Проектирование детектора последовательности импульсов в LabVIEW для оценочного модуля DE FPGA Board

Цель работы:

Целью этой лабораторной работы является разработка конечного автомата (finite state machine – FSM) с использованием структуры case. Результаты проектирования вы проконтролируете на оценочном модуле DE FPGA и системе NI ELVIS.

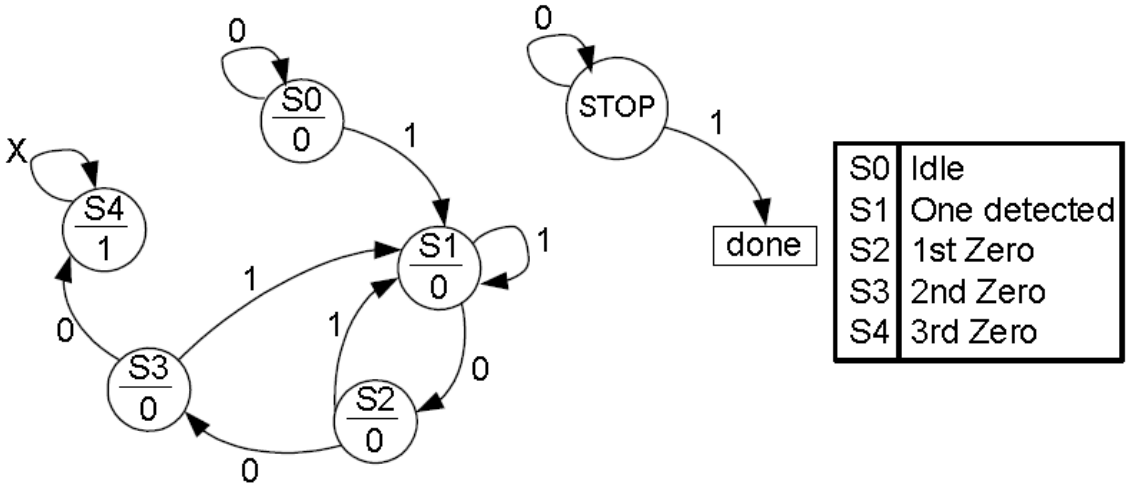
Задание на проектирование:

Разработать детектор для обнаружения последовательности 1000. Детектор последовательности имеет два входа – вход данных (data) и вход останова (Stop), а также один выход (Y). Выход Y принимает значение 1 при обнаружении требуемой входной последовательности и остается в этом состоянии, пока не будет нажата кнопка Stop, независимо от состояния входов.

Выполнение задания:

Для проектирования устройства нужно разработать граф (диаграмму) состояний. Граф состояний, соответствующий заданию, изображен ниже. Он состоит из 5 вершин (состояний), обозначенных S0÷S4, обозначения абстрактные и не отражают целевое назначение проектируемого устройства. Потом в нашем устройстве мы состояниям присвоим значимые имена, используемые в проекте (они приведены в таблице). В диаграмме состояний показан также автономный путь для кнопки STOP, при нажатии на которую выполнение задачи прекращается.

Тактирование FPGA осуществляется высокой частотой, и будет сложно, управляя входами, наблюдать промежуточные состояния выхода. Поскольку наблюдение промежуточных состояний важно, мы снизим частоту тактирования. Один из способов снижения частоты – использовать более низкочастотный генератор и сообщить об этом компилятору LabVIEW. Другой способ предоставляет функция Loop Timer из субпалитры Timing, которую можно настроить на требуемую задержку. В нашем проекте мы используем задержку в 1000 ms (1 секунда), это обеспечит достаточное время для приема входных данных, а мы сможем увидеть процесс функционирования детектора.

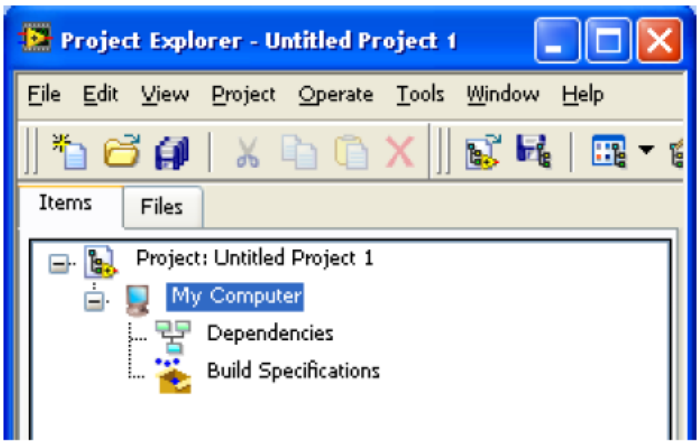


Для быстрого освоения LabVIEW выполните курс “Learn LabVIEW in 3 hours” (http://www.ni.com/academic/learn_LabVIEW/)

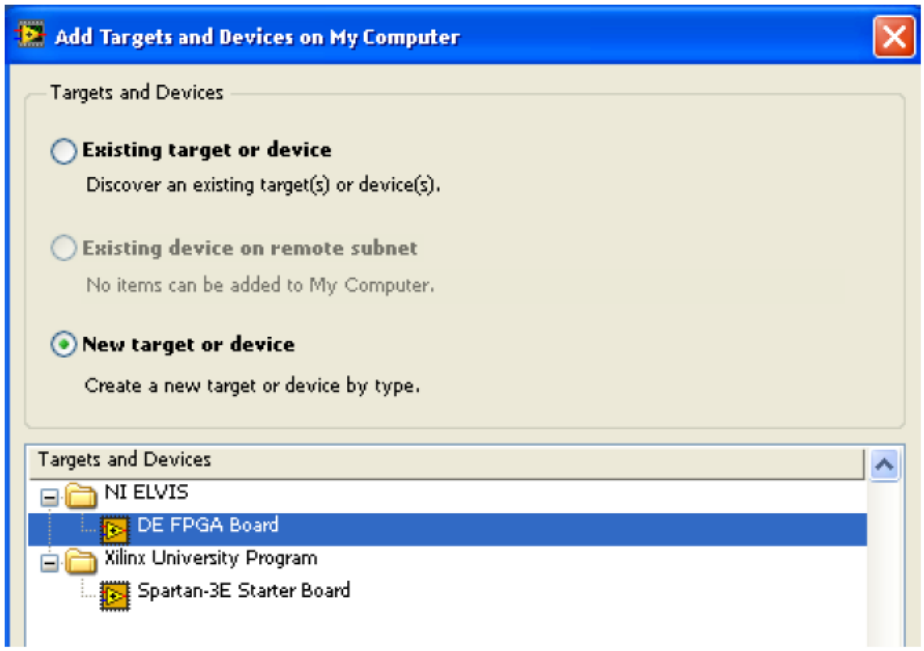
Раздел 1: разработка устройства

Порядок выполнения:

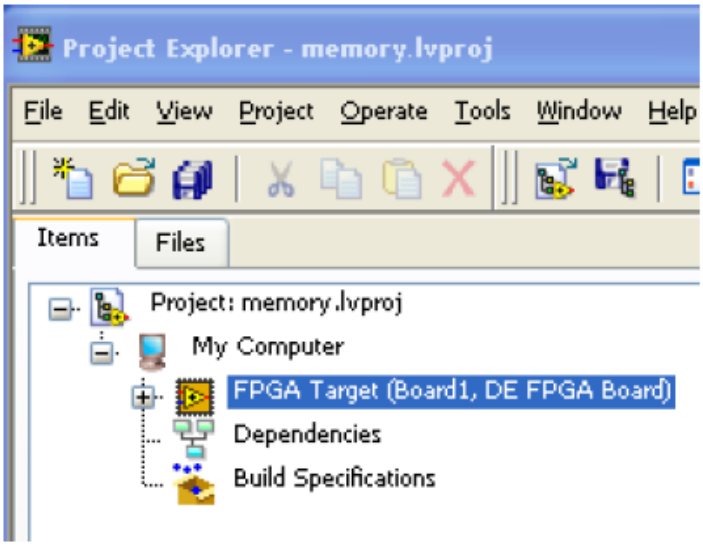
- Разархивируйте файл **sequencedetect_Lab.zip** в папку **c:\NI\LabVIEW_Labs**
- Запустите на исполнение LabVIEW **(Start>>All Programs>>National Instruments>>LabVIEW)**
- Выберите вариант **File>>New Project**
- В окне *Project Explorer* щелкните правой кнопкой мыши по пункту **My Computer** и выберите **New>>Targets and Devices**



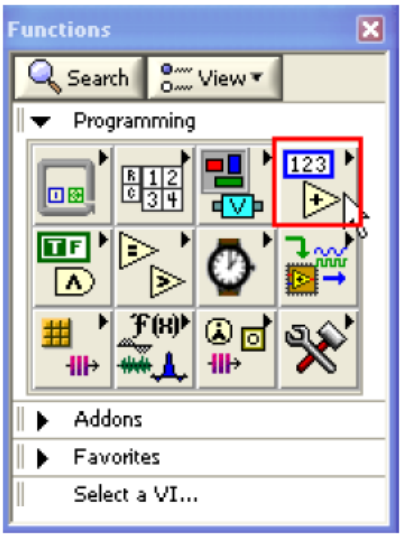
- Установите флажок в позиции *New target or device* и выберите **DE FPGA Board** в секции **NI ELVIS**



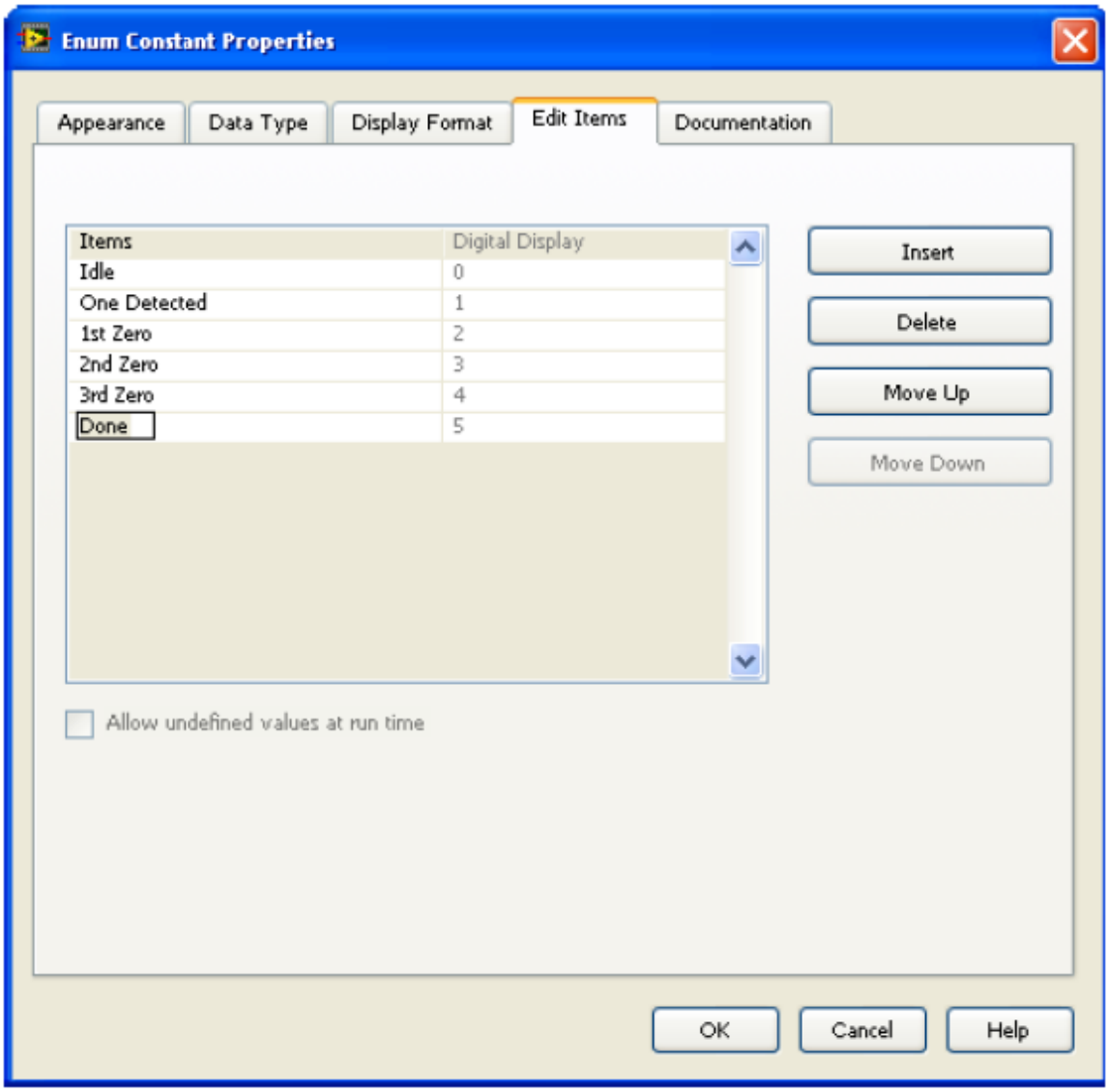
- Щелкните по кнопке OK
- Выберите в меню **File>>Save** и сохраните проект под именем **sequencedetect** в папке **c:\NI\LabVIEW_Labs\ sequencedetect_Lab**
- В окне *Project Explorer* щелкните правой кнопкой мыши по пункту **FPGA Target (Board1, DE FPGA Board**) и выберите **New VI**



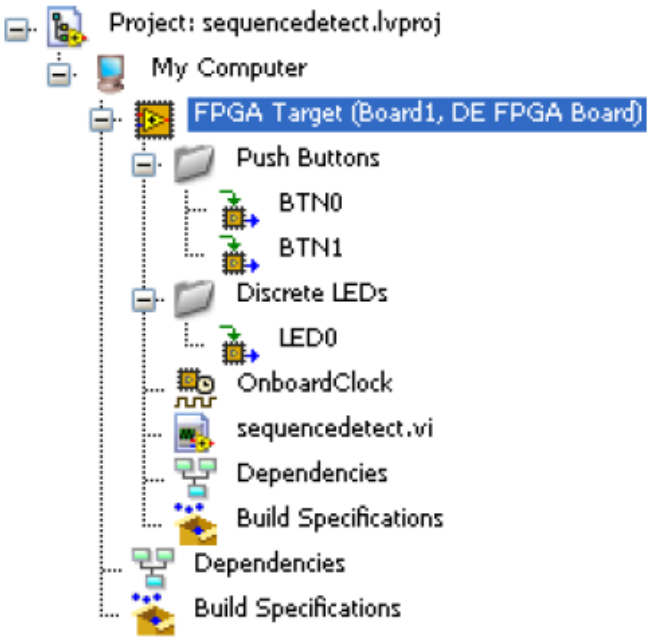
- Откройте окно **Block Diagramm** и щелкните правой кнопкой мыши где-нибудь на белом поле (в рабочей области). В палитре функций откройте субпалитру **Numeric**



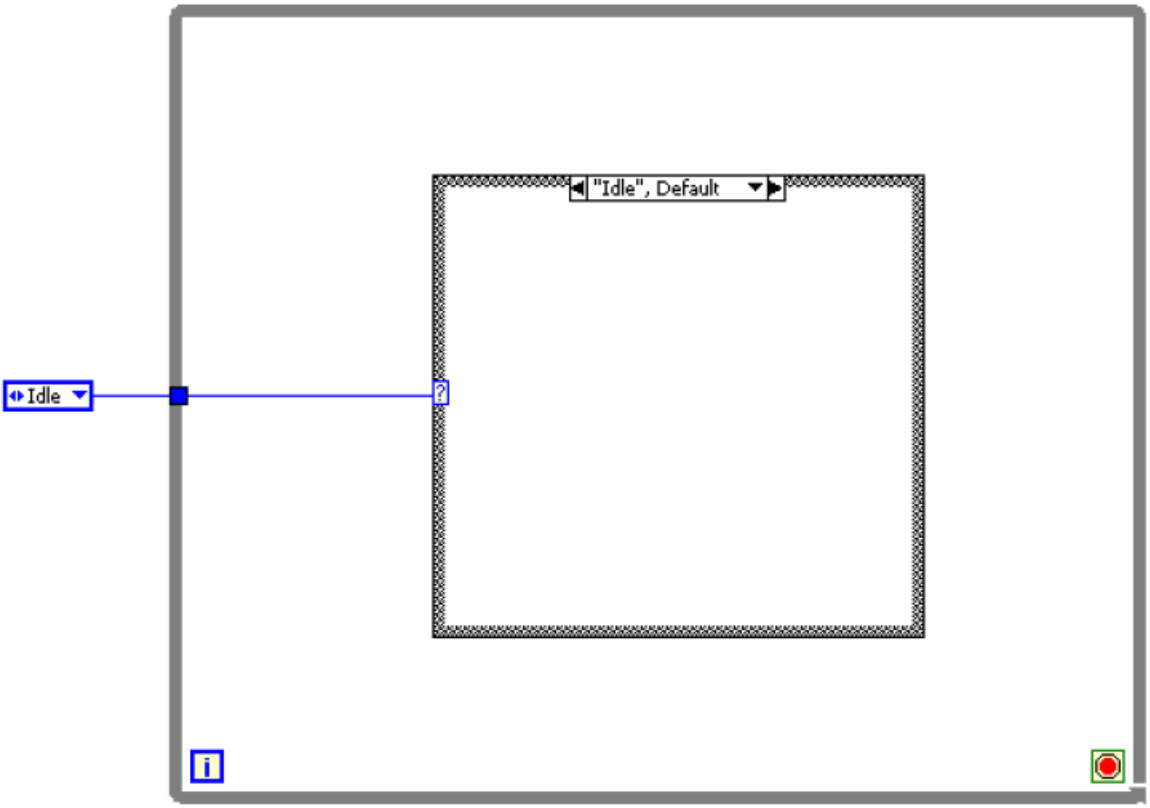
- Выберите **Enum Constant** и поместите ее в рабочей области блок-диаграммы
- Наведите курсор на установленный элемент и обратите внимание, что он принял вид указателя. Щелкните правой кнопкой мыши и выберите пункт **Edit Items…** В колонке *Items* введите **Idle**, после нажатия на клавишу *Enter* курсор переместится на следующую строку. Введите **One detected** в качестве второго состояния. Продолжайте вводить состояния **1st Zero**, **2nd Zero**, **3rd Zero** и **Done**. Щелкните по кнопке OK после ввода Done.



- Выберите в меню **File>>Save** и сохраните VI *под именем* **sequencedetect** в папке **c:\NI\LabVIEW_Labs\ sequencedetect_Lab**
- Теперь добавляем каналы ввода-вывода FPGA
- В окне *Project Explorer* щелкните правой кнопкой мыши по пункту **FPGA Target (Board1, DE FPGA Board**) и выберите **New>>FPGA I/O**
- Доступные в оценочном модуле (DE FPGA Board) каналы ввода-вывода отображаются в левой части окна выбора
- В секции **Available Resources** раскройте папку **Push Buttons**, выберите **BTN0** и **BTN1** (BTN0 – в качестве входа данных, BTN1 – для завершения работы детектора последовательности). Щелкните по кнопке Add, чтобы добавить каналы ввода-вывода в проект. Аналогично, раскройте папку **Discrete LEDs**, добавьте **LED0** и щелкните по кнопке Add
- Щелкните по кнопке **OK** для подтверждения выбора и закройте окно выбора
- Ниже на рисунке показано, как должно выглядеть окно **Project Explorer** с внесенными дополнениями

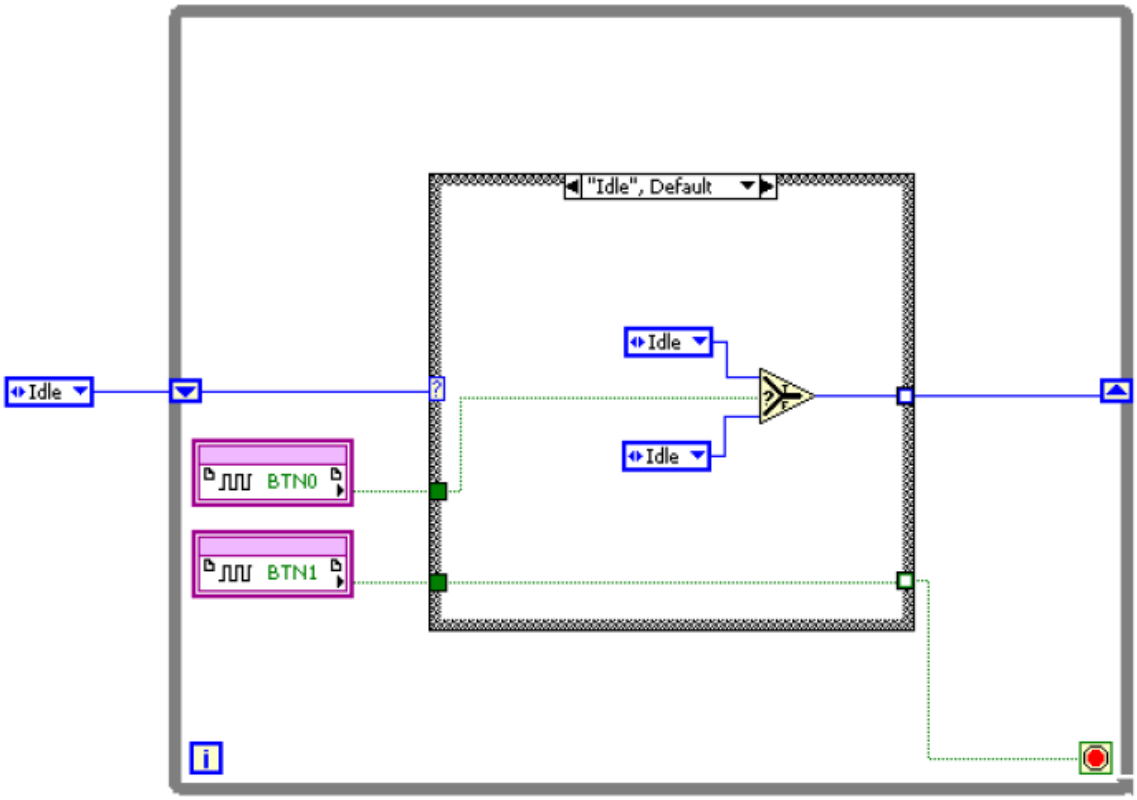


- В окне блок-диаграммы поместите цикл **while loop**, а в него структуру **case**
- Константу Enum, расположенную вне цикла, соедините с селектором выбора структуры case, проведя проводник внутрь цикла while
- Удостоверьтесь, что выбран фрейм Idle структуры case



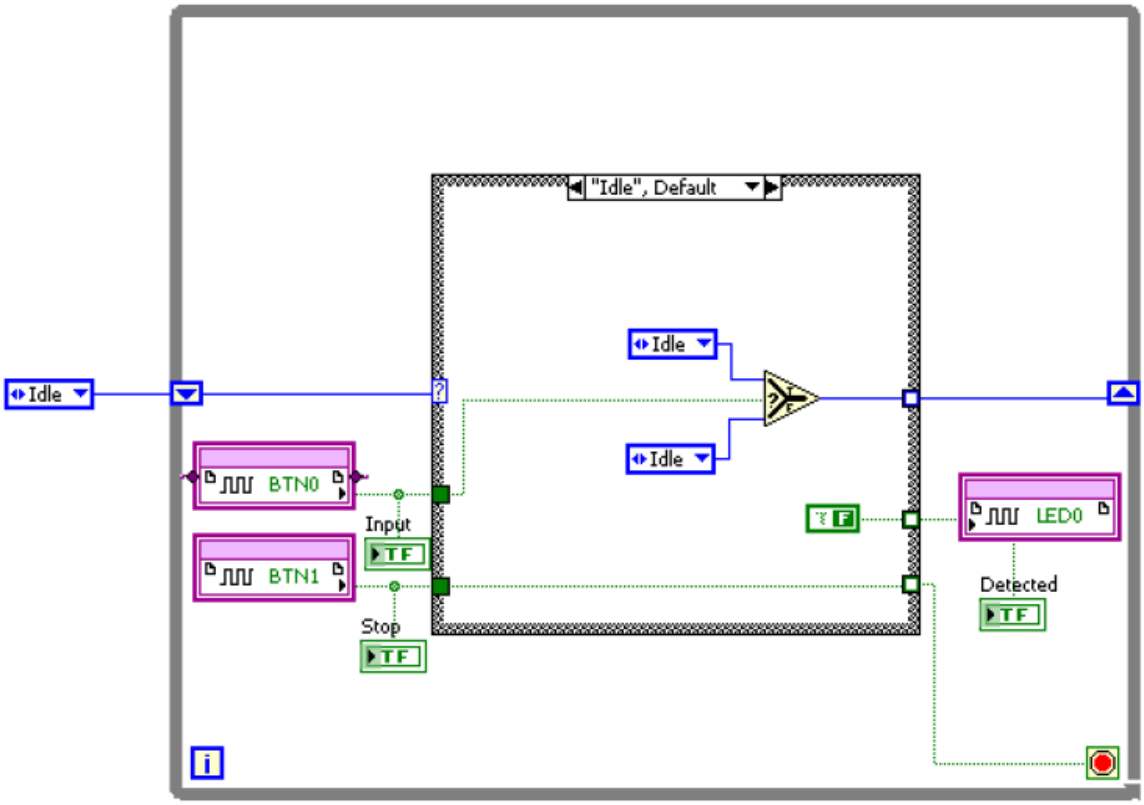
- Выберите входной туннель цикла while, к которому подключена константа Enum, щелкните по нему правой кнопкой мыши и выберите **replace with shift register…**

- В окне **Project Explorer** выберите по одному элементы *Buttons* и перетащите их на рабочую область блок-диаграммы внутрь цикла while, но вне структуры case
- Поместите внутрь структуры case функцию **select** из палитры **comparison**
- Соедините выход функции **select** через границу структуры case с правым терминалом сдвигового регистра (shift register)
- Скопируйте и вставьте константу Enum для подключения ко входам true и false функции select
- Выполните соединения, как показано ниже

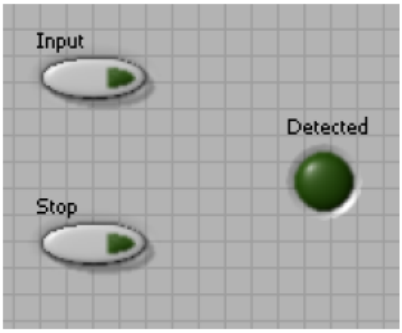


Мы также хотим контролировать выходной сигнал, который должен обновляться при каждом состоянии, и его можно получить из структуры case

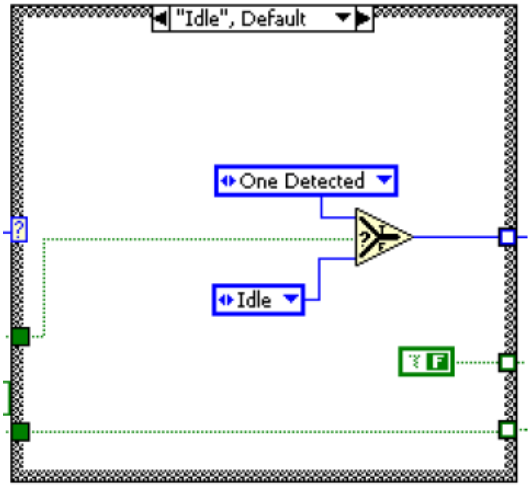
- Добавьте константу **False** из палитры **Boolean** в структуру case
- В окне Project Explorer из папки **Discrete LEDs** выделите элемент **LED0** и перетащите его внутрь цикла while, но снаружи структуры case
- Соедините константу **False** из структуры case с элементом LED0
- Создайте индикаторы для элементов BTN0, BTN1 и LED0, присвойте им имена **Input**, **Stop** и **Detected** соответственно
- Упорядочьте, если необходимо, блок-диаграмму
- Теперь блок-диаграмма должна выглядеть, как показано на рисунке ниже



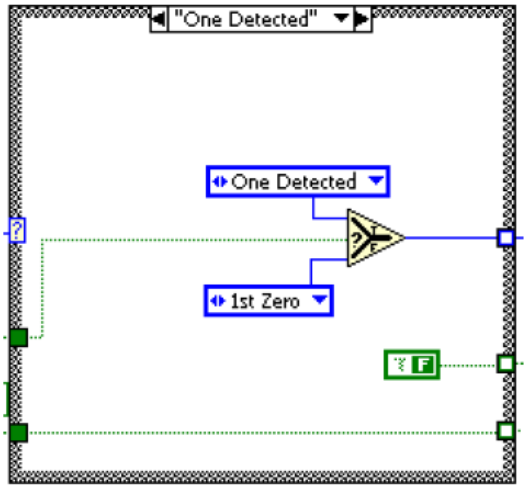
- Переключитесь на лицевую панель (Ctrl-E), на ней видны две входные кнопки и индикатор выходного сигнала. Если хотите, можете изменить их форму, размеры и положение



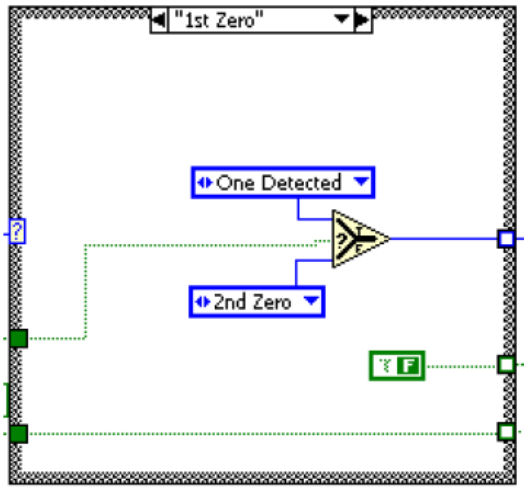
- Сохраните VI
- Теперь нам нужно продублировать фреймы структуры case и присвоить им состояния, используя булевские константы, в соответствии с требуемой функцией
- Щелкните правой кнопкой мыши по границе структуры case и выберите **Add structure for every case**
- Щелкните вверху по индикатору текущего фрейма и выберите вариант **Idle**. Щелкните по константе, соединенной со входом true функции select, и выберите **One Detected**. Оставьте булевскую константу **False** для выходного туннеля case, как индикатор того, что искомая последовательность не обнаружена. Фрейм структуры case должен выглядеть так



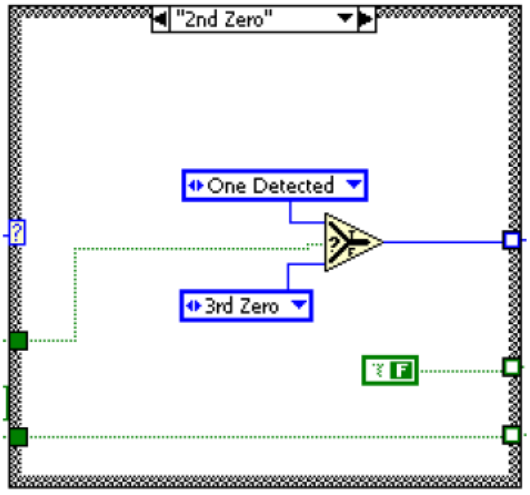
- Щелкните вверху по индикатору текущего фрейма и выберите вариант **One Detected**. Этот фрейм должен выглядеть так



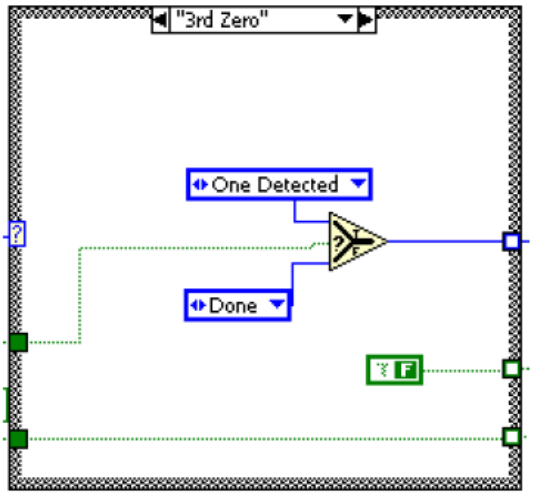
- Щелкните вверху по индикатору текущего фрейма и выберите вариант **1st Zero**. Этот фрейм должен выглядеть так



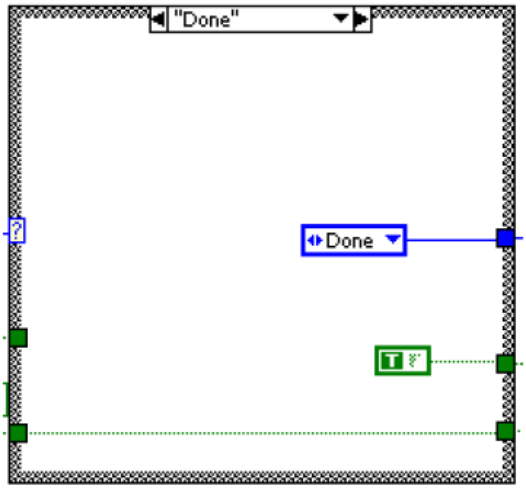
- Щелкните вверху по индикатору текущего фрейма и выберите вариант **2nd Zero**. Этот фрейм должен выглядеть так



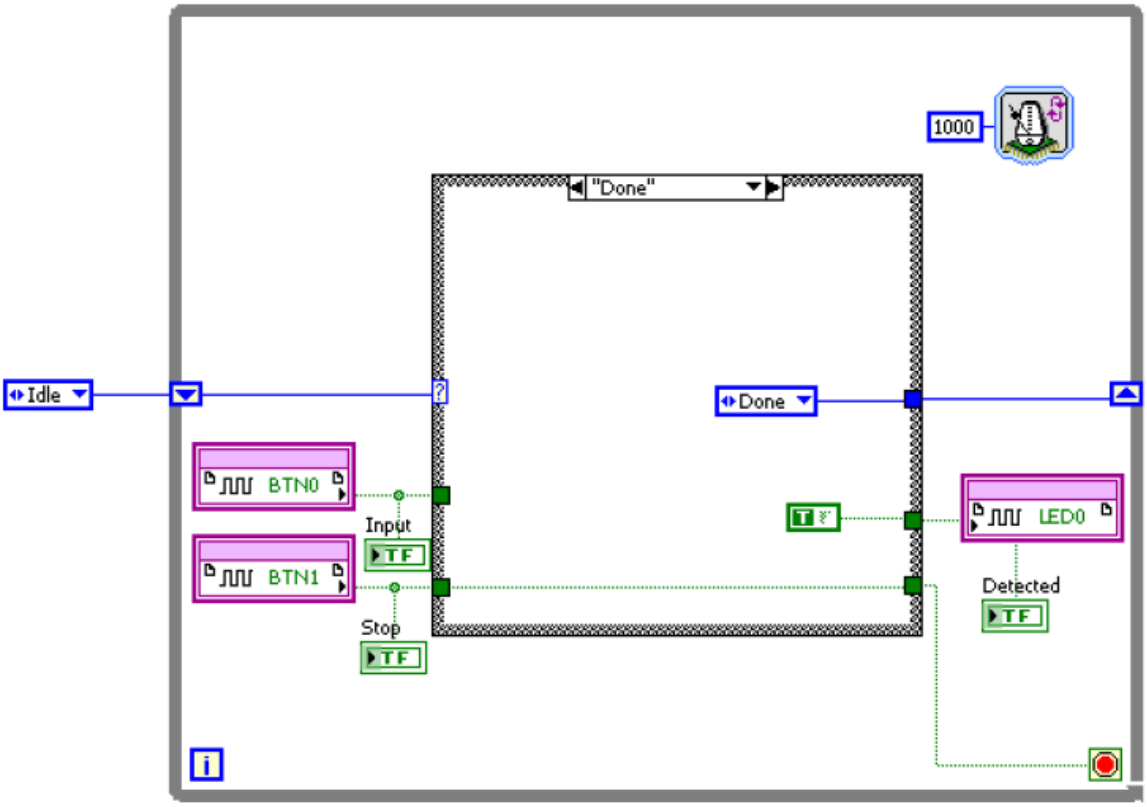
- Щелкните вверху по индикатору текущего фрейма и выберите вариант **3rd Zero**. Этот фрейм должен выглядеть так



- Щелкните вверху по индикатору текущего фрейма и выберите вариант **Done**. Этот фрейм должен выглядеть так



- Наконец, из палитры **Timing** поместите узел **Loop Timer** в цикл while и установите единицу счета **mSec**. Создайте константу на входе узла Loop Timer и присвойте ей значение **1000**, поскольку нам нужно увеличить время до1секунды, это необходимо, чтобы устройство успевало реагировать на нажатия физических кнопок
- На этом этапе проектирования блок-диаграмма должна выглядеть, как показано на рисунке ниже

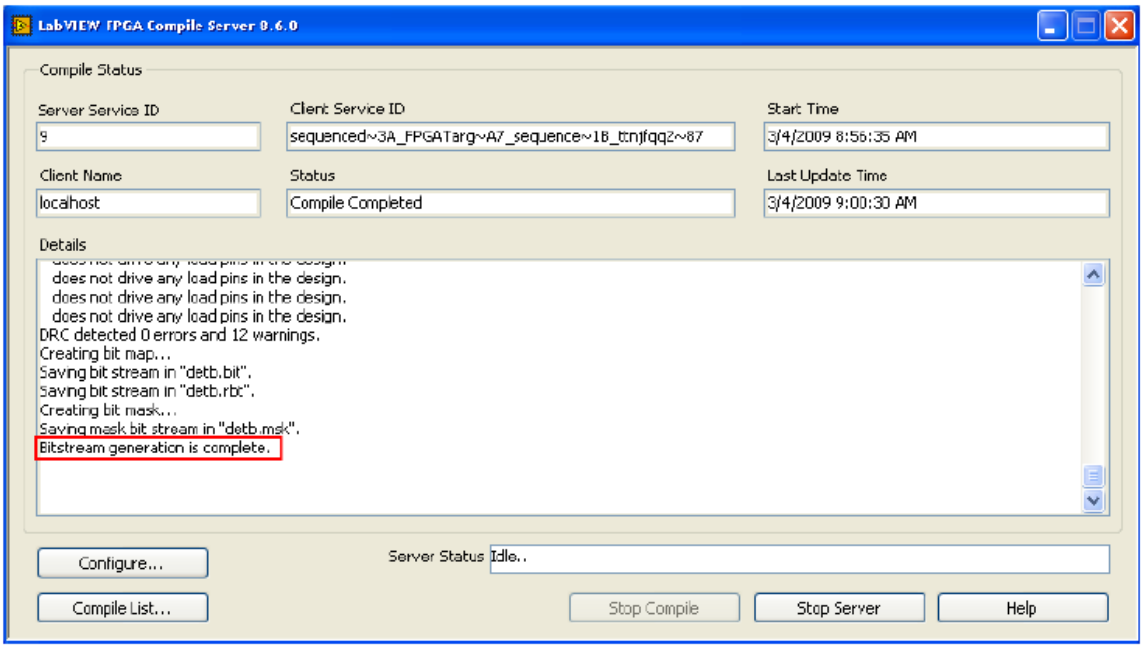


- Сохраните VI

Раздел 2: Проверка результатов проектирования с помощью оценочного модуля

Порядок выполнения:

- Подключите оценочный модуль, USB кабель и включите питание модуля
- Щелкните по кнопке **Run** () или выполните команду меню **Operate>>Run**
- Как только завершится генерация двоичного кода **Bitstream**, о чем появится сообщение в окне хода компиляции **Compile Server**, закройте окно щелчком по кнопке **X** в правом верхнем углу окна. Щелкните по кнопке **ОК**, чтобы закрыть сводное окно состояния компиляции



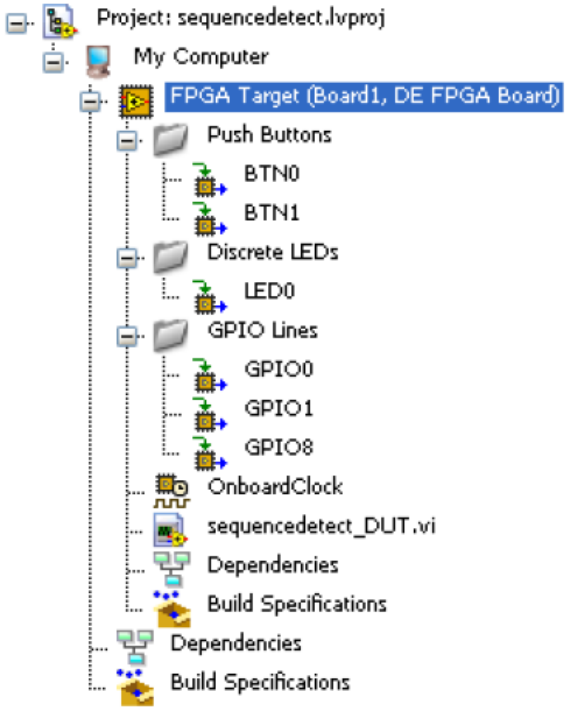
FPGA сконфигурируется, об этом свидетельствует подсветка кнопка RUN и включится светодиод DONE на модуле, и вы можете нажать кнопку BTN1 (Stop) на модуле, чтобы остановить выполнение задачи. Если вы остановили выполнение задачи, то можете перезапустить устройство щелчком по кнопке Run. Вы можете нажимать на кнопку BTN0 (Input), изменяя состояние на 1. Старайтесь удерживать кнопку нажатой достаточно долго (около 1 секунды), соответствующий индикатор на лицевой панели включается, сигнализируя о регистрации входных данных. Вы можете проверить функционирование спроектированного устройства на различных комбинациях данных

- После завершения тестирования щелкните по кнопке Stop () для прерывания моделирования

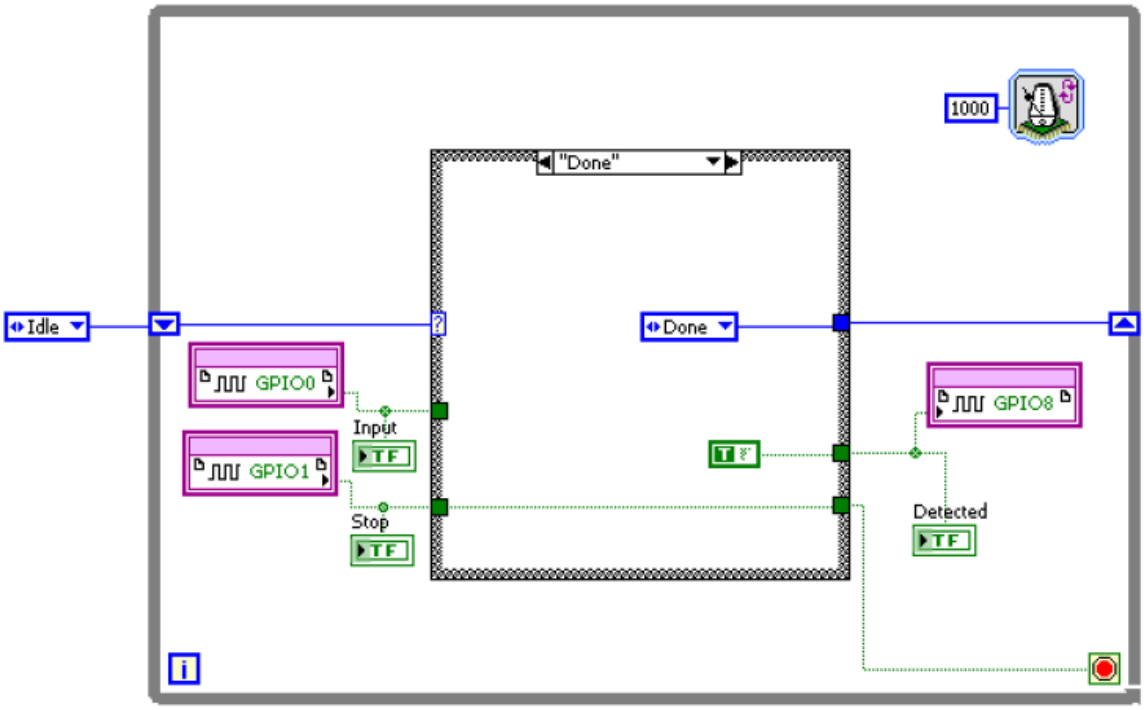
Раздел 3: Проверка результатов проектирования с помощью VI тестера

Порядок выполнения:

- Сохраните VI под именем **sequencedetect_DUT.vi**
- На блок-диаграмме удалите узлы ввода-вывода BTN0, BTN1 и LED0, оставив соответствующие индикаторы
- В окне Project Explorer щелкните правой кнопкой мыши по пункту **FPGA Target** и выберите **New>>FPGA I/O**
- В открывшемся окне отройте папку линий **GPIO**, выделите линии **GPIO0**, **GPIO1**, и **GPIO8** и добавьте их в проект, а затем щелкните по кнопке **ОК**
- Окно Project Explorer должно выглядеть так



- Выделите добавленные линии ввода-вывода GPIO и перетащите их на блок-диаграмму в цикл while loop
- Подключите узлы GPIO0 и GPIO1 вместо SW0 и SW1
- Щелкните правой кнопкой мыши по узлу GPIO8 и измените его назначение для записи.
- Соедините выход устройства со входом GPIO8
- Ниже показана полученная блок-диаграмма

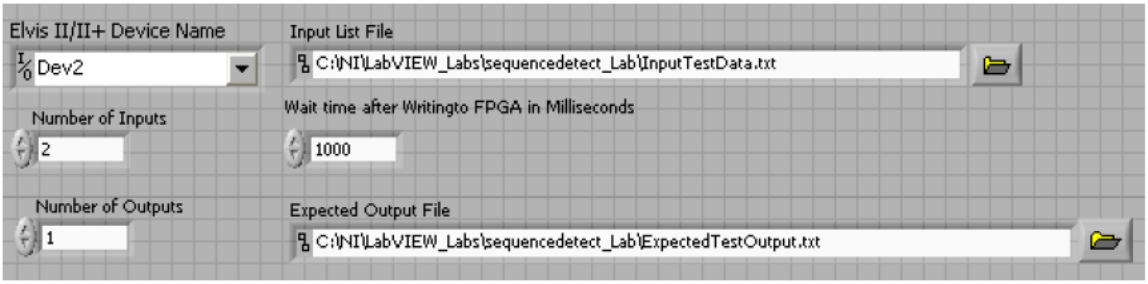


- Сохраните VI
- Щелкните по кнопке **Run**. Начнется компиляция проекта. Когда завершится генерация битового массива, щелкните по кнопке OK, чтобы закрыть диалоговое окно. FPGA сконфигурирован
- В окне Project Explorer щелкните правой кнопкой мыши по ветви **My Computer**, выберите **Add>>File** и из каталога проекта добавьте **Simple FPGA Tester.vi**
- В окне Project Explorer щелкните дважды по **Simple FPGA Tester.vi**, откроется окно VI
- Ознакомьтесь с блок-диаграммой и разберитесь с ее содержанием
- Обратите внимание, что Digital Writer использует каналы DIO с 8 по 15 [канал 15 - младший] для записи в FPGA, а Digital Reader использует каналы DIO с 0 по 7 [канал 7 - младший] для чтения откликов из FPGA
- Проводниками для макетной платы физически соедините контакты разъемов согласно следующей таблице

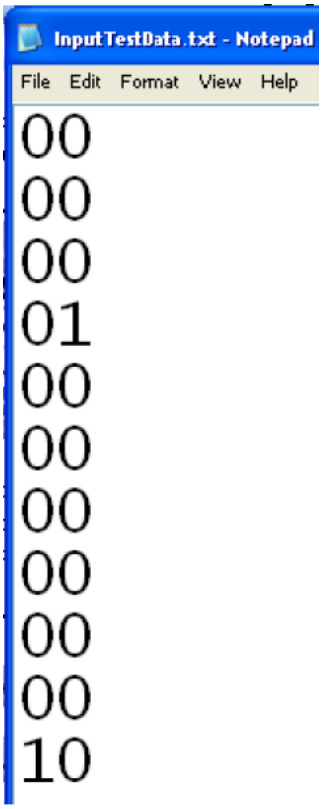
BB5 Connector	BB2 Connector
DIO15	GPIO0
DIO14	GPIO1
DIO7	GPIO8

- Щелкните по стрелке в поле **Elvis II/II+ Device Name** и выберите устройство, соответствующее Elvis II

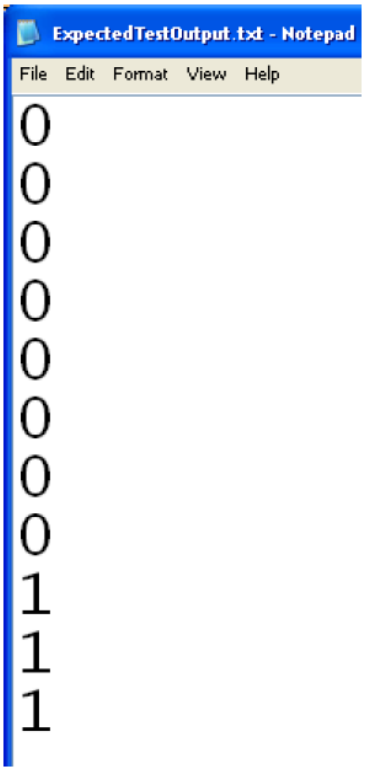
- Измените **Number of Inputs** на **2** и **Number of Outputs** на **1**
- Измените время Wait на **1000** ms
- Щелкните по кнопке Browse для файла Input и добавьте **InputTestData.txt**
- Таким же образом добавьте **ExpectedTestOutput.txt** с ожидаемыми состояниями выходов. На этом этапе проектирования лицевая панель должна выглядеть похожей на изображенную ниже



Файл входных данных – текстовый, его можно открыть/создать любым текстовым редактором. На следующем рисунке показано содержимое этого файла

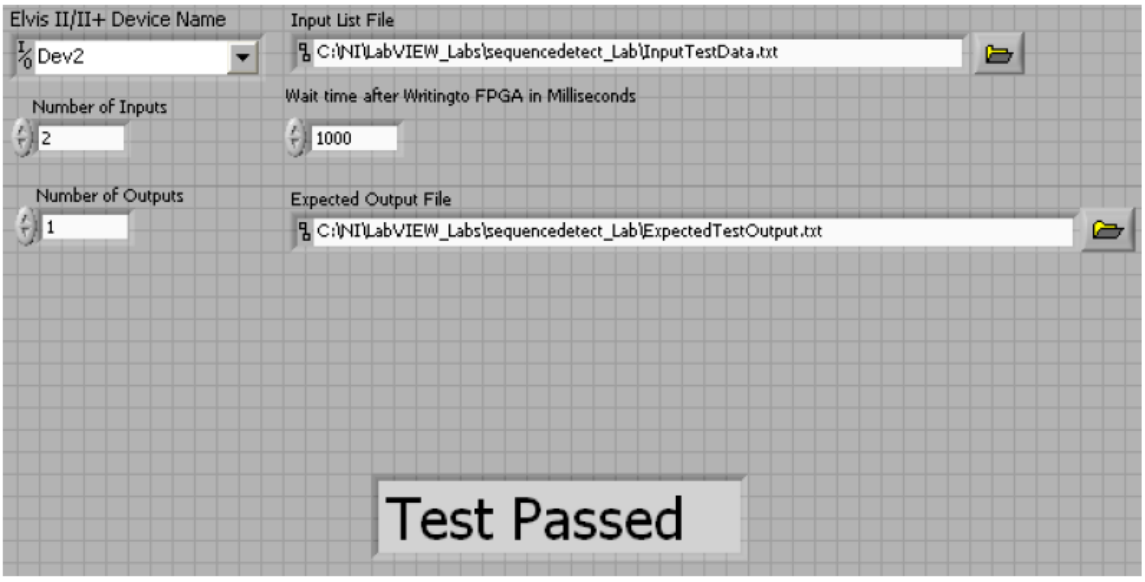


Обратите внимание, что количество входов меньше 8, для старших каналов дополнение нулями выполняется в тестируемом VI. В соответствии с этим файлом входных данных каналам (DIO) с 8-го по 13-й автоматически присваиваются нули, каналу 14 присвоен левый крайний бит (Stop), каналу 15 присвоен крайний правый бит (Input). Файл выходных данных – тоже текстовый, и его тоже можно открыть/создать любым текстовым редактором. На следующем рисунке показано содержимое этого файла



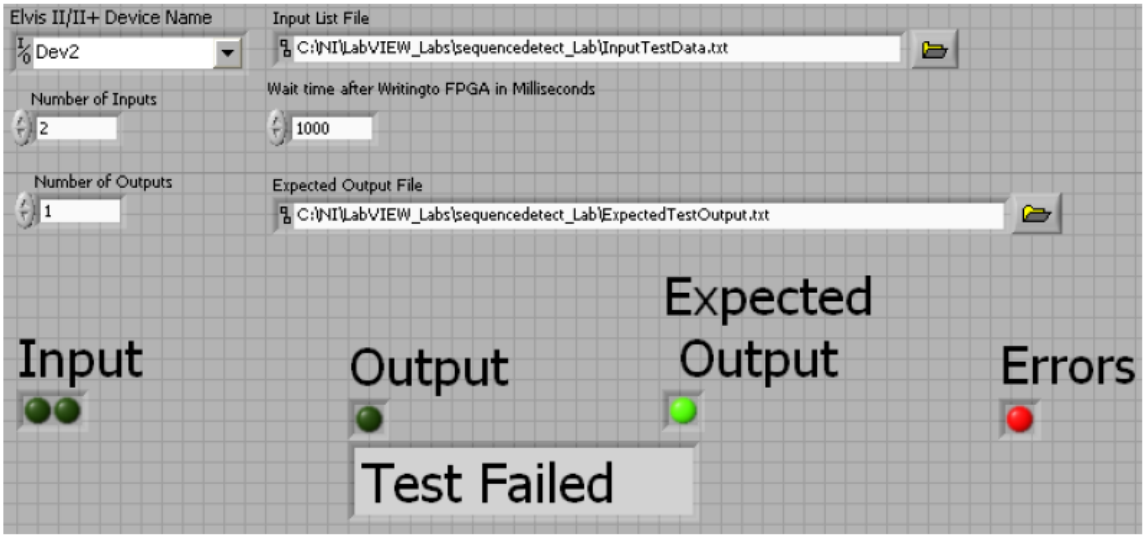
В соответствии с этим выходным файлом каналам (DIO) с 0-го по 6-й автоматически присвоено значение ноль, а каналу 7 назначен столбец, представленный в файле (выход детектора последовательности)

- Удостоверьтесь, что **sequencedetect_DUT.vi** продолжает исполняться
- Щелкните по кнопке Run на лицевой панели тестера. Тест запустится на исполнение, а его результаты отображаются на лицевой панели



- Теперь измените содержимое файла ожидаемых выходных реакций. Измените восьмой набор выходных сигналов с 0 на 1, при этом тест должен показать ошибку. Сохраните изменения
- Выключите и включите питание ELVIS и оценочного модуля FPGA

- Запустите sequencedetect_DUT еще раз, запустите тестер и обратите внимание, что тестер сообщает об ошибке (Failed). Он также сообщает, где ошибка, какие были входные сигналы, ожидаемые выходные сигналы, какие были реальные выходные сигналы, включается индикатор ошибки



- Остановите исполнение **sequencedetect_DUT**, закройте проект, закройте LabVIEW, выключите питание NI ELVIS II и оценочного модуля

Выводы:

В этой работе вы научились проектировать конечные автоматы с использованием структуры case. Вы протестировали проект с помощью оценочного модуля DE FPGA Board и системы NI ELVIS. Чтобы визуально наблюдать промежуточные состояния выходов, тактовая частота устройства было снижена.