$. Отметим, что АЦП может обрабатывать входную информацию в виде

однополярного аналогового напряжения с пределами 0…$u_{вх.max}$ и двуполярного $\pm u_{вх.max}/2$;

– абсолютная разрешающая способность ЗМР = Δu (значение младшего разряда) – среднее значение минимального изменения входного сигнала $u_{вх}$, обуславливающего увеличение или уменьшение выходного кода на единицу. Значение ЗМР определяется разрядностью выходного кода и диапазоном входного напряжения;

– абсолютная погрешность δ_i преобразования в конечной точке шкалы есть отклонение реального максимального значения входного сигнала $u_{вх.max}$ от максимального значения идеальной характеристики L АЦП (см. рис. 1, *а*). Обычно δ_I измеряется в ЗМР;

– максимальная частота преобразования (десятки и сотни килогерц);

– время преобразования входного сигнала: $t_{пр.max} \le (1/2)\Delta t$.

Состав АЦП в отличие от ЦАП может изменяться в значительной степени в зависимости от выбранного метода преобразования и способа его реализации. Наибольшее распространение получили три основных метода: последовательного счёта, поразрядного кодирования и считывания.

Метод последовательного счёта основан на уравновешивании входной величины суммой одинаковых по величине эталонов (суммой шагов квантования). Момент уравновешивания определяется с помощью одного компаратора, а количество эталонов, уравновешивающих входную величину, подсчитывается с помощью счётчика.

Метод поразрядного кодирования (уравновешивания) предусматривает наличие нескольких эталонов (часто реализованных в виде уравновешивающего сдвигающего регистра), обычно пропорциональных по величине степеням числа 2, и сравнение этих эталонов с аналоговой величиной. Сравнение начинается с эталона старшего разряда. В зависимости от результата этого сравнения формируется значение старшего разряда выходного кода. Если эталон больше входной величины, то в старшем разряде ставится 0 и далее производится уравновешивание входной величины следующим по значению эталоном. Если эталон равен или меньше входной величины, то в старшем разряде выходного кода ставится 1 и в дальнейшем производится уравновешивание разности между входной величиной и первым эталоном.

Наибольшим быстродействием обладают преобразователи,

построенные по методу считывания. *Метод считывания* подразумевает наличие $2^n - 1$ эталонов при n-разрядном двоичном коде. Входная аналоговая величина одновременно сравнивается со всеми эталонами. В результате преобразования получается параллельный код в виде логических сигналов на выходах $2^n - 1$ компараторов.

3. ВАРИАНТ РЕАЛИЗАЦИИ АЦП ПОСЛЕДОВАТЕЛЬНОГО СЧЁТА

В качестве примера рассмотрим структурную схему АЦП последовательного счёта с ЦАП в цепи обратной связи (рис. 1, *б*) и вариант её реализации (рис. 2). По сигналу "Пуск" на вход обнуленного счётчика СТ начинают подаваться импульсы генератора тактовой частоты ГТИ (см. рис. 1, *б*). По мере поступления этих импульсов растёт входной код ЦАП и ступенчато повышается напряжение $u_{цап}$ на его выходе, причем уровень ступени соответствует шагу квантования Δu входного напряжения $u_{вх}$ АЦП.

Процесс преобразования заканчивается, когда напряжение $u_{цап}$ станет чуть больше входного напряжения $u_{вх}$ АЦП, поданного на вход ОУ, на котором собран компаратор. При этом работа счётчика прекращается, а на его выходе устанавливается код A_i, являющийся цифровым эквивалентом напряжения $u_{вх}$.

Согласно рассмотренной структурной схеме АЦП на рис. 2 приведен вариант реализации модели 4-разрядного АЦП последовательного счёта с ЦАП, состоящего из операционного усилителя **OPAMP1** и резистивной матрицы **R1**, …, **R4** со взвешенными сопротивлениями. Переключатели **Key1**, …, **Key4** в схеме (при разомкнутом ключе **Space)** служат для проверки работы счётчика **CT**, а осциллограф **XSC1** – для снятия осциллограмм напряжения с выхода ЦАП и входа компаратора.

При запуске моделирования АЦП сформированные генератором **E1** импульсы подаются на вход счётчика **CT,** число которых последовательно высвечивается на 7-сегментном индикаторе. Выходные поразрядные сигналы со счётчика поступают также на входы логического анализатора **XLF1** и входы резистивной матрицы **R1**, …, **R4**, а суммарное напряжение с матрицы – на вход ОУ. Ступенчатое напряжение $u_{ЦАП}$ с выхода **OPAMP1** (рис. 3) подаётся на вход компаратора, собранного на операционном усилителе **OPAMP2**. На этот же вход подано постоянное напряжение $u_{вх}$ с генератора **E7** через делитель **R6-R7**. В момент, когда указанные напряжения сравниваются, компаратор срабатывает, на элемент И (**AND***)* подаётся логический 0 и прекращается работа счётчика, а на индикаторе высвечивается цифровой код (число

шагов квантования), соответствующий уровню $u_д(k)$.

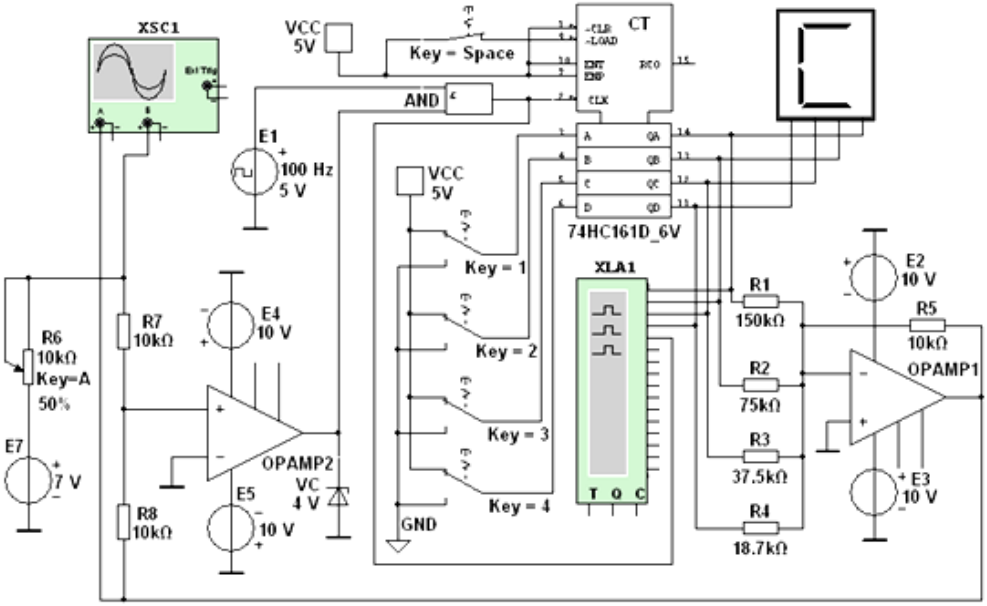


Рис. 2

Анализ временных диаграмм сигналов с выхода счётчика и осциллограмм напряжений с входов компаратора (см. рис. 3) показывает, что счётчик прервал счёт с приходом двенадцатого тактового импульса, поэтому на 7-сегментном индикаторе высветилось число C_{16} (12_2) (см. рис. 2).

Погрешность преобразования зависит от шага квантования (высоты ступени напряжения $u_{цап}$), погрешности в формировании ступенчатого напряжения $u_{цап}$ и ошибки компаратора в определении равенства $u_{вх}$ и $u_{цап}$. Время преобразования непостоянно и зависит от уровня напряжения $u_{вх}$. При заданном числе разрядов АЦП время преобразования определяется числом периодов счетных импульсов.

По структуре построения ИМС АЦП подразделяют на АЦП с применением ЦАП и без них. К БИС АЦП без ЦАП, например ИМС КР572ПВ2, К107ПВ2 и др., относят АЦП последовательного счёта с двойным интегрированием (на первом такте – входного напряжения, на втором – эталонного напряжения с преобразованием результатов интегрирования во временной интервал и в эквивалентный цифровой код) для сглаживания импульсных помех, повышения точности и помехозащищённости данного типа АЦП [12].

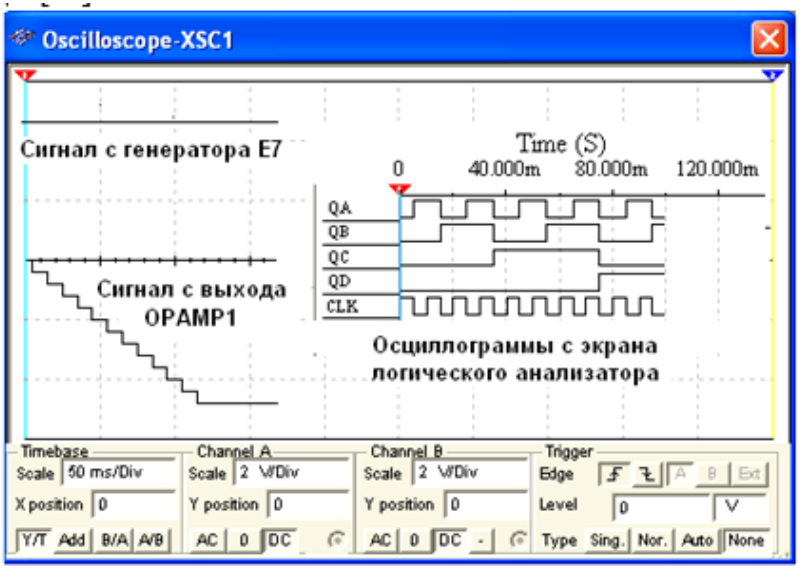


Рис. 3

Платы АЦП/ЦАП, например, модели LTC российской компании ЗАО "Л-КАРД", широко применяют в цифровых измерительных приборах, в системах и устройствах обработки и отображения информации, в автоматических системах контроля и управления, в устройствах ввода-вывода информации ЭВМ и т. д.

Основные направления развития АЦП – повышение быстродействия основных узлов, в частности, компараторов до 5…10 нс, повышение их точности до 0,05…0,005%, увеличение разрядности преобразователей до 24, использование микропроцессоров в преобразователях.

УЧЕБНЫЕ ЗАДАНИЯ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ИХ ВЫПОЛНЕНИЮ

Задание 1. **Собрать** на рабочем поле среды MULTISIM схему для испытания *аналого-цифрового преобразователя* с ЦАП (рис. 4) и **установить** в диалоговых окнах компонентов их параметры или режимы работы. **Скопировать** схему (рис. 4) на страницу отчёта.

В схему (рис. 4) включены собственно библиотечный 8-разрядный АЦП (**ADC**); источники опорного напряжения **E1 и E2** (подключены к входам **Vref+** и **Vref-** АЦП); генератор **E4** для синхронизации работы (подключен к входу **SOC**) и разрешения (вход **OE**) на выдачу двоичной информации на выходы **D0**, …, **D7** АЦП, с которыми соединены входы логического анализатора

XLA1 и пробники **X0**, …, **X7**; функциональный генератор **XFG1** в качестве источника входного сигнала u_{Bx} (подключен к входу **Vin**); ЦАП (**DAC**) и осциллограф **XSC1**. Выход **EOC** служит для передачи двоичной информации АЦП, например, на ЭВМ.

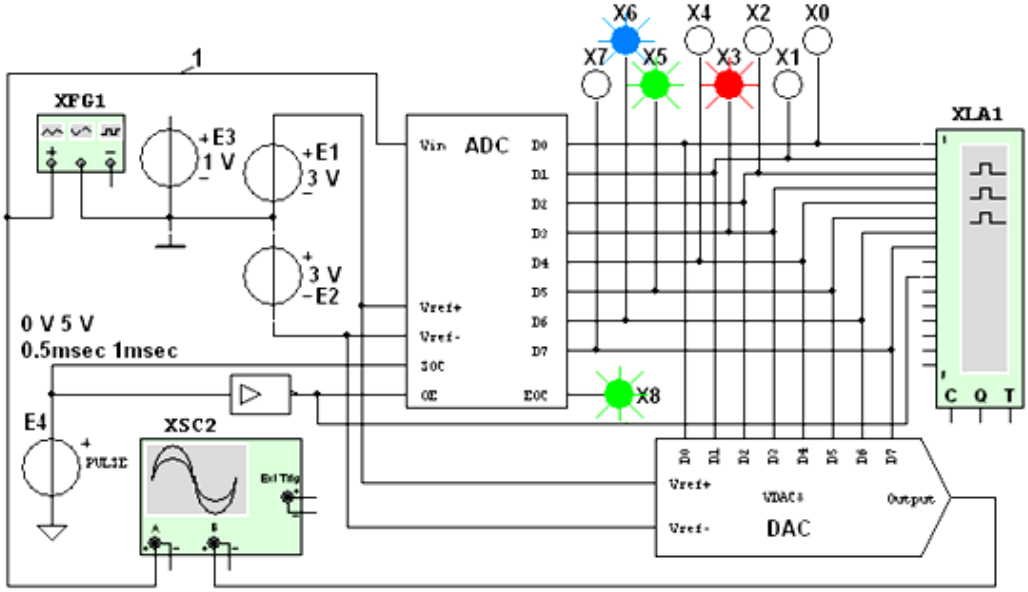


Рис. 4

Задание 2. **Исследовать** точность преобразования АЦП уровней входного напряжения u_{Bx} в цифровой код с помощью пробников **X0**, …, **X7**, логического анализатора **XLA1**, а также ЦАП и осциллографа **XSC1**.

С этой целью:

– временно **удалить** провод 1 (см. рис. 4) и подключить вход **Vin** АЦП к положительному полюсу источника постоянного напряжения **E3**;

– **составить** таблицу, аналогичную табл. 1, в первый столбец которой записать уровни напряжения

u_{Bx} = 0,1; 0,2; 0,5; 1,0; 1,5; 2,0; 2,4; -0,5; –1,0: –2,0 В,

поочерёдно задаваемые в диалоговом окне генератора **E3**;

– **установить** в диалоговых окнах генераторов **E1** и **E2** ЭДС E_1 = 2,5 В, и ЭДС E_2 = –2,5 В;

– **запустить** программу моделирования АЦП и **заносить** в поля составленной таблицы значения напряжения $u_{Bых(ЦАП)}$ с выхода ЦАП, измеряемые на экране осциллографа с помощью визирной линии; двоичный эквивалент $D_{(2)}$ преобразуемого напряже-

ния, определяемый по свечению пробников **X7**, …, **X0**; шестнадцатеричный код $D_{(16)}$, считываемый с дисплея анализатора **XLA1**;

– получаемые с выхода АЦП десятичные инверсные сигналы $D_{(10)инв}$ **пересчитать** на неинверсные $D_{(10)}$ по выражению

$D_{(10)} = D_{(10)инв} - 128$

и **занести** в соответствующие столбцы таблицы;

– расчётные десятичные эквиваленты $D_{(10)расч}$ двоичного кода $D_{(2)}$ на выходе АЦП при заданном значении входного напряжения $u_{вх}$ **определить** по формуле

$D_{(10)расч} = 256\,u_{вх}/(E_1 + |-E_2|)$,

и **занести** во второй справа столбец таблицы;

– **рассчитать** погрешности измерения напряжения по выражению

$\Delta U\% = 100(u_{вых(ЦАП)} - u_{вх})/u_{вх}$

и **занести** в правый столбец таблицы.

В качестве примера в табл. 1 приведены данные измерений при моделирования АЦП при E_1 = 3 В и E_2 = –3 В, которые близки к расчётным значениям. Так, при $E_1 = |E_2| = 3$ В и $u_{вх} = E_3 = 1$ В расчётный десятичный эквивалент $D_{(10)расч} = 256 \cdot 1/6 \approx 42{,}67$ при измеренном $D_{(2)}$ = 10101010 и $D_{(10)}$ = 42. При этом погрешность измерения составила 3,56%.

Таблица 1

$u_{вх}$, В	$u_{вых(ЦАП)}$, В	$D_{(2)}$	$D_{(16)}$	$D_{(10).инв}$	$D_{(10)}$	$D_{(10)расч}$	$\Delta U\%$
0,1	0,09375	10000100	84	132	4	4,27	6,25
0,5	0,5156	10010101	95	149	21	21,33	3,12
1,0	0,9644	10101010	AA	170	42	42,67	3,56
2,0	2,017	11010101	D5	213	85	85,34	0,85
2,5	2,484	11101010	EA	234	106	106,67	0,64
2,9	2,906	11111011	FB	251	123	123,74	0,21
-1,0	-0,9844	01010101	55	85	-43	-42,67	3,56

Задание 3. **Исследовать** процесс преобразования входного напряжения треугольной формы в цифровые коды, а затем с помощью ЦАП – в ступенчатое напряжение, аппроксимирующее напряжение $u_{вх}$.

Для этого:

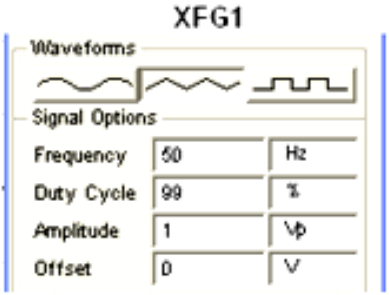
– **удалить** провод, соединяющий выход генератора **E3** с входом **Vin** АЦП, и **восстановить** провод 1, соединяющий выход "+" функционального генератора **XFG1** с входом **Vin** АЦП (см. рис. 4);

– **установить** параметры генератора **XFG1** (рис.

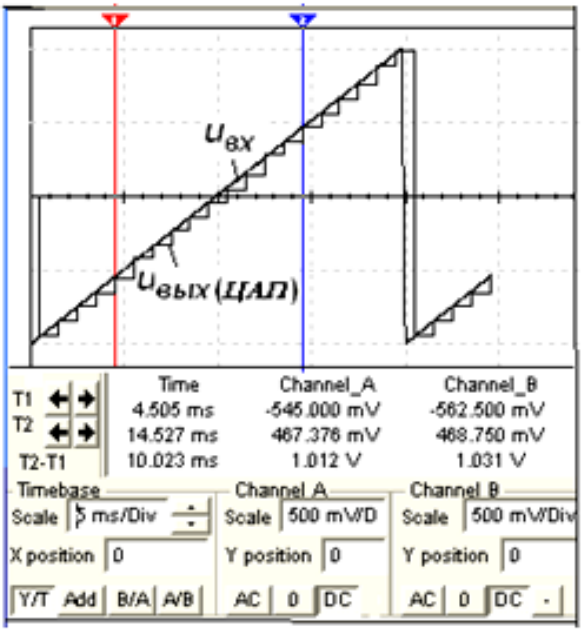
5, *а*): напряжение треугольной формы со скважностью N = 99 и амплитудой 1 В (диапазон от –1 В до 0,98 В) и его частоту $f_г$ = 50 Гц;

– **запустить** программу моделирования АЦП;

– **получить** и **скопировать** на страницу отчета осциллограмму входного напряжения $u_{вх}$, осциллограмму ступенчатого напряжения $u_{вых(ЦАП)}$ с выхода ЦАП (см. рис. 5, *б*), и временные диаграммы сигналов с выходов **D0**, …, **D7** АЦП, поступающих на входы логического анализатора **XLA1** и являющимися двоичными эквивалентами дискретных отсчётов $u_{вх}(k\Delta t)$ входного напряжения (рис. 6);



а)



б)

Рис. 5

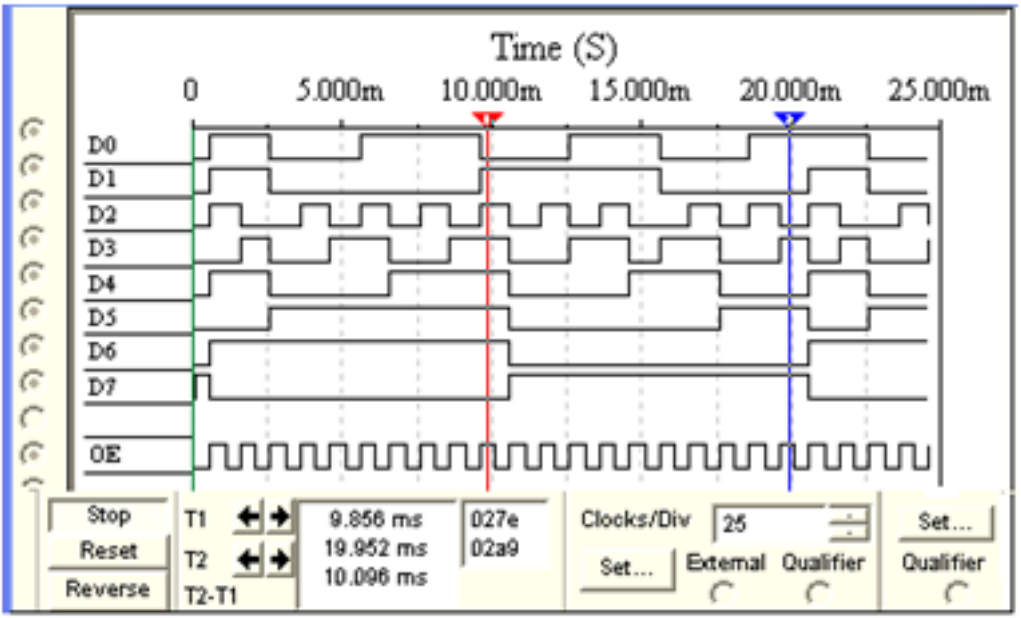


Рис. 6

– воспользовавшись визирными линиями, **провести анализ** формирования напряжения $u_{Вых(ЦАП)}$, аппроксимирующего входное напряжение $u_{Вх}$, в частности, **измерить** напряжение и высоту его ступеней в разные моменты преобразования (с интервалом в 1 мс в моменты положительного перепада тактового импульса синхронизации) и **сравнить** их с отсчётами $u_{Вх}(k\Delta t)$ напряжения $u_{Вх}$.

Так, при частоте синхронизации f_C = 1 кГц и частоте пилообразного напряжения $f_Г$ = 50 Гц образовалось на выходе ЦАП двадцать ступеней напряжения $u_{Вых(ЦАП)}$, средняя высота которых равна $U_{СТ} \approx$ 93,7 мВ при расчётном значении $\Delta u = u_{Вх.max}/(N + 1)$ = 1,98/21 = 94 мВ. Первая ступень высотой 66 мВ сформировалась по истечении 0,5 мс с момента включения моделирования при уровне входного напряжения $u_{Вх}$ = –93,4 мВ, вторая – при $u_{Вх}$ = –0,849 В высотой 93,75 мкВ и и т. д.

Задание 4 (выполняется факультативно или по указанию преподавателя). **Исследовать** процесс преобразования АЦП входного синусоидального напряжения в цифровые коды, а затем с помощью ЦАП – в ступенчатое напряжение.

С этой целью:

– **щёлкнуть мышью** на кнопке "Синусоидальное напряжение" генератора **XFG1** (см. рис. 5, *а*) и **установить** частоту напряжения $f_г$ = 25 Гц, а затем, при остановке моделирования, $f_г$ = 5 Гц с изменением времени развёртки лучей осциллографа с 10 мс/дел на 50 мс/дел. **Сместить** вверх на 0,6 деления осциллограмму входного напряжения $u_{вх}$ (рис. 7);

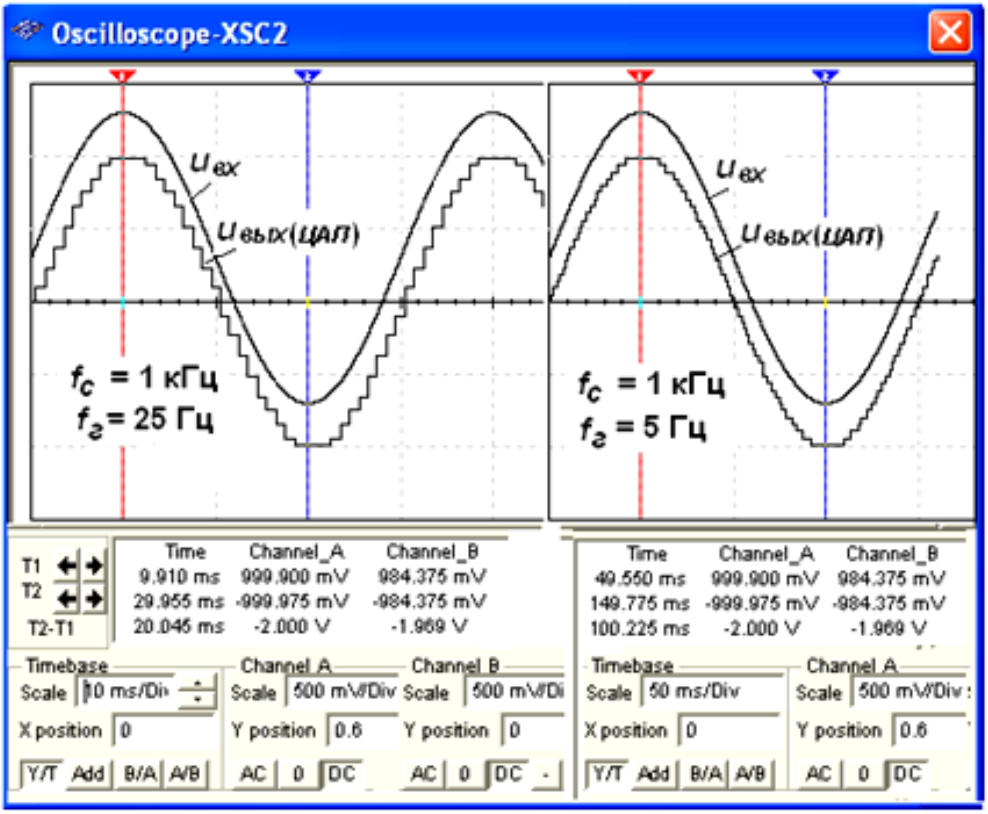


Рис. 7

– **измерить** напряжение $u_{вых(ЦАП)}$ и высоту его ступеней в разные моменты преобразования и **сравнить** их с отсчётами напряжения $u_{вх}(k\Delta t)$ входного напряжения $u_{вх}$ для моментов положительного перепада тактового импульса синхронизации.

Двоичные эквиваленты отсчетов напряжения $u_{вх}(k\Delta t)$ с выходов АЦП преобразуются с помощью ЦАП в аналоговый ступенчатый сигнал $u_{вых(ЦАП)}$ (см. рис. 7). При этом с уменьшением частоты сигнала увеличивается число ступеней и преобразованная кривая хорошо аппроксимирует входной сигнал. Высота ступеней

переменная, от 46 мВ до 141 мВ, так как интервал дискретизации Δt при заданной частоте синхронизации постоянный. Особенно заметна верхняя и нижняя ступени с отклонением от амплитуды входного напряжения приближённо на 15,5 мВ, так как на интервалах дискретизации около амплитуд скорость изменения напряжения минимальная.

СОДЕРЖАНИЕ ОТЧЁТА

1. Наименование и цель работы.
2. Перечень приборов, использованных в экспериментах, с их краткими характеристиками.
3. Изображение электрической схемы для испытания аналого-цифрового преобразователя.
4. Копии осциллограмм и временных диаграмм сигналов с разных узлов схемы, отображающие работу исследуемого АЦП.
5. Таблица с результатами измерений и расчётов входных отсчетов входного напряжения и выходных кодов АЦП.
6. Выводы по работе.

ТЕСТОВЫЕ ЗАДАНИЯ К РАБОТЕ

1. Укажите **назначение** АЦП.

- ○ Для преобразования кодов
- ○ Для преобразования цифрового кода N в пропорциональное аналоговое значение напряжения $u(N)$
- ○ Для преобразования постоянного напряжения, заданного на тактовом интервале, в двоичный код
- ○ Для преобразования информации из последовательной во времени формы представления в параллельную форму

2. Укажите **формулу Котельникова**, с помощью которой определяют шаг дискретизации Δt аналогового сигнала.

○ $\Delta t \le 1/2f_m$ ○ $\Delta t \le 1/f_m$ ○ $\Delta t \le t_{ex}/2^{N+1}$ ○ $\Delta t \le t_{ex}/2^{N-2}$

(f_m – максимальная частота спектра аналогового сигнала; t_{ex} – длительность аналогового сигнала; N – число уровней квантования)

3. Определите понятие "**абсолютная разрешающая способность**" АЦП.

- ○ Это число уровней квантования, делённое на количество разрядов выходного кода
- ○ Это наибольшее значение отклонения аналогового сигнала от расчётного
- ○ Это среднее значение минимального изменения входного сигнала, обуславливающего увеличение или уменьшение выходного кода на единицу
- ○ Это время преобразования отсчёта входного сигнала

4. Укажите, можно ли подавать на входы V_{ref+} и V_{ref-} АЦП **разные** (по модулю) **напряжения**?

○ Да ○ Нет

5. Укажите, можно ли **свести к нулю** погрешность квантования аналогового сигнала посредством выбора параметров устройства, например за счёт увеличения разрядности АЦП?

○ Да ○ Нет

6. Укажите, какую **погрешность** квантования имеет 8-разрядный АЦП при напряжениях на входах $V_{ref+} = 2$ В, $V_{ref-} = 0$ и отсчёте входного напряжения $u_{ex}(k\Delta t) = 1$ В?

○ ±4,15 мВ ○ ±3,91 мВ ○ ±3,15 мВ ○ ±2,25 мВ ○ ±1,95 мВ

7. Укажите **десятичный эквивалент** двоичного кода на выходе 8-разрядного АЦП, если опорные напряжения $V_{ref+} = 2$ В, $V_{ref-} = -2$ В, а входное напряжение $u_{ex} = 0{,}5$ В.

○ 48 ○ 32 ○ 16 ○ 8

8. Выберите из приведенных ниже значений минимально необходимые **значения опорных напряжений** $\pm V_{ref}$ для преобразования синусоидального напряжения $u_{ex}(t) = 1{,}41\sin\omega t$.

○ ±1 В ○ ±2 В ○ ±3 В ○ ±4 В ○ ±5 В

9. Укажите значение расчётного **шестнадцатеричного кода** 16-разрядного АЦП, если на его вход подано напряжение $u_{ex}(k\Delta t) = 0{,}25$ В при $\pm V_{ref} = \pm 2$ В.

○ 1000 ○ FFF ○ 10000 ○ FFFF ○ FFA

10. Укажите **выражение**, с помощью которого определяют десятичный эквивалент двоичного кода на выходе 14-разрядного АЦП

○ $D = 256u_{ex}/(V_{ref+} + |-V_{ref-}|)$ ○ $D = 16384u_{ex}/(V_{ref+} + |-V_{ref-}|)$

○ $D = 4096u_{ex}/(V_{ref+} + |-V_{ref-}|)$ ○ $D = 65536u_{ex}/(V_{ref+} + |-V_{ref-}|)$

11. Укажите, как изменится **выходной код** АЦП при неизменном входном u_{ex} и опорных напряжениях $V_{ref+} = 2$ В и $V_{ref-} = -2$ В, если установить $V_{ref-} = 0$?

○ Его значение уменьшится в 2 раза ○ Не изменится

○ Его значение увеличится в 2 раза ○ Сменится на инверсный.

12. Укажите характер изменения **общей погрешности** преобразования входного сигнала при увеличении разрядности АЦП.

○ Погрешность преобразования уменьшится ○ Не изменится

○ Погрешность преобразования увеличится ○ Нет правильного ответа

13. Укажите перспективные **направления** развития АЦП.

□ Повышение быстродействия основных узлов АЦП, в частности, компараторов

□ Увеличение частоты генератора тактовых импульсов

□ Применение стабилизированных источников опорного напряжения

□ Уменьшение разрядности преобразователя напряжение-код (до 4…6)

□ Использование микропроцессоров в преобразователях

14. Укажите, какие **операции** необходимо выполнить при аналого-цифровом преобразовании?

○ Ограничение уровня и дискретизацию по времени аналогового сигнала

○ Тактируемое интегрирование входного сигнала и сравнение полученного результата с эталонами

○ Дискретизацию по времени аналогового сигнала, квантования по уровню его отсчётов и кодирование квантованных уровней

○ Дискретизацию по времени аналогового сигнала, квантование по уровню для подачи на вход ЦАП

15. Укажите, обладает ли способ последовательного счёта аналого-цифрового преобразования наибольшим быстродействием?

○ Да ○ Нет

ЛАБОРАТОРНАЯ РАБОТА 5

ПОЛЬЗОВАТЕЛЬСКИЕ ИНТЕРФЕЙСЫ

ЦЕЛЬ РАБОТЫ

Ознакомление с основными элементами управления.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ И РАСЧЕТНЫЕ ФОРМУЛЫ

Кнопки

Кнопкой называется элемент управления, всё взаимодействие пользователя с которым ограничивается одним действием – нажатием. Эта формулировка, кажущаяся бесполезной и примитивной, на самом деле довольно важна, поскольку переводит в гордое звание кнопок многие элементы управления, которые как кнопки по большей части не воспринимаются (об этом позже).

Командные кнопки

Нажатие на такую кнопку запускает какое-либо явное действие, поэтому правильнее называть такие кнопки «кнопками прямого действия». С другой стороны, из-за тяжеловесности этого словосочетания им всегда пренебрегают.

Рис. 30. Всё это командные кнопки, они же кнопки прямого действия (включая

гипертекстовую ссылку справа).

В интернете кнопка должна быть оформлена как текстовая ссылка,

если она перемещает пользователя на другой фрагмент контента,

и как кнопка – если она запускает действие

Размеры и поля

Чем больше кнопка, тем легче попасть в нее курсором. Одновременно пользователю должно быть трудно нажать не на ту кнопку. Добиться этого можно либо изменением состояния кнопки при наведении на неё курсором, либо установлением пустого промежутка между кнопками.

Текст и пиктограммы.

Все руководства по разработке интерфейса с изумительным упорством требуют снабжать командные кнопки названиями, выраженными в виде глаголов в форме инфинитива (Прийти, Увидеть, Победить).

Помимо текста, на кнопках можно выводить пиктограммы. Эта возможность редко используется в ПО, но очень широко в

интернете. Формально, на таких кнопках пиктограммы не очень хороши из-за того, что они обычно должны передавать пользователям идею *действия* (т.е. глагол), а действие плохо передается пиктограммами.

Пиктограммы хороши для тех кнопок, для которых пиктограммы нарисовать легко, и для тех кнопок, которые нужны особенно часто (при этом качество пиктограммы особого значения не имеет, важно только различия пиктограмм между собой). С другой стороны, единство и согласованность интерфейса требует, чтобы если уж есть пиктограммы, то уж везде; если же это невозможно, лучше вообще изъять пиктограммы из кнопок, поскольку их эффект невелик, а трудозатраты, уходящие на их создание, значительны.

Кнопки доступа к меню

Существует много ситуаций, когда раскрывающийся список не помещается в отведенное для него место, поскольку текст в списке слишком велик. Первое, что приходит в голову, это вставить кнопку, нажатие на которую будет вызывать меню. Рассмотрим недостатки такого решения:

Во-первых, недостаток кнопки будет проявляться в том, что, поскольку, по условиям задачи, текст не будет виден, значение кнопки будет менее понятным, чем контекстное меню безо всякой кнопки.

Во-вторых, само использование кнопки в таком исполнении не совсем правильно, поскольку нарушается принцип единообразия: пользователь нажал на кнопку, а действия как такового и нет (не считать же действием появление меню). В интернете это еще проходит, поскольку там кнопки могут и не выглядеть как кнопки, будучи оформлены как ссылки; в этом случае противоречия не возникает.

Суммируя, можно смело сказать, что использовать кнопку для инициирования показа меню можно, но стыдно. Не высший класс.

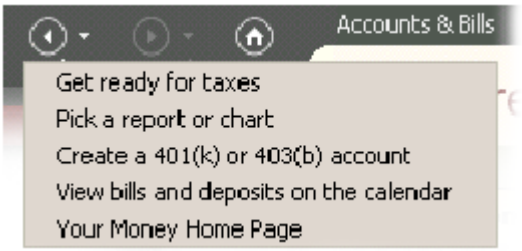


Рис. 33. Пример очень хорошего использования кнопки доступа к меню. Большинство пользователей в большинстве случаев не думают ни о каком меню, сразу нажимая на кнопку и инициируя действие. В то же время они получают возможность без труда сэкономить несколько нажатий, если обратятся к меню.

Существуют определенные ситуации, когда такие кнопки очень хороши. **Для этого только нужно сделать так, чтобы кнопка была одновременно и командной кнопкой, и показывала меню.** Для этого нужно сделать две вещи:

- Во-первых, нужно разделить кнопку на две области, одна из которых запускает действие, а другая открывает меню.

Во-вторых, нужно организовать такой контекст, при котором результат нажатия на кнопку всегда будет понятным.

Например, это очень хорошо работает с кнопками **Вперед и Назад**.

Осталось сказать немного. Во-первых, на области, вызывающей меню, обязательно должно находиться изображение направленной вниз стрелки.

Во-вторых, эта область должна находиться справа на кнопке, чтобы изображение стрелки не мешало воспринимать текст или пиктограмму на кнопке.

Чекбоксы и радиокнопки

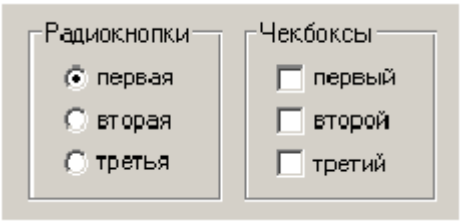


Рис. 34. Пример радиокнопок и чекбоксов.

Они являются кнопками отложенного действия, т.е. их нажатие ни при каких обстоятельствах не должно инициировать какое-либо немедленное действие. С их помощью пользователи вводят параметры, которые скажутся после, когда действие будет запущено иными элементами управления.

Нарушать это правило нельзя ни при каких обстоятельствах, поскольку это серьезно нарушит сложившуюся ментальную модель пользователей. В этом заключается общность чекбоксов и радиокнопок, теперь поговорим о различиях.

Главное различие заключается в том, что группа чекбоксов дают возможность пользователям выбрать любую комбинацию параметров, радиокнопки же позволяют выбрать только один па-

раметр. Это сближает эти элементы со списками множественного и единственного выбора соответственно.

В группе не может быть меньше двух радиокнопок (как можно выбрать что-либо одно из чего-либо одного?).

У чекбокса есть три состояния (выбранное, не выбранное, смешанное), а у радиокнопки только два, поскольку смешанного состояния у неё быть просто не может (нельзя совместить взаимоисключающие параметры).

В группе радиокнопок как минимум одна радиокнопка должна быть проставлена по умолчанию.

Всякий раз, когда пользователю нужно предоставить выбор между несколькими параметрами, можно использовать либо чекбоксы, либо радиокнопки (или списки, но о них позже). Если параметров больше двух, выбор прост: если параметры можно комбинировать, нужно использовать чекбоксы (например, текст может быть одновременно *и* жирным *и* курсивным); если же параметры комбинировать нельзя, нужно использовать радиокнопки (например, текст может быть выровнен *или* по левому, *или* по правому краю).

Если же параметров всего два и при этом параметры невозможно комбинировать (т.е. либо ДА, либо НЕТ), решение более сложно.

И чекбоксы и радиокнопки крайне желательно расставлять по вертикали, поскольку это значительно ускоряет поиск нужного элемента

Внешний вид. Традиционно сложилось так, что чекбоксы выглядят как квадраты, а радиокнопки – как кружки. Нарушать это правило нельзя.

Желательно вертикально располагать чекбоксы и радиокнопки в группе, поскольку это облегчает поиск конкретного элемента.

Текст подписей. Каждая подпись должна однозначно показывать эффект от выбора соответствующего элемента. Поскольку как радиокнопки, так и чекбоксы, не вызывают немедленного действия, формулировать подписи к ним лучше всего в форме существительных, хотя возможно использование глаголов (если изменяется не свойство данных, а запускается какое-либо действие). Подписи к стоящим параллельно кнопкам лучше стараться делать примерно одинаковой длины. Все подписи обязаны быть позитивными (т.е. не содержать отрицания). Повторять одни и те же слова, меняя только окончания подписей (например, «Показывать пробелы» и «Показывать табуляции»), в нескольких

кнопках нельзя, в таких случаях лучше перенести повторяющееся слово в рамку группировки. Если подпись не помещается в одну строку, выравнивайте индикатор кнопки (кружок или квадрат) по первой строке подписи.

Вариант для панелей инструментов

Как чекбоксы, так и радиокнопки, бывают двух видов: описанные выше стандартные, и предназначенные для размещения на панелях инструментов.

Рис. 35. Пример чекбоксов и радиокнопок на панели инструментов.

Справа расположены чекбоксы (шрифт может быть и жирным и курсивным), справа радиокнопки (абзац может быть выровнен либо по левому, либо по правому краю). Обратите внимание, что визуально чекбоксы и радиокнопки не различаются. У них есть определенный недостаток: они не различаются внешне

Списки

Все часто используемые списки функционально являются вариантами чекбоксов и радиокнопок.

Списки бывают пролистываемыми и раскрывающимися, причем пролистываемые могут обеспечивать как единственный (аналогично группе радиокнопок), так и множественный выбор (a la чекбокс); раскрывающиеся же работают исключительно как радиокнопки.

Но сначала необходимо рассказать об общих свойствах всех списков.

Ширина

Ширина списка как минимум должна быть достаточна для того, чтобы пользователь мог определить различия между элементами. В идеале, конечно, ширина всех элементов должна быть меньше ширины списка, но иногда это невозможно.

Пиктограммы.

Уже довольно давно в ПО нет технических проблем с выводом в списках пиктограмм отдельных элементов. Однако практически никто этого не делает. Это плохо, ведь пиктограммы обеспечивают существенное повышение субъективной привлекательности интерфейса и быстрее вызываются из ДВП, нежели слова.

Раскрывающиеся списки

Самым простым вариантом списка является раскрывающийся список. Помимо описанных выше родовых достоинств списков,

раскрывающиеся списки обладают одним существенным достоинством. Оно заключается в том, что малая высота списка позволяет с большой легкостью визуально отображать команды, собираемые из составляющих.

Рис. 36. Пример визуальной сборки команды из составляющих.

Данный метод значительно проще для понимания, нежели, например, ввод положительного значения для смещения вверх и отрицательного значения для смещения вниз без поддержки раскрывающимся списком.

Пролистываемые списки

Другим, более сложным вариантом списка является пролистываемый список. Пролистываемые списки могут позволять пользователям совершать как единственный, так и множественный выбор. Одно требование применимо к обоим типам списков, остальные применимы только к одному типу.

Размер. По-вертикали в список должно помещаться как минимум четыре строки, а лучше восемь. Напротив, список, по высоте больший, нежели высота входящих в него элементов, и соответственно, содержащий пустое место в конце, смотрится неряшливо. Требование выводить полоски прокрутки в больших списках кажется моветоном, но забывать о нем не следует.

Списки единственного выбора.

Список единственного выбора является промежуточным вариантом между группой радиокнопок и раскрывающимся списком. Он меньше группы радиокнопок с аналогичным числом элементов, но больше раскрывающегося списка. Соответственно, использовать его стоит только в условиях «ленивой экономии» пространства экрана.

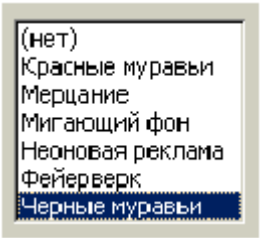


Рис. 37. Список единственного выбора.

Обратите внимание, что в ситуациях, когда все элементы помещаются в список без пролистывания, список работает в точности как группа радиокнопок.

Списки множественного выбора.

С точки зрения дизайна интерфейсов, списки множественного выбора интересны, прежде всего, тем, что их фактически нет в интернете. Технически создать список множественного выбора непроблематично, для этого в HTML есть даже специальный тег. Из-за такой убогой реализации списков браузерами, использовать их, как правило, оказывается невозможно. Приходится использовать чекбоксы1.

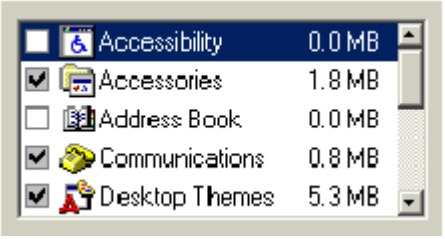


Рис. 38. Список множественного выбора с чекбоксами.

Гораздо лучше обстоят дела в ПО. Возможность безболезненно выводить в списке чекбоксы позволяет пользователям без труда пользоваться списками, а разработчикам – без труда эти списки создавать.

Комбобоксы

Комбобоксами (combo box), называются гибриды списка с полем ввода: пользователь может выбрать существующий элемент, либо ввести свой. Комбобоксы бывают двух видов: раскрывающиеся и расширенные. Оба типа имеют проблемы.

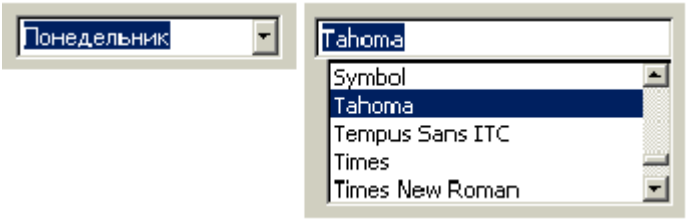


Рис. 39. Раскрывающийся комбобокс с установленным фокусом ввода (слева)

и расширенный комбобокс (справа).

У раскрывающегося комбобокса есть проблемы. Во-первых, такие комбобоксы выглядят в точности как раскрывающиеся

списки, визуально отличаясь от них только наличием индикатора фокуса ввода (да и то, только тогда, когда элемент выделен). Это значит, что полноценно пользоваться ими могут только сравнительно продвинутые пользователи. В этом нет особой проблемы, поскольку комбобоксом все равно можно пользоваться, как обычным списком.

Поля ввода

Вместе с командными кнопками, чекбоксами и радиокнопками, поля ввода являются основой любого интерфейса. В результате требований к ним довольно много.

Размеры. Основная часть требований к полям ввода касается размера. Понятно, что размер по вертикали должен быть производным от размера вводимого текста – если текста много, нужно добавить несколько строк (нарушением этого правила регулярно грешат форумы, заставляющие пользователей вводить сообщения в поля ввода размером с ноготь).

С размерами по горизонтали интереснее. Конечно, ширина поля должна соответствовать объему вводимого текста, поскольку гораздо удобнее вводить текст, который *видишь*. Менее очевидным является другое соображение: ширина поля ввода не должна быть больше объема вводимого в поле текста, поскольку частично заполненное поле выглядит как минимум неряшливо.

Ширина поля ввода не должна быть больше максимальной длины строки

Отдельной проблемой является ограничение вводимого текста. С одной стороны, ограничение хорошо для базы данных. С другой стороны, всегда найдутся пользователи, для которых поле ввода с ограничением вводимых символов окажется слишком маленьким. Поэтому этот вопрос нужно решать применительно к конкретной ситуации.

Код активации

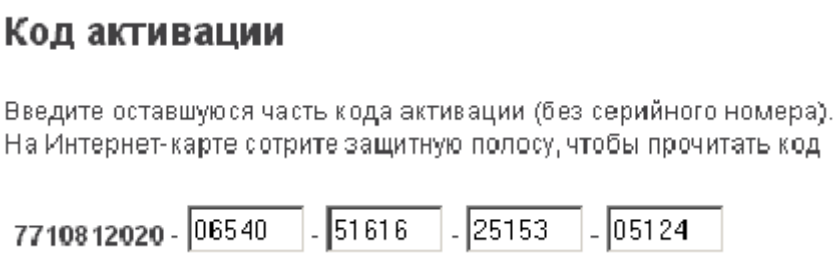


Рис. 40. Пример полей ввода, больших объема вводимых в них информации.

Мало того, что такие поля выглядят неряшливо, так они

ещё и обманывают пользователей, показывая, что пользователь ввел не всю информацию. Вдобавок, после заполнения поля приходится самому перемещать фокус ввода, хотя с этим справилась бы и система.

Если же суммировать информацию из двух предыдущих абзацев, можно определить ***самую большую ошибку, которую разработчики допускают при создании полей ввода. Всякий раз, когда ширина поля ввода больше максимального объема вводимого в него текста, при этом объем вводимого текста ограничен, пользователи неприятно изумляются, обнаружив, что они не могут ввести текст, хотя место под него на экране имеется.***

Соответственно, делать поле ввода шире текста нельзя вообще.

Подписи.

Вопрос «где надо размещать подписи к полям ввода?» является одним из самых популярных среди программистов: битвы сторонников разных подходов, хоть и бескровны, но значительны. Аргументов и подходов тут множество.

Обычно действует следующее простое правило: в часто используемых экранах подписи должны быть сверху от поля (чтобы их было легче не читать), в редко же используемых подписи должны быть слева (чтобы всегда восприниматься и тем самым сокращать количество ошибок).

Подписи к полям ввода имеют определенное отличие от других подписей. В полях ввода подписи можно размещать не рядом с элементом, а внутри него, что позволяет экономить пространство экрана. Подпись при этом выводится в самом поле ввода, точно так же, как и текст, который в него нужно вводить. Необходимо только отслеживать фокус ввода, чтобы при установке фокуса в поле убирать подпись. Это решение, будучи нестандартным, плохо работает в ПО, но неплохо работает в интернете.

Крутилки

Крутилка (spinner, little arrow) есть поле ввода, не такое универсальное, как обычное, поскольку не позволяет вводить текстовые данные1, но зато обладающее двумя полезными возможностями.

Рис. 41. Крутилка.

Во-первых, чтобы ввести значение в крутилку, пользователю не обязательно бросать мышь и переносить руку на

клавиатуру (в отличие от обычного поля ввода). Поскольку перенос руки с место на место занимает сравнительно большое время (в среднем почти половину секунды и сбивает фокус внимания, отсутствие нужды в клавиатуре оказывается большим благом. Во всяком случае, случаи ввода значения в крутилку с клавиатуры достаточно редки, т.е. пользователи воспринимают крутилки целиком и полностью положительно.

Во-вторых, при вводе значения мышью система может позволить пользователям вводить только корректные данные, причем, что особенно ценно, в корректном формате. Это резко уменьшает вероятность человеческой ошибки. Таким образом, использование крутилок для ввода любых численных значений более чем оправдано.

Ползунки

Как и ранее описанные элементы управления, ползунки позволяют пользователям выбирать значение из списка, не позволяя вводить произвольное значение. Возникает резонный вопрос: зачем нужен ещё один элемент управления, если аналогичных элементов уже полно. Ответ прост: ползунки незаменимы, если пользователям надо дать возможность выбрать значение, стоящее в хорошо ранжирующемся ряду, если:

- ✓ значений в ряду много
- ✓ нужно передать пользователям ранжируемость значений.
- ✓ необходимо дать возможность пользователям быстро выбрать значение из большого их количества (в таких случаях ползунок оказывается самым эффективным элементом, хотя и опасен возможными человеческими ошибками).

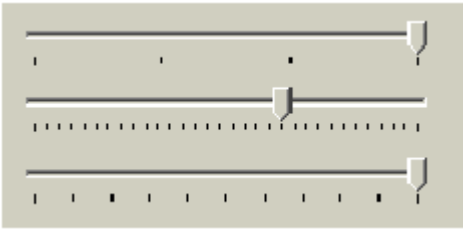


Рис. 42. Примеры ползунков.

Видно, что количество параметров в ползунке может быть весьма значительным (хотя расстояние между градациями не может быть меньше половины ширины ползунка, в случае необходимости вместить дополнительные значения можно просто увели-

чить ширину линейки).

Ползунки имеют интересный аспект. Их можно также использовать для выбора текстовых параметров, но только в случаях, когда эти параметры можно понятным образом отранжировать. Случаев таких немало, например, «завтрак», «обед» и «ужин», при отсутствии внешней связи ранжированию поддаются вполне.

Меню

При упоминании термина меню применительно к интерфейсу, большинство людей немедленно представляют стандартные раскрывающиеся меню. В действительности, понятие меню гораздо шире. Меню – это метод взаимодействие пользователя с системой, при котором пользователь выбирает из предложенных вариантов, а не предоставляет системе свою команду.

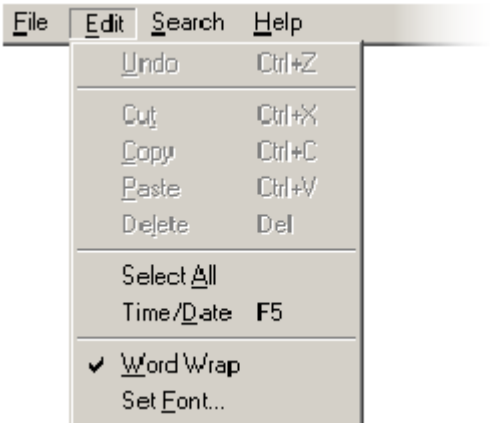


Рис. 43. Стандартное выпадающее меню.

Соответственно, диалоговое окно с несколькими кнопками (и без единого поля ввода) также является меню.

Рис. 44. Это тоже меню.

В настоящее время систем, которые не использовали бы меню в том или ином виде, практически не осталось. Объясняется это просто. Меню позволяет снизить нагрузку на мозги пользователей, поскольку для выбора команды не надо вспоминать, какая именно команда нужна и как именно её нужно использовать – вся (или почти вся) нужная информация уже содержится на экране. Вдобавок, поскольку меню ограничивает диапазон действий пользователей, появляется возможность в значительной мере изъять из этого диапазона ошибочные действия. Более того: меню показывает пользователям объем действий, которые они могут совершить благодаря системе, и тем самым обучают пользователей (в одном из исследований было обнаружено даже, что меню является самым эффективным средством обучения1). Таким образом, в большинстве систем меню является объективным благом (они неэффективны, в основном, в системах с внешней средой или течением времени).

Типы меню

Существуют несколько различных таксономий меню, но основной интерес представляют только две из них. Первая таксономия делит меню на два типа:

- ✓ Статические меню, т.е. меню, постоянно присутствующие на экране. Характерным примером такого типа меню является панель инструментов.
- ✓ Динамические меню, в которых пользователь должен вызвать меню, чтобы выбрать какой-либо элемент. Примером является обычное контекстное меню.

В некоторых ситуациях эти два типа меню могут сливаться в один:

например, меню, состоящее из кнопок доступа к меню (см. стр. 68), могут работать и как статические (пользователи нажимают на кнопки) и как динамические (пользователи вызывают меню).

Вторая таксономия также делит меню на два типа:

- ✓ Меню, разворачивающиеся в пространстве (например, обычное выпадающее меню). Всякий раз, когда пользователь выбирает элемент нижнего уровня, верхние элементы остаются видимыми.
- ✓ Меню, разворачивающееся во времени. При использовании таких меню элементы верхнего уровня (или, понимая шире, уже пройденные

элементы) по тем или иным причинам исчезают с экрана. Например, в предыдущей иллюстрации диалоговое окно с меню перекрыло элемент управления, которым это меню было вызвано.

Каждый тип меню в обеих таксономиях имеет определенные недостатки.

Статические меню из первой таксономии, как правило, обеспечивают меньшую скорость работы, лучше обучают пользователей, но зато занимают место на экране. Напротив, с динамическими меню ситуация обратная. Во второй таксономии первый тип (меню, разворачивающиеся в пространстве) обеспечивает большую поддержку контекста действий пользователей, но эта поддержка обходится в потерю экранного пространства. Второй тип более бережно использует пространство, но зато хуже поддерживает контекст.

Реальность, впрочем, оказывается несколько шире обеих таксономий. Например, мастер (см. «Последовательные окна» на стр. 99), являясь и динамическим меню из первой таксономии, и разворачивающимся во времени меню из второй, не оказывается более быстрым, чем, например, раскрывающееся меню. Но объем и специфика входящих в него элементов управления не позволяют, как правило, сделать из него какое-либо другое меню, например, раскрывающееся.

Поэтому очень полезно научиться анализировать влияние и взаимопроникновение разных типов меню, а также осознавать их место в интерфейсе. Например, контекстное меню на ином уровне абстракции оказывается временным (т.е. динамическим) диалоговым окном, только с нестандартной структурой. Понимание этой структуры позволяет определить, какие элементы управления, помимо кнопок, можно использовать в таком меню, чтобы оно обрело как достоинства меню, так и достоинства диалогового окна. К сожалению, объем этой книги не позволяет более полно описать эту тему. Поэтому в этом разделе будут описаны только главные и контекстные меню.

Устройство меню

На эффективность меню наибольшее влияние оказывают устройство отдельных элементов и их группировка. Несколько менее важны другие факторы, такие как выделение элементов и стандартность меню.

Устройство отдельных элементов

Самым важным свойством хорошего элемента меню является его название. Название должно быть самым эффективным из

возможного. В отличие от кнопок в диалоговых окнах, элементы главного меню практически никогда не несут на себе контекста действий пользователя, просто потому, что в любой момент времени доступны все элементы. Это значит, что к наименованию элементов меню нужно подходить весьма тщательно, тщательней, нежели ко всему остальному. Впрочем, помимо тщательности (и таланта, к слову говоря) нужно ещё кое-что. Обязательно нужно убедиться, что выбранное название понятно целевой аудитории. Сделать это просто – пользователю нужно сообщить название элемента и попросить его сказать, что этот элемент меню делает.

Нелишне заметить, что функциональность, не отраженная названием элемента, с большой степенью вероятности не будет найдена значительной частью аудитории. Поэтому не стоит умещать в диалоговое окно какую-либо функцию, если её существование в этом окне невозможно предсказать, глядя на соответствующий элемент меню.

Не делайте элементов меню, часть функциональности которых не влезает в текст элемента

Особо стоит остановиться на склонении текста. В отличие от диалоговых окон, в которых кнопки прямого и отложенного действия выглядят и действуют по-разному, в меню нет четкой разницы между этими элементами. Единственным способом разграничения этих элементов является текст, так что нужно очень тщательно подходить к тому, чтобы элементы, запускающие действия, были глаголами в форме инфинитива (как командные кнопки). Впрочем, часто глагол приходится выкидывать вообще, чтобы переместить значимое слово ближе в начало текст элемента. Нужно это, чтобы повысить скорость распознавания. Повысить её можно всего одним способом: главное (т.е. наиболее значимое) слово в элементе должно стоять в элементе первым. Обратите внимание, что короткий текст элемента, без сомнения, быстро читаясь, совершенно необязательно быстро распознается. Поэтому не стоит безудержно сокращать текст элемента: выкидывать нужно все лишнее, но не более.

Пиктограммы в меню

Пиктограммы в меню, если они повторяют пиктограммы в панели инструментов, обладают замечательной способностью обучать пользователей возможностям панели. Помимо этого они здорово ускоряют поиск известного элемента и точность его выбора, равно как и общую разборчивость меню. Таким образом, пиктограммы в меню объективно хороши (только стоят дорого, к сожалению). Это очевидный факт. Теперь менее очевидный: пик-

тограммы лучше работают, когда ими снабжены не все элементы. Когда все элементы имеют пиктограммы, разборчивость каждого отдельного элемента падает: в конце концов, пиктограммы всех ненужных в данное время элементов являются визуальным шумом. Когда же пиктограммами снабжены только самые важные элементы, их разборчивость повышается (а разборчивость остальных не понижается), при этом пользователям удается легче запоминать координаты элементов («элемент сразу под второй пиктограммой»).

Не снабжайте пиктограммами все элементы меню, снабжайте только самые важные

Переключаемые элементы.

Особого внимания заслуживают случаи, когда меню переключает какие-либо взаимоисключающие параметры, например, показывать или не показывать палитру. Тут есть несколько возможных способов. Можно поместить перед переключателем галочку, показывая, что он включен (если же элемент снабжен пиктограммой, можно её утапливать). Заранее скажу, что это лучший метод. Можно не помещать галочку, зато инвертировать текст элемента: например, элемент Показывать сетку превращается в Не показывать сетку. Это плохо по многим причинам. Во-первых, в интерфейсе желательно не употреблять ничего негативного: в меньшей степени потому, что негативность слегка снижает субъективное удовлетворение; в большей степени потому, что она снижает скорость распознавания текста (главное слово не первое, нужно совершить работу, чтобы из отрицания вычислить утверждение). Во-вторых, если изъять «не» и переформулировать одно из состояний элемента, пользователям будет труднее осознать, что два разных элемента на самом деле есть один элемент. Таким образом, галочка предпочтительнее. Всегда формулируйте текст в интерфейсе без использования отрицаний

Предсказуемость действия.

Пользователей нужно снабжать чувством контроля над системой. Применительно к меню это значит, что по виду элемента пользователи должны догадываться, что произойдет после выбора. Сделать это неимоверно трудно, поскольку на экране нет места под такие подсказки. Можно сделать только одно, но сделать это нужно обязательно: нужно показать пользователям, какой элемент запускает действие или меняет параметр, а какой открывает окно с продолжением диалога. Почти во всех ОС стандартным индикатором продолжения диалога является многоточие после текста элемента, так что пользоваться этим признаком стоит

везде, включая интернет. Также необходимо показывать, какой элемент срабатывает сразу, а какой открывает элементы меню нижнего уровня (в любой ОС это делается автоматически, в интернете нужно не забывать делать это вручную).

Это же правило касается и гипертекстовых ссылок вообще (они тоже меню). Пользователи испытывают значительно большее чувство контроля, когда имеют возможность предсказать, куда их ссылка приведет (при этом снижается количество ошибочных переходов). Таким образом, нестандартные ссылки (т.е. ссылки на другой сайт, на почтовый адрес, на файл, на узел FTP, на долго загружающуюся страницу и т.д.) полезно снабжать характерными для них признаками, например, ссылку на почтовый адрес пиктограммой письма.

Группировка элементов

Второй составляющей качества меню является группировка его элементов. В большинстве меню группировка оказывает не меньшее значение при поиске нужного элемента, нежели само название элемента, просто потому, что даже идеальное название не сработает, если элемент просто нельзя найти.

Чтобы уметь эффективно группировать элементы в меню, нужно знать ответы на три вопроса: зачем элементы в меню нужно группировать, как группировать элементы и как разделять группы между собой.

Зачем элементы в меню нужно группировать.

Меню, группы элементов в котором разделены, сканируется значительно быстрее обычного, поскольку в таком меню больше «точек привязки» (точно также как и в меню с пиктограммами). К тому же наличие явных разделителей многократно облегчает построение ментальной модели, поскольку не приходится гадать, как связаны между собой элементы. Наконец, в объемных меню группировка элементов облегчает создание кластеров в кратковременной памяти, благодаря чему всё меню удается пометить в кратковременную память.

Как группировать элементы.

Каждый знает, или, во всяком случае, догадывается, что элементы в меню нужно группировать максимально логично. Поспорить с этим утверждением нельзя, но от этого его проблематичность не уменьшается.

Взаимоисключающие элементы желательно помещать в отдельный уровень иерархии

Дело в том, что существует множество типов логики. Есть логика разработчика, который знает все функции системы. Есть

логика пользователя, который знает только меньшую часть. При этом практика показывает, что эти типы логики в значительной мере не совпадают. Поскольку пользователи важнее, нужно сгруппировать меню в соответствии с их логикой.

Для этого используется очень простой и надежный метод, называемый карточной сортировкой (см. стр. 120).

Как разделять группы между собой.

Существует два основных способа разделять группы: между группами можно помещать пустой элемент (разделитель) или же размещать отдельные группы в разных уровнях иерархии.

Второй способ создает более четкое разделение: в меню Файл, например все элементы более близки друг другу (несмотря на разделители), чем элементы других меню. В то же время выбор конкретного способа диктуется результатами карточной сортировки, так что интерес представляет только вопрос «как должны выглядеть и действовать разделители».

Для разграничения групп традиционно используют полоски. Это надежное, простое решение, другой разговор, что с дизайнерской точки зрения полоски плохи, поскольку представляют собой визуальный шум. Гораздо правильнее, но и труднее, использовать только визуальные паузы между группами, как это сделано, например, в MacOS X.

Глубина меню.

Наличие многих уровней вложенности в меню приводит к там называемым «каскадным ошибкам»: выбор неправильного элемента верхнего уровня неизбежно приводит к тому, что все следующие элементы также выбираются неправильно. При этом широкие меню больше нравятся пользователям. Поэтому большинство разработчиков интерфейсов стараются создавать широкие, а не глубокие меню1.

К сожалению, у широких меню есть недостаток: они занимают много места. Это значит, что, начиная с определенного количества элементов, меню физически не сможет оставаться широким, оно начнет расти в глубину. Возникает проблема, которую надо решать. Итак, проблема заключается в том, что велика вероятность каскадных ошибок. Чтобы снизить их число, нужно повысить вероятность того, что пользователи будут правильно выбирать элементы верхних уровней. Чтобы повысить эту вероятность, нужно заранее снабдить пользователей контекстом.

При перемещении по меню пользователь действует по определенному алгоритму:

1 Выбирая элемент первого уровня, он выбирает элемент,

«нужность» которого кажется ему максимальной.

2 После выбора он видит список элементов второго уровня, при этом он оценивает вероятность соответствия всех элементов второго уровня его задаче и одновременно выбирает наиболее вероятный элемент. При этом в уме он держит контекст, т.е. название элемента первого уровня.

3 Если ни один из элементов не кажется пользователю достаточно вероятным, пользователь возвращается на первый уровень.

4 Если какой-то элемент удовлетворяет пользователя, он выбирает его и получает список элементов третьего уровня. Действия из второго и третьего шагов повторяются применительно к новым элементам меню.

Видно, что действия пользователя при поиске нужного элемента отчетливо цикличны, при этом на каждом шаге есть вероятность ошибок. При этом с каждым новым уровнем меню объем контекста, который приходится держать в голове, непрерывно возрастает. При этом, если пользователь всё-таки не находит нужного элемента, весь этот контекст оказывается ненужным. Хранение же контекста, даже не засчитывая усилия, затрачиваемые на выбор элемента, есть довольно существенная работа. Её объем лучше уменьшить.

Теперь рассмотрим другой вариант: пользователь по самому элементу может предугадать его содержимое, т.е. при поиске элемента в меню не столько оценивает контекст, сколько просто ищет нужный элемент. Эта возможность есть в любом случае, поскольку элемент имеет хоть сколько-нибудь значимый идентификатор (т.е. его название). Но она, как правило, довольно слаба и почти всегда допускает неоднозначность. Усилить её можно наличием аннотации к каждому элементу, но эту аннотацию никто не будет читать.

Есть другой метод, и этот метод есть лучшее, что дал интернет науке о проектировании интерфейсов: в качестве аннотации к элементу можно показывать наиболее популярные элементы следующего уровня.

Контекстные меню

Преимущество контекстных (всплывающих) меню заключается в том, что они полностью встраиваются в контекст действий пользователей: не нужно переводить взгляд и курсор в другую область экрана, практически не нужно прерывать текущее действие для выбора команды. При этом они не занимают места на экране, что всегда ценно. С другой стороны, из-за того, что они

не находятся всё время на экране, они практически неспособны чему-либо научить пользователя.

Не делайте контекстные меню единственным способом вызова какой*либо функции

Поскольку основной причиной появления контекстных меню является стремление максимально повысить скорость работы пользователей, на их размер и степень иерархичности накладываются определенные ограничения. Если меню будет длинным, пользователям придется сравнительно долго возвращать курсор на прежнее место, так что привлекательность нижних элементов окажется под вопросом. Поэтому лучше сокращать размер контекстных меню до разумного минимума (порядка семи элементов).

К тому же не надо забывать, что главное меню *не всегда* перекрывает выделенный (т.е. актуальный объект), а контекстное меню – *почти всегда* (как-никак оно вызывается на самом объекте). В большинстве же случаев перекрытие актуального объекта нежелательно (сбивается контекст). Мы не можем сделать в этой ситуации ничего, кроме как уменьшить размер меню, в расчете, что маленькое меню будет перекрывать малое количество информации. Разумеется, если точно известно, что оперируемый объект совсем уж мал, сокращать объем меню бесполезно. Другая особенность контекстных меню – иерархия. В обычном меню иерархия имеет хотя бы одно достоинство: при обучении она позволяет упорядочивать элементы меню и тем самым делать его понятнее. В контекстных же меню обучающая функция не играет никакой роли, поскольку такими меню пользуются только опытные пользователи. Иерархия элементов теряет свое единственное достоинство, не теряя ни одного недостатка. Поэтому делать иерархические контекстные меню можно, ничего плохого в этом нет, но необходимо сознавать, что вложенными элементами почти никто не будет пользоваться (тем более что вложенность сбивает контекст действий).

Система сначала должна показывать максимально релевантную информацию, затем всё остальное

Последнее отличие контекстных меню от обычных заключается в том, что в них очень важен порядок следования элементов. В главном меню не обязательно стремиться к тому, чтобы наиболее часто используемые элементы были самим первыми – все равно курсор придется возвращать к рабочему объекту, так что разницы в дистанции перемещения курсора практически нет. В контекстном же меню ситуация обратная – чем дальше нужный элемент от верха меню, тем больше придется двигать курсор.

Поэтому правило релевантности в таких меню действует в полной мере.

Окна

Поскольку разработка интерфейса заключается в основном в том, чтобы правильно помещать правильные элементы управления в правильные диалоговые окна или экраны, окна требуют не меньше заботы, чем элементы управления.

Типы окон

Современная наука знает несколько типов окон, а именно:

_ главные окна программы

_ окна документа

_ режимные диалоговые окна (о разнице между режимными и безрежимными окнами

_ безрежимные диалоговые окна

_ палитры

_ окна браузера (поскольку используемая в интернете технология существенно отличается от технологии ПО, этот тип окон стоит несколько особняком).

При этом доля отдельных типов в общем пироге со временем изменяется: окна документов, как будет показано ниже, отмирают, заменяясь окнами программ, режимные диалоговые окна сменяются безрежимными, а безрежимные, в свою очередь, палитрами. Интересно, что идея палитр тоже клонится к закату (палитры сменяются панелями инструментов, причины этого опять-таки рассмотрены ниже), так что в будущем, скорее всего, в ПО останутся только окна программ, панели инструментов и безрежимные диалоговые окна (которые разработчики поленятся переделывать).

Задание. **Запустить** 3 разнотипных интерфейса (например, один из программных продуктов Microsoft Office, сайт и приложение). **Проанализировать** какие основные элементы управления используются в них. **Оценить** пригодность каждого элемента. **Дать** свою характеристику интерфейсам используя психофизические характеристики и факторы оценки пользовательских интерфейсов.

СОДЕРЖАНИЕ ОТЧЕТА

1. Наименование и цель работы.

2. Перечень программных продуктов использованных при анализе.

3. Выводы по работе.