Основы Цифровой Электроники

Проф. Барри Патон,

Университет Далхауси, Канада

Оригинальное издание: Март 1998

В переводе на русский язык: Сентябрь 2002

Код продукта 321948A-01

Авторские права

Торговые марки:

Дополнительная информация

Если у Вас есть какие-либо вопросы или пожелания относительно этого руководства, пожалуйста, смотрите следующую ссылку в Интернете: http://sensor.phys.dal.ca/Digital Electronics/.

Обратная связь

Российское представительство National Instruments с благодарностью примет ваши рецензии, отзывы, замечания и пожелания по поводу содержания, качества перевода и оформления русского издания данного учебного курса.

Наша контактная информация:

117049 г.Москва, Ленинский проспект 1/2, офис 1013
тел/факс +7 (095) 238 7139

электронная почта: *ni.russia@ni.com, books2002@labview.ru*
Интернет: *http://ni.com/russia, http://www.labview.ru*

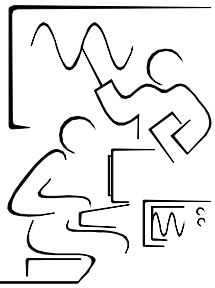
Штаб-квартира корпорации National Instruments:

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 1 (512) 794 0100

Международные представительства:

Австралия 03 9879 5166, Австрия 0662 45 79 90 0, Бельгия 02 757 00 20, Бразилия 011 284 5011, Канада (Калгари) 403 274 9391, Канада (Онтарио) 905 785 0085, Канада (Квебек) 514 694 8521, Китай 0755 3904939, Дания 45 76 26 00, Финляндия 09 725 725 11, Франция 01 48 14 24 24, Греция 30 1 42 96 427, Германия 089 741 31 30, Гонконг 2645 3186, Индия 91805275406, Израиль 03 6120092, Италия 02 413091, Япония 03 5472 2970, Корея 02 596 7456, Мексика 5 280 7625, Мексика (Монтеррей) 8 357 7695, Нидерланды 0348 433466, Новая Зеландия 09 914 0488, Норвегия 32 27 73 00, Польша 0 22 528 94 06, Португалия 351 1 726 9011, Россия 7 095 238 7139, Сингапур 2265886, Испания 91 640 0085, Швеция 08 587 895 00, Швейцария 056 200 51 51, Тайвань 02 2528 7227, Великобритания 01635 523545

Содержание



Содержание 3

Введение 6

Занятие 1. Логические элементы. 8

Логический элемент И 8

Логические элементы ИЛИ и Исключающее ИЛИ 9

Отрицание 9

Логические элементы НЕ-И, НЕ-ИЛИ, НЕ-Исключающее ИЛИ 10

Построение одних логических элементов из других 10

Логические элементы с более чем двумя входами 11

Наложение маски 12

Приложение: Селектор данных 13

Определите логический элемент 14

Состав библиотеки виртуальных приборов, использованных на занятии 1 (показан в порядке упоминания) 14

Занятие 2. Шифраторы и дешифраторы. 15

Игральная кость 16

Счетчик по модулю 6 (или счетчик с коэффициентом пересчета 6) ... 17

Шифратор 18

Виртуальная игральная кость 19

Состав библиотеки виртуальных приборов, использованных на занятии 2 (показан в порядке упоминания) 20

Занятие 3. Операция суммирования по модулю 2 21

Расширение сумматора 23

Двоично-десятичное представление (ДДП) 26

Задание повышенной сложности 27
Состав библиотеки виртуальных приборов, использованных на занятии 3 (показан в порядке упоминания) 27

Занятие 4. Оперативная память: Регистр D-типа. 28

Сдвиговые регистры 29
Задание повышенной сложности 31
Кольцевые счетчики 31
Состав библиотеки виртуальных приборов, использованных на занятии 4 (показан в порядке упоминания) 32

Занятие 5. Генераторы псевдослучайных чисел. 33

Шестибитовый ГПСЧ 33
8-битовый генератор псевдослучайных последовательностей 34
8-битовый генератор псевдослучайных чисел 36
Шифрование цифровых данных 38
Состав библиотеки виртуальных приборов, использованных на занятии 5 (показан в порядке упоминания) 39

Занятие 6. JK-триггер типа «ведущий-ведомый». 40

Двоичные счетчики 42
8-битовый двоичный счетчик 44
Задание повышенной сложности 44
Резюме 45
Состав библиотеки виртуальных приборов, использованных на занятии 5 (показан в порядке упоминания) 45

Занятие 7.Цифро-аналоговый преобразователь. 46

Что такое ЦАП? 46
Арифметико-логическое устройство 48
Моделирование реального ЦАП 48
Генераторы сигналов 50
Специальные ЦАП 51
Фигуры Лиссажу 52
Состав библиотеки виртуальных приборов, использованных на занятии 7 (показан в порядке упоминания) 53

Занятие 8. Аналого-цифровой преобразователь. Часть 1. 54

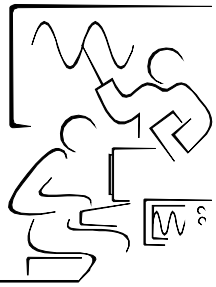
Назначение аналого-цифрового преобразователя..... 54
Интегрирующий АЦП..... 55
Задание повышенной сложности 57
Следящие АЦП..... 57
Состав библиотеки виртуальных приборов, использованных на занятии 8 (показан в порядке упоминания) 60

Занятие 9. Аналого-цифровой преобразователь. Часть 2. 61
Моделирование АЦП последовательных приближений..... 63
Резюме 65
Состав библиотеки виртуальных приборов, использованных на занятии 9 (показан в порядке упоминания) 65

Занятие 10. Семисегментные цифровые дисплеи. 66
Семисегментный индикатор 66
Задание повышенной сложности 70
Состав библиотеки виртуальных приборов, использованных на занятии 10 (показан в порядке упоминания..... 71

Занятие 11. Последовательная передача данных. 72
Последовательный передатчик данных 73
Преобразователь аналогового напряжения в последовательный код 76
Задание повышенной сложности 77
Состав библиотеки виртуальных приборов, использованных на занятии 11 (показан в порядке упоминания) 78

Занятие 12. Центральный процессор..... 79
Работа арифметико-логического устройства. 80
Накапливающий сумматор..... 81
Операция сложения 83
Двоичный счетчик..... 83
Задание повышенной сложности 85
Состав библиотеки виртуальных приборов, использованных на занятии 12 (показан в порядке упоминания) 85



Введение

Цифровая электроника – один из фундаментальных курсов, входящих в радиотехнику и многие другие научно-образовательные дисциплины. Большое разнообразие логических и числовых индикаторов, управляющих элементов, структур и функций программирования делают пакет LabVIEW превосходным инструментом для визуализации и демонстрации многих фундаментальных принципов цифровой электроники. Модульный принцип работы, применяемый в LabVIEW, используется и при создании сложных цифровых интегральных схем из более простых схем, которые в свою очередь строятся на основе базовых логических элементов. Данная книга создана как учебный курс, который может применяться в аудиторном практикуме, для самостоятельных занятий или в лаборатории.

Последовательность занятий повторяет порядок изложения материала, принятый в большинстве учебников по электронике. На первых шести занятиях изучаются базовые схемы на основе логических элементов, кодирующих устройств, двоичного сложения, D-вентилей, кольцевых счетчиков и JK-триггеров. Многие виртуальные приборы могут применяться как для демонстрации на занятиях, так и в лабораторных работах.

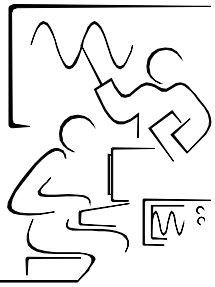
На последних шести занятиях изучаются более сложные темы, такие как цифро-аналоговые и аналогово-цифровые преобразователи, семисегментные индикаторы, последовательная передача данных, процессоры. Для лучшего понимания эти темы изучаются в контексте лабораторных работ по цифровой электронике, где сравниваются реальные интегральные схемы с их моделями, выполненными в среде LabVIEW. В каждом таком примере вы можете улучшать модели, используя многофункциональную плату ввода-вывода фирмы National Instruments, которая необходима для взаимодействия виртуальных приборов среды LabVIEW, таких как цифровой и аналоговый ввод-вывод, с окружающими реальными устройствами схемы.

Занятия 2, 5 и 12 ориентированы на изучение конкретных устройств. На них демонстрируются принципы работы кодирующих схем, цифрового шифрования и процессора. Эти занятия могут являться основой для семинаров или научных практикумов.

Объединив эти занятия, можно продемонстрировать связь устройств высокого уровня и определенных базовых логических элементов.

Например, принцип работы процессора зависит от концепции регистров и двух операций ввода.

В данном пособии используется набор виртуальных приборов LabVIEW. Содержание пособия также содержится на компакт-диске, так что вы можете выбрать наиболее удобную для вас форму представления материала.



Занятие 1.
Логические элементы.

Логические элементы – это базовые блоки цифровых логических схем. Они могут открываться или закрываться, позволяя или отказывая пропускать логический сигнал. На основе небольшого количества основных логических элементов (И, ИЛИ, исключающее ИЛИ, НЕ) может быть построено громадное количество логических функций.

Логический элемент И

Базовый логический элемент И состоит из двух входов и выхода. Если два входа назвать соответственно A и B, то выход (обычно называют Q) находится в состоянии «включено» только тогда, когда *оба* входа A и B находятся в состоянии «включено».

В цифровой электронике состояние «включено» часто представляется в виде 1, а состояние «выключено» в виде 0. Соотношение между входными и выходными сигналами представляют в виде таблицы истинности, в которой сопоставляются все возможные состояния входов и результирующих выходов. Для логического элемента И существуют четыре возможные комбинации входного состояния: *A*=0, *B*=0; *A*=0, *B*=1; *A*=1, *B*=0 и *A*=1, *B*=1. Эти значения представлены в следующей таблице истинности в левом и среднем столбцах. Выход логического элемента И отображен в правом столбце.

Таблица 1-1. Таблица истинности для логического элемента И.

A	*B*	*Q* = *A* И *B*
0	0	0
0	1	0
1	0	0
1	1	1

В среде LabVIEW вы можете определять состояние логического входа переключением логического выключателя, а логический светодиодный индикатор может показывать состояние выхода. Поскольку в среде LabVIEW элемент И является одной из основных встроенных функций, то вы легко можете создать простейший виртуальный прибор,

демонстрирующий работу этого логического элемента, присоединив два выключателя к его входу и светодиодный индикатор к его выходу.

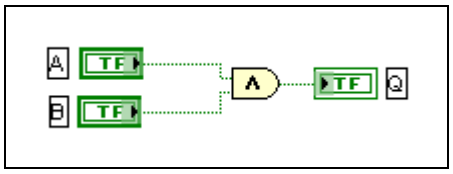


Рисунок 1-1. Функция LabVIEW «И», присоединенная к входным и выходному терминалам.

Запустите виртуальный прибор **AND gate.vi** из библиотеки **Chap 1.llb**. Нажмите на две входные кнопки и наблюдайте изменения выходного индикатора. Проверьте таблицу истинности, показанную выше.

Логические элементы ИЛИ и Исключающее ИЛИ

Логический элемент ИЛИ – это также элемент с двумя входами и одним выходом. В отличие от элемента И, на выходе получим 1, когда один или другой вход или сразу оба входа равны 1. Выход элемента ИЛИ только тогда равен 0, когда оба входа равны нулю.

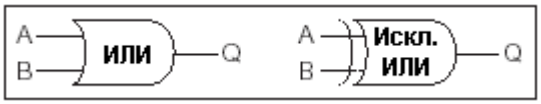


Рисунок 1-2. Обозначения логических элементов ИЛИ и Исключающее ИЛИ.

Связанный с предыдущим логический элемент исключающее ИЛИ дает на выходе 1, когда один *и только один* из входов равен 1. Другими словами, выход этого элемента равен 1, когда состояния входов различны.

Отрицание

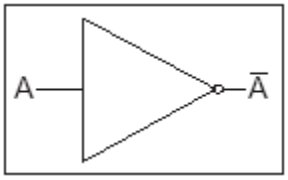


Рисунок 1-3. Логический элемент НЕ.

Еще более простым элементом является логический элемент НЕ. Он имеет один вход и один выход. Состояние выхода всегда противоположно входному (*логическое отрицание* или *инверсия*).

Логические элементы И-НЕ, ИЛИ-НЕ, Исключающее ИЛИ-НЕ

Операция отрицания очень полезна. Поэтому в добавок к трем логическим элементам (И, ИЛИ и исключающее ИЛИ) с двумя входами, которые мы выше обсуждали, обычно пользуются еще тремя. Они во всем идентичны первым, за исключением того, что на их выходе добавлена операция логического отрицания (инверсии). Эти элементы называются И-НЕ, ИЛИ - НЕ и исключающее ИЛИ - НЕ. Их символическое изображение такое же, как и для элементов без отрицания, только на выходе рисуется маленький кружок:

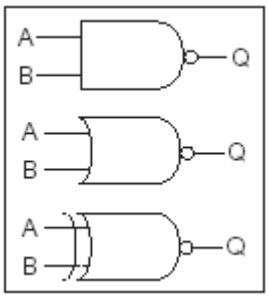


Рисунок 1-4. Инвертированные логические элементы И. ИЛИ и исключающее ИЛИ.

Запустите виртуальный прибор **Truth table.vi**. Выберите логический элемент и попробуйте все комбинации входов A и B, чтобы заполнить следующую таблицу истинности.

Таблица 1-2. Таблица истинности для основных логических элементов.

A	B	И	ИЛИ	Искл. ИЛИ	И – НЕ	ИЛИ - НЕ	Искл. ИЛИ - НЕ
0	0	0					
0	1	0					
1	0	0					
1	1	1					

Построение одних логических элементов из других

Используя несколько логических элементов И-НЕ, вы можете построить все остальные базовые логические элементы. Например, можно построить элемент НЕ, соединяя вместе входы элемента И-НЕ.

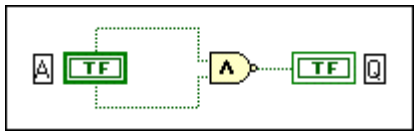


Рисунок 1-5. Логический элемент НЕ, построенный из элемента И – НЕ.

Похожим образом легко построить элемент И из двух элементов И-НЕ:

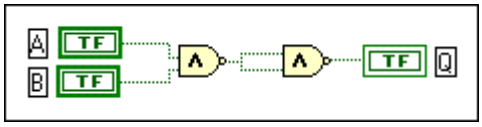


Рисунок 1-6. Логический элемент И, построенный из двух элементов И-НЕ.

Для элемента ИЛИ вам потребуется три элемента И-НЕ:

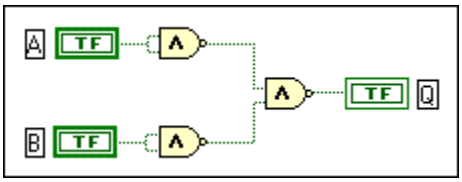


Рисунок 1-7. Логический элемент ИЛИ, построенный из трех элементов И - НЕ.

Придумайте виртуальный прибор, демонстрирующий, что элемент исключающее ИЛИ можно построить из четырех элементов НЕ-И. Для справки используйте виртуальный прибор **XOR from NAND.vi** из библиотеки **Lab01.llb**.

Логические элементы с более чем двумя входами

Хотя в пакете LabVIEW содержаться все основные двухвходовые логические элементы, вы можете использовать больше входов. Например, таблица истинности элемента И, которую мы рассматривали выше, можно представить в следующем виде, обобщенном на случай трех входов:

Таблица 1-3. Таблица истинности для трехвходового логического элемента И.

A	*B*	*C*	*A* **И** *B* **И** *C*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Из двух двухвходовых логических элементов И вы можете построить виртуальный прибор, реализующий логический элемент И с тремя входами:

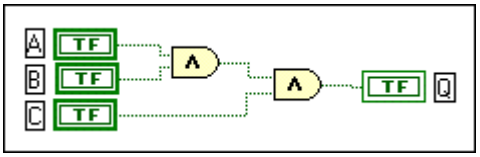


Рисунок 1-8. Модель логического элемента И с тремя входами.

Откройте виртуальный прибор **3 AND.vi**. Обратите внимание на его иконку и выходные коннекторы. С помощью них этот виртуальный прибор можно сделать полноценной подпрограммой.

Наложение маски

Простейшей примером того, как базовые логические элементы могут применяться вместе, является понятие наложения маски. Для иллюстрации этого понятия ниже представлена таблица истинности логического элемента И, названия столбцов в которой изменены.

Таблица 1-4. Таблица истинности для логического элемента И с одним входом в виде маски.

A	*Маска*	*A И B*	Результат	
0 1	0 0	0 1	A – заблокирован	Логический элемент – «закрыт»
0 1	1 1	0 1	A – неизменен	Логический элемент – «открыт»

Таблица истинности доказывает, что элемент И можно использовать в качестве электронного ключа.

Это утверждение легко демонстрируется с помощью LabVIEW:

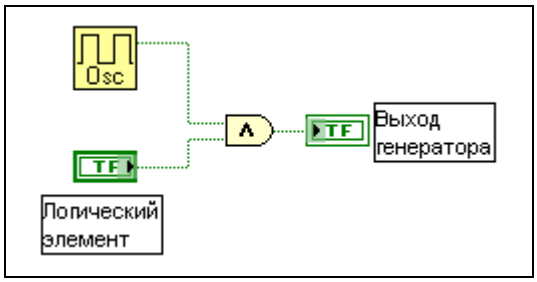


Рисунок 1-9. Логический элемент И, работающий в режиме электронного ключа.

Запустите виртуальный прибор **E-switch.vi**, чтобы наблюдать в действии электронный ключ. Используя понятие наложения маски, вы также можете увидеть таблицы истинности для других логических элементов. В следующей таблице reset означает «сбросили в состояние 0», а set – «установили в состояние 1».

Таблица 1-5. Таблица истинности для логических элементов И, ИЛИ, Искл. ИЛИ с одним входом в виде Маски.

A	*Маска*	И	ИЛИ	Искл. ИЛИ
0 1	0 0	Reset A	A – неизменен	A – неизменен
0 1	1 1	A – неизменен	Set A	A - инвертирован

Таким образом, здесь сведены три полезные функции. Чтобы установить состояние 1, используйте элемент ИЛИ с маской 1. Чтобы установить состояние 0, используйте элемент И с маской 0. Чтобы инвертировать состояние, используйте элемент ИЛИ - НЕ с маской 1.

Приложение: Селектор данных

Другим простейшим применением основных логических элементов является селектор данных, в котором один цифровой вход отбирает тот или иной поток цифровых данных:

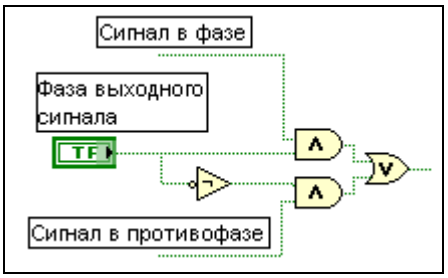


Рисунок 1-10. Цифровой селектор данных, построенный на основе базовых логических элементов.

В пакете LabVIEW есть встроенная функция, называемая **Select**, которая имитирует подобную операцию. Таким образом, вы можете реализовать предыдущий прибор в виде:

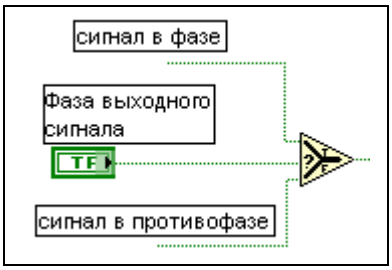


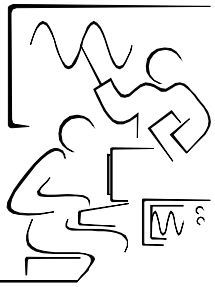
Рисунок 1-11. Модель цифрового селектора данных.

Определите логический элемент

Элементы, представленные в этой главе, составляют основы всей цифровой электроники. Поэтому необходимо хорошее знание таблиц истинности этих логических элементов. В качестве повторения пройденного материала, проверьте себя с помощью виртуального прибора **Name that gate.vi**.

Состав библиотеки виртуальных приборов, использованных на занятии 1 (показан в порядке упоминания)

- **AND gate.vi** (двухвходовая операция И)
- **Truth table.vi** (для элементов И, ИЛИ, исключающее ИЛИ, И-НЕ, ИЛИ-НЕ, исключающее ИЛИ-НЕ)
- **XOR from NAND.vi**
- **3 AND.vi** (трехвходовая операция И)
- **Masking.vi** (демонстрация)
- **E-switch.vi** (электронный ключ)
- **Data select.vi** (селектор данных на основе базовых логических элементов)
- **Data select2.vi** (реализация селектор данных, используя функцию LabVIEW **Select**)
- **Oscillator.vi** (подпрограмма, используемая в ВП **Data select.vi**)
- **Name that gate.vi** (проверка ваших знаний)



Занятие 2.
Шифраторы и дешифраторы.

Шифратор преобразует состояние устройства в двоичное представление, состоящее из нулей и единиц. Рассмотрим поворотный переключатель, имеющий десять позиций, которые применяются для ввода чисел от 0 до 9. Каждая позиция может быть закодирована единственной двоичной последовательностью. Например, позицию 7 можно закодировать как 0111. Дешифратор выполняет обратное преобразование – из двоичных кодов в выходные коды.

Рассмотрим случай игральной кости. На каждой из ее шести граней имеется одно из следующих изображений, представляемых числами 1-6.

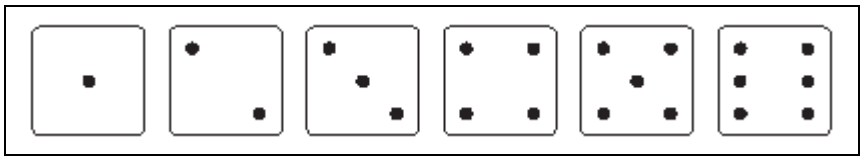


Рисунок 2-1. Шесть граней игральной кости.

Данные изображения граней игральной кости традиционны. Все их можно реализовать посредством семи индикаторов, расположенных в H-образном порядке:

Рисунок 2-2. Расположение индикаторов.

Зажигая нужные индикаторы, вы можете создать изображение любой из шести граней игральной кости.

При более внимательном рассмотрении видно, что существуют четыре особых изображения, из которых можно сформировать рисунок каждой грани. Назовем эти изображения А, Б, С и Д:

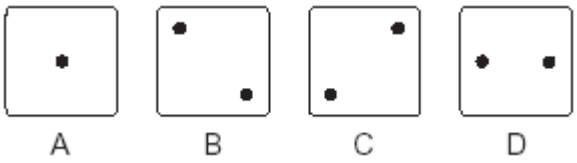


Рисунок 2-3. Четыре основных рисунка.

Если вы распишете таблицу истинности для события наличия или отсутствия этих базовых элементов в зависимости от граней игральной кости, то станет понятным назначение этих элементов.

Таблица 2-1. Основные состояния, используемые для изображения каждой грани.

Грань	*A*	B	C	D
1	√			
2		√		
3	√	√		
4		√	√	
5	√	√	√	
6		√	√	√

Базовый рисунок А используется для всех нечетных номеров граней (1, 3 и 5). Рисунок Б присутствует в изображении всех граней кроме первой. Рисунок С находится а гранях с номерами 4, 5 и 6. Рисунок Д используется в представлении только шестой грани.

Игральная кость

Для создания виртуальной игральной кости расположите на лицевой панели четыре переключателя и семь светодиодных индикатора в Н-образном порядке. На панели диаграмм соедините входы индикаторов таким образом, чтобы изобразить четыре базовых рисунка А, Б, С и Д. Тогда четыре переключателя на лицевой панели смогут моделировать включение и выключение базовых изображений.

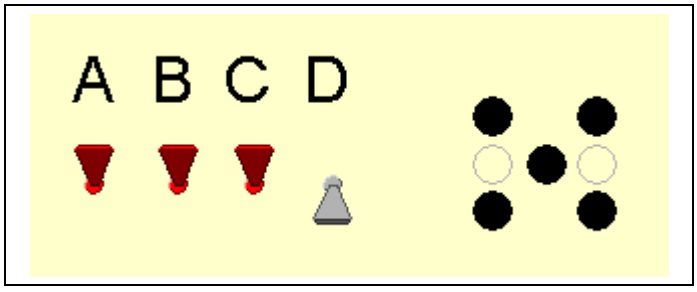


Рисунок 2-4. Лицевая панель прибора, моделирующего игральную кость.

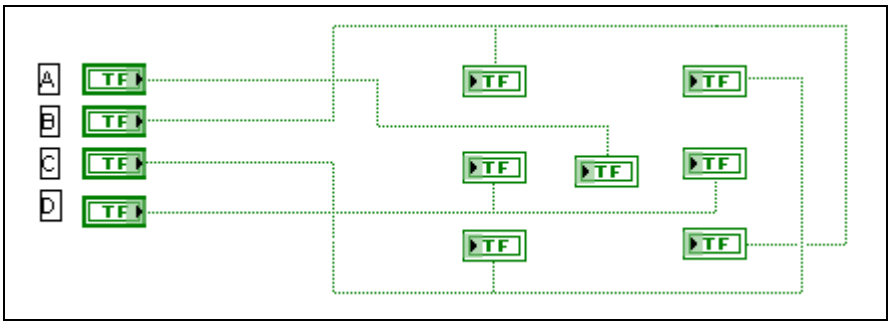


Рисунок 2-5. Диаграммная панель прибора, моделирующего игральную кость.

Запустите программу **Display.vi** и наблюдайте реализацию виртуальной игральной кости.

Счетчик по модулю 6 (или счетчик с коэффициентом пересчета 6)

Счетчиком по модулю 6 может являться любой счетчик с шестью особыми состояниями, повторяющимися последовательно друг за другом. Простейший счетчик такого типа можно создать, используя трехэлементный сдвиговый регистр, в котором выход последнего элемента инвертируется и подается на вход первого элемента.

Создайте новый виртуальный прибор. Расположите три светодиодных индикатора на лицевой панели. Они будут показывать состояния элементов сдвигового регистра под названиями Q1, Q2 и Q3. На панели диаграмм используйте сдвиговый регистр с тремя элементами, каждый из которых соединен с соответствующим светодиодным индикатором. Для замедления работы прибора вы можете использовать функцию ожидания **Wait**. Каждый раз, когда вызывается этот виртуальный прибор, возвращается следующая по порядку величина. Находясь на лицевой панели, вы можете отождествить входы/выходы с элементами коннекторной панели (правый верхний угол экрана). Сохраните этот прибор в качестве подпрограммы под названием **Rotate.vi**.

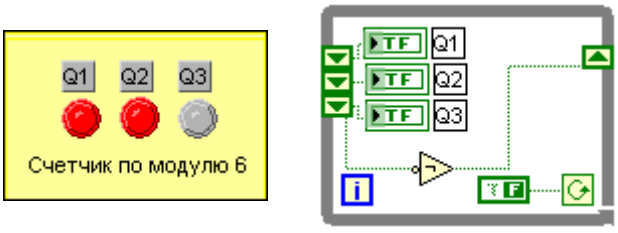


Рисунок 2-6. Лицевая и диаграммная панели прибора **Rotate.vi**.

Ниже показана таблица истинности для счетчика по модулю 6. Запустите программу семь раз, чтобы увидеть ее работу.

Таблица 2-2. Таблица истинности для счетчика по модулю 6.

Цикл	Q1	Q2	Q3
1	0	0	0
2	1	0	0
3	1	1	0
4	1	1	1
5	0	1	1
6	0	0	1
7	0	0	0

После шести отсчетов вид индикаторов повторяется, что и подтверждает, что это счетчик по модулю шесть.

Шифратор

Заранее не известно – какому входному сигналу соответствует определенный отсчет. Однако небольшая предусмотрительность делает этот выбор проще:

Таблица 2-3. Схема кодирования игральной кости.

#	Q1	Q2	Q3	Q1’	Q2’	Q3’
6	0	0	0	1	1	1
4	1	0	0	0	1	1
2	1	1	0	0	0	1
1	1	1	1	0	0	0
3	0	1	1	1	0	0
5	0	0	1	1	1	0

Например, каждый выходной сигнал имеет три единичных и три нулевых состояния. Один из таких выходных сигналов, например Q3, может означать нечетные состояния 1, 3, 5. Тогда другой, например Q2′, может означать набор состояний 4, 5, 6. После этого при помощи двух найденных соответствий можно легко декодировать два из четырех базовых рисунков. Два оставшихся рисунка можно декодировать особой комбинацией состояний счетчика. Для этой цели можно использовать инвертер и трехвходовый логический элемент И. Тогда событие НЕ 1

(базовый рисунок B) декодируется при помощи комбинации Q1 & Q2 & Q3, а оставшееся базовое состояние «6» декодируется с помощью Q1′ & Q2′ & Q3′.

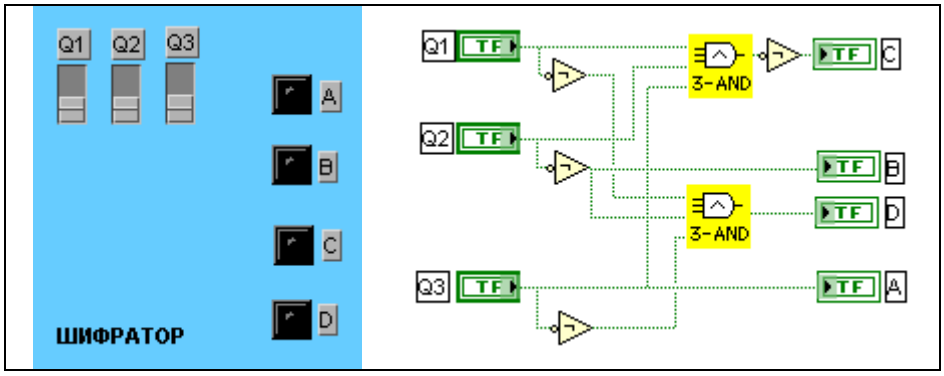


Рисунок 2-7. Лицевая и диаграммная панели прибора **Encode.vi**.

Шифратор можно создать, размещая на лицевой панели три логических индикатора и четыре светодиодных индикатора. Соединение выходов проводится таким образом, чтобы воплотить в схеме все то, что было сказано в предыдущем абзаце.

Виртуальная игральная кость

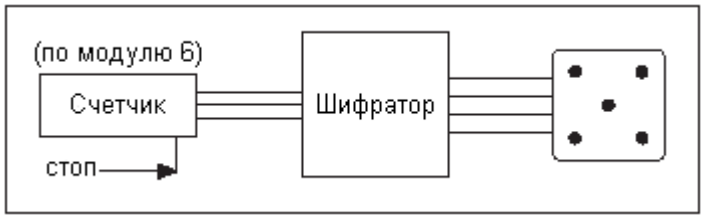


Рисунок 2-8. Схема реализации виртуальной игральной кости.

Чтобы заставить вращаться виртуальную игральную кость, используется быстрый циклический счетчик, последовательно пробегающий шесть состояний. Работа счетчика циклически повторяется до тех пор, пока на него не поступит команда остановки. Какое состояние будет на выходе счетчика в этот момент, то и считается результатом вращения. Частота циклов счетчика, большая 1 кГц, обеспечивает случайность процесса.

Виртуальный прибор, исполняющий функции шифратора, конвертирует три выходных сигнала счетчика в четыре сигнала, задающих базовые изображения. Те в свою очередь «зажигают» индикаторы на виртуальной игральной кости, создавая соответствующий выходной код.

Теперь можно привести простейший пример собрания воедино всех компонентов – счетчика, шифратора и графического представления. Это – виртуальный прибор **Dice.vi**. Подобно тому, как вы собираете электронные схемы из логических элементов, вентилей, переключателей

и отображающих элементов, пакет LabVIEW имитирует этот процесс, создавая сложные логические функции из простых.

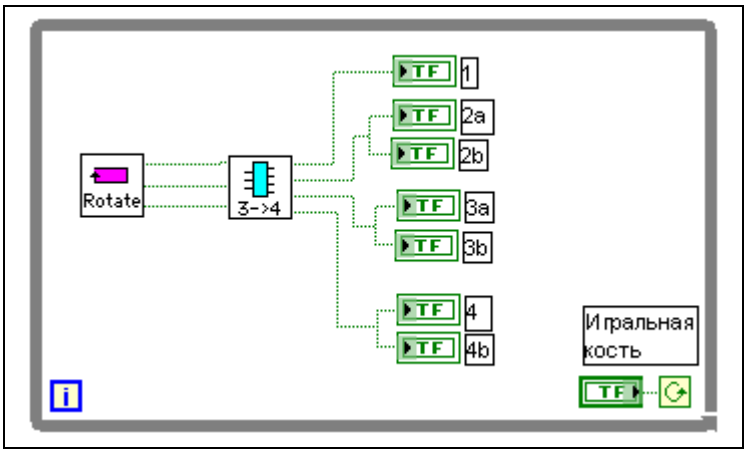


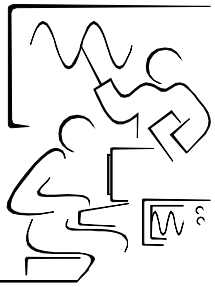
Рисунок 2-9. Диаграммная панель прибора **Dice.vi**. Обратите внимание на схожесть этого рисунка с предыдущим.

А сейчас нажмите переключатель на лицевой панели и посмотрите, как работает этот виртуальный прибор.

Состав библиотеки виртуальных приборов, использованных на занятии 2 (показан в порядке упоминания)

- **Display.vi** (светодиодные индикаторы для виртуальной кости)
- **Rotate.vi** (счетчик по модулю 6)
- **Encoder.vi** (преобразует коды счетчика в выходные коды)
- **3 AND.vi** (подпрограмма, используемая в **Encoder.vi**)
- **Dice.vi** (виртуальная игральная кость)

Занятие 3.
Операция суммирования по модулю 2 (двоичное сложение).



Пред тем, как детально излагать данный раздел, полезно посмотреть некоторые особенности операции суммирования по модулю 2 (двоичного сложения). Так же как и при сложении в десятичной системе, при добавлении 0 к числу, последнее не меняется: 0 + 0 = 0 и 1 + 0 = 1. Однако когда вы складываете 1 + 1 в двоичной системе, результат равен не 2 (такого символа не существует в двоичной системе счисления), а 10, то есть единица на месте двойки, а ноль на месте единицы. Если записать сложение вертикально, то как бы необходимо произносить вслух: «Один и один равно двум, пишем ноль, а единицу сдвигаем».

$$\begin{array}{r} 1 \\ +\ 1 \\ \hline 10 \end{array}$$

Рисунок 3-1. Однобитовое сложение.

Ниже представлена таблица истинности для однобитового сложения. В ней два входных столбца для каждого слагаемого – A1 и A2, и два выходных столбца, один для значения суммы в этом же разряде (Sum), а второй для смещенного бита (Carry):

Таблица 3-1. Таблица истинности для операции Сложение.

A1 + A2	= СУММА +	Перенос Разряда
0 0	0	0
0 1	1	0
1 0	1	0
1 1	0	1

Какой из базовых логических элементов может помочь вам осуществить такую операцию? Заметим, что если произвести операцию A1 исключающее ИЛИ A2, то на выходе получим значение суммы Sum в этом же разряде. Если же произвести операцию A1 И A2, то получим значение

смещенного бита. Так что реализация этой таблицы истинности для однобитового сложения при помощи LabVIEW имеет следующий вид:

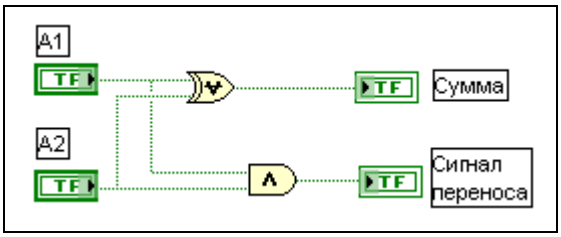


Рисунок 3-2. Полусумматор, построенный из логических элементов И и Исключающее ИЛИ.

Цифровой блок, построенный таким способом, называется (одноразрядным) сумматором с двумя входами или полусумматором. Термин «полусумматор» отображает тот факт, что пока такой блок генерирует сигнал для индикации переноса в соседний старший разряд, он не может принять сигнал переноса с сумматора младшего разряда.

«Полный» сумматор имеет три входа. Вдобавок к входам для двух слагаемых, у него есть вход сигнала переноса, который добавляет бит, перенесенный из предыдущего столбца, как это показано для среднего столбца в следующем примере:

```
  101
+ 101
 1010
```

Рисунок 3-3. Трехбитовое сложение.

Поэтому таблица истинности для 1-битового полного сумматора имеет три входа и, следовательно, восемь возможных состояний:

Таблица 3-2. Таблица истинности для сложения с учетом переноса бита из предыдущего разряда.

Бит из предыдущего разряда	A1	A2	Сумма	Перенос разряда
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Отметим, что все три входа полностью эквивалентны, то есть полный сумматор просто складывает состояния трех входов. Один из возможных способов построения 1-битового полного сумматора состоит в одновременном использовании двух полусумматоров:

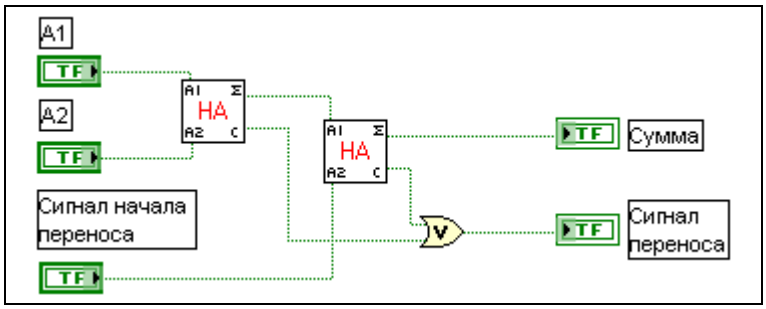


Рисунок 3-4. Полный сумматор на базу двух полусумматоров.

Отметим простоту схемы соединений при использовании полусумматоров.

Расширение сумматора

Комбинируя 1-битовые сумматоры, вы можете придумать устройство, которое бы складывало многобитовые двоичные числа. Каждый такой сумматор совершает процедуру сложения для одного «столбца» суммы чисел, таких как

$$\begin{array}{r} 1011 \\ +0010 \\ \hline 1101 \end{array}$$

Рисунок 3-5. Четырехбитовое сложение (11 + 2 = 13)

Например, 4-битовый сумматор можно создать при помощи LabVIEW следующим образом:

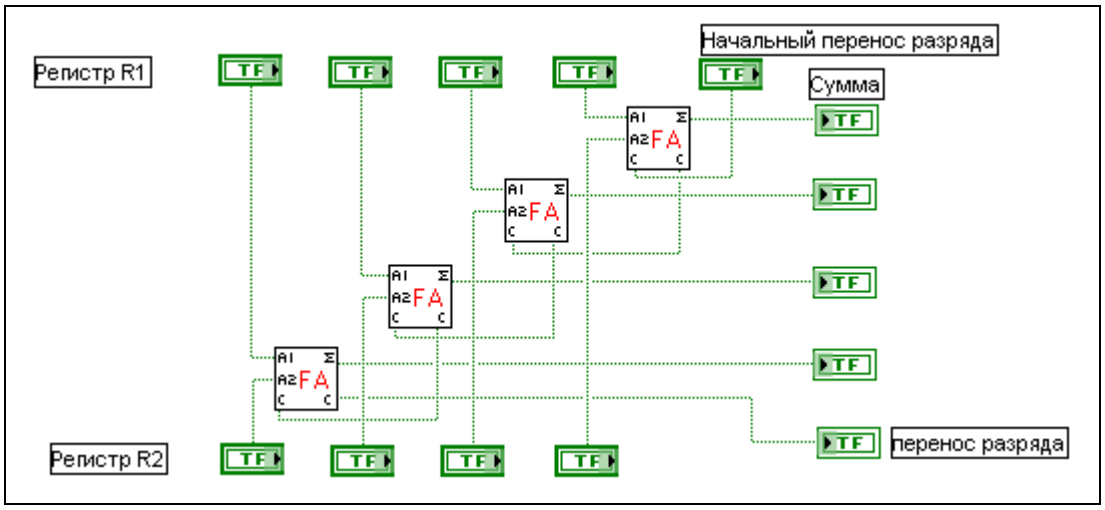


Рисунок 3-6. Диаграммная панель прибора, моделирующего 4-битовое сложение.

Отметим, что в данном виртуальном приборе используются четыре однобитовых полных сумматора. Если с помощью этой схемы вы хотите складывать только 4-битовые числа, то сумматор самого младшего разряда может быть полусумматором. Использование же всех полных сумматоров позволяет схеме 4-битового сумматора иметь кроме двух входов для 4-битовых чисел также и вход для сигнала переноса. Запустите **Four-bit Adder1.vi** и наблюдайте сложение двух 4-битовых числа. В виртуальном приборе использовались две подпрограммы: **Full Adder.vi**, показанная на рисунке 3-4, и and **Half Adder.vi**, показанная на рисунке 3-2.

Как вы уже, наверное, заметили, схема соединения элементов прибора достаточно сложное и будет еще сложнее, если вы расширите сумматор для работы с числами с большим количеством бит. Используя структуру LabVIEW – цикл с фиксированным числом итераций – вместе со сдвиговым регистром, можно значительно упростить схему соединений:

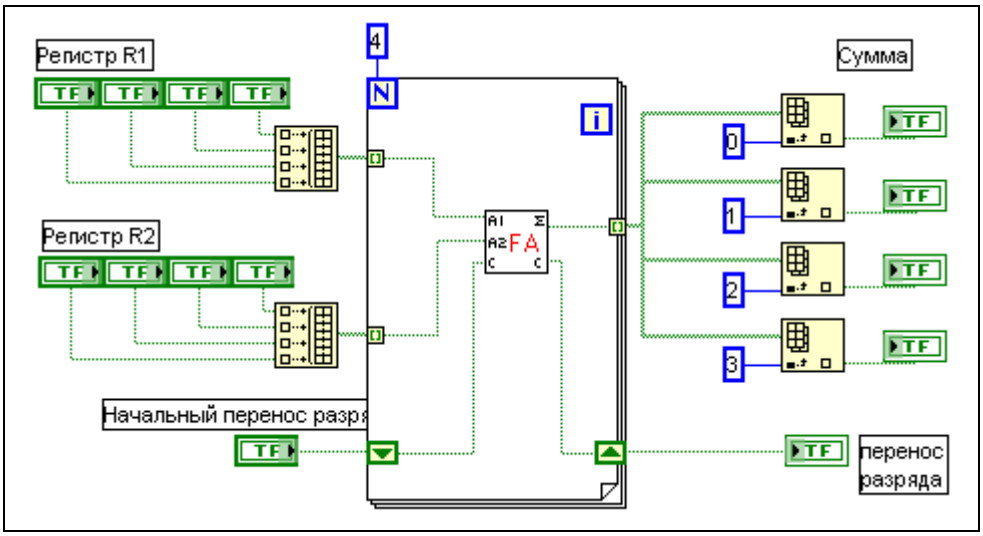


Рисунок 3-7. 4-битовое сложение с использованием структуры LabVIEW – «Массивы» (**Four-Bit Adder2.vi**).

Отметим, что перед тем, как попасть в цикл, четыре независимых бита объединяются в матрицу. Число итераций цикла – четыре. На каждой итерации добавляется пара битов, начиная с самого младшего двоичного разряда. На первой итерации на вход сигнала переноса 1-битового сумматора поступает бит, введенный с лицевой панели. При следующих итерациях происходит последовательный перенос битов с предыдущей итерации. Запустите обе версии виртуального прибора и установите, что их работа идентична.

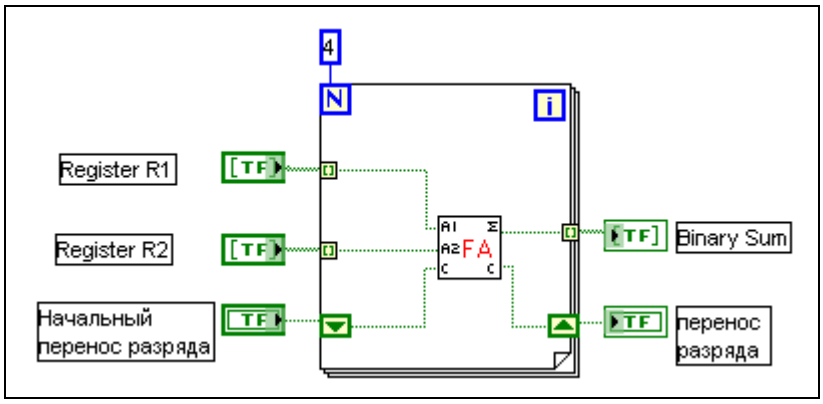


Рисунок 3-8. 4-битовое сложение с использованием входных и выходного массивов.

Существует также третья версия предыдущего виртуального прибора, называемая **Four-bit Adder3.vi**, идентичная изображенной на рисунке 3-7 за исключением того, что входы и выходы отображаются в виде булевых матриц. Отметим, что в булевых матрицах самый младший двоичный разряд располагается слева, а самый старший разряд – справа. Эта версия виртуального прибора сконфигурирована в виде подпрограммы,

поэтому вы можете скомбинировать два таких прибора для построения 8-битового сумматора. Отметим, что каждое 8-битовое (1-байтовое) слагаемое разделяется на два «куска» – полубайта. После этого, два «самых младших» полубайта отправляются на один 4-битовый сумматор, а два «самых старших» полубайта поступают на другой 4-битовый сумматор.

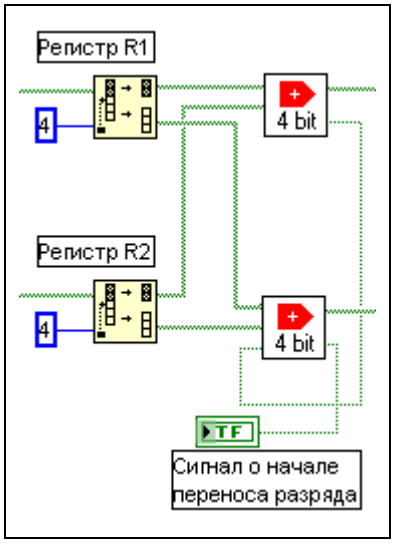


Рисунок 3-9. 8-битовый сумматор, построенный на базе двух 4-битовых сумматоров.

Двоично-десятичное представление (ДДП)

Не все арифметические операции с числами совершаются с помощью прямой перекодировки в двоичное представление. Наряду с ним используется двоично-десятичное представление (ДДП). При ДДП любое десятичное число раздельно кодируется четырьмя битами:

Таблица 3-3. ДДП представление чисел от 0 до 9.

Число	ДДП	Число	ДДП
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

ДДП можно рассматривать как подмножество полной двоичной системы счисления, в котором используются только состояния с 0000 до 1001 (с 0 до 9). Например,

$$42_{10} = 0100\ 0010_{ДДП}$$

Обратите внимание, что этот результат отличается от чисто двоичного представления, которое в данном случае привело бы к результату

$$42_{10} = 00101010_2$$

Очевидно, что ДДП содержит лишние биты, потому что существует целое множество 4-битовых наборов, которые не используются для кодирования десятичных чисел. Такая бесполезная трата становится более заметной для больших чисел. Двух байтов (16 бит) достаточно для кодирования любых десятичных чисел в диапазоне 0-65535 при использовании двоичного представления, в то время как те же два байта охватят числа в диапазоне 0-9999 при использовании ДДП. Преимущество ДДП заключается в том, что преобразованные данные имеют десятичное представление.

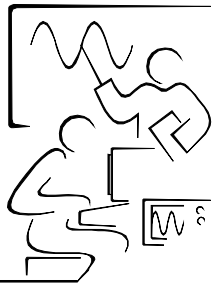
Задание повышенной сложности

Придумайте ДДП шифратор, преобразующий числа в диапазоне 0-9 на входе в 4-битовое двоично-десятичное представление на выходе. Создайте ДДП дешифратор, который бы преобразовывал обратно. Соберите одноразрядный ДДП сумматор.

Состав библиотеки виртуальных приборов, использованных на занятии 3 (показан в порядке упоминания)

- **Half Adder.vi** (1-битовое суммирование)
- **Full Adder.vi** (1-битовое суммирование с входным сигналом переноса)
- **Four-bit Adder1.vi** (суммирование двух 4-битовых числа с входным сигналом переноса)
- **Four-bit Adder2.vi** (упрощенный вариант)
- **Four-bit Adder3.vi** (используется булевы матрицы для ввода и вывода)
- **Eight-bit Adder.vi** (используются два 4-битовых сумматора)

Занятие 4.
Оперативная память:
Регистр D-типа.



На первых трех занятиях мы изучали комбинационные схемы, в которых входное состояние полностью определяло выходное. Поведение таких схем не зависит от предыстории, или по-другому, от того, каким образом вы создали текущее состояние. Это значит, что в подобные схемы нельзя встроить функцию запоминания. Большинство цифровых операций являются последовательными, то есть событие Б должно возникнуть после события А. Более того, в цифровых компьютерах события не только последовательные, но и синхронизированные с помощью некоторого внешнего таймера. Синхронные логические устройства – это устройства, в которых выходное состояние изменяется только после получения сигнала таймера. На следующих нескольких занятиях мы увидим, как использование дополнительных синхронных логических устройств позволяет создать память в цифровых схемах, делая возможным конструирование многих интересных цифровых устройств.

Одной из простейших запоминающих схем является «защелка» данных или регистр D-типа. Когда на синхронизирующий вход такого устройства приходит сигнал, оно запоминает состояние на своем входе и передает это состояние на выход. Даже в том случае, если входной сигнал изменяется, выходное состояние остается неизменным до тех пор, пока не поступит запрос на обновление. Обычно вход регистра D-типа обозначают буквой D, а выход буквой Q. Команда обновления поступает через синхронизирующий вход в форме изменения уровня сигнала от высокого к низкому или от низкого к высокому. Такие устройства называются устройствами, тактируемые фронтом (сигнала), в которых состояние выхода следует за входным состоянием, в то время, когда уровень синхронизирующего сигнала высокий.

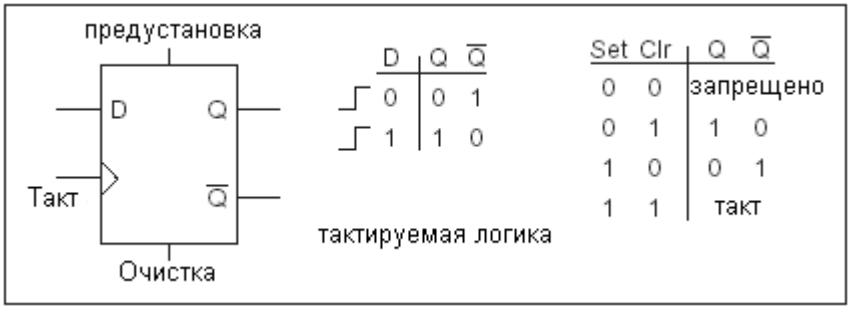


Рисунок 4-1. Обозначение регистра D-типа и его таблица истинности.

Когда получен синхронизирующий сигнал, данные на входе D передаются на выход Q или $\overline{Q}$. Таблица истинности для D-регистра, запускаемого фронтом сигнала, показана справа в схематическом виде. Некоторые из D-регистров имеют также вход установки в исходное состояние и вход сигнала сброса, которые позволяют состоянию выхода быть высоким или низким независимо от синхронизирующего сигнала. В обычном режиме работы состояния этих двух входов поддерживаются на высоком уровне, так что в работу синхронной логики они не вмешиваются. Однако выходы Q и $\overline{Q}$ можно установить в известное состояние, используя входы установки в исходное состояние и сигнала сброса, когда синхронная логика не задействована.

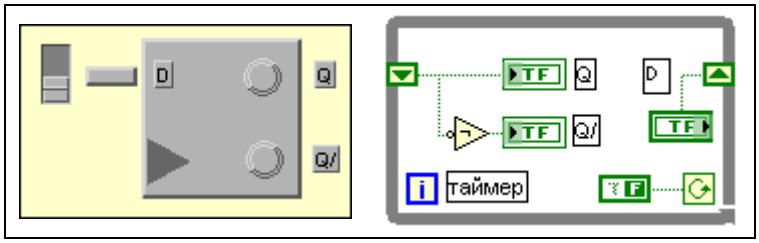


Рисунок 4-2. Модель регистра D-типа.

В пакете LabVIEW вы можете смоделировать регистры D-типа, используя сдвиговые регистры в цикле по условию. Блок со стрелкой вверх – это вход D, а блок со стрелкой вниз – это выход Q. Булево дополнение получается соединением инвертора с выходом Q. Синхронизирующий вход аналогичен индексу цикла [i]. Для установки состояния выхода в исходное состояние или его обнуления можно использовать булевскую константу вне цикла. В виртуальном приборе **D Latch.vi**, показанном выше, управляющий элемент цикла ни с чем не соединен. Это сделано для того, чтобы программа выполнялась только один раз, когда ее вызывают.

Сдвиговые регистры

В цифровой электронике под сдвиговым регистром понимается набор 1-битовых ячеек памяти, в котором каждый бит обновляется под действием синхронизирующего импульса. При этом состояние данной ячейки заменяется состоянием соседней.

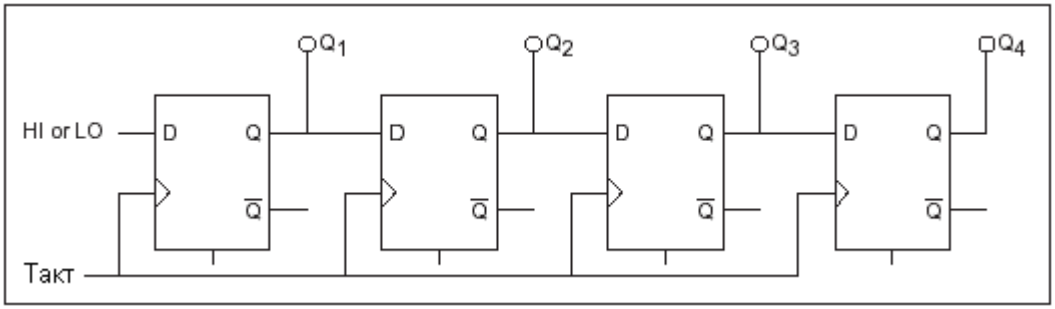


Рисунок 4-3. 4-битовый сдвиговый регистр.

Ячейки на концах имеют только по одному соседу. Состояние входа D первой ячейки определяется внешним источником (Высокое или Низкое), а выходное состояние последней ячейки Q_4 определяет выход сдвигового регистра. Здесь представлен пример 4-битного сдвигового регистра, у которого начальное выходное состояние – [0000] и входное – [1]:

Тактовый цикл	Q1	Q2	Q3	Q4
n	0	0	0	0
n+1	1	0	0	0
n+2	1	1	0	0
n+3	1	1	1	0
n+4	1	1	1	1

Чтобы смоделировать сдвиговый регистр D-типа, к обыкновенному D-регистру, смоделированному выше при помощи LabVIEW, добавляются некоторые элементы. В качестве примера здесь приведен 4-битный регистр. Последовательность, изображенную выше, реализует виртуальный прибор **Shift.vi**.

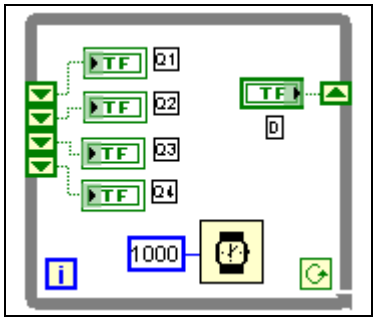


Рисунок 4-4. Блок-схема 8-битового сдвигового регистра.

Не составляет труда добавить элементы, чтобы имитировать сдвиговые регистры большой размерности. Следующий виртуальный прибор **Bucket.vi**, моделирует «пожарную цепочку», в которой бит, поступивший

на вход D, двигается дальше до конца по цепочке, где выходит и теряется после прохождения Q_8.

Рисунок 4-5. Лицевая панель прибора, моделирующего 8-битовый сдвиговый регистр.

Задание повышенной сложности

Придумайте виртуальный прибор, в котором после того, как бит прошел последнюю ячейку, новый бит поступает на вход D, и процесс повторяется до бесконечности.

Кольцевые счетчики

Если выходное состояние сдвигового регистра подать назад на вход, то, в конце концов, после *n* тактирующих циклов, состояние на выходе повторится. В этом случае сдвиговый регистр становится счетчиком. Название «кольцевой счетчик» отображает тот факт, что выходной бит возвращается назад на вход. В простейшем 4-битовом счетчике состояние последнего выхода Q_4 возвращается прямо на вход сдвигового регистра D.

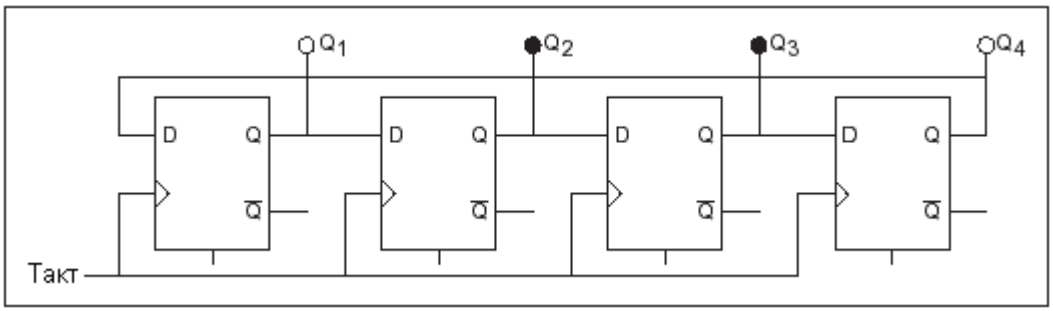


Рисунок 4-6. 4-битовый кольцевой счетчик, построенный при помощи микросхем.

В примере, показанном выше, выходы были предварительно установлены в состояние [0110]. Запустите виртуальный прибор **Rotate.vi**. Наблюдайте, что выходное состояние циклически меняется, начиная от [0110], через [0011], [1001], [1100] и снова до [0110]. Возвращение к исходному состоянию занимает четыре цикла, следовательно, это кольцевой счетчик по модулю 4. В случае, когда эти четыре выхода соединены с элементами, управляющими током шагового двигателя, каждое изменение состояний

выхода приводит к повороту ротора двигателя на один шаг. Двигатель с 400-шаговым разрешением будет поворачиваться на 0.9 градуса каждый раз, когда вызывается счетчик. Немного изменив кольцевой счетчик, мы получим счетчик с инверсной связью – счетчик Джонсона. В таком счетчике вместо состояния выхода берется его булево дополнение $\overline{Q}$ и подается на вход. Измените виртуальный прибор **Rotate.vi**, чтобы он выполнял такую функцию, и сохраните его под именем **Switch Tail Ring Counter.vi**.

Чему равен коэффициент пересчета счетчика с инверсной связью?

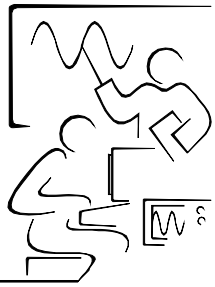
Кольцевые счетчики часто используются в случаях, когда необходимо повторение событий с постоянной скоростью. Запустите виртуальный прибор **Billboard.vi**, показанный ниже. Он моделирует бегущие друг за другом огоньки.

Используя ползунок, вы можете установить необходимую скорость вращения, а 16 булевых констант на диаграммной панели определяют последовательность изменения огоньков.

Состав библиотеки виртуальных приборов, использованных на занятии 4 (показан в порядке упоминания)

- **D Latch.vi** (моделирование регистра D-типа)
- **Shift.vi** (4-битовый сдвиговый регистр)
- **Bucket.vi** (моделирование 8-битового сдвигового регистра)
- **Rotate.vi** (4-битовый кольцевой счетчик)
- **Billboard.vi** (16-битовый кольцевой счетчик в роли генератора бегущих огоньков)

Занятие 5. Генераторы псевдослучайных чисел.



На предыдущем занятии были представлены простейшие кольцевые счетчики. Они рассматривались как один из способов построения счетчиков по модулю n. На данном занятии будет рассматриваться работа обратной связи, когда комбинация более поздних стадий процесса подается назад на входной логический элемент. Если выбирается правильная комбинация, тогда выходной сигнал имеет максимальную длину (то есть коэффициент пересчета счетчика равен 2^N –1). Для 8-битового счетчика N = 8, тогда 2^N –1 = 255. Такие схемы, часто называемые генераторами псевдослучайных чисел (ГПСЧ), имеют несколько интересных особенностей. Оказывается, что сформированная последовательность случайна в небольшом диапазоне, но, фактически, повторяется через 2^N –1 циклов. Более того, данный набор появляется только однажды среди всех 2^N –1 наборов, составляющих последовательность.

Генераторы псевдослучайных последовательностей и чисел находят широкое применение в компьютерной безопасности, криптографии, тестировании аудиосистем, проверке ошибок в двоичном коде и секретной связи.

Шестибитовый ГПСЧ

В следующей схеме выходные сигналы пятого и шестого регистра D-типа поступают на вход логического блока НЕ-Исключающее ИЛИ, выходное состояние которого подается на сдвиговый регистр. Предполагается, что состояния всех выходов – нулевые.

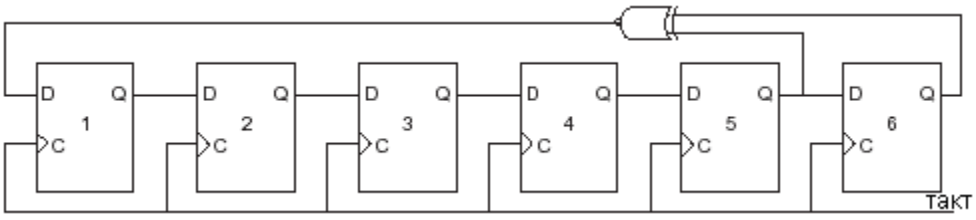


Рисунок 5-1. 6-битовый ГПСЧ, построенный на базе шести регистров D-типа и логического элемента Исключающее ИЛИ.

Когда Q5 и Q6 равны нулю, то выход блока НЕ-Исключающее ИЛИ (см. занятие 1) равен 1. Эта величина подается на сдвиговый регистр – вход

D1. По команде тактового генератора все биты сдвигаются вправо. Следовательно, начальное значение (000000) становится (100000). Нетрудно мысленно пробежать несколько циклов, чтобы увидеть, что значения выходов Q1...Q6 принимают следующую последовательность:

(000000)

(100000)

(110000)

(111000)

После 63 циклов последовательность возвращается к исходному состоянию (000000). Такую схему легко смоделировать при помощи LabVIEW:

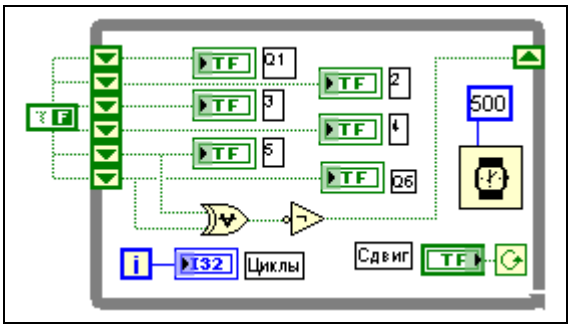


Рисунок 5-2. Моделирование 6-битового ГПСЧ.

В цикле По Условию располагается шестиэлементный сдвиговый регистр. Элемент Исключающее ИЛИ и инвертор используются для создания элемента НЕ-Исключающее ИЛИ, входы которого присоединяются к Q5 и Q6. Индекс отслеживает число циклов, а задержка на 500мс позволяет наблюдать работу генератора. Когда запустите этот виртуальный прибор – **6PRNG.vi** – обратите внимание, что циклы 0 и 63 имеют одинаковые состояния (все биты – нулевые).

8-битовый генератор псевдослучайных последовательностей

В 8-битовом генераторе случайных последовательностей используются выходы Q4, Q5, Q6 и Q8, которые поступают на логический элемент НЕ-Исключающее ИЛИ. После этого формируется числовая последовательность максимальной длины $2^N - 1 = 255$.

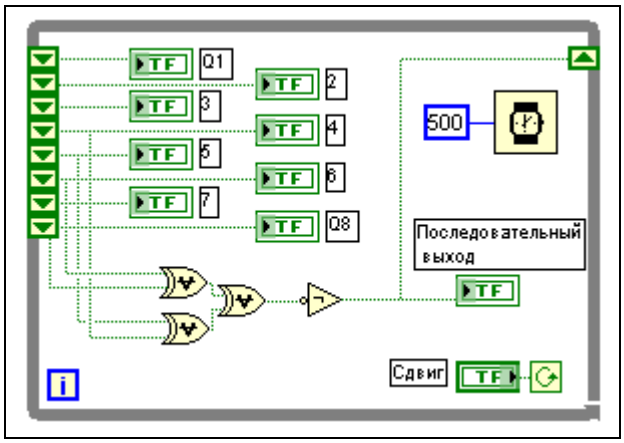


Рисунок 5-3. Модель 8-битового ГПСЧ.

Как и в предыдущем примере, выходное состояние можно параллельно наблюдать на восьми светодиодных индикаторах. Дополнительно, псевдослучайная последовательность, состоящая из единиц и нулей, выводится на последовательный выход.

Многие цифровые схемы необходимо тестировать сигналами, состоящими из всевозможных комбинаций нулей и единиц. Такой сигнал обеспечивает случайная булева последовательность, снимаемая с последовательного выхода. В такой конфигурации схему называют генератором псевдослучайных битовых последовательностей (ГПСБП). На светодиодном индикаторе лицевой панели показанного выше виртуального прибора **PRBS0.vi** можно увидеть такую последовательность.

Рисунок 5-4. Лицевая панель прибора, моделирующего 8-битовый ГПСБП.

Наиболее удобный способ увидеть битовую последовательность заключается в построении графика. Логические сигналы конвертируются в численные величины – 0 или 1 и затем отображаются на графическом индикаторе виртуального прибора. Здесь приведен пример отображения первых пятидесяти битов, сгенерированных виртуальным прибором **PRBS.vi**, в виде такого графика.

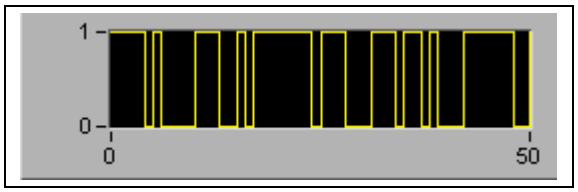


Рисунок 5-5. Последовательный выход ГПСБП.

Сигналы, сгенерированные ГПСБП, используются для тестирования лазеров, применяемых в системах связи. Это необходимо из-за того, что под действием определенной последовательности нулей и единиц лазер может заблокироваться. Также это может произойти из-за превышения уровнем логического сигнала допустимых пределов. Излучение лазера детектируется фотодиодом, затем конвертируется в цифровой сигнал и подается на один из входов компаратора. В то же время, логическая последовательность, управляющая генерацией лазера, поступает на другой вход компаратора. Таким образом, любые ошибки при передаче или из-за блокировки могут быть выявлены.

Теперь легко проверить, что битовая последовательность повторяется точно через 255 циклов. С этой целью в приборе **PRBS2.vi** последовательности изображаются на двух графиках. Изменяя диапазон второго графика с 255 до 305, можно наблюдать повторяющуюся структуру генерации ГПСБП.

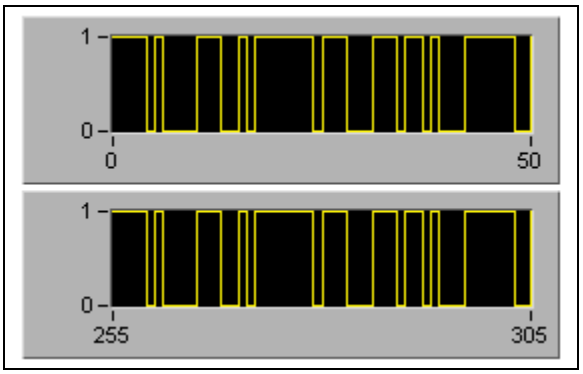


Рисунок 5-6. Сравнение первых 50 битов, сгенерированных ГПСБП, с битами от 255 до 305.

8-битовый генератор псевдослучайных чисел

Дополнение в виде аналого-цифрового преобразователя позволяет конвертировать псевдослучайную цифровую последовательность на параллельных выходах генератора в число. При двоичном преобразовании биты параллельных выходов Q1...Q8 имеют весовые коэффициенты 1, 2, 4, 8, 16, 32, 64, и 128. В следующем примере численные величины отображаются на трехразрядном индикаторе и на графике лицевой панели.

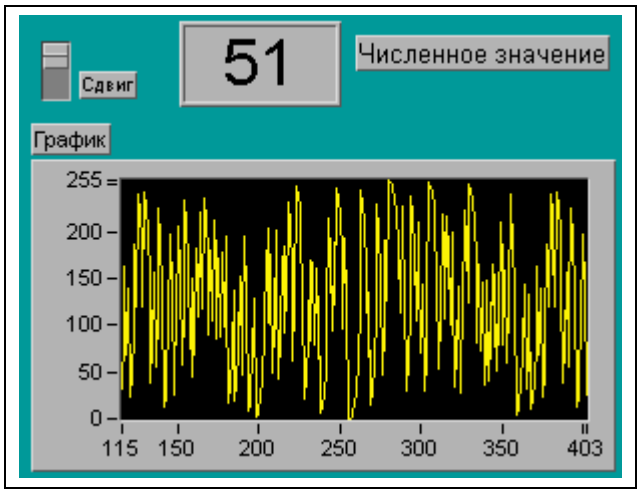


Рисунок 5-7. Числовой выходной сигнал 8-битового ГПСЧ.

Запустив программу **PRNG.vi**, можно наблюдать последовательность чисел, сгенерированную ГПСЧ. В этой последовательности присутствуют все числа от 0 до 254. При более внимательном рассмотрении видно, что каждое число появляется лишь однажды. Является ли появление определенной последовательности случайным?

Следующая диаграммная панель показывает реализацию при помощи LabVIEW 8-битового ГПСЧ. Отметим, что блок DAC служит для отображения численных значений булевых параллельных выходов.

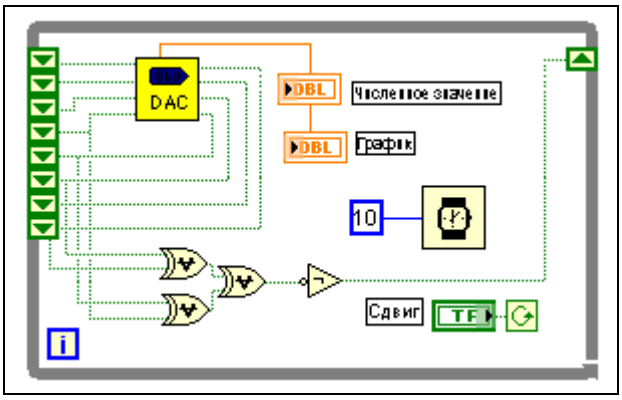


Рисунок 5-8. Модель 8-битового ГПСЧ с графическим индикатором на выходе.

Графический формат очень удобен для отображения аналоговой последовательности. В коротком диапазоне (10-30) чисел появление определенного выходного сигнала случайно. Оно и является случайным с математической точки зрения. Если такой сигнал преобразовать с аналоговый (например, в напряжение), то его осциллограмма будет представлять собой белый шум. Использование ГПСЧ в тестировании аудио схем обусловлено тем, что такой шум повторяется через $2^N - 1$

циклов. Усилители наподобие цифровых вентилей могут иметь кратковременную, но никак не долговременную память. Для тестирования аналоговых схем применяются сигналы, снимаемые с аналоговых выходов ГПСЧ. Сигнал, прошедший через схему сравнивается с уровнями, рассчитанными по последовательности ГПСЧ. Любое отклонение (ошибки) может свидетельствовать о неполадках в тестируемой схеме.

Шифрование цифровых данных

Чаще всего передача данных проходит в форме обмена символами ASCII. Добавление контрольного (двоичного) разряда четности к 7-битовому ASCII коду приводит к 8-битовому дискретному числу. Для сохранения секретности в банкоматах, электронных замках дверей, компьютерных паролях используются данные в форме ASCII. При этом применяется и некоторая форма шифрования.

Для шифрования ASCII данных весьма полезным оказывается 8-битовый ГПСЧ. До сих пор для начала генерации ГПСЧ использовались начальные условия по умолчанию для сдвигового регистра. Реально же последовательность может начинаться с любой величины за исключением неразрешенного состояния 11111111. Предположим, что начальное значение было 01111010 или 122 в численном виде или $7A в шестнадцатеричном виде или z в ASCII. Тогда последовательность с таким начальным номером воспроизводит себя обычным образом, повторяясь через 255 циклов. Ниже показано представление сгенерированных величин в виде булевой матрицы, начиная с некоторого индекса (7) и еще шесть следующих значений. Отметим, что через 255 циклов плюс этот индекс (7 + 255 = 262) последовательности совпадут, и, следовательно, они предсказуемы.

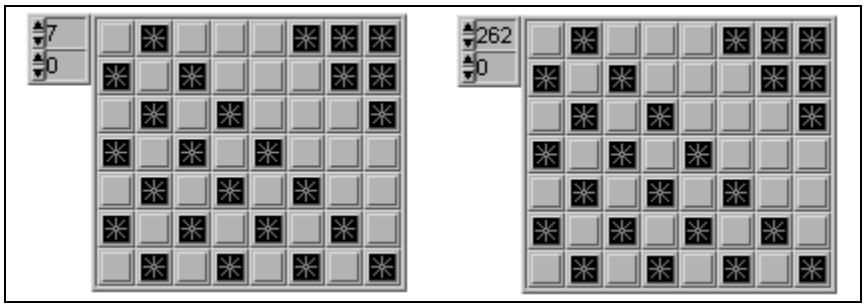


Рисунок 5-9. Сравнение первых 8-битовых чисел, сгенерированных 8-битовым ГПСЧ, с числами, сгенерированными в циклах от 262 до 268.

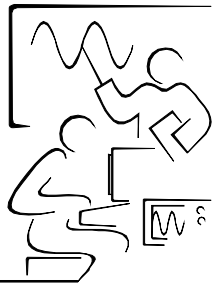
Предположим, что мы используем PIN или пароль в форме уникального численного кода N. Генератор псевдослучайных чисел устанавливается в начальное состояние символом ASCII кода. После этого пробегая N циклов, ГПСЧ конвертирует этот символ в зашифрованный. Воспользуемся предыдущим примером. Если PIN код равен 257, то буква

“z” будет зашифрована как “X”. Следовательно, для каждого символа в послании будет сформирован новый символ. Получатель, зная алгоритм шифрования и PIN код, сможет расшифровать это послание.

Состав библиотеки виртуальных приборов, использованных на занятии 5 (показан в порядке упоминания)

- **6PRNG.vi** (6-битовый ГПСЧ)
- **PRBS0.vi** (генератор псевдослучайных битовых последовательностей)
- **PRBS.vi** (8-битовый ГПСБП с последовательным выходом на график)
- **PRNG.vi** (с графическим индикатором выхода)
- **PRNG7.vi** (8-битовый ГПСЧ с выходами в виде матрицы)
- **DAC8.vi** (подпрограмма 8-битового АЦП)

Занятие 6.
JK-триггер типа «ведущий-ведомый».



Одним из наиболее важных тактируемых логических устройств является JK-триггер типа «ведущий-ведомый». В отличие от регистра D-типа, который «помнит» только до тех пор, пока не пришел другой тактовый импульс, JK-триггер обладает настоящей памятью. Когда напряжение на входах J и K низкого уровня, состояния выходов Q и $\overline{Q}$ при тактировании остаются неизменными. Таким образом, информация может быть помещена в выходном бите и храниться до тех пор, пока не будет запрошена в будущем. Если на входах J и K действуют противоположные сигналы (0,1) или (1,0), то с приходом очередного тактового импульса выход Q воспроизведет значениеJ-входа, то есть будет низкого или высокого уровня. Фактически, помещая между входами J и K инвертер, мы получим регистр D-типа. Принципиальная схема JK-триггера и его таблица истинности показаны ниже. Отметим, что JK-триггер может быть установлен в определенное (исходное) состояние прямым вводом определенных сигналов без использования тактируемой логики (прямая логика).

Рисунок 6-1. Схематическое изображение JK триггера и его таблица истинности.

Первая строка таблицы истинности в случае синхронной логики описывает состояние памяти, а другие две комбинации описывают фиксированные состояния. Новое здесь только то, что существует четвертая комбинация (1,1), которая порождает переключаемое состояние. С приходом тактового импульса выходное состояние меняется с 1 на 0 или с 0 на 1. Эту дополнительную функцию часто называют переключателем битов, а триггер (когда состояния J и K входов поддерживаются Высокими) – триггером со счетным входом. Поскольку происходит одно переключение

за один цикл, необходимо два тактовых импульса для возращения выхода в исходное состояние. Запустите программу **Binary1.vi** и наблюдайте работу триггера со счетным входом.

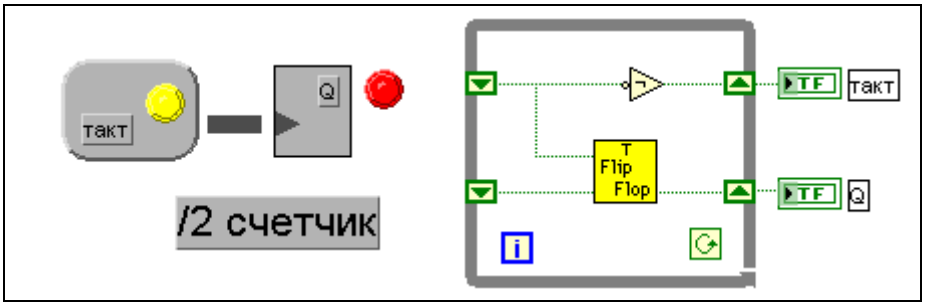


Рисунок 6-2. Модель счетчика-делителя на 2, с использованием подпрограммы T Flip-Flop.

Каждый раз, когда нажимается кнопка Run, тактовый импульс изменяет состояние с Низкого на Высокое или наоборот.

Сколько раз необходимо нажать кнопку Run, чтобы циклически поменять выходной бит – Низкий – Высокий – Низкий?

Может быть легче для более правильного наблюдения использовать кнопку Run Continuously. Так как для повторения выходного состояния необходимы два тактирующих импульса, триггер со счетным входом делит тактовую частоту на 2 и поэтому называется двоичным счетчиком с делением на два.

В пакете LabVIEW (см. панель диаграмм и подпрограмму T flip-flop) триггер со счетным входом моделируется при помощи структуры Варианта, помещенной в цикл по условию. Верхний сдвиговый регистр и инвертер моделируют генератор тактовых импульсов.

Если выход одного триггера со счетным входом использовать как синхронизирующий вход для второго триггера со счетным входом, то результирующая частота сигнала на выходе пары триггеров будет поделена на 4. Запустите виртуальный прибор **Binary2.vi**.

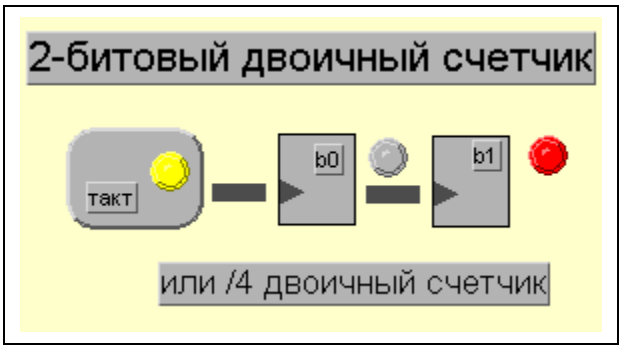


Рисунок 6-3. Моделирование счетчика-делителя на 4.

Если выход первого триггера брать с весом 1, а второго с весом 2, то соответствующие величины в десятичной системе счисления при тактировании образуют следующую последовательность: 0,1,2,3, 0,1,2,3, 0,1,2,3, и т.д. Это двоичный счетчик по модулю 4. Обратите внимание на то, каким образом на диаграммной панели выход первого триггера складывается с тактирующим сигналом, чтобы сформировать входной сигнал для второго триггера.

Двоичные счетчики

Двоичные счетчики создаются из J-K триггеров подачей на (J,K) входы всех триггеров логической единицы (уровень сигнала Высокий) и подключением выхода каждого предыдущего триггера к тактовому входу последующего. Тактирующий сигнал поступает в цепочку через вход синхронизации первого триггера. Далее результирующий сигнал двигается по цепочке.

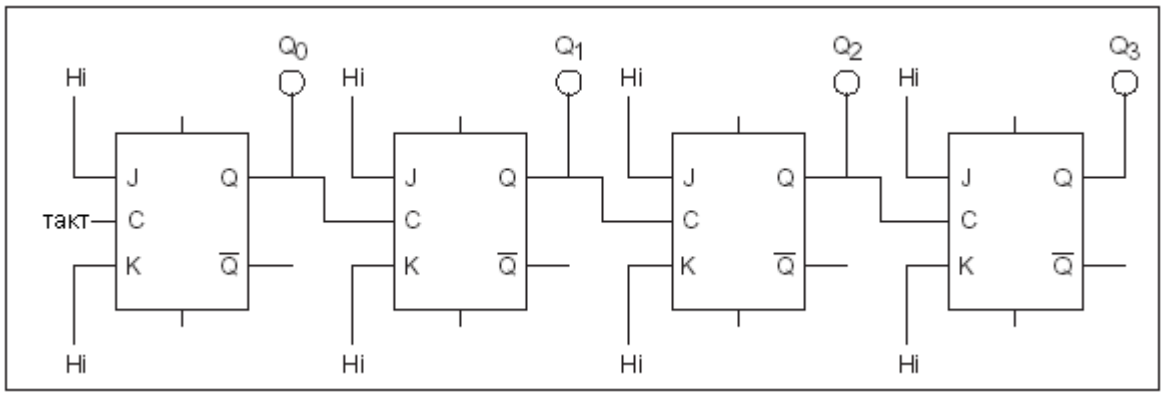


Рисунок 6-4. 4-битовый счетчик, построенный на основе JK триггеров.

В такой конфигурации частота тактирующего сигнала делится на 2 каждый раз, когда сигнал проходит через JK-триггер. Четыре последовательно включенных триггера делят на 2^4 или 16.

Запустите виртуальный прибор **Binary4.vi**, моделирующий представленный двоичный счетчик. Нажимая на кнопку Run, наблюдайте работу счетчика-делителя на 16. Четыре двоичных состояния (Q_3, Q_2, Q_1, Q_0) изображаются на светодиодных индикаторах, а эквивалентное значение в десятичной форме изображено в виде числа на лицевой панели. Кроме этого, на отдельных графиках представлены четыре временные диаграммы для выходов $Q_0 – Q_3$.

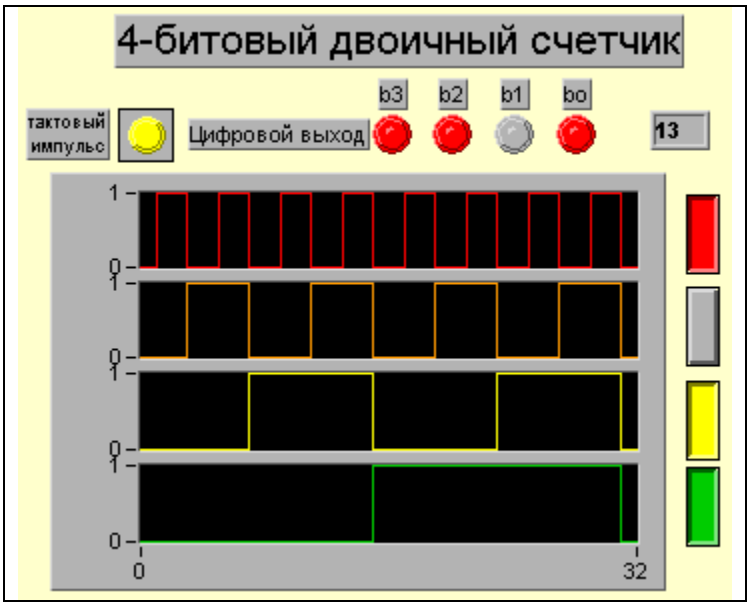


Рисунок 6-5. Моделирование 4-битового двоичного счетчика.

Наблюдайте последовательность и заполните следующую таблицу истинности:

Таблица 6-1. 4-битовая двоичная последовательность, сгенерированная счетчиком, и эквивалентная десятичная величина.

Тактовый цикл	Q_3	Q_2	Q_1	Q_0	Десятичный эквивалент
0	0	0	0	0	0
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16	1	1	1	1	15

Заполненная таблица отобразит все двоичные комбинации для 4-битового двоичного счетчика. Если выходам Q_0, Q_1, Q_2, и Q_3 присвоить весовые коэффициенты 2^0, 2^1, 2^2 и 2^3, то все числа 0–15 в двоичном представлении можно отобразить на четырех выходах. Посмотрите на диаграммную панель и обратите внимание, каким образом вычисляются эквивалентные десятичные величины.

В шестнадцатеричной системе счисления 16 состояний (0-15) обозначаются как 0…9 и A…F. Такое представление более компактно, и его проще запомнить, чем длинные комбинации битов двоичного представления. Биты с большей длиной разбиваются на группы из четырех битов, называемых полубайтом. Каждый полубайт кодируется как один шестнадцатеричный символ.

Например, 16-битовое число 1101 0111 0011 1100 кодируется в HEX как $D73C.

8-битовый двоичный счетчик

Для данных с более высокими разрядами применяется логическое расширение 4-битового двоичного счетчика. Встроенные контроллеры используют внутреннюю 8-битовую шину данных, а современные процессоры используют 16- и 32-битовые тракты прохождения данных. Программа **Binary8.vi** наглядно демонстрирует двоичную счетную последовательность в виде байта на восьми светодиодных индикаторах и 8-битовую временную диаграмму. Запустите этот прибор в непрерывном режиме для наблюдения двоичных чисел от 0 до 255. Временная диаграмма показывает, что частота сигнала на каждой стадии в два раза меньше, чем на предыдущей. Выходные частоты сигналов на стадиях Q_0...Q_7 – f/2, f/4, f/8, f/16, f/32, f/64, f/128 и f/256. Здесь f – тактовая частота.

Для различных операций двоичные счетчики необходимо сбрасывать в состояние 0 (все биты 0) или устанавливать в состояние 1 (все биты 1). Таблица истинности для JK-триггера, показанная выше, имеет прямые входы, обеспечивающие такую возможность (прямая логика). Синхронная (тактируемая) логика активизируется, когда входы сигналов установки в "0" и "1" имеют Высокий уровень. Низкий уровень сигнала на входе установки в "1" или "0" заставляет выход триггера становится 1 или 0 соответственно. Эти операции взаимно исключающие, следовательно, состояние (00) запрещено. Виртуальный прибор **Bin8_Reset.vi** обеспечивает функцию установки в исходное состояние 8-битового двоичного счетчика. Запустите эту программу в непрерывном режиме. Нажатием на кнопку Reset счетчик устанавливается в исходное состояние. Эта операция применяется для счетчиков с нечетным коэффициентом пересчета и в аналого-цифровых преобразователях.

Задание повышенной сложности

Придумайте двузначный двоичный счетчик, который бы считал от 0 до 99.

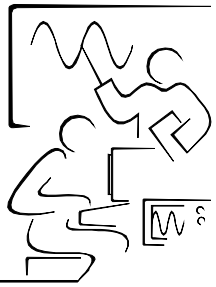
Резюме

Двоичные счетчики являются базовой компонентой цифровых электронных схем. Они применяются для счетчиков по модулю n различных типов, для генерации субгармоник тактирующих сигналов, а также в многоразрядных устройствах, таких как АЦП и ЦАП.

Состав библиотеки виртуальных приборов, использованных на занятии 5 (показан в порядке упоминания)

- **Binary1.vi** (счетчик–делитель на 2)
- **Binary2.vi** (счетчик–делитель на 4)
- **Binary4.vi** (счетчик-делитель на 16 с графическим индикатором)
- **Binary8.vi** (счетчик-делитель на 256 с графическим индикатором)
- **Bin8_Reset.vi** (8-битовый счетчик с кнопкой возврата в исходное состояние)
- **FlipFlop.vi** (подпрограмма, применяемая во всех приборах этой главы)

Занятие 7. Цифро-аналоговый преобразователь.



Цифроаналоговый преобразователь (сокращенно ЦАП) – это одна из наиболее важных схем сопряжения, применяемых для организации связи между аналоговыми и цифровыми устройствами. ЦАП является основой многих электронных схем и устройств, включая цифровые вольтметры, графопостроители, осциллографы и многие другие устройства, управляемые компьютером. На этом занятии изучается цифроаналоговый преобразователь, его модификации, а также его применение для генерации сигналов.

Что такое ЦАП?

ЦАП – это электронное устройство, преобразующее цифровой логический сигнал в аналоговый сигнал. Выходное напряжение ЦАП - это набор битов входного сигнала, взвешенных определенным образом:

$$DAC = \sum_{i=0} w_i \, b_i \, ,$$

где w_i – весовой коэффициент, b_i – значение бита (1 или 0), i – индекс номера бита. В случае двоичной схемы взвешивания, когда $w_i = 2^i$, полное выражение для 8-битового ЦАП запишется в виде:

$$DAC = 128\ b_7 + 64\ b_6 + 32\ b_5 + 16\ b_4 + 8\ b_3 + 4\ b_2 + 2\ b_1 + 1\ b_0 \, .$$

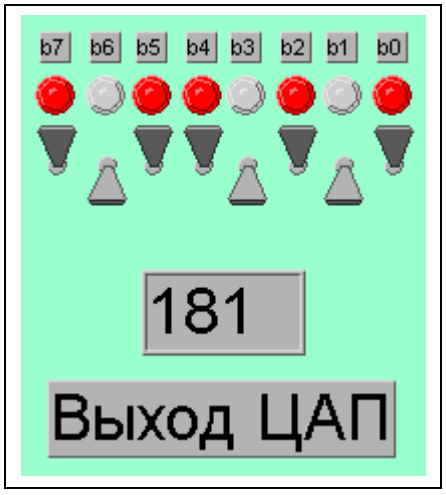


Рисунок 7-1. Моделирование 8-битового ЦАП.

В предыдущей модели, виртуальный прибор **DAC.vi** демонстрирует процесс преобразования. Восемь булевых переключателей на лицевой панели устанавливают входные биты от b_0 до b_7. Когда программа запущена, восемь светодиодных индикаторов отображают величину входного байта. Выходной сигнал отображается в виде числового индикатора. На диаграммной панели виден алгоритм работы 8-битового конвертора, реализованного при помощи пакета LabVIEW.

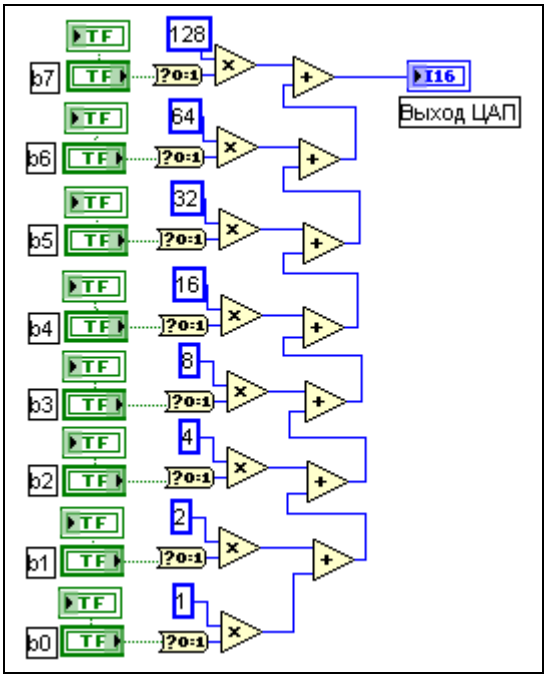


Рисунок 7-2. Диаграммная панель **DAC.vi**.

Для генерации выходного сигнала ЦАП в данной модели используются функции сложения и умножения. Обратите внимание на значок диаграммной панели прибора, моделирующего естественным способом преобразование двоичной величины (булевы уровни) в аналоговую (численную) величину.

Запустите виртуальный прибор **DAC.vi** и наблюдайте отношения между двоичными кодами и эквивалентными числовыми величинами. Этот виртуальный прибор используется также в качестве подпрограммы для других приборов с целью преобразования 8-битового сигнала в соответствующую числовую величину. Чтобы ознакомиться с применением ЦАП, рассмотрите модель инструкции 8-битового сложения в микрокомпьютерном чипе.

Арифметико-логическое устройство

Арифметико-логическое устройство отвечает за все арифметические и логические операции, возникающие внутри центрального процессора (ЦПУ) компьютерного чипа. Рассмотрим инструкцию сложения

СЛОЖИТЬ R1,R2

которая складывает содержимое регистров 1 и 2 и сохраняет полученный результат в сумматоре. Восемь булевых переключателей и индикаторов моделируют 8-битовые регистры R1 и R2. Девять светодиодных индикаторов показывают величину в сумматоре и любое переполнение в двоичном разряде переноса. Три идентичных подпрограммы **DAC.vi** конвертируют содержимое трех регистров в соответствующие числовые величины.

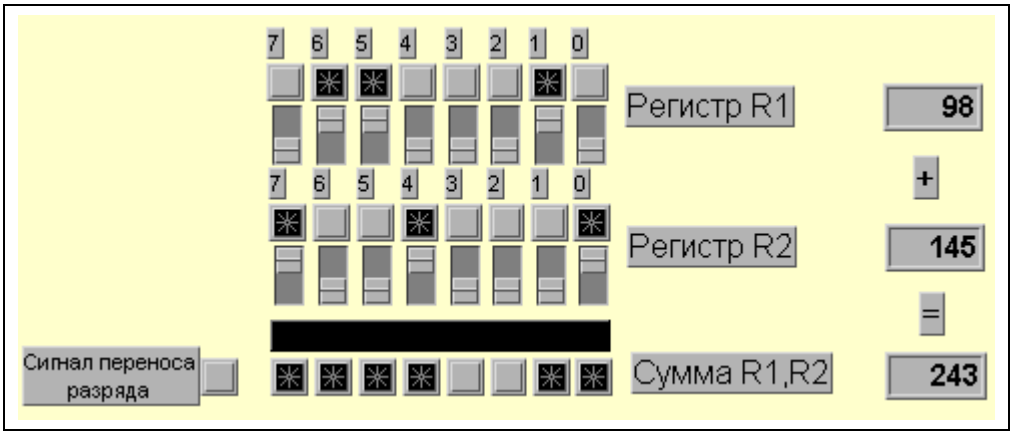


Рисунок 7-3. Моделирование 8-битового сумматора.

Запустите программу **ADD R1,R2.vi** и наблюдайте в действии 8-битовое двоичное сложение. Попробуйте сложение простых последовательностей вида 1+1 с более сложными, такими как $EF +$3. Наблюдайте работу индикатора переноса разряда. Этот виртуальный прибор может складывать величины с большим числом битов, например 16-битовые числа. На диаграммной панели данного моделирующего прибора можно увидеть совместное использование модулей двоичного сложения, построенных на занятии 3, и подпрограмм **DAC.vi**.

Моделирование реального ЦАП

Микросхема Motorola MC1408 является представителем 8-битовых цифроаналоговых преобразователей, выходной ток которых, *i*, прямо пропорционален цифровому входному напряжению. Передаточная функция, которую можно найти в описании данной схемы, имеет вид:

$$i = K \{A1/2 + A2/4 + A3/8 + A4/16 + A5/32 + A6/64 + A7/128 + A8/256\},$$

где цифровые входы A_i могут принимать значения 0 или 1, и A_1 – самый старший разряд. A_8 – самый младший двоичный разряд. Константа пропорциональности $K = V_{ref} / R14$. Здесь опорное напряжение, равное +5 В, вызывает опорный ток величиной 5 В/3.9 кОм = 1.28 мА через резистор R14. Максимальное значение генерируемого тока получается, когда на все входы поступает цифровой сигнал (т.е. состояние каждого входа равно 1). Оно равно 0.996 * 1.28 мА = 1.275 мА.

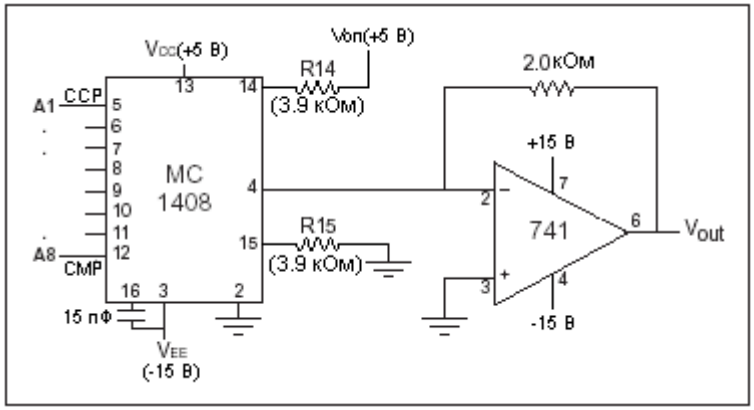


Рисунок 7-4. Схема 8-битового ЦАП, собранная из подходящих микросхем.

Операционный усилитель MC741, сконфигурированный для выполнения функции преобразователя тока в напряжение, конвертирует выходной ток ЦАП в напряжение $V_{out} = – i R$. При величине резистора обратной связи 2 кОм максимальное выходное напряжение равно – 2.55 В, а чувствительность 10 мВ/бит. Такое соотношение очень удобно, поскольку максимальное двоичное число на входе, когда состояние всех входов равно 1, имеет эквивалентное десятичное значение 255.

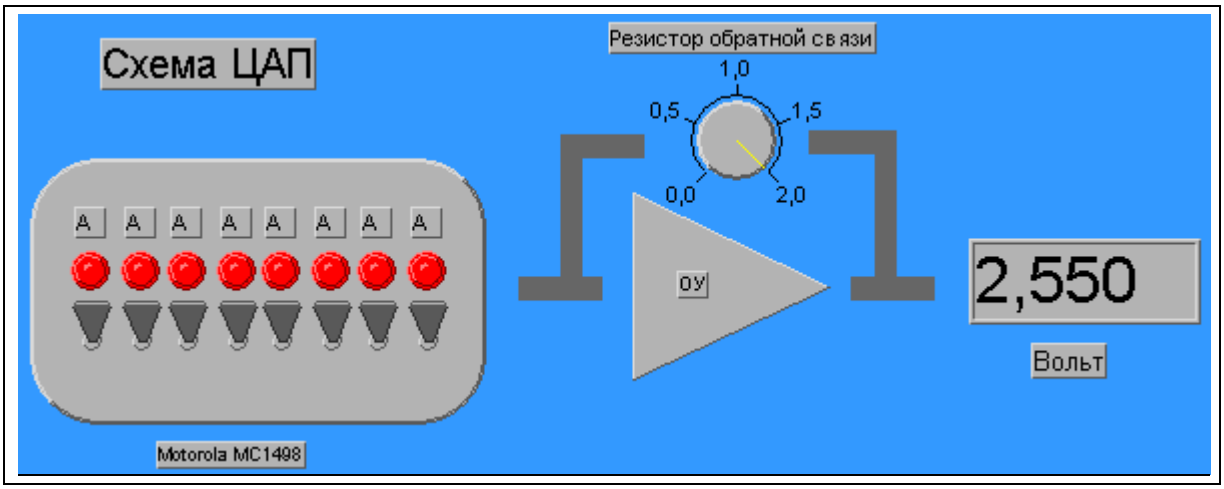


Рисунок 7-5. Моделирование 8-битового ЦАП, принципиальная схема которого показана на рис. 7-4.

Запустите и изучите виртуальный прибор **MC1408.vi**, который моделирует схему ЦАП, построенную на чипе 1408. Убедитесь, что разрешение ЦАП (то есть изменение выходного напряжения, при изменении входного сигнала на 1 бит) равно 10 мВ. Подстраивая резистор обратной связи, можно масштабировать выходное напряжение на любую удобную величину (например, 1 вольт). Обратите внимание на различие диаграммной панели этого прибора и **DAC.vi**. Если у вас есть ЦАП MC1408 и операционный усилитель 741, то можете сравнить данную модель с работой реального устройства, принципиальная схема которого показана выше.

Генераторы сигналов

Для генерации аналогового сигнала можно использовать любую последовательность битов, подаваемых с постоянной скоростью на вход ЦАП. Простейшая последовательность получается на выходе 8-битового двоичного счетчика. С ее помощью генерируется цифровое пилообразное напряжение в пределах от 0 до 2.55 В. Для демонстрации такого процесса необходимо присоединить ВП **Binary8.vi**, представленный на занятии 5, к **DAC.vi**. После этого выходной сигнал подается на графический индикатор. Скорость нарастания графика определяется частотой счета: больше частота – больше наклон. Колебательный модуль генерирует тактовый сигнал. Когда происходит переполнение счетчика от (11111111) до (00000000), аналоговое напряжение быстро спадает от 255 до 0. Такой цифровой сигнал называют ступенчатым, поскольку он похож на ступеньки лестницы.

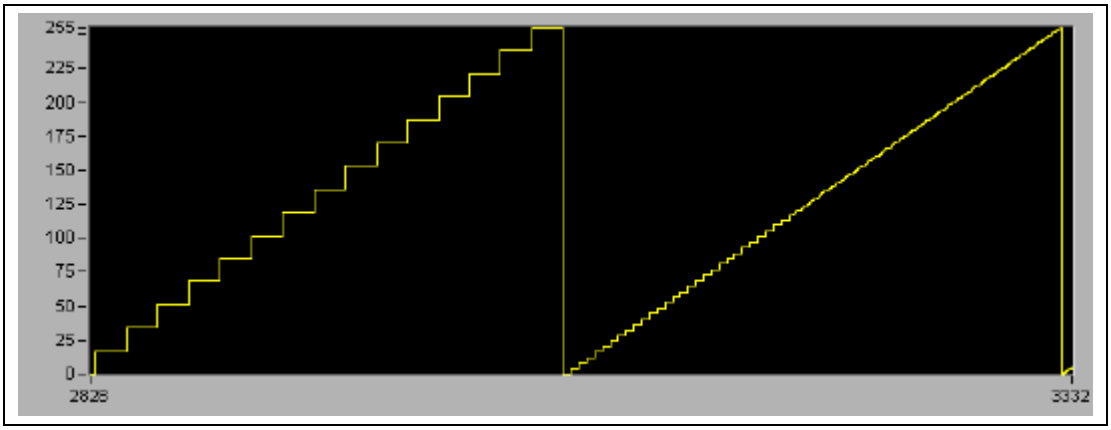


Рисунок 7-6. Выходные сигнала 4, 6 и 8-битовых ЦАП.

При увеличении числа битов ЦАП амплитуда ступеньки уменьшается. Выходной сигнал 4-битового ЦАП имеет 15 ступеней, 6-битовый ЦАП – 63 ступени, а 8-битовый – 255 ступеней. В предыдущем примере, **DAC Resolution.vi**, демонстрируется увеличение разрешения при увеличении числа битов. В пределе, когда число битов возрастает от 16 до 20, цифровой сигнал очень хорошо аппроксимирует аналоговую пилообразную кривую. Взгляните на выходное напряжение ВП **DAC8/12.vi**, демонстрирующего дополнительное разрешение при изменении типа ЦАП с 8 до 12-битового. Для проведения инженерных и научных исследований чаще всего требуется, по крайней мере, 12-битовое разрешение.

Специальные ЦАП

В беззнаковой двоичной арифметике все числа положительные. В арифметике со знаком для отображения знака числа используется самый старший двоичный разряд (0 – положительное и 1 – отрицательное). В этом случае 256 двоичных чисел 8-битового ЦАП разделяются на положительные от 0 до 127 и отрицательные от – 128 до – 1. Виртуальный прибор **DAC+/-.vi** демонстрирует выходной сигнал обоих знаков.

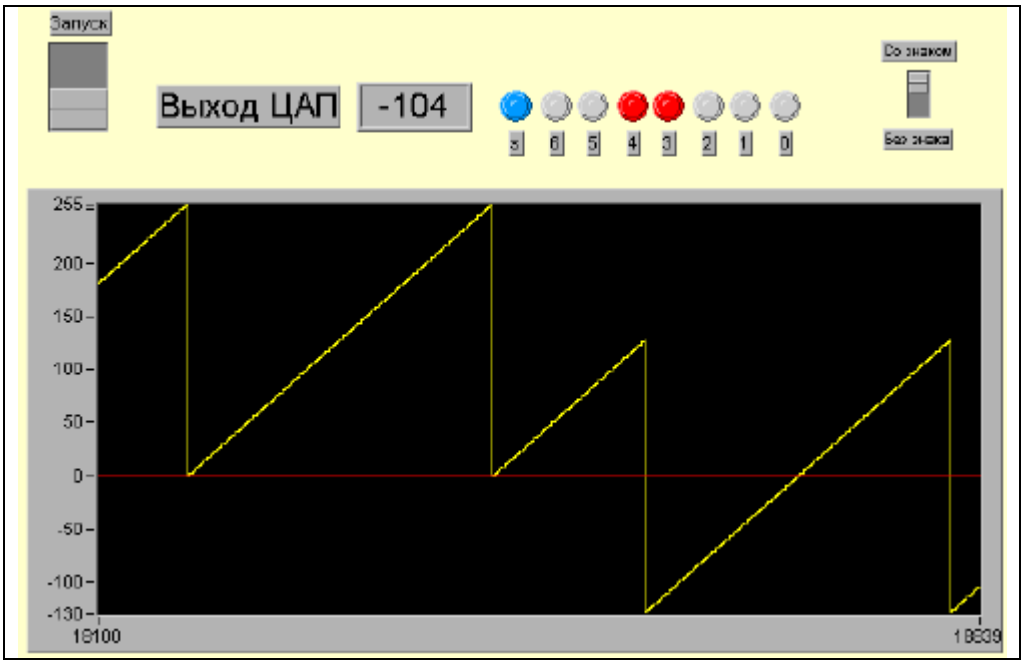


Рисунок 7-7. Выходные сигналы ЦАП одной и двух полярностей.

Обратите внимание, что интервал изменения величины Y одинаков в обоих случаях.

Фигуры Лиссажу

Если один из двух гармонически связанных сигналов подать на ось X, а другой на ось Y, то в результате будет построена интересная кривая, часто называемая фигурой Лиссажу. Зная числа точек пересечения горизонтальной линии и графика, и вертикальной линии и графика, вы можете найти отношение двух частот путем деления этих чисел первого на второе. В следующем примере показаны четыре точки пересечения на горизонтальной линии и две на вертикальной, что дает отношение частот 2:1. Вдобавок при условии совершенной гармоничности двух сигналов, фигура Лиссажу может дать информацию о сдвиге фаз между сигналами. Запустите виртуальный прибор **Lissajous1.vi** и исследуйте фазовое соотношение двух гармонически связанных сигнала.

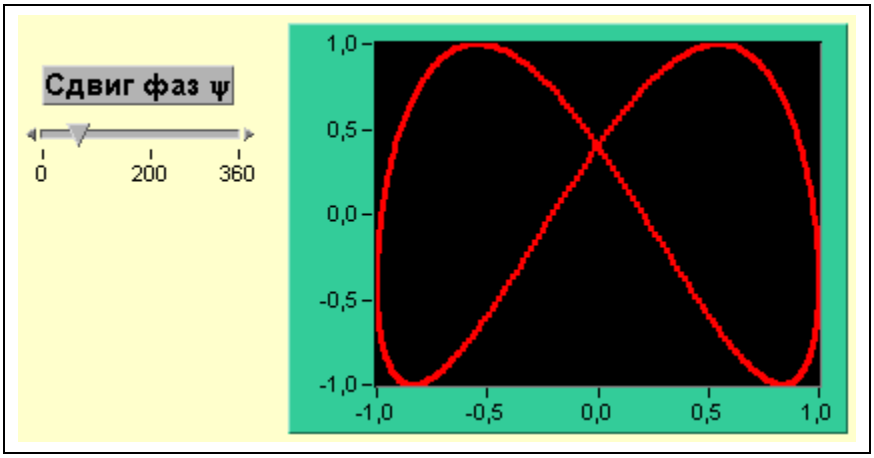
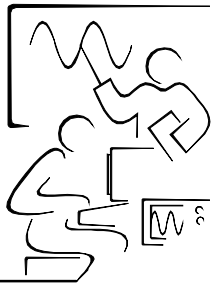


Рисунок 7-8. Модель прибора, строящего фигуру Лиссажу с $w_2 = 2w_1$.

Состав библиотеки виртуальных приборов, использованных на занятии 7 (показан в порядке упоминания)

- **DAC.vi** (модель 8-битового ЦАП)
- **ADD R1,R2.vi** (8-битовый двоичный сумматор)
- **MC1408.vi** (моделирование интегральной схемы ЦАП Motorola 1408)
- **DAC Resolution.vi** (моделирование 4-битового, 6-битового и 8-битового ЦАП)
- **DAC+/-.vi** (беззнаковый и имеющий знак ЦАП)
- **Lissajous.vi** (модель построения фигуры Лиссажу)
- **DAC8/12.vi** (разрешение 8-битового и 12-битового ЦАП)
- **DAC12.vi** (подпрограмма, используемая в **DAC8/12.vi**)
- **BIN_RST.vi** (8-битовый счетчик с обнулением)
- **Half Adder.vi** (подпрограмма, используемая в **ADD R1,R2.vi**)
- **Full Adder.vi** (подпрограмма, используемая в **ADD R1,R2.vi**)
- **FlipFlop.vi** (подпрограмма, используемая в **ADD R1,R2.vi**)

Занятие 8. Аналого-цифровой преобразователь. Часть 1.



Аналого-цифровой преобразователь (сокращенно АЦП) – это второй ключевой элемент, обеспечивающий взаимодействие аналоговых и цифровых устройств. АЦП является основой цифровых вольтметров, цифровых авометров, многоканальных анализаторов, осциллографов и многих других приборов. Существует несколько различных типов АЦП. Наиболее распространенными являются интегрирующие, следящие и преобразователи последовательного приближения. На данном занятии мы изучим работу интегрирующих и следящих аналого-цифровых преобразователей.

Назначение аналого-цифрового преобразователя

Назначение АЦП заключается в генерации двоичного цифрового кода, пропорционального входному аналоговому сигналу. Основной процесс преобразования показан на следующей диаграмме:

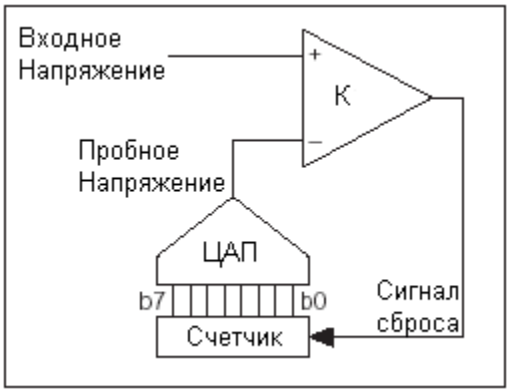


Рисунок 8-1. Схема 8-битового аналого-цифрового преобразователя.

Счетчик создает пробную двоичную последовательность, которая конвертируется в аналоговое напряжение при помощи цифроаналогового преобразователя. ЦАП является базовым элементом многих схем АЦП, а его принцип работы обсуждался на занятии 7 (кстати, сейчас самое время узнать принцип его работы, если вы не знакомы с устройством ЦАП). После этого пробное напряжение сравнивается с входным сигналом. Если входное напряжение больше пробного сигнала, счетчик увеличивает значение, чтобы приблизить пробный сигнал к уровню входного

напряжения. Если же входной сигнал меньше пробного, счетчик уменьшает свое выходное значение с тем, чтобы уровень пробного сигнала приблизился к уровню входного. Этот процесс продолжается до тех пор, пока компаратор изменит знак. В этот момент уровень пробного сигнала будет в пределах одного отсчета от уровня входного напряжения. При увеличении числа разрядов счетчика будет увеличиваться и разрешение ЦАП такого типа.

Интегрирующий АЦП

В интегрирующем АЦП для генерации пилообразного пробного напряжения используются двоичный счетчик и цифроаналоговый преобразователь. В настоящем примере 8-битовый счетчик, **Binary Counter.vi**, работающий в режиме возрастания, в совокупности с 8-битовым ЦАП, **DAC.vi** (рассмотрен на предыдущем занятии), генерируют пробный сигнал. Его уровень непрерывно возрастает от 0 до 255, если прибор работает в режиме свободного запуска. Однако, когда уровень пробного сигнала становится больше (или, как в данном примере, равен) уровню входного сигнала, компаратор изменяет знак и работа останавливается.

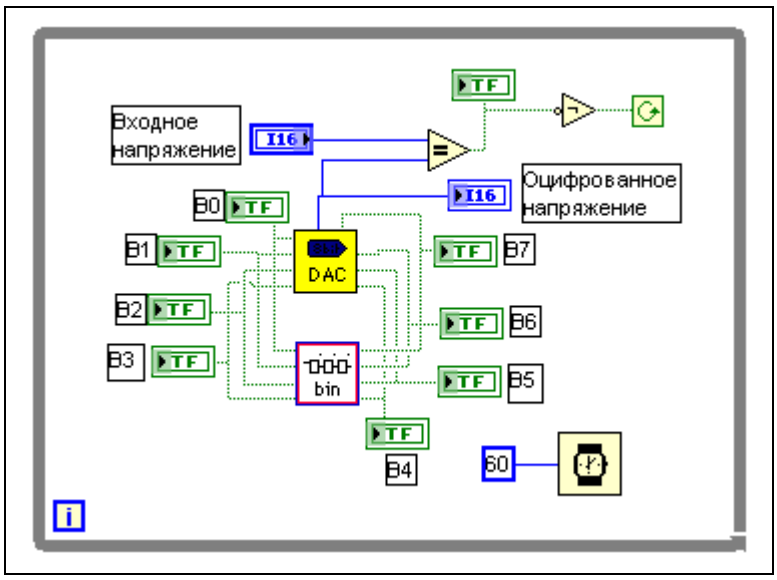


Рисунок 8-2. Моделирование 8-битового интегрирующего АЦП.

Последнее число, записанное в разрядах (b7-b0), является оцифрованной величиной уровня входного напряжения. При моделировании было выбрано время задержки в 60 мс для того, чтобы глаз успевал следить за процессом. Функция сравнения моделируется при помощи LabVIEW функции **Equal**.

Запустите виртуальный прибор **Ramp.vi**. Попробуйте выставить другие значения входного сигнала. При этом обратите внимание, что время преобразования зависит от уровня входного сигнала.

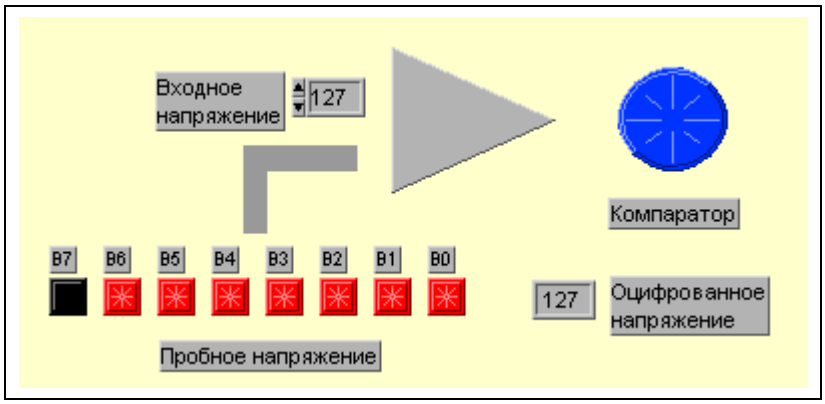


Рисунок 8-3. Лицевая панель 8-битового АЦП. Светодиодный индикатор компаратора меняет свое состояние, когда уровень пробного напряжения превышает уровень входного сигнала.

В следующем примере, **Ramp4.vi**, двоичный счетчик работает в свободном режиме. Всякий раз, когда пробный сигнал становится больше входного, компаратор меняет знак. Такое пересечение пилообразного сигнала с входным напряжением можно наблюдать на графическом индикаторе. Двоичная величина счетчика в точке пересечения – это оцифрованный сигнал. Изменение состояния компаратора свидетельствует о пересечении.

Чтобы смоделировать действительно пилообразный сигнал АЦП, необходимо, чтобы при изменении состояния компаратора происходил возврат счетчика в исходное состояние. Для этого постой двоичный счетчик заменяется на двоичный счетчик с обнулением, который изучался на шестом занятии. Запустите виртуальный прибор **Ramp2.vi** и наблюдайте его работу. Обратите внимание: как только уровень пробного сигнала достигает входного, двоичный счетчик возвращается в исходное состояние, а пилообразный цикл начинается снова. На индикаторе, показанном ниже, уровень входного сигнала менялся трижды.

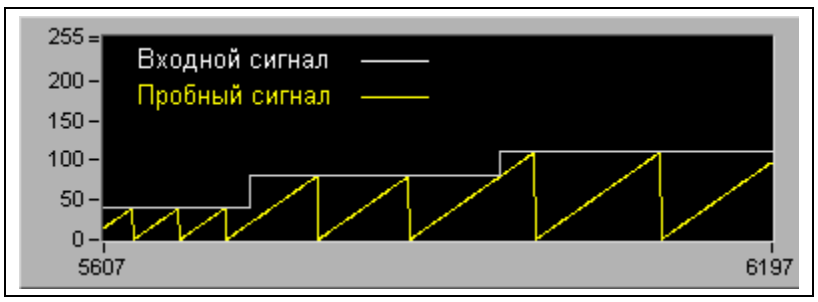


Рисунок 8-4. Графический индикатор, отображающий работу интегрирующего АЦП.

Интересная особенность, уникальная для интегрирующего АЦП, заключается в том, что длительность преобразования зависит от амплитуды входного сигнала. Сигналы с маленькой амплитудой оцифровываются быстрее сигналов с большой амплитудой. Таким образом, время преобразования зависит от амплитуды входного сигнала и тактовой частоты схемы. Для 8-битового ЦАП различие времен преобразования может не вызывать проблем при тактовой частоте в несколько мегагерц, однако для 12-битового ЦАП это является проблемой.

Так же хорошо интегрирующий АЦП функционирует со счетчиками, работающими в режиме убывания, когда генерируются числа от 255 до 0. Изменение состояния компаратора снова сигнализирует о том, что уровень пробного сигнала, определяемого величиной счетчика, равен уровню входного сигнала.

Задание повышенной сложности

Придумайте интегрирующий АЦП, в котором используется счетчик, работающий в режиме убывания.

Возможно ли использование счетчика, работающего в режиме возрастания/убывания, для отслеживания уровня входного сигнала?

Да, возможно. Такая техника преобразования называется следящим АЦП. Для него характерно наименьшее из всех АЦП время преобразования.

Следящие АЦП

Основная задача следящего АЦП – быстро приблизиться к уровню входного сигнала, используя некоторую технику, такую как генерация пилообразного напряжения. В точке, определяемой пересечением пилообразного и входного сигналов, следящий алгоритм заканчивает свою работу.

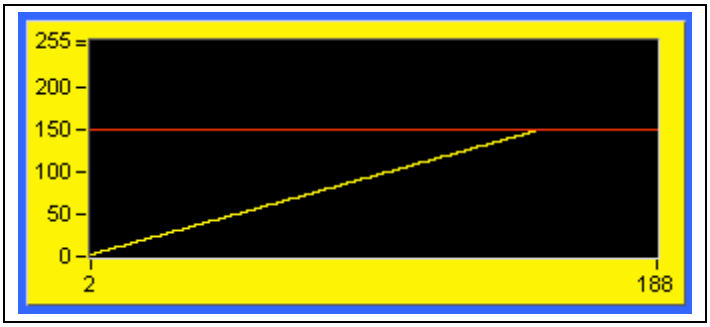


Рисунок 8-5. Следящий АЦП быстро приближается к уровню входного сигнала, прежде чем начать слежение.

Следящий алгоритм достаточно прост:

если

уровень пробного сигнала больше уровня входного сигнала,
уменьшить число счетчика на единицу,

если
уровень пробного сигнала меньше уровня входного сигнала,
увеличить число счетчика на единицу

повторять бесконечно.

В следующем примере используется генерация положительного пилообразного напряжения для быстрого приближения к уровню входного сигнала величиной 150. Как только уровень входного сигнала достигнут, следящий алгоритм заканчивает свою работу.

Увеличивая масштаб графика по вертикали, можно увидеть следящий алгоритм в действии.

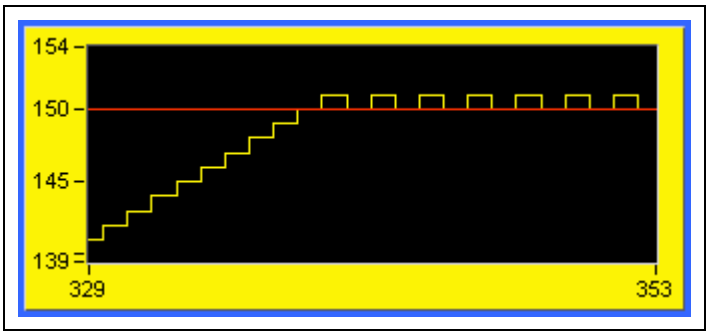


Рисунок 8-6. Выходное напряжение следящего АЦП при постоянном входном сигнале.

Однако, если уровень входного сигнала меняется, АЦП должен вернуться к генерации пилообразного напряжения, чтобы нагнать изменение входного сигнала. В том случае, когда тактовый генератор достаточно быстр, отслеживание происходит оперативно. Но если входной сигнал меняется слишком быстро, оцифрованный сигнал пропадает до тех пор, пока пробный сигнал снова нагонит входной. В действительности существует предельная отслеживающая скорость АЦП. Она ограничивает максимальную частоту изменения входного сигнала.

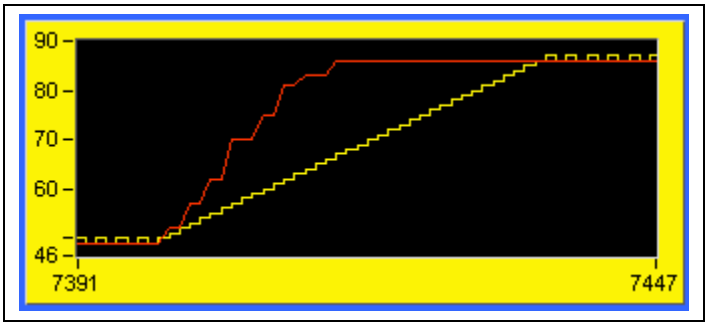


Рисунок 8-7. Быстрое изменение входного сигнала приводит к подстройке пробного напряжения до нового уровня.

Поскольку в следящем АЦП используется счетчик, работающий в режиме возрастания/убывания, то для отслеживающего алгоритма возникает подобная задача, когда входной сигнал внезапно спадает ниже уровня пробного напряжения. Следящий АЦП возвращается к генерации спадающего пилообразного напряжения (рисунок 8-8) до тех пор, пока пробное напряжение достигнет уровня входного сигнала.

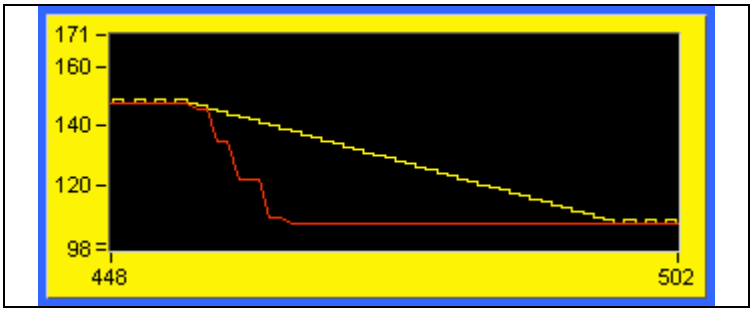


Рисунок 8-8. Отрицательное изменение входного сигнала обуславливает спадание пробного напряжения к новому уровню.

Для демонстрации этой техники и генерации всех графиков, представленных выше, используется виртуальный прибор **Tracking ADC.vi**. Сам алгоритм, показанный на диаграммной панели, очень прост. Его выполнение обеспечивает функция LabVIEW Select и сдвиговый регистр Цикла по условию.

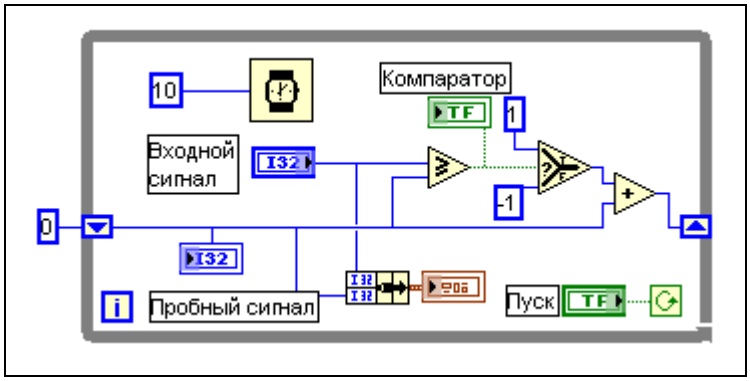


Рисунок 8-9. Виртуальный прибор, моделирующий следящий АЦП.

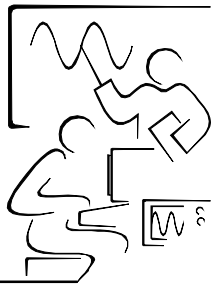
Функция ожидания **Wait** имеет параметр 100мс. Это сделано для того, чтобы пользователь мог наблюдать за работой пробора с лицевой панели. Для увеличения графика в процессе работы прибора необходимо переопределить масштаб вертикальной оси. Это можно сделать при помощи инструмента Управление. Чтобы наблюдать процесс слежения

при изменяющемся уровне входного сигнала, уменьшите постоянную времени ожидания **Wait** до 1мс.

Состав библиотеки виртуальных приборов, использованных на занятии 8 (показан в порядке упоминания)

- **Ramp.vi** (8-битовый интегрирующий АЦП, в котором процесс преобразования замедлен для облегчения наблюдения)
- **Ramp4.vi** (интегрирующий АЦП без обратной связи с компаратором)
- **Ramp2.vi** (8-битовый интегрирующий АЦП с графическим индикатором)
- **Tracking ADC1.vi** (Следящий АЦП)
- **Binary Counter.vi** (подпрограмма – 8-битовый двоичный счетчик)
- **BIN_RST.vi** (подпрограмма – 8-битовый двоичный счетчик с внешним обнулением)
- **DAC.vi** (подпрограмма – 8-битовый ЦАП)
- **FlipFlop.vi** (подрограмма)

Занятие 9. Аналого-цифровой преобразователь. Часть 2.



На предыдущем занятии для генерации пробного сигнала в интегрирующем и следящем АЦП использовались двоичные счетчики, работающие в режиме возрастания и возрастания/убывания. В другом распространенном АЦП для генерации пробного напряжения используется регистр последовательных приближений (РПП). АЦП такого типа существенно быстрее интегрирующих АЦП и имеют постоянное и известное время преобразования. В РПП применяется схема взвешивания, в которой каждый бит генерируется в последовательности от самого старшего разряда (ССР) до самого младшего разряда (СМР).

Алгоритм работы РПП заключается в следующем:

1. Обнуление регистра последовательных приближений и установка выхода АЦП в ноль.
2. Установка самого старшего разряда РПП:

   Если $V_{ЦАП}$ больше $V_{вх}$, тогда обнулить этот разряд;

   Если же $V_{ЦАП}$ меньше $V_{вх}$, оставить в этом разряде единицу.
3. Повторить шаг 2 для следующего ССР до тех пор, пока все *n* разрядов РПП будут установлены и проверены.
4. После *n* циклов цифровой канал на выходе РПП будет иметь значение оцифрованной величины входного сигнала.

Наиболее удобно работу алгоритма можно посмотреть с помощью графиков входного сигнала и выходного напряжения ЦАП, сгенерированного РПП. Предположим, что на вход АЦП подается сигнал амплитудой 153. Число 153 можно записать в виде 128 + 16 + 8 + 1. Тогда в двоичном представлении оно запишется как (читать справа налево)

$$153_{10} = (10011001)_2$$

В алгоритме РПП утверждается, что ССР, имеющий величину 128, должен быть проверен первым. Поскольку 128 меньше 153, этот разряд остается. Таким образом, лучшая оценка после первого цикла - (10000000). На следующей итерации следующий ССР, имеющий значение 64, добавляется к предыдущей оценке (то есть, 128 + 64 = 192). Поскольку 192 больше 153, этот бит обнуляется, поэтому предыдущая оценка остается неизмененной. В следующем цикле следующий разряд со

значением 32 приводит к пробной величине 128 + 32 = 160. Снова оценка больше входного сигнала, так что этот бит обнуляется, а наилучшая оценка остается (10000000). На следующей итерации разряд со значением 16 приведет к 128 + 16 = 144. Эта величина меньше 153, так что в данном разряде остается единица. После четырех циклов наилучшая оценка – это (1001 0000). Оставшиеся циклы можно увидеть при помощи модели аналого-цифрового преобразователя с последовательным приближением.

На рисунке, показанном ниже, изображена точная временная диаграмма этого процесса. Непрерывная линия изображает пробное напряжение в каждом цикле, а пунктирная – уровень входного сигнала 153. После четырех циклов работы прибор выдает окончательную двоичную величину, отображаемую на восьми светодиодных индикаторах.

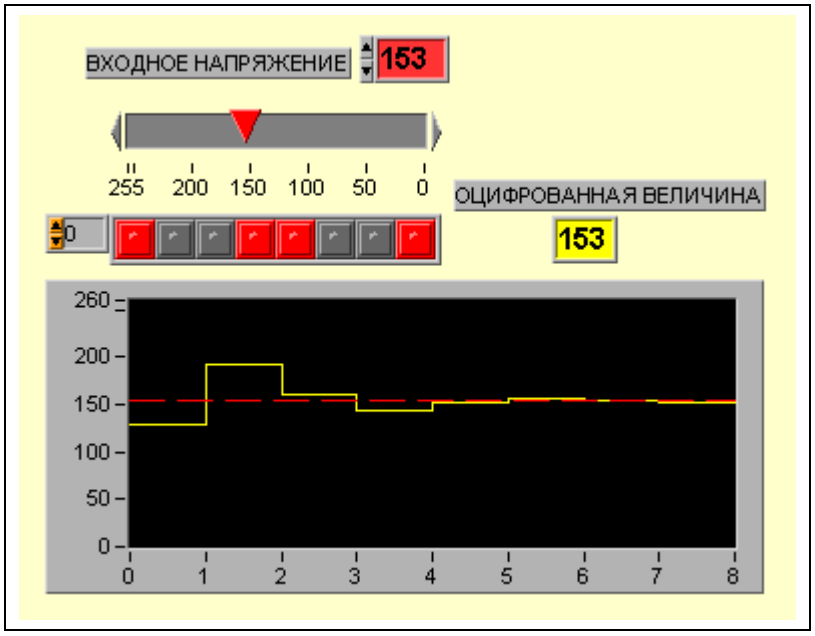


Рисунок 9-1. Пробное напряжение при последовательном приближении с целью оцифровать входной сигнал.

Запустите виртуальный прибор **SAR.vi** в непрерывном режиме. Для изменения уровня входного сигнала можно использовать инструмент Управление. При этом пробный сигнал, генерируемый РПП, послушно следует за входным напряжением, выдавая оцифрованное значение последнего в любом случае за 8 циклов. Время генерации выходного сигнала ЦАП под действием РПП устанавливает фундаментальный предел скорости работы АЦП. Большинство преобразователей, построенных на основе РПП, имеют время преобразования меньше 100мс.

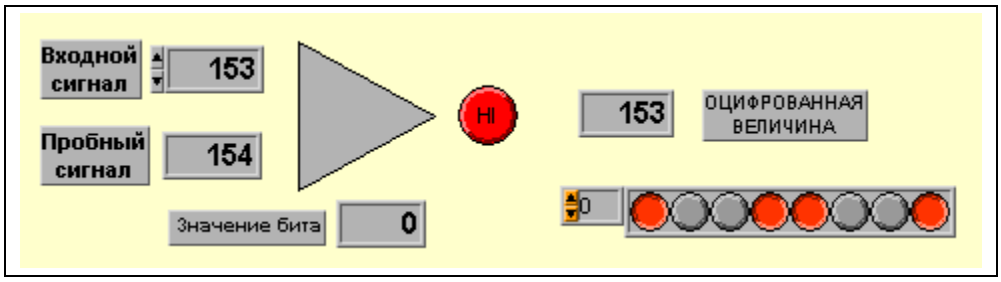


Рисунок 9-2. Оцифрованная величина РПП в АЦП, представленная в виде булевой матрицы.

Работа второго прибора **SAR0.vi** замедлена с целью облегчения наблюдения за функционированием каждого цикла. Уровень входного сигнала установлен в 153. Выходной сигнал ЦАП – это цифровое пробное напряжение, значение каждого бита которого складывается с предыдущей наилучшей оценкой, как это обсуждалось выше. Оцифрованная выходная величина – это наилучшая оценка входного напряжения после восьми циклов. По мере выполнения этого алгоритма матрица булевых индикаторов показывает текущую величину наилучшей оценки, а после 8 циклов эта матрица покажет конечную оцифрованную величину.

Моделирование АЦП последовательных приближений

Моделирование РПП в среде LabVIEW достаточно сложно, как сложна и реальная микросхема РПП. Поэтому диаграммная панель виртуального прибора **SAR0.vi** будет обсуждаться в два этапа. Сначала алгоритм работы РПП, затем двоичное представление с помощью булевой матрицы.

Биты пробного сигнала формируются последовательным 8-кратным делением числа 256 на 2 в сдвиговом регистре. Величина, отображаемая на индикаторе «Значение бита», будет пробегать значения (128, 64, 32,16, 8, 4, 2 и 1) по мере изменения индекса цикла от 0 до 7. Девятый цикл необходим для записи начальных значений в сдвиговые регистры. Пробная величина образуется добавлением нового значения бита к предыдущей наилучшей оценке. Функция сравнения решает, записывать ли текущий бит в наилучшую оценку. После восьми циклов работы РПП наилучшая оценка становится оцифрованным уровнем сигнала.

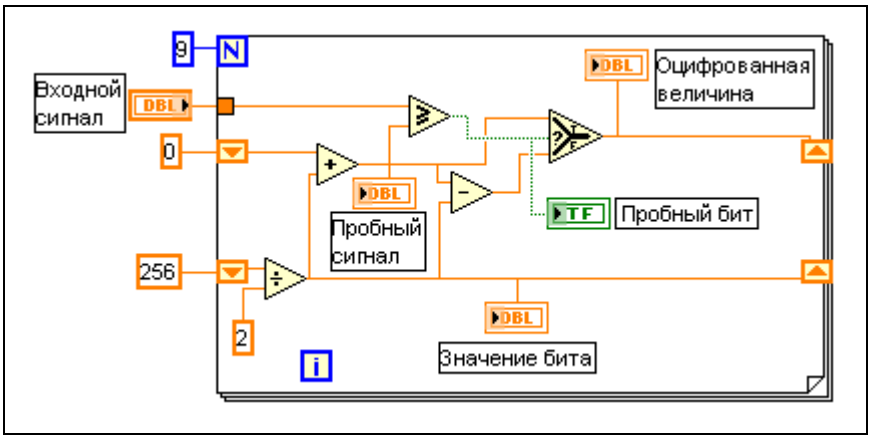


Рисунок 9-3. Моделирование РПП при помощи сдвиговых регистров.

Для создания двоичного представления наилучшей оценки используется логический сумматор в форме булевого сдвигового регистра. Пробный бит, имеющий любое значение, записывается в матрицу после каждого цикла. Это реализуется при помощи функции LabVIEW «Добавить элемент матрицы». Булева величина «Истина» или «Ложь» записывается в булеву матрицу под индексом, определяемым счетчиком цикла. В самом начале все восемь элементов матрицы установлены в состояние «Ложь», чтобы обеспечить выключение всех светодиодных индикаторов в матрице.

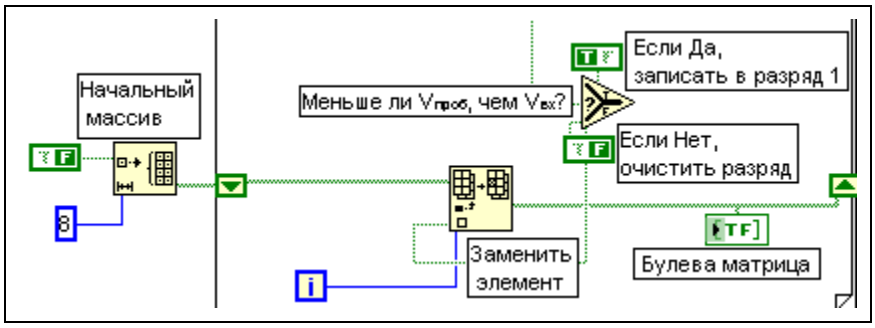


Рисунок 9-4. Моделирование РПП при помощи массивов.

Как только определена наилучшая оценка, оцифрованная двоичная величина будет показана на лицевой панели. После восьми циклов определение двоичной величины заканчивается. Ее эквивалент в десятичном виде показывается на цифровом индикаторе.

Функция LabVIEW для обработки строк **Format and Strip** форматирует любую строковую переменную в число в соответствии с выбранным кодом преобразования. В виртуальном приборе **SAR_Hex.vi** строковая переменная, состоящая из двух букв и представляющая шестнадцатеричное число от $00 до $FF, конвертируется в десятичное число от 0 до 255 и цифруется при помощи РПП-алгоритма. Испытайте работу этого прибора.

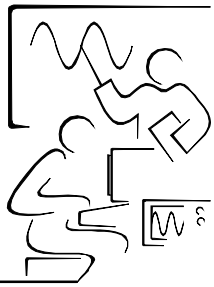
Выводы

На прошедших двух занятиях была изучена и продемонстрирована работа аналого-цифровых преобразователей трех типов. Интегрирующий АЦП концептуально наиболее прост, но имеет недостаток. Его время преобразования различно и пропорционально амплитуде входного сигнала. Следящие АЦП – это наиболее быстрые преобразователи, но они работают быстро только в случае, если входной сигнал не претерпевает резких изменений. Наилучший преобразователь – это АЦП последовательного приближения с известным и постоянным временем преобразования.

Состав библиотеки виртуальных приборов, использованных на занятии 9 (показан в порядке упоминания)

- **SAR.vi** (АЦП с регистром последовательных приближений)
- **SAR0.vi** (замедленная версия предыдущего виртуального прибора)
- **SAR_Hex.vi** (АЦП последовательных приближений с шестнадцатеричным входом)

Занятие 10. Семисегментные цифровые дисплеи.



Цифровые дисплеи являются связующим звеном между цифровым миром нулей и единиц и числами, с которыми работают люди. Мы уже видели как параллельные комбинации нулей и единиц могут представлять двоичные, шестнадцатеричные или десятичные числа. В наиболее простых приборах используются цифровые дисплеи, состоящие из семи индикаторов (сегментов), для представления чисел от 0 до 9. Каждый сегмент управляется одним битом. Комбинация включенных и выключенных сегментов может изобразить все числа от 0 до 9 и еще ряд букв, таких как A, b, c, d, E и F.

Семисегментный индикатор

В дисплее с семью сегментами используются семь независимых светоизлучающих диода, сконфигурированных в виде числа 8, как это показано на рисунке ниже:

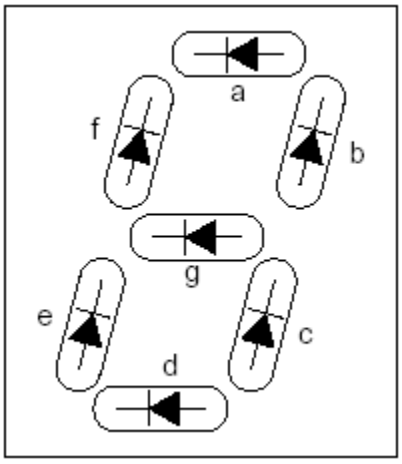


Рисунок 10-1. Семисегментный дисплей, собранный из семи светодиодных индикаторов.

Независимые сегменты расположены по часовой стрелке и обозначены буквами a, b, c, d, e, f. Последний сегмент (g) представляет собой центральную перекладину. Когда на светодиод приложено прямое смещение, он излучает свет. Сам сегмент образуется расположением светодиода вертикально или горизонтально. Многие выходные устройства, такие как параллельные порты компьютера, работают с

восьмиразрядными сигналами. В некоторых семисегментных дисплеях употребляется восьмой светодиод в форме точки для индикации разделения десятичных разрядов.

Запустите виртуальный прибор **7 Segment.vi**, моделирующий семисегментный дисплей. Попробуйте различные комбинации переключателей.

Сколько букв алфавита можно изобразить таким образом?

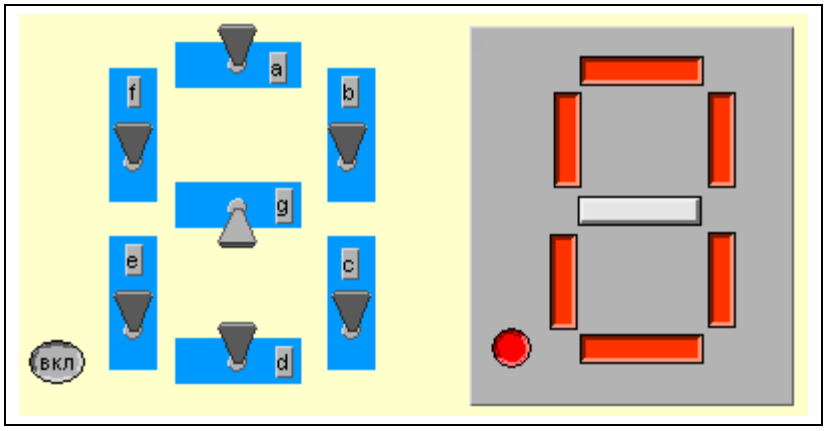


Рисунок 10-2. Модель семисегментного дисплея.

Входные биты 0-7 вводятся при помощи восьми булевых переключателей. Соответствующие сегменты дисплея традиционно обозначаются буквами от a до g и dp (десятичный разряд). Переключатель самого младшего разряда присоединен к сегменту a, следующий бит присоединен к сегменту b и т.д. Самый старший разряд – седьмой – часто присоединяют к восьмому светодиоду и используют как разделитель десятичных разрядов. Манипулируя переключателями, можно отобразить все числа и несколько букв. После экспериментирования с дисплеями попробуйте набрать по одной букве сообщение: “help call 911”.

Большинство дисплеев с семью сегментами управляются шифратором, который преобразовывает двоично-кодированный полубайт в десятичное число. При помощи этого числа, в свою очередь, выбирается подходящий семисегментный код. Первый этап при моделировании – преобразование полубайта в число от 0 до 15. Эту процедуру моделирует виртуальный прибор **Bin->Digit.vi**.

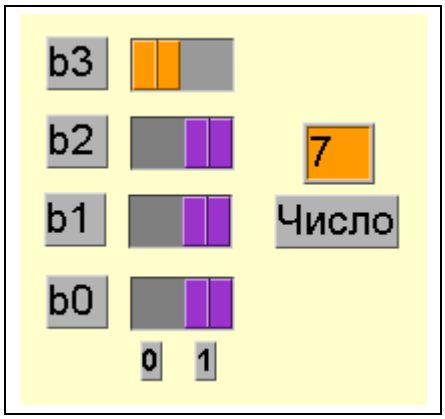


Рисунок 10-3. Лицевая панель программы, преобразующей 4-битовые двоичные числа в деятичные.

Принцип работы 4-битового цифро-аналогового преобразователя можно увидеть на диаграммной панели этого виртуального прибора.

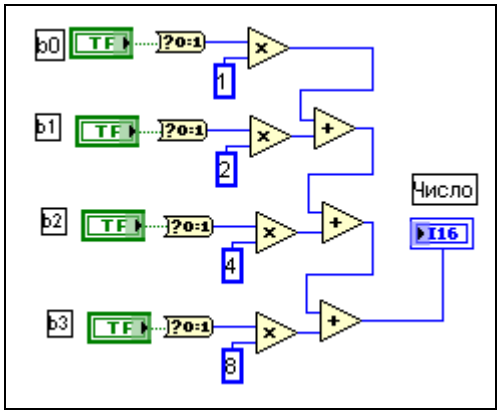


Рисунок 10-4. Диаграммная панель 4-битового цифроаналогового преобразователя.

Следующий этап – преобразование чисел от 0 до 15 в подходящий код для семисегментного дисплея. Для чисел от 10 до 15 используются буквы от A до F шестнадцатеричного представления. В виртуальном приборе **Encoder Hex.vi** это действие реализуется при помощи структуры Варианта LabVIEW. Терминал данных селектора структуры Варианта соединен с ручкой управления, которая предназначена для выполнения функции выбора определенного целого числа. Число 0 формирует семисегментный код для отображения нуля. Число 1 формирует код для отображения единицы и так далее до F. Булевы постоянные внутри каждой структуры Варианта инициализированы таким образом, чтобы генерировать правильный семисегментный код.

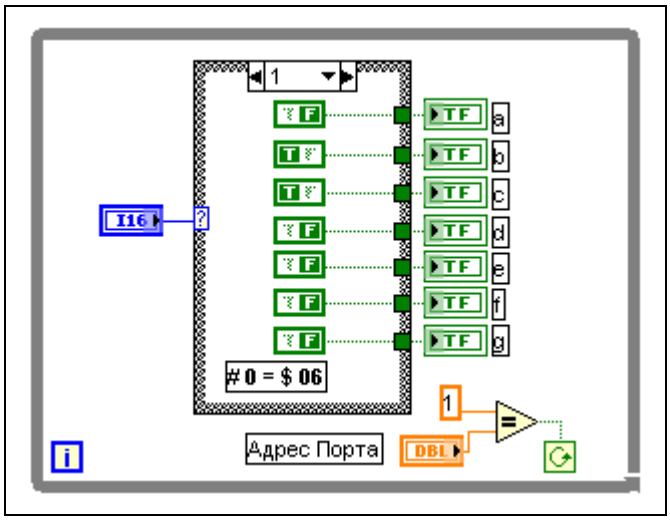


Рисунок 10-5. Виртуальный прибор, преобразующий число в семисегментный код.

Шестнадцатеричное число внутри рамки – это шестнадцатиричное выражение байта, необходимое для представления числа #n.

Каждый порт имеет уникальный адрес, который нужно выбрать перед началом обмена данными с внешними устройствами. Чтобы получить доступ к порту, необходимо ввести правильный адрес на лицевой панели. В данной модели адрес выполняет функцию включения/выключения прибора.

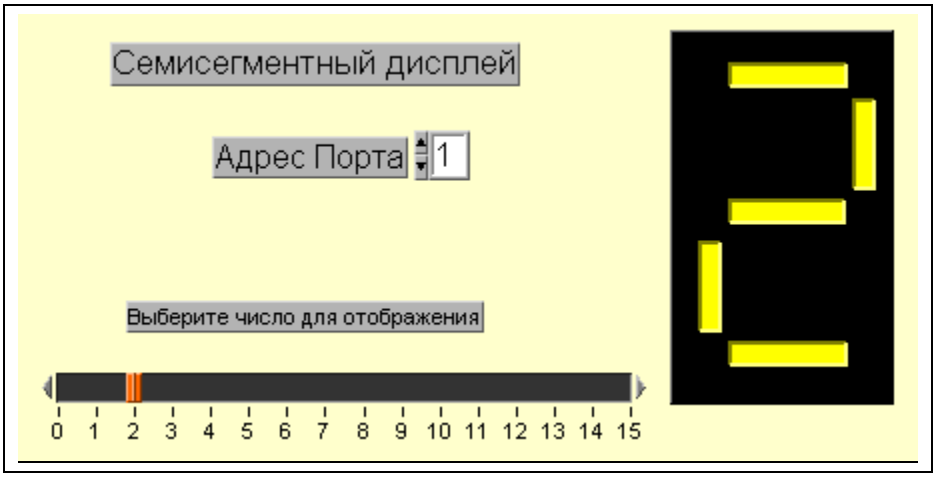


Рисунок 10-6. Индикатор преобразователя шестнадцатеричного числа в семисегментный код.

Выберите адрес порта 1 и запустите **Encoder.vi**. С помощью инструмента Управление нажмите и перемещайте ползунок, выбирая нужное число от 0 до 15. Любые числа из этого диапазона будут отображены в виде семисегментного шестнадцатиричного символа.

Для создания шифратора, переводящего двоичное число в семисегментный код с последующим отображением его на дисплее, можно объединить два виртуальных прибора - **Bin->Digit.vi** и **Encoder.vi**.

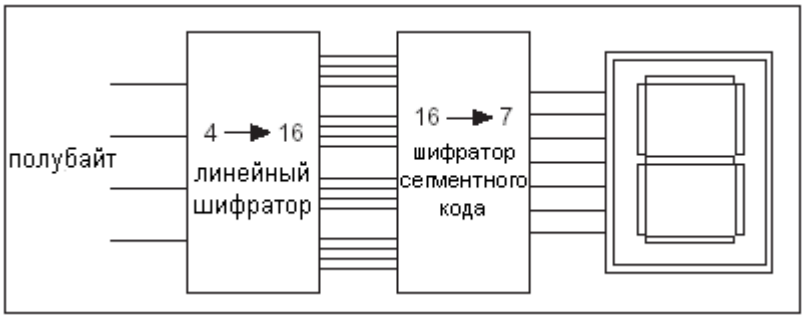


Рисунок 10-7. Схема преобразователя двоичного числа в семисегментный код.

В таком приборе входной сигнал должен быть 4-битовым двоичным числом, а выходной сигнал – семисегментным кодом, адекватно отображающим это число. Сначала полубайт преобразуется в один из шестнадцати выходных сигналов. Затем этот сигнал выбирает подходящий семисегментный код, преобразуясь в изображение на семисегментном дисплее. Запустите виртуальный прибор **Display7.vi**, реализующий этот алгоритм.

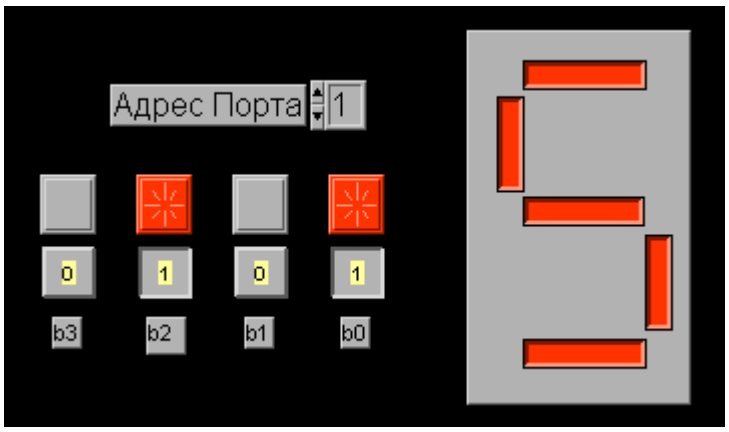


Рисунок 10-8. Лицевая панель преобразователя двоичного числа в семисегментный код.

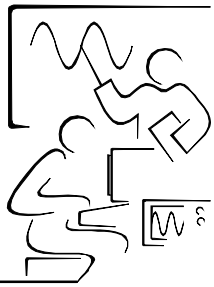
Задание повышенной сложности

Придумайте двухразрядный счетчик, который бы считал от 0 до 99. Для этого модифицируйте 8-битовый двоичный счетчик, представленный на шестом занятии.

Состав библиотеки виртуальных приборов, использованных на занятии 10 (показан в порядке упоминания

- **7Segment.vi** (модель LabVIEW для семисегментного дисплея)
- **Bin->Digit.vi** (4-битовый цифро-аналоговый преобразователь)
- **Encoder Hex.vi** (семисегментный дисплей для работы с шестнадцатеричными числами)
- **Display7.vi** (шестнадцатеричный семисегментный дисплей с преобразователем двоичных чисел)

Занятие 11. Последовательная передача данных.



Многие приборы, контроллеры и компьютеры оснащены последовательным интерфейсом. Возможность взаимодействовать с этими устройствами посредством последовательного интерфейса открывает новую область применимости инструментов измерения и управления. Стандартный интерфейс последовательной передачи данных, RS-232, определяет порядок следования битов и форму сигнала во времени и пространстве. Для передачи данных между компьютером и внешним устройством необходимы, по крайней мере, три линии связи: передачи, приема и опорная (земля).

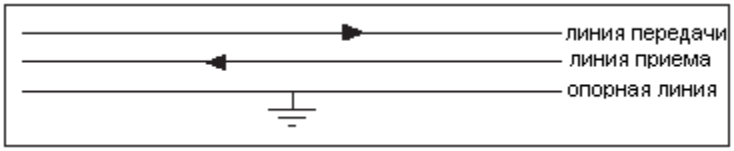


Рисунок 11-1. Линии последовательной передачи данных.

При последовательной передаче данных сигнал с высоким уровнем называется Меткой, а сигнал низкого уровня – Паузой. В обычной работе выходной сигнал в линии передачи имеет высокий уровень и часто обозначается как 1 или в среде LabVIEW – логической переменной Истина. Передатчик сообщает Приемнику, что собирается отправлять данные, переводя линию передачи в состояние Пауза (0). Такой спадающий фронт (изменение уровня сигнала от высокого к низкому) сигнализирует приемнику, чтобы он был готов принимать данные. В протоколе RS-232 все биты данных пересылаются и удерживаются в течение определенного промежутка времени. Этот период времени является обратной величиной скорости передачи, измеренной в Бодах, то есть частоты передачи данных, измеряемой в единицах бит/секунда. Например, скорость передачи данных 300 Бод определяет временной интервал 1/300 секунды или 33.3 мс. В начале каждого такого интервала времени напряжение в линии передачи становится высоким или низким и остается в таком состоянии в течение этого интервала. Все вместе, то есть перевод с уровня на уровень и пребывание на них, формирует последовательный сигнал.

Рассмотрим 8 битовые данные (байт) $3A (или в двоичном виде 00111010). При последовательной передаче протокол требует, чтобы

самый младший разряд, b0, передавался первым, а самый старший, b7, последним. Условно принято, что время течет слева направо, следовательно, байт будет передаваться в виде (01011100), то есть в обратном порядке.

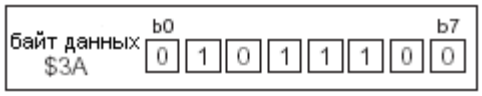


Рисунок 11-2. При последовательной передаче данных самый младший бит (b0) передается первым

Также в протоколе требуется, чтобы каждый байт данных был окружен двумя специальными битами, в начале стартовый бит (пауза), а в конце – стоповый бит (метка).

Рисунок 11-3. Связывающие биты – Стартовый и Стоповый – обрамляют байт данных.

Теперь, при добавлении этих обрамляющих битов, для отправки одного байта данных необходимы десять тактовых интервалов. Если каждый байт представляет символ ASCII, то для посылки одного символа необходимо отправить десять последовательных битов. Например, модем марки 9600 Baud может передавать данные со скоростью 960 символов в секунду. Если представить сигнал при последовательной передаче данных по протоколу RS-232 на временной диаграмме, то байт $3A выглядит следующим образом:

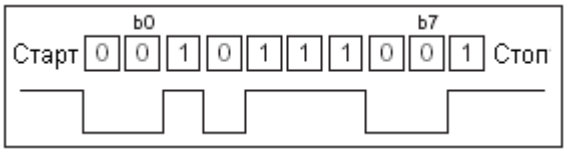


Рисунок 11-4. Форма сигнала при последовательной передаче байта $3A

Последовательный передатчик данных

С помощью пакета LabVIEW можно сконструировать последовательный передатчик данных. Для этого необходим 10-битовый сдвиговый регистр и цикл задержки для моделирования скорости двоичной передачи (в бодах). Запустите виртуальный прибор **Serial.vi**.

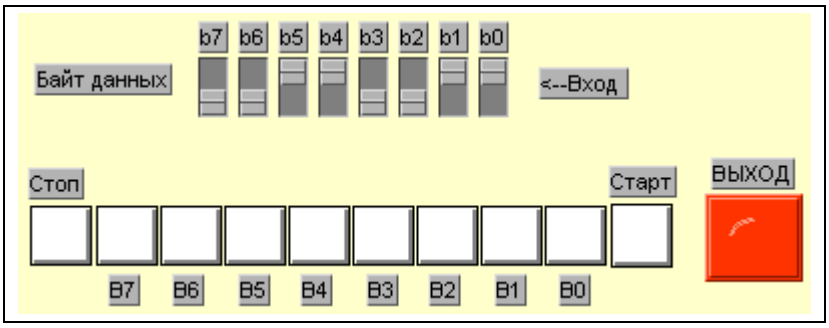


Рисунок 11-5. Моделирование последовательного передатчика данных.

При помощи восьми переключателей на лицевой панели прибора вы можете загрузить байт данных в сдвиговый регистр. Обратите внимание, что самый старший разряд в шестнадцатеричной системе располагается слева. Следовательно, число $33 вводится как (00110011). Однако данные выходят в обратном порядке, когда самый младший разряд идет первым. Последовательный выход отображается на большом квадратном светодиодном индикаторе. Вначале идет Метка. Перед выполнением программы все биты данных и окружающие их биты показаны нулями. Как только кнопка запуска нажимается, символ $33 загружается в сдвиговый регистр, стоповый бит становится логической единицей, а стартовый нулем. Выходной бит немедленно становится логическим нулем, сигнализируя о начале передачи данных. После задержки (1/скорость передачи), на выход поступает следующий бит.

На блок-диаграмме можно увидеть алгоритм работы передатчика.

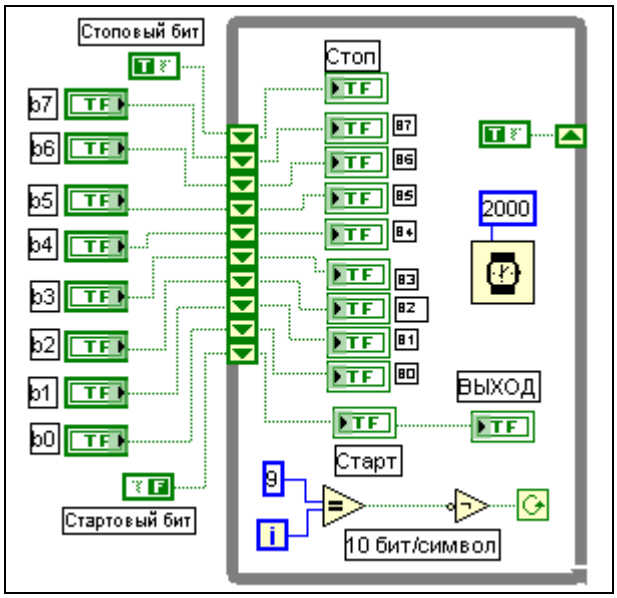


Рисунок 11-6. Блок-диаграмма прибора, моделирующего последовательный передатчик данных.

Первый бит, который выводится (Старт), инициализирован к состоянию Пауза (0) логической переменной Ложь. Следующие восемь битов – это байт данных в последовательности от самого младшего к самому старшему разряду. Последний элемент в сдвиговом регистре (стоповый бит), инициализирован к состоянию Метка (1) логической переменной Истина. После запуска виртуальный прибор выполняется циклически 10 раз. В каждом цикле на выход последовательно выводится один бит. Элемент «ожидание» моделирует тактовый интервал или скорость в единицах 1/бод.

Как только данные поступают в линию передачи, сдвиговый регистр заполняется единицами. Это необходимо для обеспечения состояния «Метка» в конце передачи данных, то есть после 10 циклов.

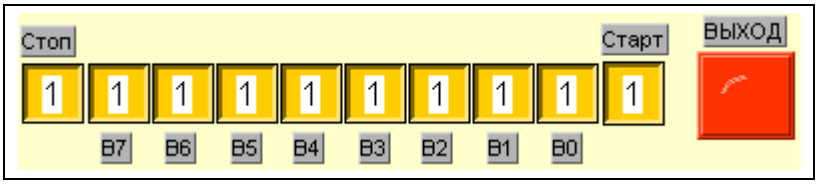


Рисунок 11-7. Буфер передатчика после отправки данных в порт.

Наиболее легко наблюдать форму сигнала при последовательной передаче данных, если подать выходной сигнал на осциллограф или самописец. Во втором виртуальном приборе, **Serial1.vi**, выходной сигнал преобразуется в число, а затем отображается на графическом индикаторе. Выбирая подходящий набор символов точек, отображающихся на графике, а также интерполяцию линии, соединяющей эти точки, можно получить график, похожий на тот, который получается в осциллографе. После этого при больших тактовых интервалах вы можете наблюдать форму сигнала, посылаемого при последовательной передаче данных.

На следующих примерах показываются формы сигналов для чисел $00 (00000000), $55 (01010101) и $FF (11111111).

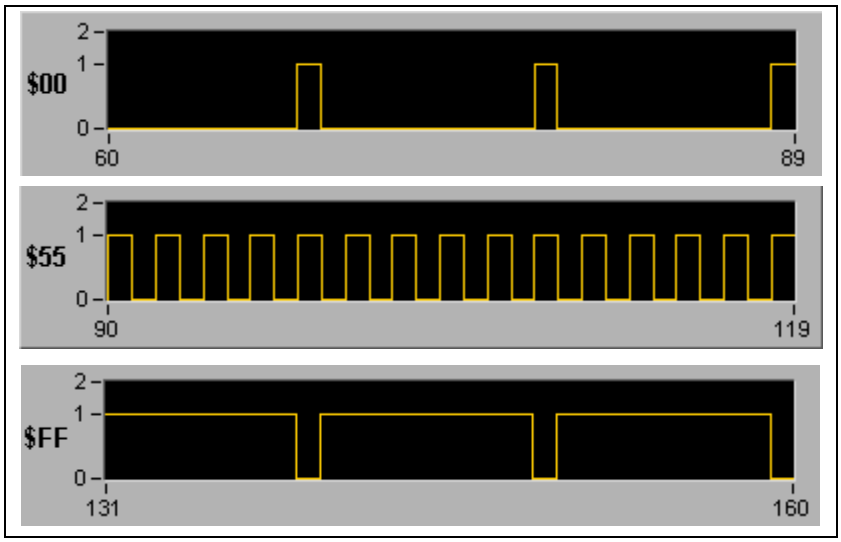


Рисунок 11-8. Форма сигналов при последовательной повторной передаче данных.

Отметим, что в промежуточном случае $55, генерируется сигнал прямоугольной формы.

Построенную модель преобразователя параллельного кода в последовательный можно сохранить как подпрограмму и использовать в других виртуальных приборах. В общих чертах в этой модели будут иметься восемь двоичных входов для параллельного ввода байта данных, двоичный выход для последовательного потока данных и числовая матрица для построения графика сигнала.

Преобразователь аналогового напряжения в последовательный код

В первом примере аналоговый ввод моделировался числами. На эти числа накладывалось ограничение: они не должны выходить за пределы 0-255. Виртуальный прибор **RampADC.vi**, представленный на девятом занятии, преобразовывал аналоговый сигнал в 8-битовое двоичное число. Оно, в свою очередь, подавалось на преобразователь параллельного кода в последовательный. Для наблюдения сигнала этот код заполнял матрицу и затем отображался на лицевой панели в виде графика.

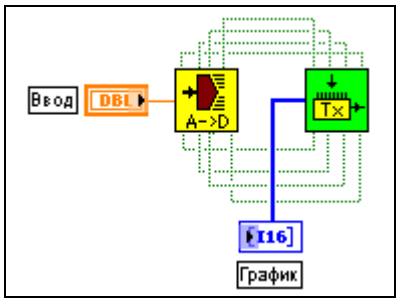


Рисунок 11-9. При моделировании последовательного передатчика данных используются подпрограммы.

Запустите виртуальный прибор **V->Serial.vi** и наблюдайте последовательные сигналы. Каждое число в пределах от 0 до 255 вырабатывает сигнал различной формы. Попробуйте ввести байты данных $00, $55 и $FF для проверки графиков, изображенных на рисунке 11-8.

Во втором примере строковая переменная ASCII, состоящая из двух символов шестнадцатеричной системы счисления, поступает на вход подпрограммы **Hex->Numeric.vi**, которая преобразует такие символы в число.

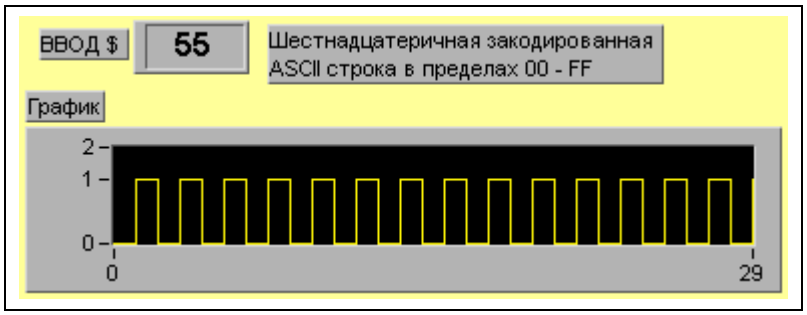


Рисунок 11-10. Генератор сигналов с шестнадцатеричным входом.

Шестнадцатеричная строковая переменная преобразуется в численную величину при помощи функции LabVIEW, называемой **Format and Strip**. После этого число поступает на предыдущий виртуальный прибор **V->Serial.vi** и отображается на индикаторе. Вспомните, что число (01010101) генерировало в **Serial1.vi** прямоугольный сигнал. В виртуальном приборе **HEX->Serial.vi** число $55 также генерирует прямоугольный сигнал.

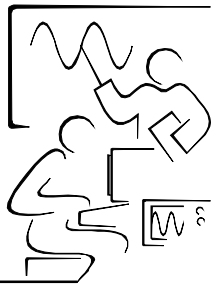
Задание повышенной сложности

Попробуйте обобщить этот виртуальный прибор для ввода произвольного 7-битового символа ASCII. Восьмой бит может выполнять функцию контрольного бита четности: четный – не четный.

Состав библиотеки виртуальных приборов, использованных на занятии 11 (показан в порядке упоминания)

- **Serial.vi** (демонстрирует последовательный передатчик данных)
- **Serial1.vi** (**Serial.vi** с выходом на графический индикатор)
- **V->Serial.vi** (моделирование последовательного передатчика данных)
- **Hex->Serial.vi** (моделирование последовательного передатчика данных с вводов шестнадцатеричных чисел)
- **RampADC.vi** (подпрограмма – 8-битовый АЦП)
- **Binary Counter.vi** (подпрограмма – 8-битовый счетчик)
- **DAC.vi** (подпрограмма – 8-битовый ЦАП)
- **FlipFlop.vi** (подпрограмма – триггер со счетным входом)
- **Hex->Numeric.vi** (подпрограмма, преобразующая шестнадцатеричное число в десятичное)
- **Serial2.vi** (подпрограмма, используемая в **Serial1.vi**, с выходом в виде числовой матрицы)

Занятие 12.
Центральный процессор.



Основной элемент любого компьютера – это центральный процессор (ЦПУ). ЦПУ связан с оперативной памятью посредством шины, передающей данные в двух направлениях. Команды, постоянные и переменные величины, используемые в программах и постоянно хранящиеся в памяти, расположены в определенной последовательности. ЦПУ обращается к этой последовательности, управляя и манипулируя адресной шиной. Специальные ячейки памяти, называемые входным/выходным (I/O) портами, передают двоичную информацию из- или во- внешний мир в форме параллельных или последовательных байтов данных. Системный тактовый генератор осуществляет управление всеми логическими элементами, триггерами и регистрами, обеспечивая, чтобы все биты были в нужное время в нужном месте и чтобы потоки данных не перекрывались. Из всех частей компьютера (ЦПУ, память, устройства ввода-вывода и тактовый генератор) процессор является наиболее важной.

ЦПУ состоит из нескольких элементов. Среди них: арифметико-логическое устройство (АЛУ), дешифратор команд, счетчик команд и целый набор внутренних ячеек памяти, называемых регистрами. При обычной работе ЦПУ счетчик команд обращается к памяти посредством адресной шины с целью получения следующей инструкции. Эта инструкция по шине данных поступает во внутренние регистры. Первая часть инструкции передается в дешифратор, который решает, какие каналы необходимо открыть и закрыть, чтобы выполнить инструкцию. В некоторых случаях инструкция содержит всю необходимую для работы информацию. В качестве примера такой инструкции можно назвать команду «очистить сумматор». В других случаях для выполнения инструкции необходима дополнительная информация, поэтому снова происходит обращение к памяти за этой информацией. В качестве примера такой инструкции можно привести команду «загрузить в Регистр 2 постоянную величину 5». Как только вся информация соберется в одном месте, инструкция выполняется посредством открывания и закрывания различных логических элементов.

Типичные команды, доступные для всех ЦПУ, состоят из простых операций с данными, уже находящимися внутри самого ЦПУ. Среди них очистка, дополнение или приращение сумматора. В более сложных инструкциях используются два внутренних регистра или данные, поступающие из памяти. На данном занятии при помощи основных

логических функций будет показано, как ЦПУ выполняет простые и некоторые более сложные операции.

Работа арифметико-логического устройства.

Арифметико-логическое устройство (АЛУ) – это набор программируемых двухвходовых логических элементов, оперирующих с параллельными массивами данных, размером 4, 8 , 16 и 32 бита. На данном занятии основное внимание будет сфокусировано на изучении 8-битовых ЦПУ. Входные регистры будут называться Регистр1 и Регистр 2, и для простоты результат операции будет загружаться в третий регистр, называемый выходным. Тип команды (И, ИЛИ, исключающее ИЛИ) мнемонически определяется как И R1, R2.

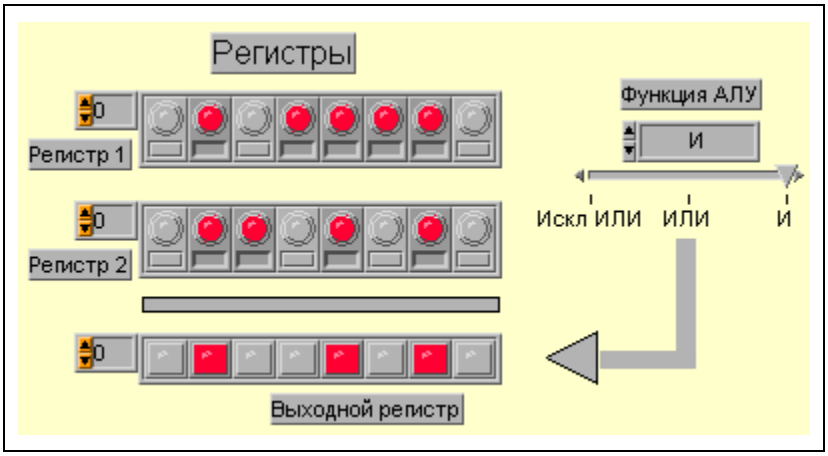


Рисунок 12-1. Моделирование арифметико-логического устройства.

При моделировании в среде LabVIEW – в виртуальном приборе **ALU0.vi** – для представления регистров R1 и R2 используются 1D массивы, состоящие из булевых переключателей. Выходной регистр представляет из себя массив булевых индикаторов. Функции (И, ИЛИ, исключающее ИЛИ) можно выбирать при помощи ползункового переключателя. Данные заносятся во входные регистры нажатием на кнопку, находящуюся ниже соответствующего бита. При нажатии на кнопку «Run» происходит выполнение выбранной логической операции.

Ниже следует ряд примеров элементарных операций ЦПУ.

Что представляют собой следующие операции:

И R1[$00], R2[$XX]

ИЛИ R1[$FF], R2[$XX]

Исключающее ИЛИ R1[$55], R2[$FF]?

Во всех указанных случаях данные, которые необходимо ввести, заключены в прямоугольные скобки [ ] и должны представлять шестнадцатиричное число типа $F3. X обозначает любой

шестнадцатиричный символ. Исследуйте представленные операции при помощи виртуального прибора **ALU0.vi**.

Операция И устанавливает все разряды выходного регистра в 0, следовательно, это действие эквивалентно команде ОЧИСТИТЬ Выходной Регистр. Операция ИЛИ устанавливает все биты выходного регистра в 1, следовательно, это действие эквивалентно команде УСТАНОВИТЬ Выходной Регистр в 1. Третья операция инвертирует разряды R1, то есть, эквивалентна операции ДОПОЛНЕНИЕ Регистра 1.

Рассмотрим команду «Загрузить в выходной регистр содержимое Регистра 1». На языке обычного программирования данную команду можно записать в виде «Выход = Регистр1». Установите R1 в **ALU0.vi** в некоторое исходное состояние и выполните операцию И R1,R2[$FF].

Еще одна интересная комбинация – Исключающее ИЛИ R1, R2 – реализует другую распространенную команду – ОЧИСТИТЬ R1. Из этих примеров становится ясным, что многие команды ЦПУ, имеющие специфическое содержание в контексте реализации программного обеспечения, выполняются при помощи основных логических элементов, представленных на занятии 1.

Накапливающий сумматор

В виртуальном приборе **ALU0.vi** команды ЦПУ выполняются посредством «выдергивания» одного бита за одну итерацию при помощи функции LabVIEW **Index Array**. После этого выполняется операция арифметико-логического устройства над этим битом. Результат передается в выходной массив под тем же самым индексом при помощи функции **Replace Array Element**. После восьми циклов каждый бит (0…7) проходит через АЛУ. После этого работа ЦПУ завершается.

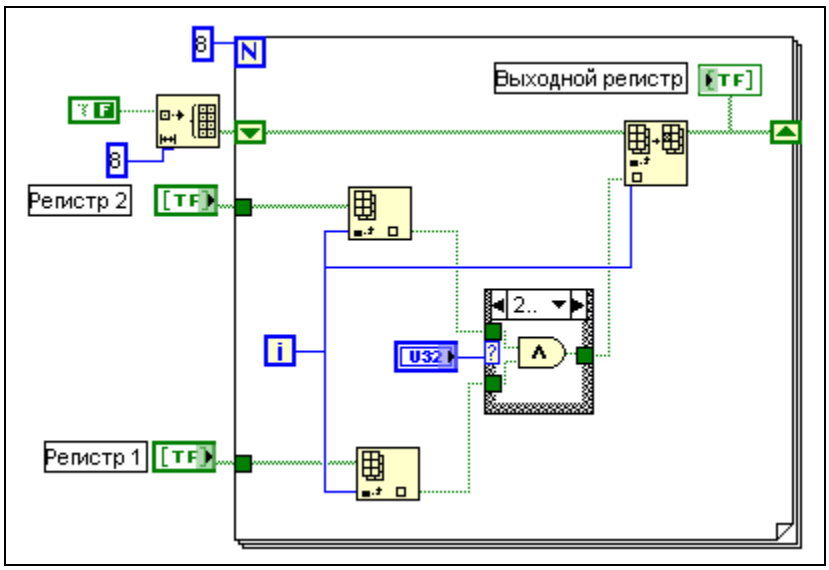


Рисунок 12-2. Прибор, моделирующий работу 8-битового АЛУ.

В среде LabVIEW нет необходимости «выдергивать» каждый бит, поскольку эта задача может быть сделана автоматически посредством отключения индексации в туннеле структуры «Цикл по Условию». Проводники данных, представляющих массивы, изображаются жирной линией, в противоположность тонкой линии, изображающей проводник передачи простых данных внутри цикла. Будьте внимательны при рассмотрении следующего примера, в котором используется эта особенность LabVIEW.

Во многих процессорах второй входной регистр, R2, соединен с выходным регистром, так что на следующей итерации выходное состояние становится входным. Такая структура позволяет упростить устройство ЦПУ, но что более важно, при этом выходной регистр становится накопителем.

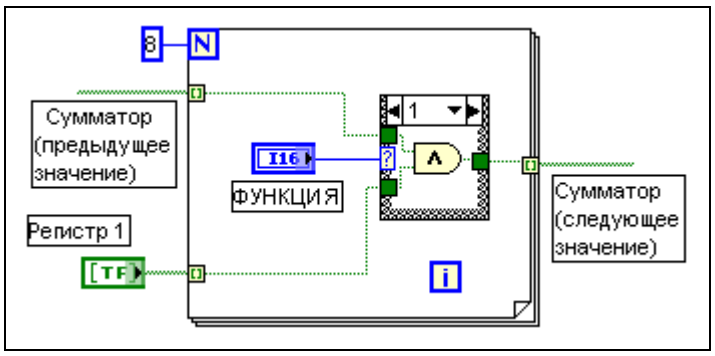


Рисунок 12-3. При моделировании АЛУ используется автоиндексация внутри туннеля цикла по условию.

В виртуальном приборе **ALU1.vi** предыдущее значение величины, накопленной в сумматоре, поступает в цикл слева, а следующее значение величины сумматора выходит справа. Такой алгоритм позволяет последовательно реализовать выполнение отдельных инструкций ЦПУ. Взгляните на следующий пример: загрузить в сумматор число 5, затем выполнить операцию взятия дополнения содержимого сумматора.

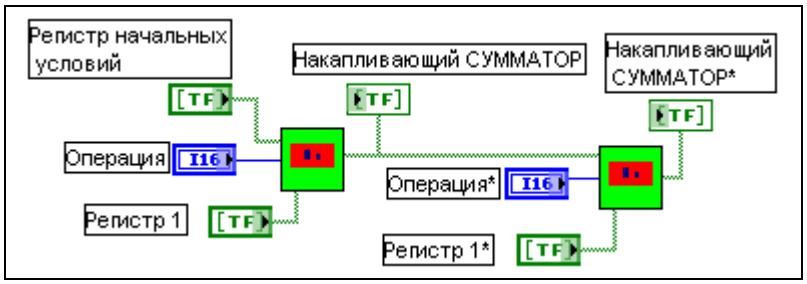


Рисунок 12-4. Виртуальный прибор, моделирующий загрузку в сумматор числа 5 и взятие дополнения содержимого сумматора.

Первая команда – загрузить в сумматор число 5 – выполняется при помощи функции И, а также наложения маски (1111 1111). Двоичное представление числа 5 (0000 0101) помещается в Регистр начальных условий, а маска $FF в R1. При выполнении операции И число 5 загружается в R1, а затем в накапливающий сумматор. Взятие операции Дополнение происходит при помощи функции Исключающее ИЛИ и маски $FF. Запустите виртуальный прибор **Prgm1.vi** и наблюдайте его работу. Дополнение содержимого сумматора находится в регистре с названием Накапливающий СУММАТОР*.

Операция сложения

Арифметико-логическое устройство выполняет не только логические операции, оно предназначено и для выполнения арифметических действий. На занятии 3 мы изучили, что при помощи функции Исключающее ИЛИ происходит добавление отдельных битов при двоичном сложении, а вычисление сигнала переноса происходит при помощи функции И. При объединении этих функций получается полусумматор (бит 1 + бит 2 = сумма + сигнал переноса (если есть)). Чтобы результат сложения сдвинуть в следующий разряд, используется полный сумматор, в котором происходит сложение битов, поступающих с двух входов, а также учитывается сигнал начала переноса из предыдущего разряда. Виртуальный прибор **ADD_c.vi** добавляет действие Сложение к операциям АЛУ. Полный сумматор, показанный ниже, складывает два входных бита, учитывая при этом сигнал о начале переноса из предыдущего разряда. Это действие моделируется при помощи булевого сдвигового регистра. Новая операция – {СУММА +1,A} – может быть теперь добавлена в структуру Варианта и к списку функций в виртуальном приборе.

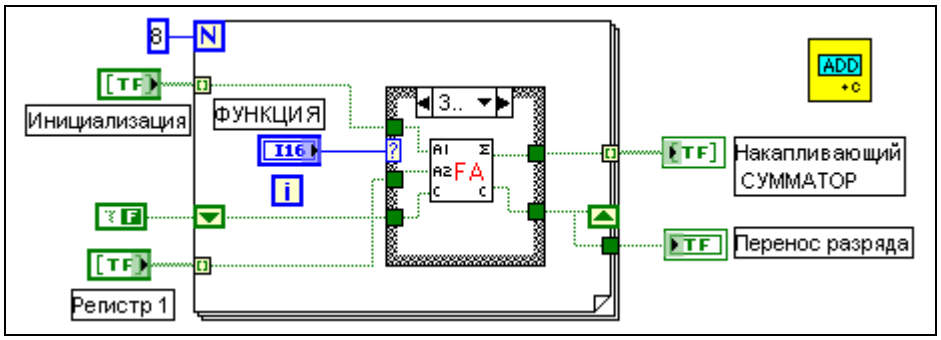


Рисунок 12-5. Операция АЛУ – сложение с переносом разряда.

Двоичный счетчик

Рассмотрим программу, генерирующую двоичные числа при помощи 8-битового двоичного счетчика. Ее работу можно описать следующим образом: после очистки сумматора, добавлять к нему единицу снова и снова в течение n раз.

На каком либо языке линейного программирования эту программу можно записать в виде:

Start	CLEAR A	: обнулить все разряды в сумматоре
Loop	INC A	: увеличить на 1 содержимое сумматора
	REPEAT Loop N	: выполнить последнюю команду n раз

Выполнение над содержимым регистра и числом $00 операции И приведет к очистке сумматора, CLEAR A. В списке функций ЦПУ операция И имеет номер 1. Операция INC A – это И 1,A – и идет под номером 3 в списке. Моделирование представлено ниже.

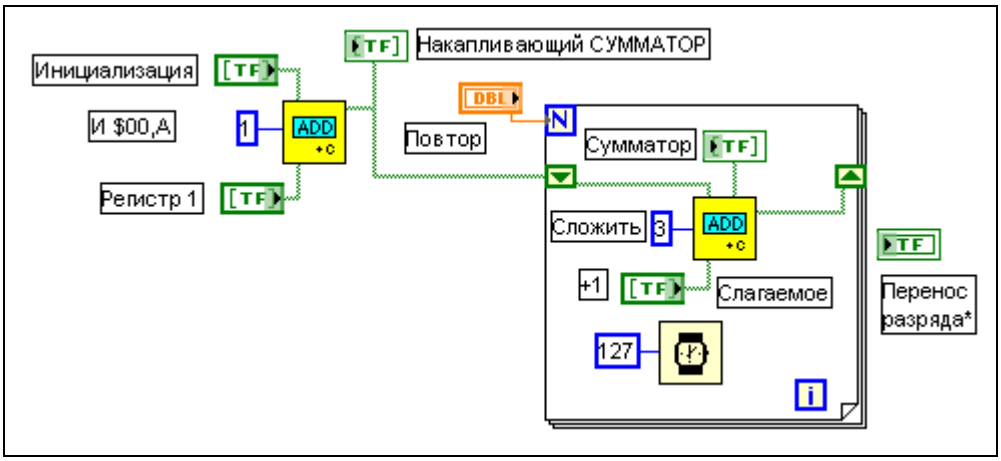


Рисунок 12-6. Виртуальный прибор, моделирующий 8-битовый счетчик.

Отметим, что терминал сигнала о переносе разряда ни с чем не соединен. Команда ПРИРАЩЕНИЕ не оказывает влияния на перенос. При соединении терминала сигнала переноса данная команда правильно запишется как И + 1,A. Запустите виртуальный прибор **Prgm2.vi** и снова наблюдайте работу счетчика. Чтобы пользователь мог с легкостью наблюдать за работой прибора, в цикл была добавлена функция Ожидание.

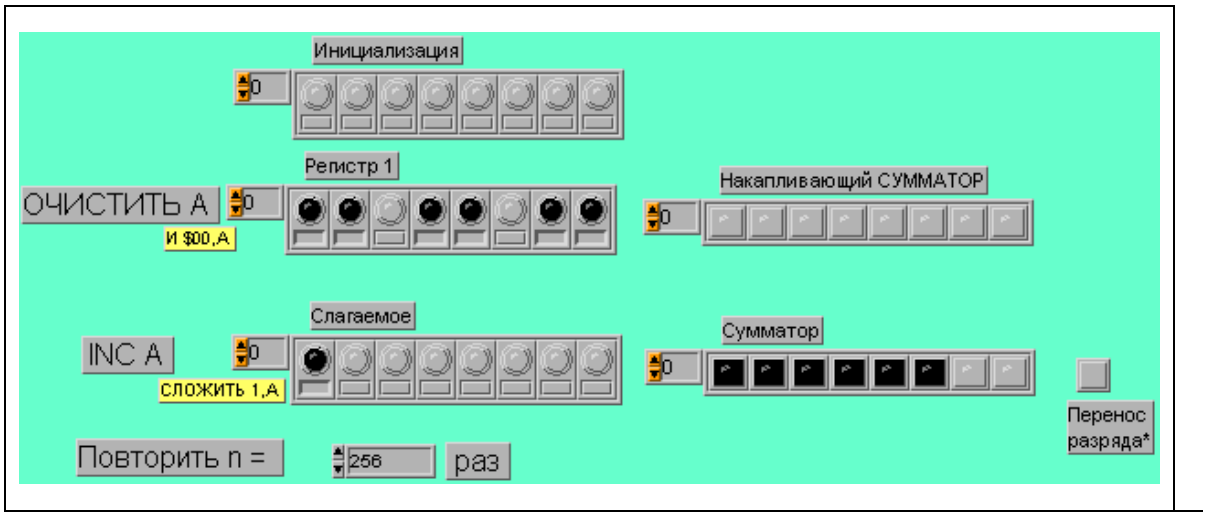


Рисунок 12-7. Лицевая панель прибора, моделирующего 8-битовый счетчик.

Задание повышенной сложности

Придумайте программу для моделирования сложения двух 16-битовых чисел.

Состав библиотеки виртуальных приборов, использованных на занятии 12 (показан в порядке упоминания)

- **ALU0.vi** (Моделирование АЛУ, логических элементов И, ИЛИ, Искл. ИЛИ)
- **ALU1.vi** (Оптимизированная модель АЛУ)
- **ADD_c.vi** (Моделирование операций АЛУ – И, ИЛИ, Искл. ИЛИ, Сложение)
- **Prgm1.vi** (Моделирование операций ЦПУ: загрузка в сумматор числа 5, взятие дополнения содержимого сумматора)
- **Prgm2.vi** (Моделирование двоичного счетчика: очистка А, СЛОЖИТЬ 1 и А)
- **Half Adder.vi** (подпрограмма, используемая при моделировании операции сложения в ЦПУ)
- **Full Adder.vi** (подпрограмма, используемая при моделировании операции сложения в ЦПУ)