Проектирование запоминающих устройств в LabVIEW для использования памяти в оценочном модуле DE FPGA Board

Цель работы:

Целью лабораторной работы является показать, как память FPGA может быть использована в качестве постоянной (ROM) и оперативной (RAM) памяти. Предлагаемый эксперимент состоит из двух частей: в первой показывается, как память FPGA может быть использована в качестве ROM, а во второй в качестве RAM. Вы также научитесь создавать локальные переменные и использовать их для обмена данными между параллельно исполняющимися задачами. Результаты проектирования вы проконтролируете на оценочном модуле DE FPGA и системе NI ELVIS.

Задание на проектирование:

В части A задания вы должны смоделировать в FPGA память ROM с организацией 16x8, инициализированную для наглядности значениями 12, 46, 23, 78, 90, 11, 22, 44, 66, 88, 100, 128, 90, 55, 34, 45. Затем для поиска в ROM необходимого содержимого используются входные переключатели. Вы должны спроектировать схему так, чтобы поиск начинался при нажатии на кнопку Go (BTN0). Поиск должен прекращаться, когда искомое содержимое будет найдено (при этом должен включиться индикатор Success) или, если будет обнаружено, что вся память проверена (при этом должен включиться индикатор Fail), о завершении поиска должен сигнализировать индикатор Done.

В части B задания вы должны смоделировать в FPGA память RAM с организацией 32x8, проинициализированную значениями по умолчанию (0). Затем для записи необходимого начального значения в первую ячейку памяти используются 8-разрядные переключатели. В следующую ячейку должно быть записано текущее значение +3. Запись начинается при нажатии на кнопку BTN0, а чтение - при нажатии на кнопку BTN1. Оба процесса – чтения и записи, а также порядок их запуска должны быть независимыми. При нажатии на кнопку BTN2 моделирование памяти должно останавливаться. Вы должны также обеспечить возможность записи других шаблонов данных путем изменения положения переключателей и нажатия кнопки BTN0 после завершения текущей процедуры записи.

Выполнение задания:

Для части A задания необходима память ROM. Такая память не использует порт для записи, но должна быть проинициализирована некоторым содержимым. В палитре функций Memory and FIFO есть две отдельные субпалитры – одна для чтения, а другая – для записи. Функции первой могут работать с разными блоками памяти, поэтому для этих функций необходимо корректно выбирать назначение памяти в соответствии с особенностями процесса. Адресация ячеек памяти у нас последовательная, это требует генерировать адрес, начиная с 0. Поскольку емкость памяти обычно равна степени двойки, наиболее эффективно генерировать адрес, используя функцию деления с вычислением частного и остатка.

Для части B задания память должна быть доступна как для записи, так и для чтения. Поэтому нам потребуются функции обеих субпалитр – и чтения, и записи. Эти функции должны выполняться независимо, поэтому находиться они должны в независимых циклах While Loop. Задание требует, чтобы и чтение, и запись могли начинаться независимо и в любой последовательности, следовательно, инициирующие эти

процессы кнопки должны находиться в независимых циклах. Последняя адресуемая при чтении ячейка памяти - 32-я, при этом чтение будет производиться из ячейки с адресом 31.

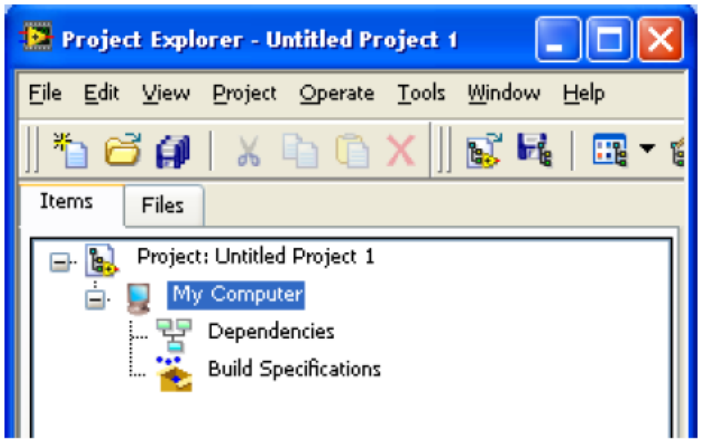
Для быстрого освоения LabVIEW выполните курс “Learn LabVIEW in 3 hours” (http://www.ni.com/academic/learn_LabVIEW/)

Часть A: постоянная память – ROM

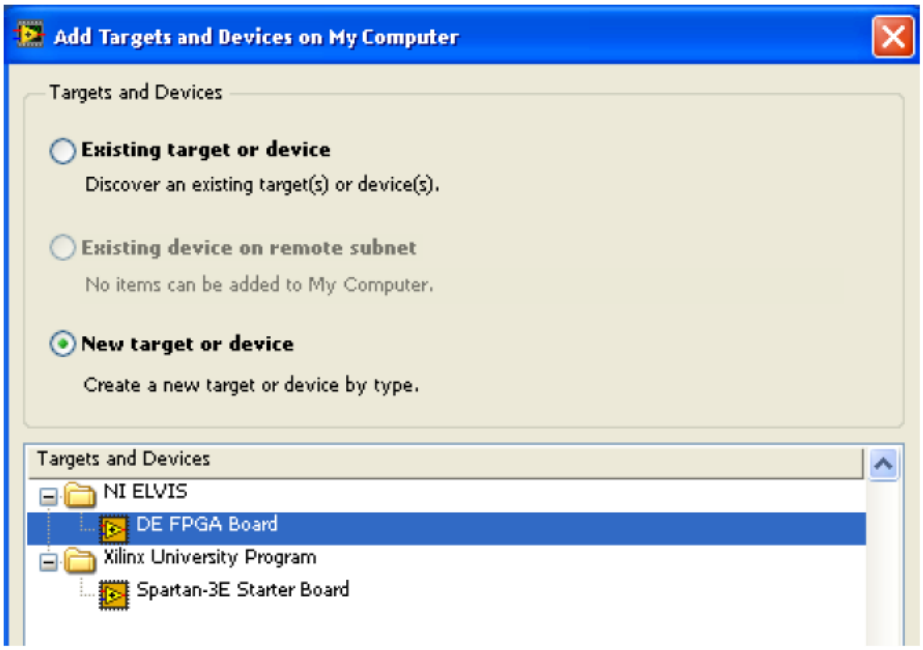
Раздел 1: разработка устройства

Порядок выполнения:

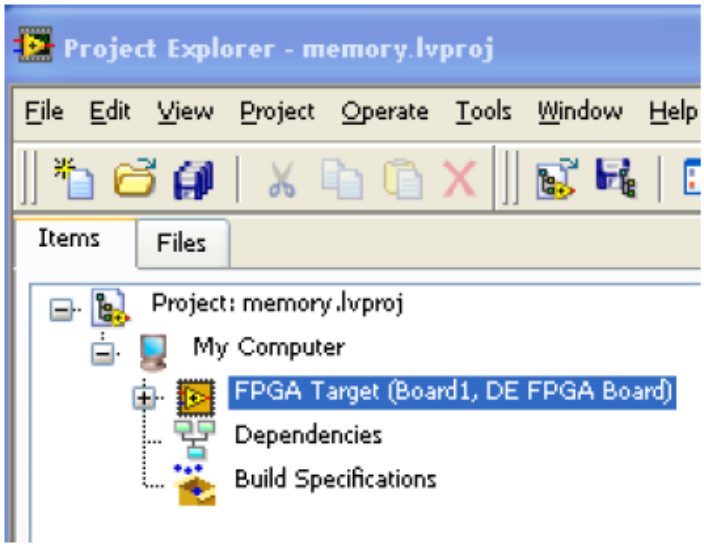
- Разархивируйте файл **counter_Lab.zip** в папку **c:\NI\LabVIEW_Labs**
- Запустите на исполнение LabVIEW **(Start>>All Programs>>National Instruments>>LabVIEW)**
- Выберите вариант **File>>New Project**
- В окне *Project Explorer* щелкните правой кнопкой мыши по пункту **My Computer** и выберите **New>>Targets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



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

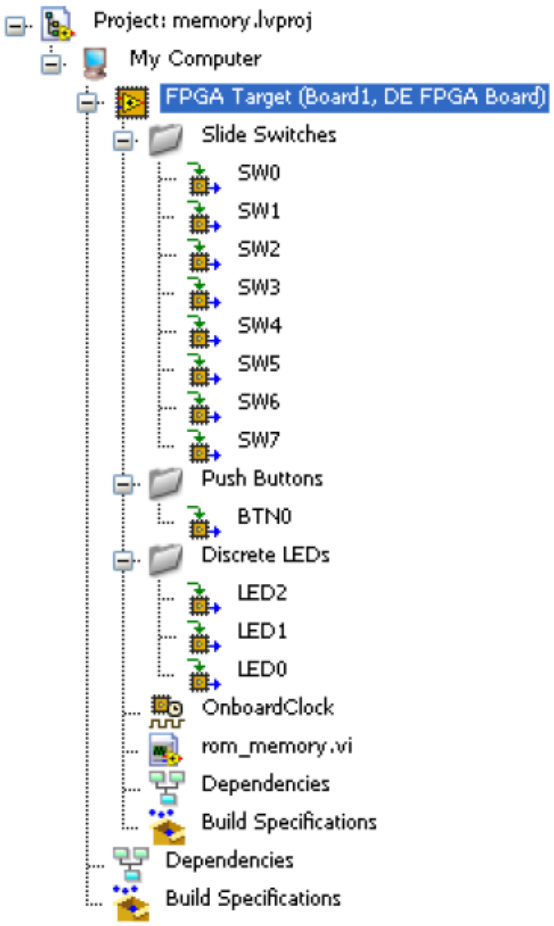


- Щелкните по кнопке OK
- Выберите в меню **File>>Save** и сохраните проект под именем **memory** в папке **c:\NI\LabVIEW_Labs\memory_Lab**
- В окне *Project Explorer* щелкните правой кнопкой мыши по пункту **FPGA Target (Board1, DE FPGA Board**) и выберите **New VI**

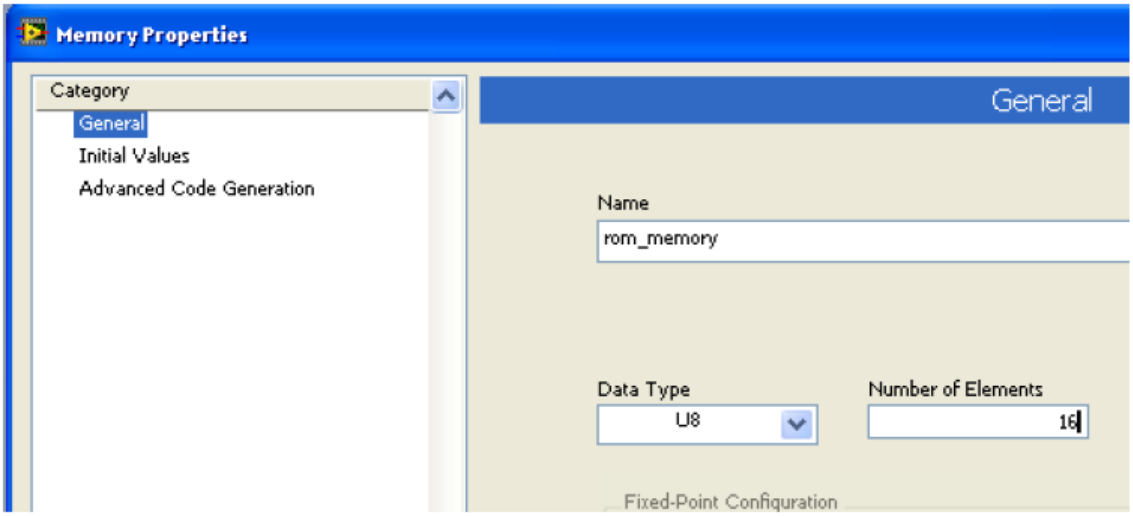


- Выберите в меню **File>>Save** и сохраните VI *под именем* **rom_memory** в папке **c:\NI\LabVIEW_Labs\memory_Lab**
- Теперь добавляем каналы ввода-вывода FPGA
- В окне *Project Explorer* щелкните правой кнопкой мыши по пункту **FPGA Target (Board1, DE FPGA Board**) и выберите **New>>FPGA I/O**
- Доступные в оценочном модуле (DE FPGA Board) каналы ввода-вывода отображаются в левой части окна выбора
- В секции **Available Resources** раскройте папку **Slide Switches**, выберите каналы **SW0÷SW7** для задания искомого шаблона и щелкните по кнопке Add, чтобы добавить каналы в проект.

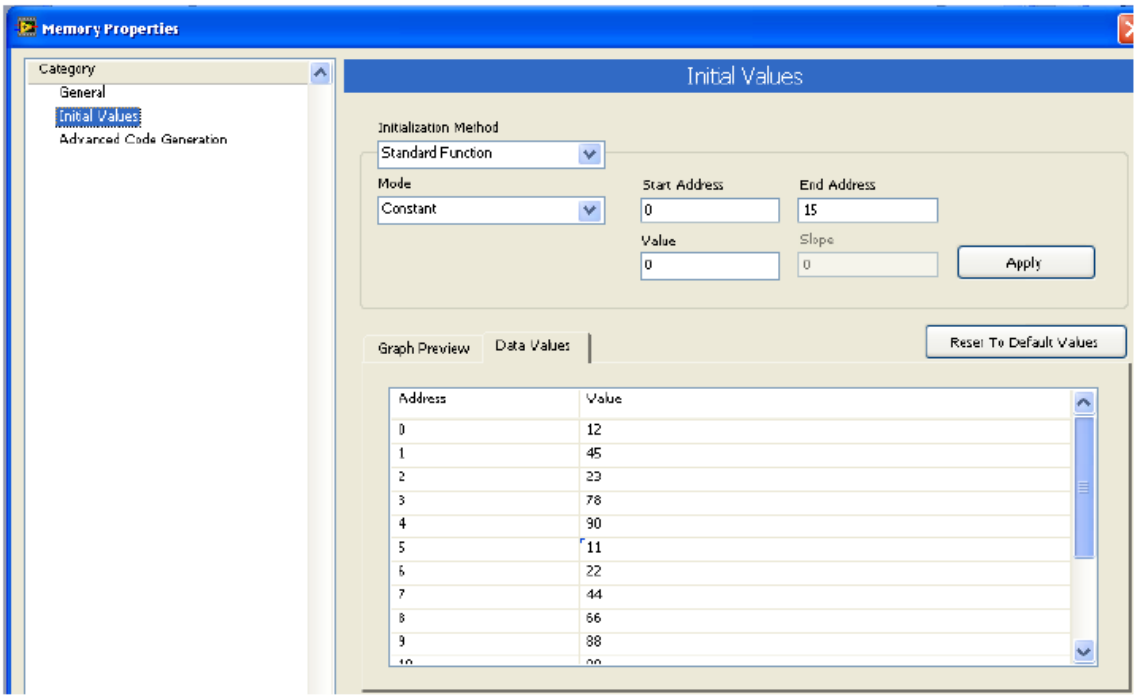
- В секции **Available Resources** раскройте папку **Push Buttons**, выберите канал **BTN0**, который будет использоваться для проверки содержимого памяти, и щелкните по кнопке Add, чтобы добавить **BTN0** в проект. Аналогично, раскройте папку **Discrete LEDs** и добавьте **LED2** в качестве индикатора **done**, **LED1** в качестве индикатора **Success** и **LED0** в качестве индикатора **Fail**
- Щелкните по кнопке **ОК** для подтверждения изменений и закройте окно выбора
- Ниже на рисунке показано, как должно выглядеть окно **Project Explorer** с внесенными изменениями



- В окне блок-диаграммы поместите цикл **while loop**
- В окне **Project Explorer** щелкните правой кнопкой мыши по пункту **FPGA Target (Board1, DE FPGA Board**) и выберите **New>> Memory**
- Откроется окно свойств Memory Properties
- В категории **General** этого окна введите имя **rom_memory**, выберите тип данных **U8** и введите **16** в поле Numbers of Elements



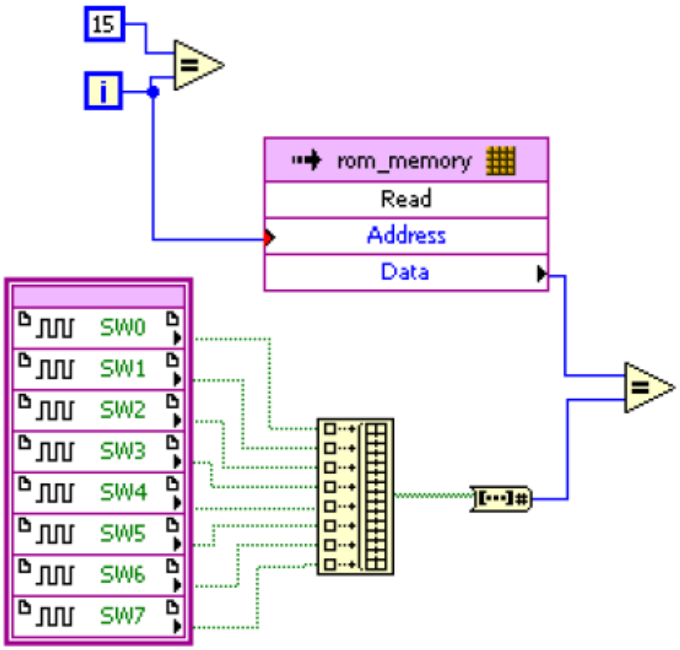
- Щелкните по категории **Initial Values** в том же окне свойств
- На открывшейся панели щелкните по закладке **Data Values**
- Щелкните по **Apply**
- В колонке **Values** General этого окна в строки с **0-й** по **15-ю** введите числа **12, 46, 23, 78, 90, 11, 22, 44, 66, 88, 100, 128, 90, 55, 34, 45**.



- Щелкните по кнопке **ОК** для подтверждения изменений и закройте окно
- Где-либо внутри цикла while loop щелкните правой кнопкой мыши для выбера из субпалитры **Memory and FIFO** функцию **Memory Read** и поместите ее в цикл
- Щелкните правой кнопкой мыши по пиктограмме функции и выберите **Select Memory>>rom_memory**. Обратите внимание, что пиктограмма блока памяти имеет один входной порт (Address) и один выходной порт (Data)

Нам необходимо спроектировать генератор адресов для последовательного обращения к ячейкам памяти до тех пор, пока не будет найдено искомое содержимое или не будет опрошена последняя ячейка памяти.

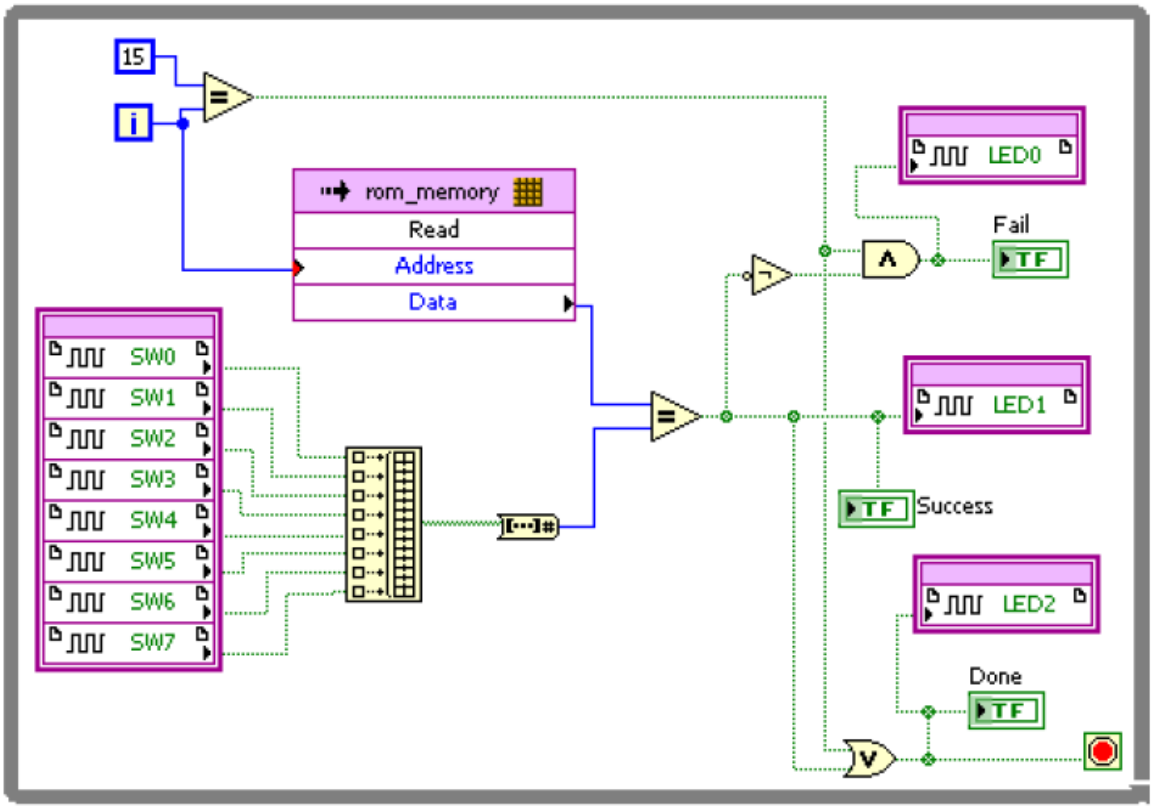
- Добавьте блок **Equal?** из субпалитры функций сравнения и соедините один из его входов с терминалом индекса цикла while loop
- Создайте константу, равную 15-ти, на втором входе блока **Equal?**
- Соедините терминал индекса цикла с портом адреса
- Где-либо внутри цикла while loop щелкните правой кнопкой мыши и поместите из субпалитры **FPGA I/O** узел **FPGA I/O Node** (). Вы можете также найти этот узел, щелкнув по кнопке “Search”
- Щелкнув по узлу и выбрав **Slide Switches>>SW0**, присвойте его линии **SW0**
- Растяните узел так, чтобы в него вошли элементы от **SW0** до **SW7**
- Из палитры **Array** поместите функцию **Build Array** и растяните ее на 8 элементов
- Подключите выходы **SW0÷SW7** ко входам функции **Build Array**
- Поместите функцию **Boolean Array to Number** из палитры **Boolean** и соедините выход функции **Boolean Array** со входом функции **Boolean Array to Number**
- Добавьте блок **Equal?** и соедините выход функции **Boolean Array to Number** с нижним входом функции **Equal?**, соедините выходной порт **Data** узла **rom_memory** с верхним входом функции **Equal?**
- На этом этапе проектирования блок-диаграмма должна выглядеть, как показано на рисунке ниже



Т.к. мы хотим останавливать выполнение задачи при обнаружении требуемого шаблона или при заполнении счетчика, нам необходимо добавить некоторую логику. Нам также нужно добавить индикацию сигналов Done, Success и Fail, подключив их к соответствующим светодиодам (LED)

- Добавьте логические элементы NOT, AND и OR и соедините в соответствии с рисунком ниже
- Поместите узлы LED0, LED1 и LED2 внутрь цикла while loop и соедините их с выходами Fail, Success, и Stop генератора (см. рисунок ниже)
- Добавьте индикаторы и обозначьте их соответствующим образом

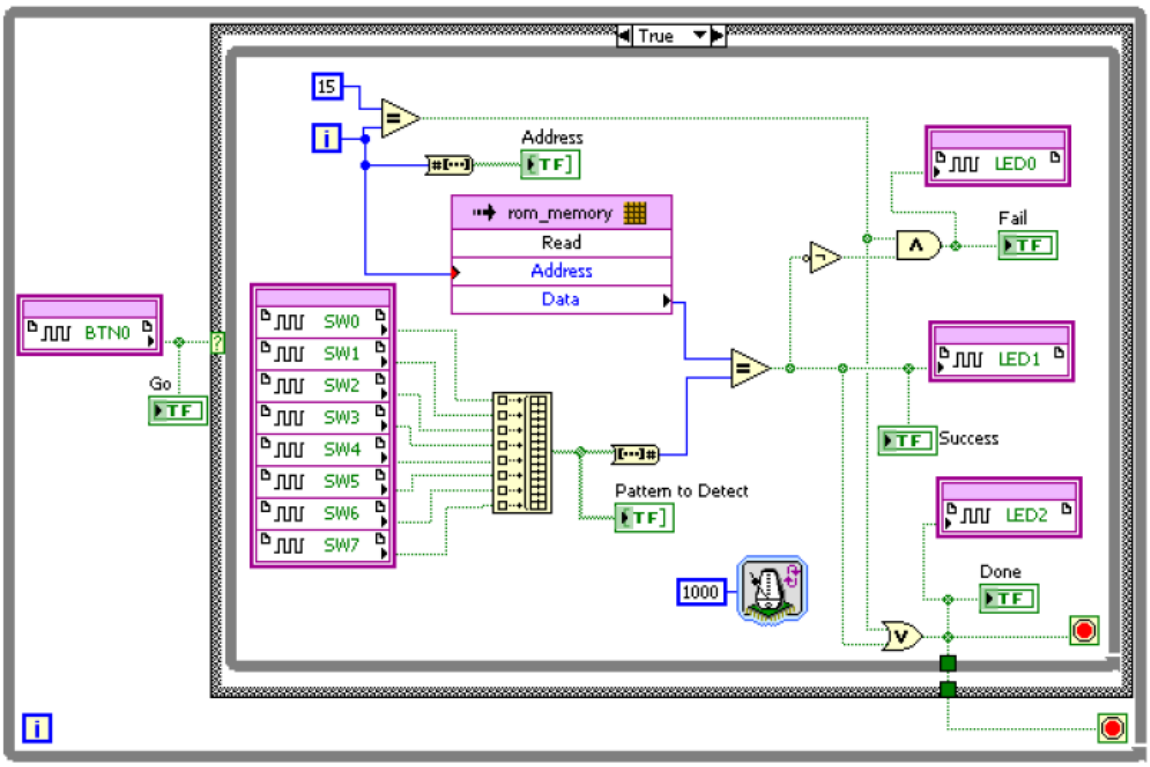
- Полученная блок-диаграмма должна быть похожа на изображенную ниже



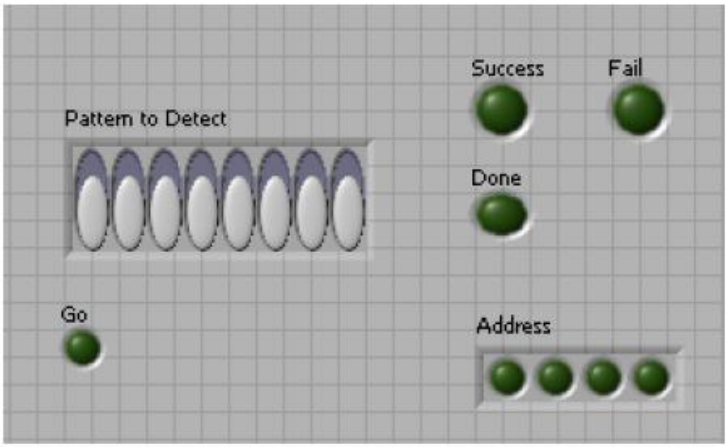
Нам нужно, чтобы изображенные выше операции выполнялись только при нажатии на кнопку BTN0. Поэтому необходимо добавить структуру Case, во фрейме True которой находился бы код, реализующий эти операции. Из фрейма False нужно подключить константу False.

- Добавьте структуру Case так, чтобы в нее поместился упомянутый код
- Выберите в окне Project Explorer элемент **BTN0** и перетащите его на блок-диаграмму так, чтобы он находился вне структуры Case
- Подключите выход BTN0 к селектору выбора (**?**) структуры Case, создайте для BTN0 индикатор и обозначьте его **Go**
- Заключите весь полученный код в еще один цикл while loop
- Соедините выход **Done** с внешним циклом while, проведя соединение через внутренний цикл while и структуру Case
- Выберите фрейм **False** структуры Case
- Во фрейме **False** структуры Case добавьте константу **False** и подключите ее к выходному туннелю Case структуры
- Опять выберите фрейм True структуры Case
- Переключитесь на лицевую панель (Ctrl-E)
- Поместите на лицевую панели массив из палитры **Array and Cluster**
- Поместите в массив круглый светодиод и растяните массив (в горизонтальном направлении), чтобы были видны 4 светодиода
- Измените имя массива на **Address**
- Щелкнув правой кнопки мыши по массиву и выбрав пункт **Visible Items>>Index Display**, скройте индикатор индекса массива

- Добавьте еще один массив и поместите в него переключатель. Растяните массив так, чтобы были видны 8 переключателей и обозначьте массив, как **Pattern to detect.** Индикатор индекса этого массива также сделайте невидимым.
- Переключитесь на блок-диаграмму
- Добавьте функцию **Number to Boolean Array** из палитры **Boolean** и подключите терминал массива Address к выходу Number to Boolean Array
- Соедините терминал индекса цикла while со входом **Number to Boolean Array**
- Щелкните правой кнопки мыши по массиву **Pattern to Detect** и измените его на индикатор
- Соедините **Pattern to Detect** с выходом функции Build Array
- Из палитры **Timing** поместите узел **Loop Timer** и установите единицу счета **ms**
- Создайте константу на входе узла **Loop Timer** и присвойте ей значение **1000** (1000 ms = 1s)
- Ниже показан результат проектирования



- Переключитесь в окно лицевой панели
- Измените положение и размеры объектов на лицевой панели, чтобы они были наглядны и понятны, как показано ниже

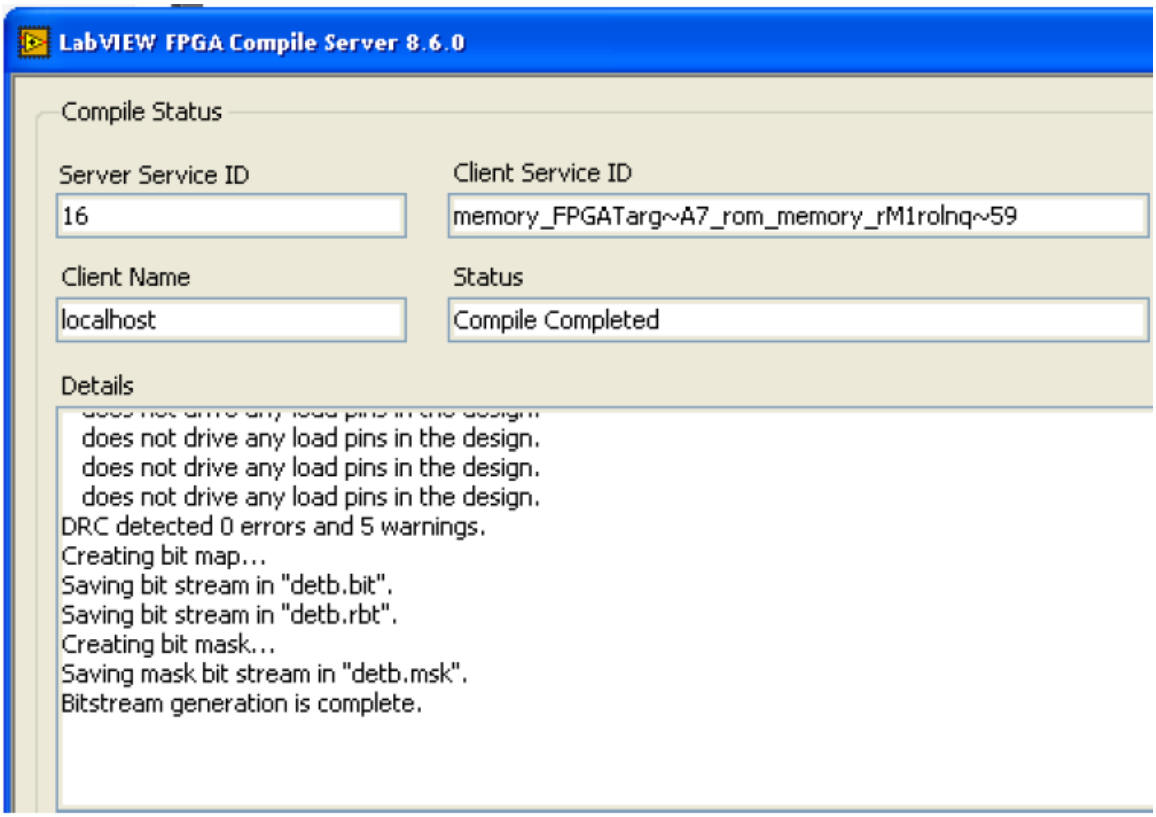


- Сохраните VI

Раздел 2: Проверка результатов проектирования с помощью оценочного модуля

Порядок выполнения:

- Подключите оценочный модуль, USB кабель и включите питание
- 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. Щелкните по кнопке **ОК**, чтобы закрыть сводное окно состояния компиляции



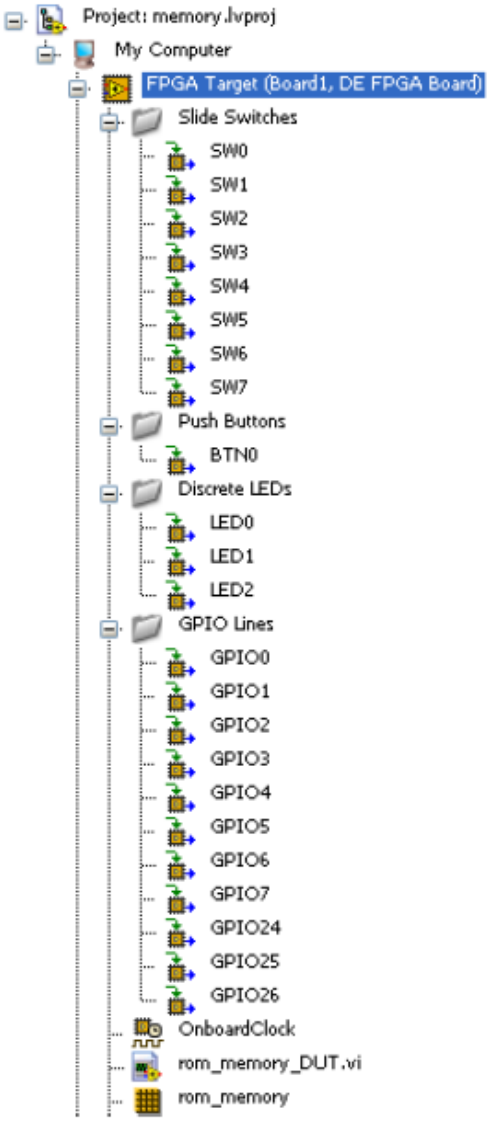
FPGA сконфигурирован – об этом свидетельствует подсветка кнопки Run. Протестировать разработанное устройство вы можете следующим образом.

Установите на оценочном модуле переключатели SW7÷SW0 в положение 10000000 (число 128, запомненное в ячейке памяти 11) и нажмите на кнопку BTN0. На лицевой панели вы должны увидеть включение индикатора Go и начало инкрементирования адреса (Address). Когда адрес станет равным 11 (1011) включатся индикаторы Done и Success, выполнение задачи остановится. Включатся также индикаторы LED1 и LED2 модуля.
Запустите выполнение задачи заново, изменив положение переключателей SW7÷SW0 на 10000001 (число 129 – в память rom не записано) и нажмите на кнопку BTN0. На лицевой панели вы должны увидеть включение индикатора Go и начало инкрементирования адреса (Address). Когда адрес станет равным 15 (1111) включатся индикаторы Done и Fail, выполнение задачи остановится. Включатся также индикаторы LED1 и LED2 модуля.
Запустите выполнение задачи еще раз, изменив положение переключателей SW7÷SW0 на 00101101 (число 45 – хранится в ячейке памяти rom по адресу 15) и нажмите на кнопку BTN0. На лицевой панели вы должны увидеть включение индикатора Go и начало инкрементирования адреса (Address). Когда адрес станет равным 15 (1111), включатся индикаторы Done и Success (заданный для поиска шаблон найден), выполнение задачи остановится.

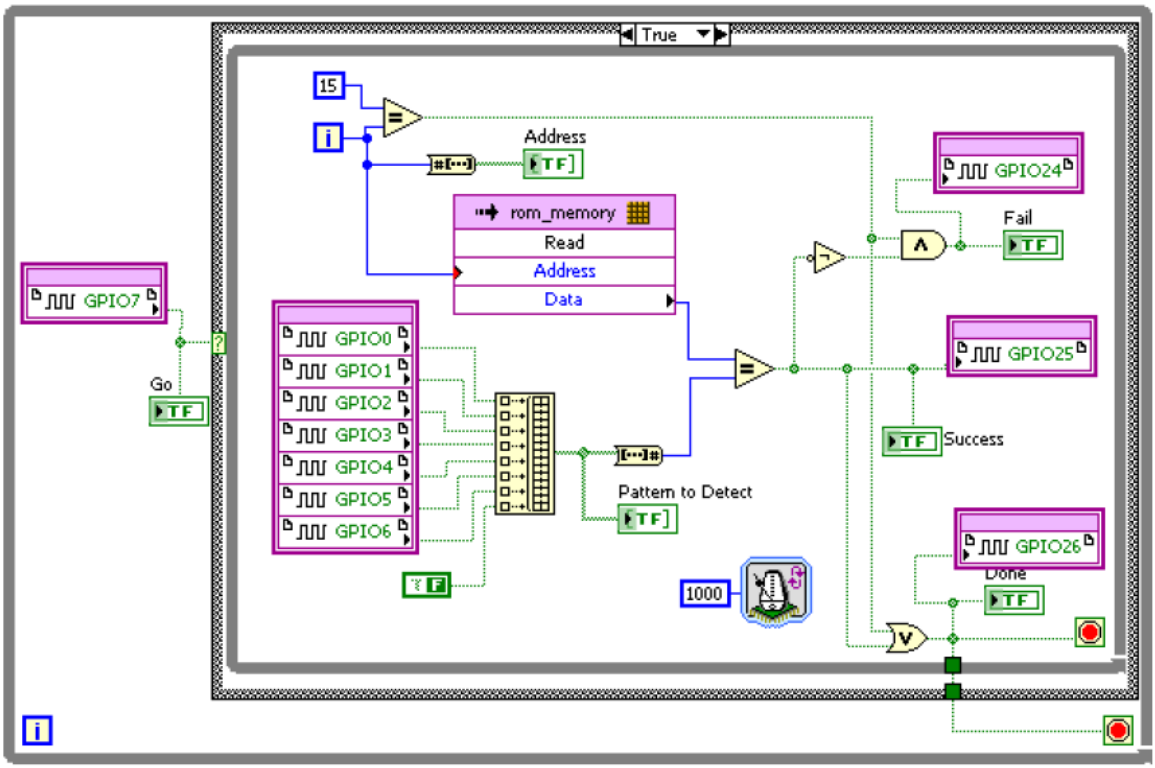
Раздел 3: Проверка результатов проектирования с помощью VI тестера

Порядок выполнения:

- Сохраните VI под именем **rom_memory_DUT.vi**
- На блок-диаграмме удалите переключатели, светодиоды и кнопки, оставив соответствующие индикаторы
- В окне Project Explorer щелкните правой кнопкой мыши по пункту **FPGA Target** и выберите **New FPGA I/O**
- В открывшемся окне откройте папку линий **GPIO**, выделите линии **GPIO0÷ GPIO7** и **GPIO24÷ GPIO26**, добавьте их в проект, а затем щелкните по кнопке **ОК**
- Окно Project Explorer должно выглядеть так



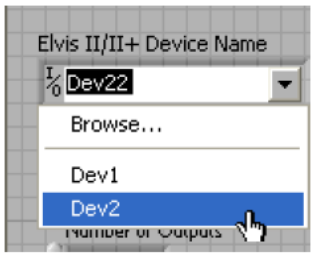
- Поместите на блок-диаграмму из субпалитры **FPGA I/O** узел **FPGA I/O Node**
- Щелкните правой кнопкой мыши по узлу и выберите **GPIO0**. Растяните узел так, чтобы в него вошли элементы от **GPIO0** до **GPIO6**
- Соедините выходы **GPIO0÷ GPIO6** с первыми 7 элементами функции **Build Array**, от которых были отключены переключатели SW
- Подключите константу **False** к 8-му элементу
- Поместите на блок-диаграмму, туда, где была кнопка BTN0, узел **FPGA I/O Node** из субпалитры **FPGA I/O**
- Щелкните правой кнопкой по добавленному узлу и выберите **GPIO7**
- Соедините выход **GPIO7** с входом селектора выбора Case структуры (см. блок-диаграмму ниже)
- В окне Project Explorer выделите линии **GPIO24÷ GPIO26** и перетащите их в цикл while loop, где были подключены светодиоды (LED)
- Щелкните правой кнопкой мыши по добавленному узлу и измените его назначение для записи
- 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 с 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 BB1/BB2 и BB5 согласно следующей таблице

BB1/BB2 Connectors	BB5 Connector
GPIO0	DIO15
GPIO1	DIO14
GPIO2	DIO13
GPIO3	DIO12
GPIO4	DIO11
GPIO5	DIO10
GPIO6	DIO9
GPIO7	DIO8
GPIO24	DIO7
GPIO25	DIO6
GPIO26	DIO5

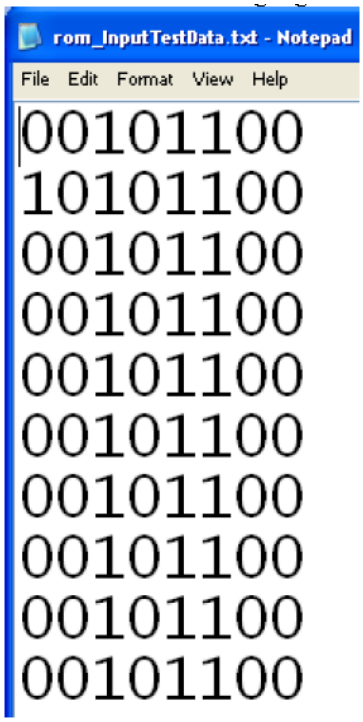
- Щелкните по стрелке в поле **Elvis II/II+ Device Name** и выберите устройство, соответствующее Elvis II



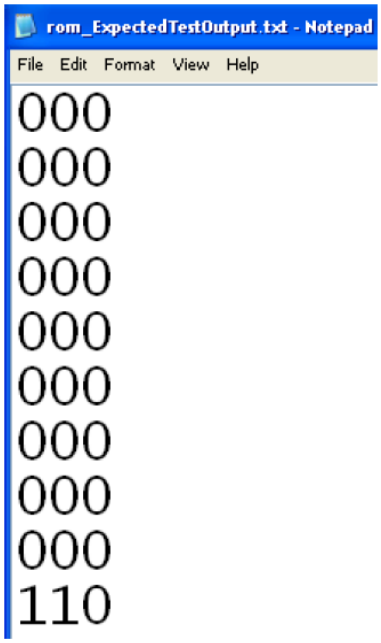
- Измените **Number of Inputs** на **8** и **Number of Outputs** на **3**
- Измените время Wait на **1000** ms
- Щелкните по кнопке Browse для файла Input и добавьте **rom_InputTestData.txt**
- Таким же образом добавьте **rom_ExpectedTestOutput.txt** с ожидаемыми состояниями выходов.

  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



В соответствии с этим входным файлом канал DIO8 назначен в качестве кнопки Go, а каналы с DIO9 по DIO15 – используются для задания шаблона поиска Pattern [6:0]. 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 с DIO0 по DIO4 автоматически присвоено значение ноль, тогда как каналы с DIO5 по DIO7 – назначены выходами Done, Success и Fail

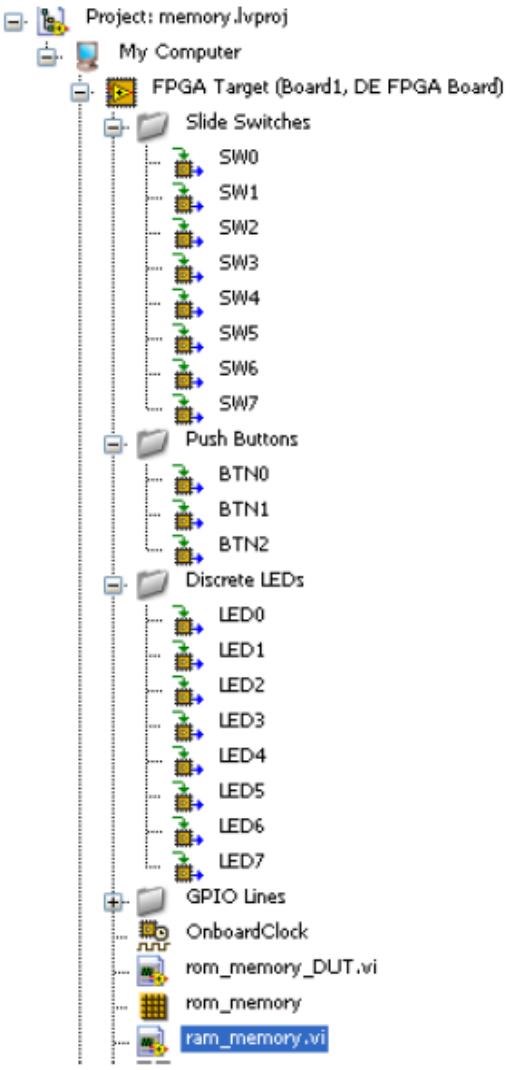
- Удостоверьтесь, что **rom_memory_DUT.vi** продолжает исполняться
- Щелкните по кнопке Run на лицевой панели тестера. Тест запустится на исполнение, а отображаемые результаты свидетельствуют об успешном завершении теста
- Теперь измените содержимое файла ожидаемых выходных реакций. Измените третий набор выходных сигналов с 000 на 001, при этом тест должен показать ошибку. Сохраните изменения
- Включите и выключите питание ELVIS и оценочного модуля FPGA
- Запустите еще раз rom_memory_DUT 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
- Если **rom_memory_DUT** продолжает работать, остановите его – проверится завершение задачи

Часть B: оперативная память – RAM

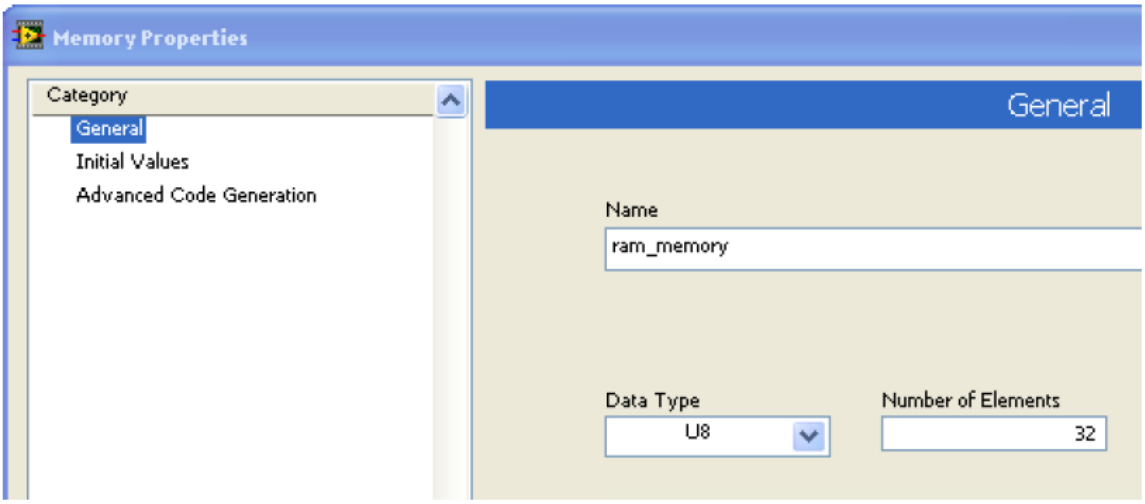
Раздел 1: разработка устройства

Порядок выполнения:

- В окне *Project Explorer* щелкните правой кнопкой мыши по пункту **FPGA Target (Board1, DE FPGA Board**) и выберите **New VI**
- Выберите в меню **File>>Save** и сохраните VI *под именем* **ram_memory** в папке **c:\NI\LabVIEW_Labs\memory_Lab**
- В окне **Project Explorer** щелкните правой кнопкой мыши по пункту **FPGA Target (Board1, DE FPGA Board**) и выберите **New>>FPGA I/O**
- В секции **Available Resources** раскройте папку **Push Buttons**, выберите **BTN1** (для запуска процесса чтения), выберите **BTN2** (для останова), и щелкните по кнопке **Add**, чтобы добавить выбранные элементы в проект. Уже включенный в проект элемент **BTN0** будем использовать для инициализации записи. Аналогично, раскройте папку **Discrete LEDs** и добавьте **LED3÷ LED7** для отображения считываемых данных
- Ниже на рисунке показано, как должно выглядеть окно **Project Explorer** с внесенными изменениями



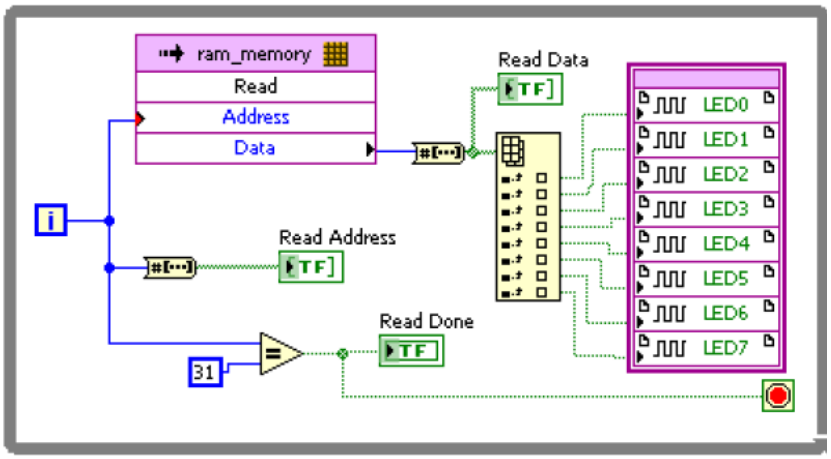
- В окне блок-диаграммы поместите цикл **while loop**
- В окне **Project Explorer** щелкните правой кнопкой мыши по пункту **FPGA Target (Board1, DE FPGA Board)** и выберите **New>>Memory**
- Откроется окно свойств Memory Properties
- В категории **General** этого окна введите имя **ram_memory**, выберите тип данных **U8** и введите **32** в поле Numbers of Elements



- Оперативную память RAM инициализировать не нужно
- Щелкните по кнопке **ОК** для подтверждения введенных значений и закройте окно свойств
- Где-либо внутри цикла while loop щелкните правой кнопкой мыши и выберите из субпалитры **Memory and FIFO** функцию **Memory Read**
- Щелкните правой кнопкой мыши по пиктограмме функции и выберите **Select Memory>>ram_memory**. Обратите внимание, что пиктограмма блока памяти имеет один входной порт (Address) и один выходной порт (Data)
- Нам необходимо спроектировать генератор адресов для последовательного обращения к ячейкам памяти, пока не будет опрошена последняя ячейка памяти. Нам нужно также отображать считываемые данные на светодиодах
- Добавьте блок **Equal?** из субпалитры функций сравнения и соедините один из его входов с терминалом индекса цикла while loop
- Создайте константу, равную 15-ти на втором входе блока **Equal?**
- Соедините терминал индекса цикла с портом адреса
- Где-либо внутри цикла while loop щелкните правой кнопкой мыши и поместите из субпалитры **FPGA I/O** узел **FPGA I/O Node** (). Вы можете также найти этот узел, щелкнув по кнопке “Search”
- Щелкнув по узлу и выбрав **Discrete LEDs >> LED0**, присвойте его линии **LED0**
- Растяните узел так, чтобы в него вошли элементы от **LED0** до **LED7** (возможно, элементы узла надо упорядочить)
- Из палитры **Array** поместите функцию **Index Array** и растяните ее на 8 элементов
- Добавьте функцию **Number to Boolean Array** и подключите ее вход к порту памяти Data
- Соедините выход функции **Number to Boolean Array** со входом функции **Index Array**

- Соедините светодиоды **LED0÷LED7** с выходами функции **Index Array**

- Добавьте функцию **Number to Boolean Array** и подключите ее вход к порту памяти **Address**
- Переключитесь на лицевую панель (Ctrl-E)
- Поместите на лицевую панель массив из палитры **Array and Cluster**
- Поместите в массив круглый светодиод и растяните массив (в горизонтальном направлении), чтобы были видны 5 светодиодов
- Измените имя массива на **Read Address**
- Щелкнув правой кнопки мыши по массиву и выбрав пункт **Visible Items>>Index Display**, скройте индикатор индекса массива
- Добавьте еще один массив и поместите в него светодиод. Растяните массив так, чтобы были видны 8 светодиодов и обозначьте массив, как **Read Data.** Индикатор индекса этого массива также сделайте невидимым.
- Переключитесь на блок-диаграмму
- Подключите Read Address и Read Data так, как показано на рисунке ниже
- Добавьте индикатор на выход функции **Equal?** и обозначьте его **Read Done**
- Соедините индикатор **Read Done** с терминалом выхода из цикла, т.к. мы хотим остановить выполнение после считывания всех ячеек памяти
- На этом этапе результат проектирования выглядит следующим образом

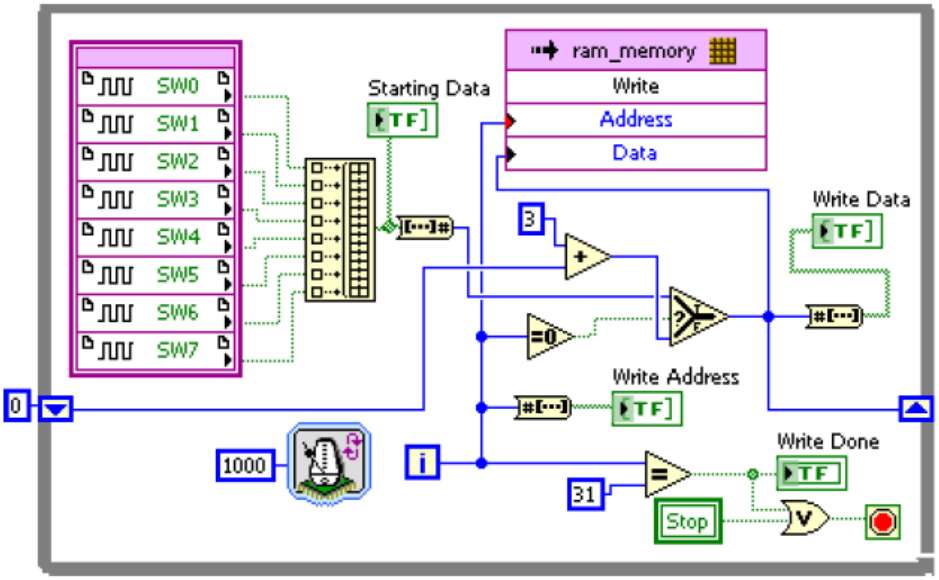


- В окне блок-диаграммы поместите еще один цикл **while loop** для процесса записи
- Где-либо внутри цикла while loop щелкните правой кнопкой мыши и выберите из субпалитры **Memory and FIFO** функцию **Memory Write**
- Щелкните правой кнопкой мыши по пиктограмме функции и выберите **Select Memory>>ram_memory**. Обратите внимание, что пиктограмма блока памяти имеет один входной порт (Address) и один выходной порт (Data)
- Нам необходимо спроектировать генератор адресов для последовательного обращения к ячейкам памяти, пока не будет опрошена последняя ячейка памяти, кроме того, мы хотим записывать в память определенный шаблон
- Добавьте блок **Equal?** и соедините один из его входов с терминалом индекса цикла while loop
- Теперь нам надо считывать состояния переключателей для начального значения шаблона, который мы хотим записывать в память

- Где-либо внутри цикла while loop щелкните правой кнопкой мыши и поместите из субпалитры **FPGA I/O** узел **FPGA I/O Node** (). Вы можете также найти этот узел, щелкнув по кнопке “Search”
- Щелкнув по узлу и выбрав **Slide Switches>>SW0**, присвойте узел линии **SW0**
- Растяните узел так, чтобы в него вошли элементы от **SW0** по **SW7**
- Из палитры **Array** поместите функцию **Build Array** и растяните ее на 8 элементов. Подключите выходы **SW0÷SW7** ко входам функции **Build Array**
- Поместите функцию **Boolean Array to Number** и соедините выход функции **Boolean Array** со входом функции **Boolean Array to Number**
- Поместите еще одну функцию **Boolean Array to Number** и соедините ее вход с портом **Address** памяти
- Переключитесь на лицевую панель (Ctrl-E)
- Поместите на лицевую панель массив из палитры **Array and Cluster**
- Поместите в массив круглый светодиод и растяните массив (в горизонтальном направлении), чтобы были видны 5 светодиодов
- Измените имя массива на **Write Address**
- Щелкнув правой кнопки мыши по массиву и выбрав пункт **Visible Items>>Index Display**, скройте индикатор индекса массива
- Добавьте еще один массив и поместите в него переключатель. Растяните массив так, чтобы были видны 8 переключателей и обозначьте массив, как **Starting Data**. Индикатор индекса этого массива также сделайте невидимым
- Переключитесь на блок-диаграмму
- Измените режим **Starting Data** на индикатор
- Соедините **Write Address** и **Starting Data**, как показано на блок-диаграмме ниже
- Добавьте индикатор на выход функции **Equal?** и обозначьте его **Write Done**
- Соедините индикатор **Write Done** с терминалом выхода из цикла, т.к. мы хотим остановить выполнение после записи во все ячейки памяти

Нам нужно записывать в память некоторые определенные значения. Для этого мы должны прибавлять 3 к начальному значению и записывать накапливаемые значения в последовательные ячейки памяти, т.е., n (начальное значение), n+3, n+6, n+9, n+12 и т.д.

- Создайте сдвиговый регистр на границе цикла while
- Поместите на блок-диаграмму узлы **Adder**, **select** и **constant** и соедините их так, чтобы реализовать требуемую функции. Полученная в результате блок-диаграмма показана ниже

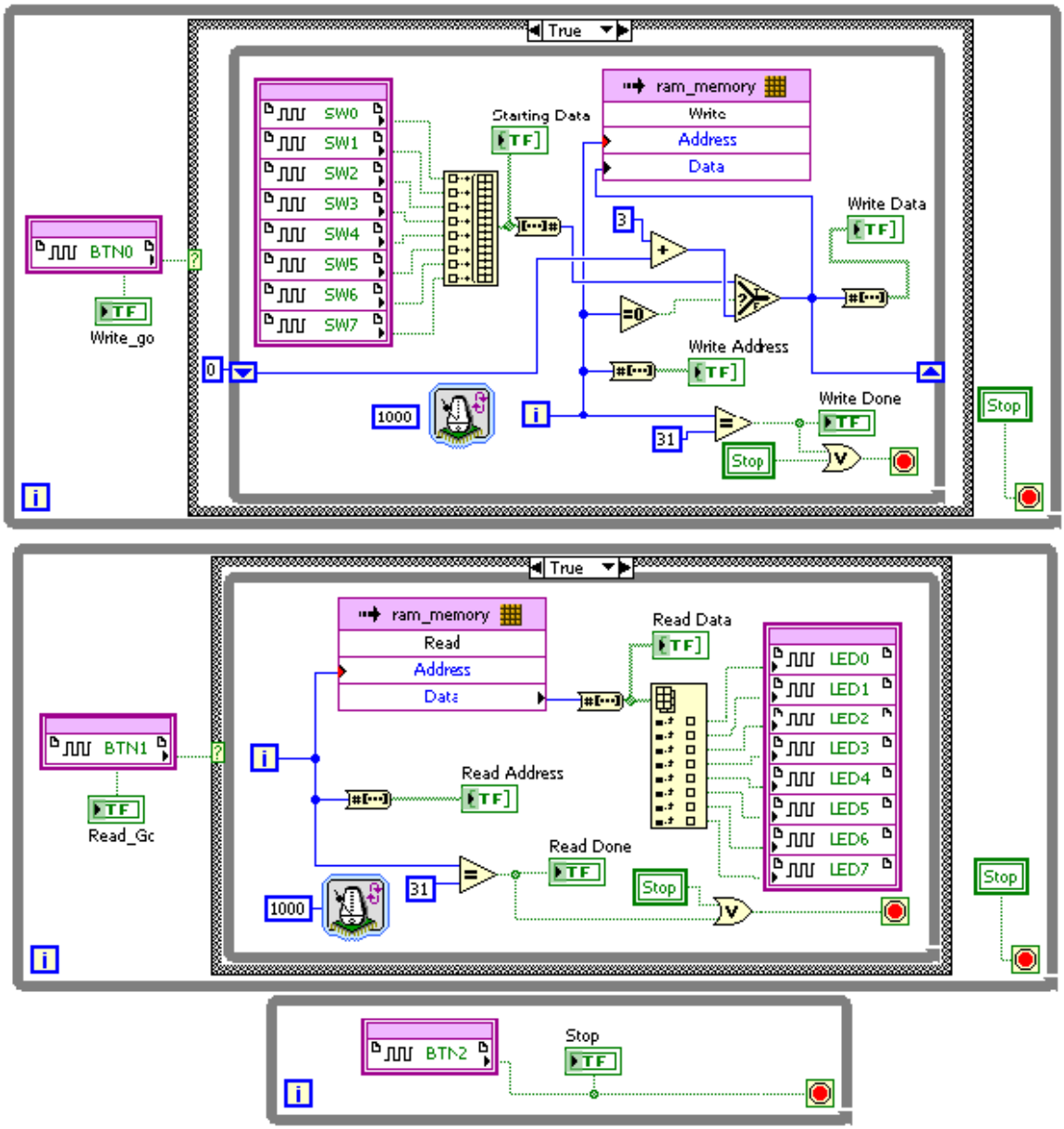


Нам нужно читать и записывать в память только при нажатии кнопок BTN1 и BTN0 соответственно. Следовательно, мы должны заключить два цикла while loop во фрейм True структур Case

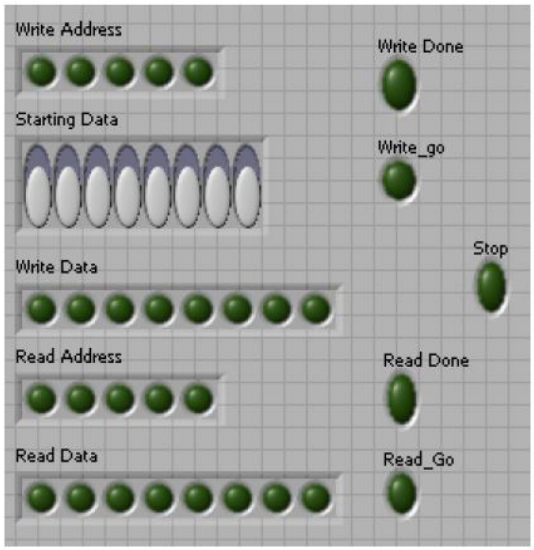
- Добавьте структуры Case вокруг каждого цикла
- Добавьте **BTN0** и подключите ее к селектору выбора структуры Case для записи. Аналогично добавьте **BTN1** и подключите ее к селектору выбора структуры Case для чтения
- Создайте индикаторы для BTN0 и BTN1 и обозначьте их **Write_Go** и **Read_Go** соответственно
- Поскольку нам нужно продолжать чтение или запись, пока не будет нажата кнопка Stop, кнопки чтения и записи должны быть заключены во внешний цикл while loop
- Создайте еще один цикл while loop, заключив в него кнопки и структуры Case для чтения и записи
- Затем создайте еще один цикл while для кнопки Stop (BTN2)
- Соедините BTN2 с терминалом выхода из цикла while
- Создайте индикатор для BTN2 и обозначьте его **Stop**

Теперь нам нужно останавливать процессы чтения и записи при нажатии кнопки BTN2

- Щелкните правой кнопкой мыши по индикатору Stop и выберите **Select>>local variable**
- Поместите локальную переменную Stop внутрь внутреннего цикла чтения
- Таким же образом создайте локальную переменную Stop внутри внутреннего цикла записи
- Измените назначение этих переменных для чтения и соедините их так, чтобы исполнение кода в циклах завершалось после обращения к последней ячейке памяти или при нажатии кнопки BTN2 (как показано на рисунке ниже)
- Аналогично создайте локальные переменные Stop для внешних циклов чтения и записи, измените назначение этих переменных для чтения и соедините их с терминалами выхода из циклов while
- Результат проектирования показан ниже



- Переключитесь на лицевую панель и упорядочьте индикаторы, как показано ниже

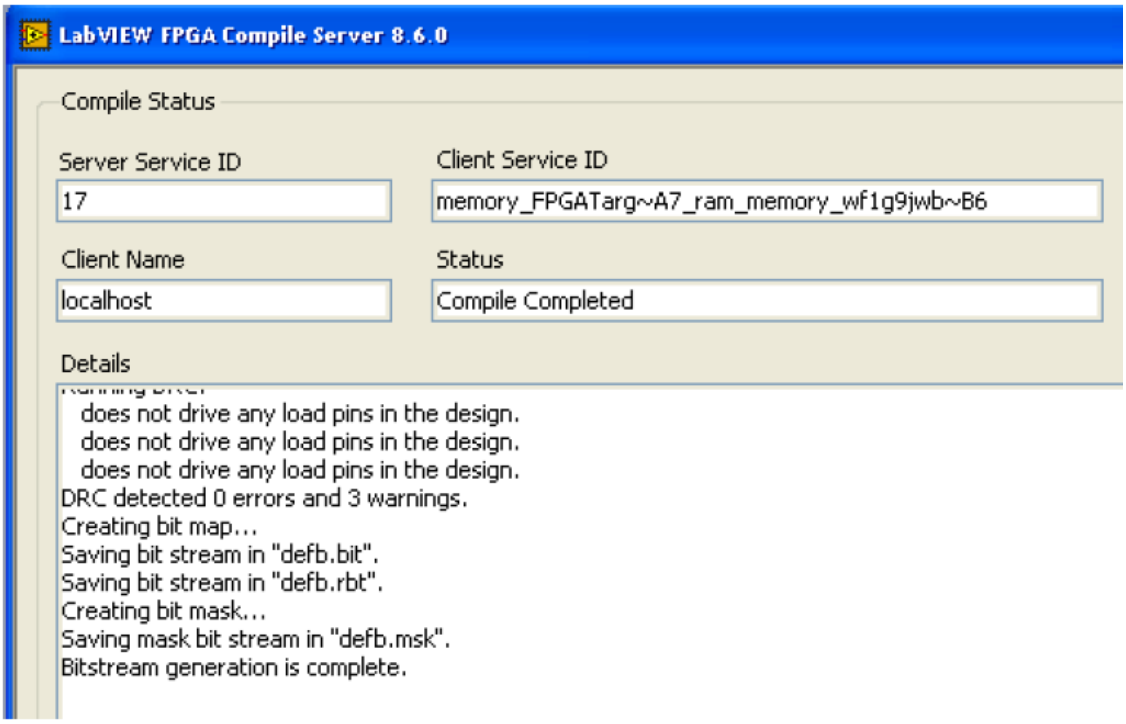


- Сохраните VI

Раздел 2: Проверка результатов проектирования с помощью оценочного модуля

Порядок выполнения:

- Подключите оценочный модуль, USB кабель и включите питание
- 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
- Щелкните по кнопке **ОК**, чтобы закрыть сводное окно состояния компиляции



FPGA сконфигурирован – об этом свидетельствует подсветка кнопки Run. Протестировать разработанное устройство вы можете следующим образом.

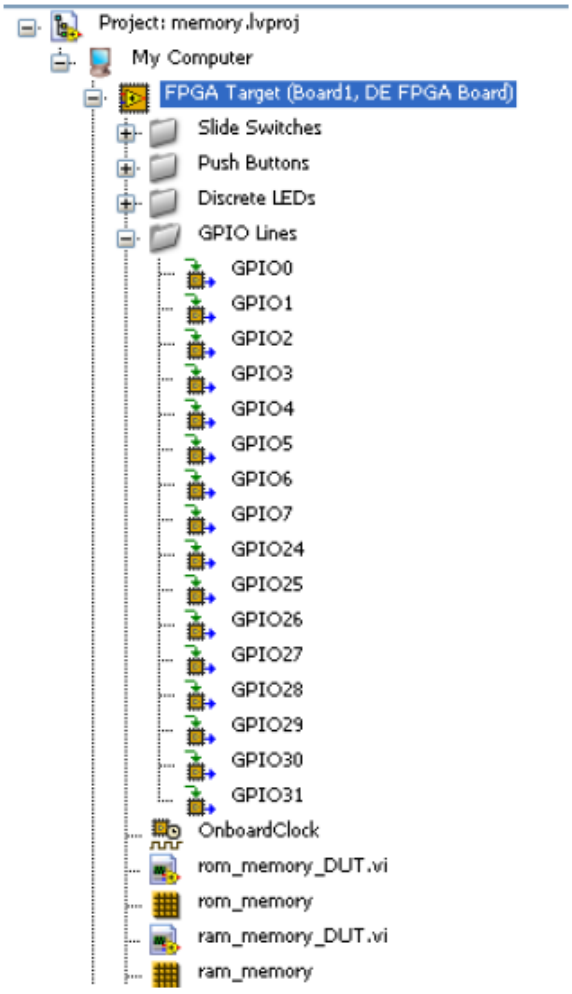
Установите на оценочном модуле переключатели SW7÷SW0 в положение 00000010 (число 2 будет запомнено в ячейке памяти 0, число 5 – в ячейке 1 и т.д.) и нажмите на кнопку BTN0. На лицевой панели вы должны увидеть включение индикатора Write_Go и начало инкрементирования адреса (Address). Кроме того, вы увидите, как изменяются записываемые данные. Когда адрес станет равным 31 (11111) включатся индикатор Write Done, индикаторы Write Go выключится и выполнение задачи записи остановится. На индикаторе записываемых данных Write Data должно отображаться число 11111010 (5F). Теперь нажмите на кнопку BTN1 – вы должны увидеть включение индикатора Read_Go, а на индикаторах Read Data текущие считываемые данные, на светодиодах оценочного модуля отображается то же самое. Когда адрес станет равным 31 (11111) на индикаторе Read Data будет отображаться число 11111010 (5F).

Вы можете изменить положение переключателей и затем нажать на кнопку Write_Go (BTN0). Вам не нужно ждать завершения процесса записи, чтобы нажать кнопку Read_Go (BTN1). Остановить выполнение задач можно в любое время, нажав кнопку (BTN2). Убедитесь, что процесс ведет себя, как ожидалось

Раздел 3: Проверка результатов проектирования с помощью VI тестера

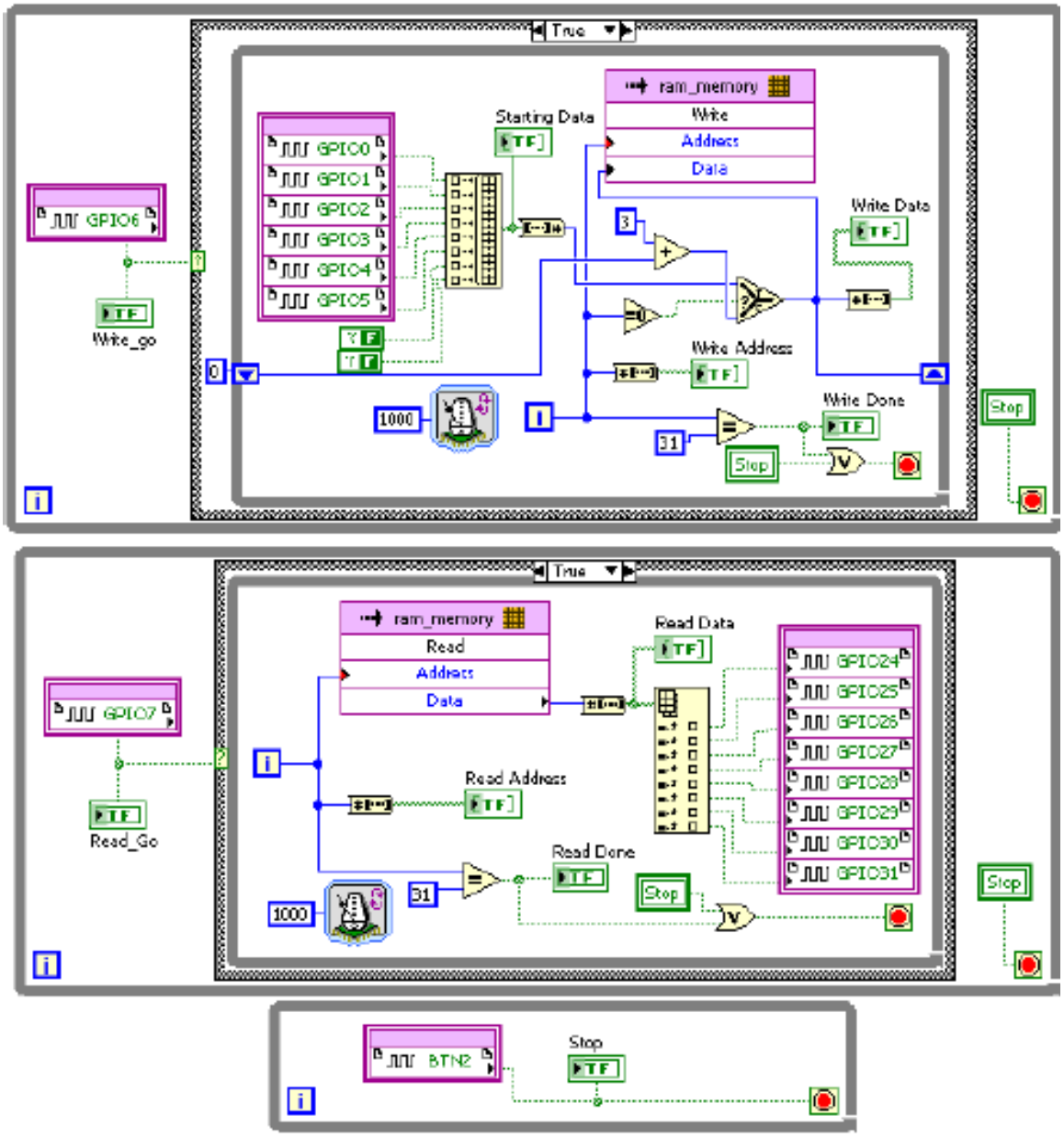
Порядок выполнения:

- Сохраните VI под именем **ram_memory_DUT.vi**
- На блок-диаграмме удалите переключатели, светодиоды и кнопки (BTN0 и BTN1), 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 **GPIO27÷ GPIO31**, добавьте их в проект, а затем щелкните по кнопке **ОК**
- Окно Project Explorer должно выглядеть так



- Поместите на блок-диаграмму из субпалитры **FPGA I/O** узел **FPGA I/O Node**
- Щелкните правой кнопкой мыши по узлу и выберите **GPIO0**. Растяните узел так, чтобы в него вошли элементы от **GPIO0** до **GPIO5**
- Соедините выходы **GPIO0÷ GPIO5** с первыми 6 элементами функции **Build Array**, от которых были отключены узлы SW
- Подключите константу **False** к 7-му и 8-му элементу

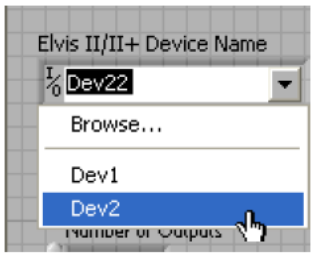
- Поместите на блок-диаграмму, туда, где был элемент BTN0, узел **FPGA I/O Node** из субпалитры **FPGA I/O**
- Щелкните правой кнопкой по добавленному узлу и выберите **GPIO6**
- Соедините выход **GPIO6** с входом селектора выбора Case структуры записи (см. блок-диаграмму ниже)
- Поместите на блок-диаграмму из субпалитры **FPGA I/O** узел **FPGA I/O Node** туда, где находился элемент BTN1
- Щелкните правой кнопкой мыши по узлу и выберите **GPIO7**
- Соедините выход **GPIO7** с входом селектора выбора Case структуры чтения (см. блок-диаграмму ниже)
- В окне Project Explorer выделите линии **GPIO24÷ GPIO31** и перетащите их в цикл while loop туда, где были подключены светодиоды (LED)
- Щелкните правой кнопкой мыши по добавленному узлу и измените его назначение для записи
- Ниже показана полученная блок-диаграмма



- Сохраните VI
- Щелкните по кнопке **Run**. Начнется компиляция проекта. Когда завершится генерация битового массива, щелкните по кнопке **ОК**, 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 BB1/BB2 и BB5 согласно следующей таблице

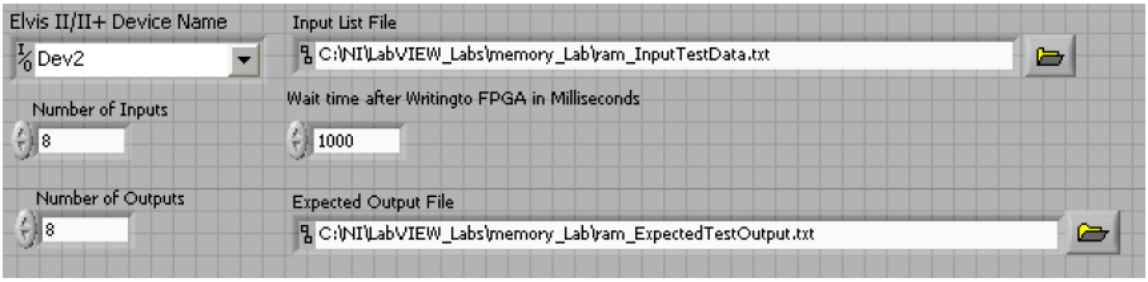
BB1/BB2 Connectors	BB5 Connector
GPIO0	DIO15
GPIO1	DIO14
GPIO2	DIO13
GPIO3	DIO12
GPIO4	DIO11
GPIO5	DIO10
GPIO6	DIO9
GPIO7	DIO8
GPIO24	DIO7
GPIO25	DIO6
GPIO26	DIO5
GPIO27	DIO4
GPIO28	DIO3
GPIO29	DIO2
GPIO30	DIO1
GPIO31	DIO0

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

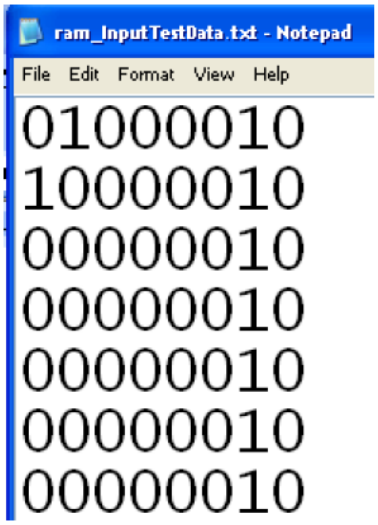


- Измените **Number of Inputs** на **8** и **Number of Outputs** на **8**
- Измените время Wait на **1000** ms
- Щелкните по кнопке Browse для файла Input и добавьте **ram_InputTestData.txt**
- Таким же образом добавьте **ram_ExpectedTestOutput.txt** с ожидаемыми состояниями выходов.

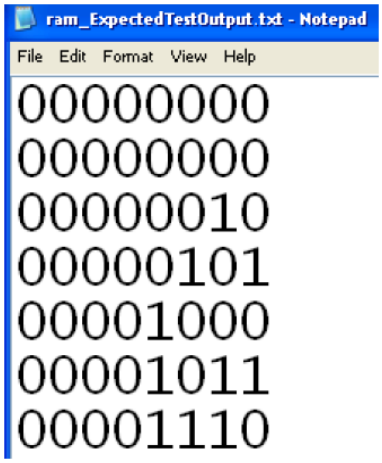
На этом этапе проектирования лицевая панель должна выглядеть похожей на изображенную ниже



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



В соответствии с этим входным файлом канал DIO8 назначен в качестве кнопки Read_Go, канал DIO9 назначен в качестве кнопки Write_Go,а каналы с DIO10 по DIO15 – используются для задания начальных данных [5:0]. 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 каналы с DIO0 по DIO7 присвоены выходам памяти [7:0]

- Удостоверьтесь, что **ram_memory_DUT.vi** продолжает исполняться
- Щелкните по кнопке Run на лицевой панели тестера. Тест запустится на исполнение, а отображаемые результаты свидетельствуют об успешном завершении теста
- Теперь измените содержимое файла ожидаемых выходных реакций. Измените третий набор выходных сигналов с 00000010 на 00000011, при этом тест должен показать ошибку. Сохраните изменения
- Выключите и включите питание ELVIS и оценочного модуля FPGA
- Запустите еще раз ram_memory_DUT и тестер, 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
- Если **ram_memory_DUT** продолжает работать, остановите его, проверив тем самым останов работы устройства

Выводы:

В этой работе вы научились проектировать базовые схемы памяти RAM и ROM. Вы также научились создавать генератор последовательно нарастающих адресов, если количество ячеек памяти ограничено степенью 2. Вы проверили результаты проектирования с помощью оценочного модуля DE FPGA Board и системы NI ELVIS.