МИНОБРНАУКИ РОССИИ

Санкт-Петербургский государственный
электротехнический университет «ЛЭТИ»

ПРОЕКТИРОВАНИЕ РЕКОНФИГУРИРУЕМЫХ СИСТЕМ НА КРИСТАЛЛЕ

Методические указания
к лабораторным работам

Санкт-Петербург
Издательство СПбГЭТУ «ЛЭТИ»
2014

УДК 004.31(075)

Проектирование реконфигурируемых систем на кристалле: метод. указания к лаб. работам / сост.: О. И. Буренева, Р. И. Грушвицкий. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2014. 48 с.

Содержат описания лабораторных работ по созданию и способам отладки систем на кристалле на базе soft-процессора NIOS II с использованием различных способов передачи данных.

Предназначены для подготовки магистров техники и технологии по программе «Микросистемные компьютерные технологии: системы на кристалле». Могут быть использованы и при подготовке специалистов и бакалавров техники и технологии по профилю «Вычислительные машины, комплексы, системы и сети», а также специалистами, занимающимися разработками электронной аппаратуры.

Утверждено
редакционно-издательским советом университета
в качестве методических указаний

Введение

Современные реконфигурируемые системы на кристалле (СнК) могут содержать до трех подсистем. Каждая подсистема имеет свои собственные принципы функционирования, структурной и технологической организации, процедуры исследования, проектирования и отладки. Речь идет:

– о подсистеме дискретной вычислительной логики (иногда называемой свободной логикой и реализуемой обычно на FPGA);

– программной подсистеме (процессоре со своей периферией);

– аналого-цифровой подсистеме (программируемых аналоговых блоков и\или ЦАП, АЦП и сопутствующего аналогового оборудования).

В соответствии с этим цикл лабораторных работ состоит из трех разделов.

В первом разделе предлагается выполнить работу, связанную с вопросами практической имплементации проектов в ПЛИС. Особое внимание уделяется организации системы тактирования в синхронных системах. Во второй части раздела при помощи средств внутрикристальной отладки (пакет SignalTAP II САПР Quartus II фирмы «Altera») требуется определить временной интервал дребезга контактов электромеханических элементов. Заключительная часть раздела – проектирование исходной системы с механизмами устранения дребезга контактов.

Во втором разделе планируется выполнение четырех работ, ориентированных на применение процессоров встраиваемых в ПЛИС (NIOS у фирмы «Altera»):

– первая работа предполагает проектирование системы, функционирование которой полностью определяется содержимым программы процессора, и поддерживается стандартным периферийным оборудованием;

– при выполнении второй работы осуществляется сравнение затрат и достигаемого быстродействия при создании решения функционально совпадающей задачи при аппаратной и программной реализации;

– третья и четвертая работы призваны показать особенности реализации операций асинхронного ввода-вывода: использование прерываний и использование контроллеров прямого доступа к памяти соответственно.

В третьем разделе, не вошедшем в данные методические указания, запланировано выполнение работ, связанных с моделированием поведения аналоговых или цифровых схем средствами пакетов аналого-цифрового моделирования.

Лабораторная работа 1.
ИМПЛЕМЕНТАЦИЯ ПРОЕКТОВ В РЕАЛЬНЫЕ ИС

Цель работы состоит в получении навыков создания проекта, его моделирования, имплементации в программируемую логическую интегральную схему (ПЛИС) с последующей внутрикристальной отладкой проекта с использованием встроенного в САПР логического анализатора.

Краткие теоретические сведения

Один из способов оценки корректности проектов основан на моделировании, предполагающем отладку проекта на модели путем анализа реакций разрабатываемых схем на стимулирующие воздействия. Несмотря на высокую вероятность обнаружения имеющихся дефектов моделирование не всегда позволяет оценить работу схемы. Более эффективны методы, основанные на экспериментах с реальным оборудованием. Получившие в последнее время широкое распространение прототипные платы разработчика, содержащие ПЛИС, позволяют организовать подобные эксперименты.

Для получения навыков работы с реальной ПЛИС предлагается провести модельную отладку проекта, представленного исходным описанием на языке VHDL, имплементацию проекта в ПЛИС учебного стенда и анализ работоспособности полученной схемы.

Отладка созданного проекта будет состоять из двух этапов: моделирования проекта и отладки на реальной схеме. Моделирование осуществляется средствами и методами, изложенными в [1].

Отладка проекта на реальной схеме будет производиться с помощью встроенного в пакет Quartus II логического анализатора *Signal Tap II Logic Analyzer*. Использование встроенного логического анализатора – это один из способов внутрикристальной отладки проекта, возможный при наличии неиспользованных ресурсов ПЛИС. Созданные в процессе проектирования логические анализаторы загружаются в ПЛИС и подсоединяются к интересующим разработчика цепям; таким образом, разработчик имеет возможность наблюдать за реальными состояниями различных сигналов, фиксируемых логическим анализатором. Signal Tap II позволяет создавать и встраивать в проект определенное число логических анализаторов, оперативно изменять условия фиксации данных в их памяти и отображать эти данные на экране компьютера.

Задание на работу

Разработать на языке VHDL описание комбинационной схемы, выполняющей функции соответствующие выбранному варианту задания. Произвести моделирование проекта, выполнить его имплементацию в ПЛИС и отладку на реальной схеме.

Последовательность выполнения работы

Этап 1. Создание проекта в САПР Quartus II

1. Подготовить директорию для проекта, запустить пакет Quartus II.

2. Создать новый проект, используя меню *File/New Project Wizard*. В появившемся диалоговом окне *New Project Wizard: Directory, Name, Top-Level Entity* указать путь к подготовленной директории и имя проекта.

Замечание: имена проектов и каталогов не должны содержать русских букв и пробелов.

3. Выбрать семейство и тип микросхемы (для используемого учебного стенда DE0 – Cylone III, EP3C16F484C6).

4. Подготовить описание схемы в соответствии с вариантом задания. Для этого создать новый VHDL-файл, используя меню *File/New*, и описать в нем комбинационную схему. Кроме используемых в схеме входных и выходных сигналов описания портов схемы должны содержать входной тактовый сигнал Clk, необходимый для работы логического анализатора. Имя файла вершины проекта должно совпадать с именем ENTITY.

5. Провести предварительную компиляцию созданного описания для проверки его на наличие синтаксических ошибок, используя меню *Processing/Analyze current file*, а затем полную компиляцию с помощью меню *Processing/Start Compilation*.

6. Выполнить «распиновку» кристалла: для каждого входного и выходного сигнала проекта назначить реальный контакт ПЛИС. Это действие выполняется при помощи утилиты *Pin Planer*, запускаемой через меню *Assignments/Pins Planner*. В окне *Pin Planer* представляется изображение выбранной ПЛИС с условными обозначениями выводов, отражающими их функциональность. Двойной щелчок на изображении вывода вызывает диалоговое окно, в котором за выводом закрепляется определенный входной или выходной сигнал проекта.

Номера необходимых для выполнения данной лабораторной работы контактов ПЛИС учебного стенда DE0 указаны в приложении.

7. После назначения контактов необходимо выполнить повторную компиляцию проекта. По завершении компиляции допустимо некоторое количество предупреждений. Результатом успешной компиляции является создание в директории проекта файла конфигурации *.sof.

8. Оценить затраты на реализацию проекта, используя команду *Processing/Compilation Report*.

9. Просмотреть RTL-вид проекта, активизировав команду – *Tools/Netlist Viewer/RTL Viewer – Tools/Technology Map View*.

10. Оценить временные характеристики реализации проекта – *Timing analyzer*.

11. Просмотреть расположение проекта в заданной ПЛИС – *Floorplan Editor*.

Этап 2. Моделирование проекта

1. Создать файл, содержащий входные тестовые воздействия, с помощью редактора временных диаграмм, воспользовавшись командой *File/New*, выбрав тип *Vector Waveform File*. Созданный файл необходимо сохранить, используя команду *File/Save As*. При этом среда проектирования предложит сохранить файл с именем, совпадающим с именем файла верхнего уровня проекта, присвоив ему расширение *.vwf. При сохранении необходимо проконтролировать состояние флажка *Add file to current project*, установка которого обеспечивает включение созданного файла в текущий проект.

2. Сформировать диаграммы входных воздействий. В открывшемся окне редактора временных диаграмм необходимо ввести входные и выходные сигналы проекта, воспользовавшись командой меню *Edit/Insert Node or Bus* или же двойным щелчком на области входных и выходных сигналов окна редактора.

Редактироваться в этом окне могут только входные сигналы, присутствующие в текущем проекте. Вектор входных воздействий, необходимый для

моделирования, вводится с помощью специального инструмента заданием для выбранных временных интервалов логических уровней, соответствующих значениям входной переменной в заданном узле проекта.

3. Запустить моделировщик САПР Quartus II, используя меню *Processing/Start Simulation*. Моделирование возможно в двух режимах: функциональное (*Functional Simulation*) и с учетом временных параметров схемы

(*Timing Simulation*). В первом случае проверяется правильность описания и логического функционирования проекта, во втором – функционирование проекта с учетом временных параметров выбранной схемы. По умолчанию задан режим *Timing Simulation*. При выборе режима *Functional Simulation* перед началом симуляции необходимо сформировать файл со списком соединений компонентов проекта, выполнив операцию *Generate Functional Simulation Netlist*.

4. Провести анализ временных диаграмм, полученных в результате моделирования, убедиться в правильности функционирования устройства.

Этап 3. Загрузка проекта в ПЛИС

1. Подключить учебную плату к компьютеру через порт USB и включить плату.

2. Запустить программатор командой меню *Tools/Programmer* и убедиться в том, что он обнаружил плату. В противном случае установить ее с помощью кнопки *Hardware setup* и в открывшемся окне выбрать *USB-Blaster*.

3. Загрузить проект в ПЛИС, нажав кнопку *Start*.

Этап 4. Отладка проекта на реальной схеме

Выполнить действия над проектом, предусмотренные процедурой отладки; убедиться, что результаты реального эксперимента не соответствуют результатам эксперимента модельного: вместо одного счетного импульса в систему проходит ряд импульсов.

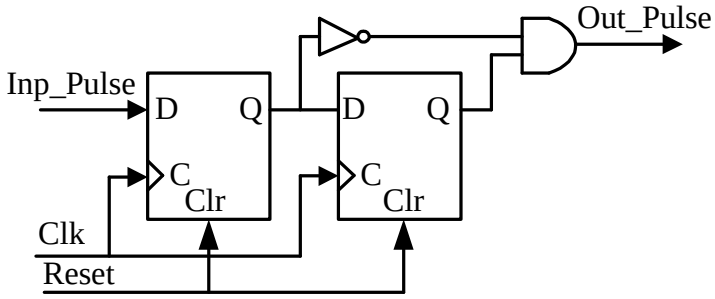


Рис. 1.1. Схема выделения одиночного импульса

Включение схемы выделения одиночного импульса (рис. 1.1) между движковым переключателем и входом разрешения счета в проекте полностью возникающие проблемы не решает: паразитные импульсы являются следствием дребезга контактов в электромеханических элементах. Возникает необходимость выяснения временных параметров (максимальной длительности) времени дребезга. Вторая часть работы заключается в определении этого временного интервала.

Этап 5. Определения интервала дребезга контактов

1. Запустить программное обеспечение логического анализатора Signal Tap II, командой меню *Tools/Signal Tap II Logic analyzer* (рис. 1.2).

2. Активизировать связь анализатора и платы нажатием кнопки *Setup* в верхней правой части анализатора, выбрав при этом *USB-Blaster*.

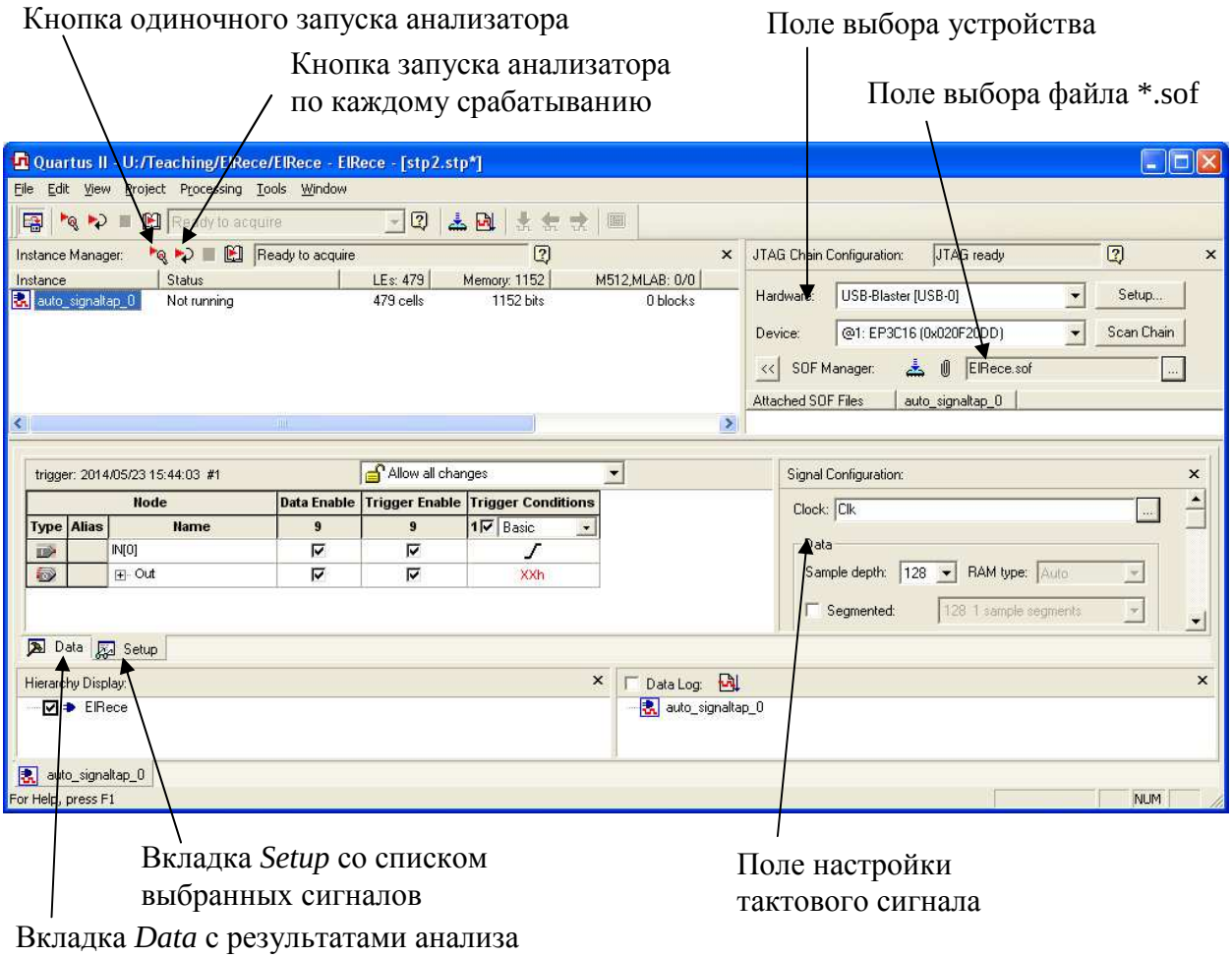


Рис. 1.2. Окно анализатора Signal Tap

3. Выбрать сигнал тактирования анализатора. Для этого в панели *Signal configuration* нажать кнопку *Browse Node Finder*, в открывшемся окне *Node Finder* установить фильтр *Signal Tap II: pre-synthesis* и нажать кнопку *List*. Из появившегося списка выбрать тактовый сигнал.

4. Создать список сигналов, необходимых для отладки схемы: во вкладке *Setup* дважды кликнуть по любому месту – откроется *Node Finder*. Убедиться, что выбран фильтр *Signal Tap II: pre-synthesis* и нажать кнопку *List*. В появившемся списке выбрать нужные для анализа сигналы.

5. Выполнить полную компиляцию проекта, в процессе которой в ответ на вопрос система предложит сохранить и добавить в проект файл *.stp, в ко-

тором содержится информация о структуре созданного логического анализатора. Вручную это можно сделать с помощью команды меню *Assignments/Settings/Signal Tap II Logic analyzer*.

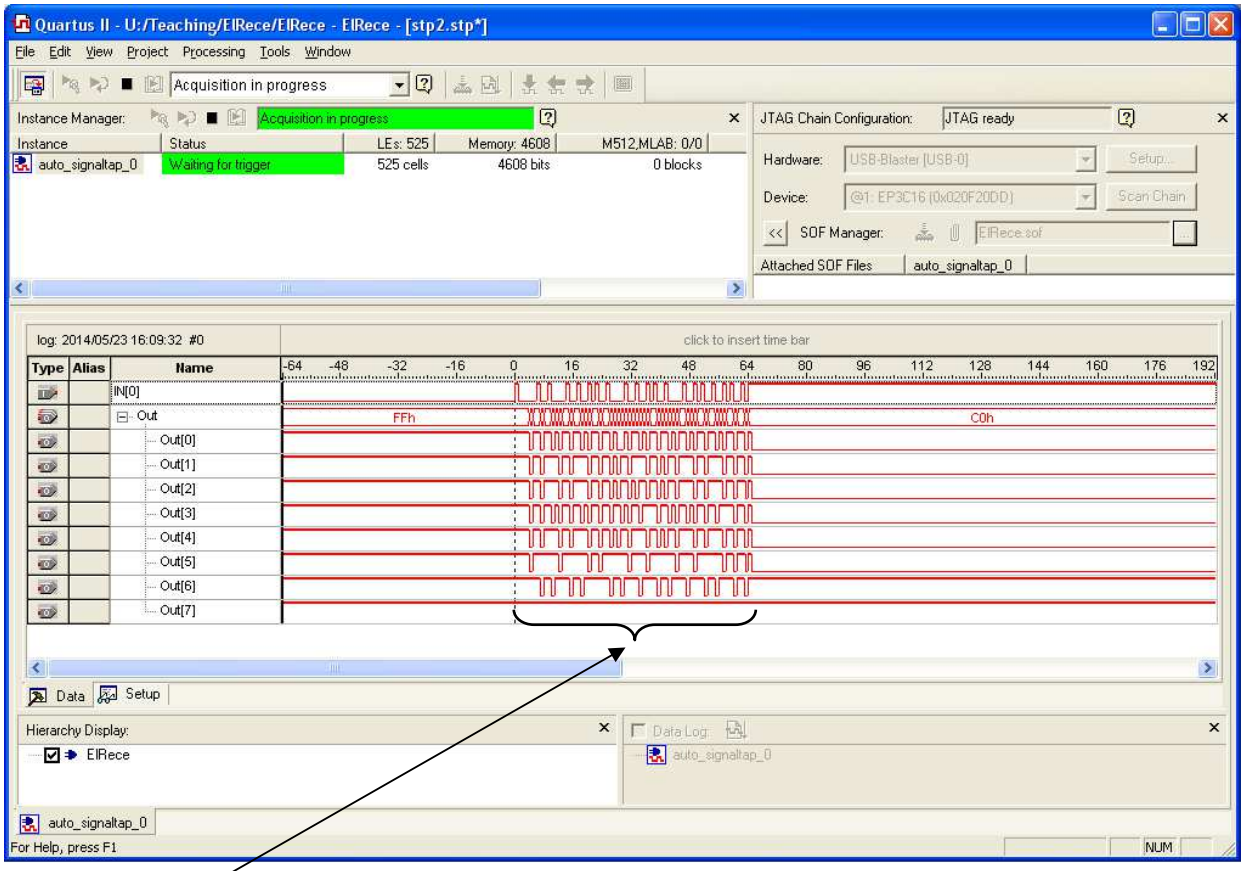


Рис. 1.3. Временная диаграмма, полученная анализатором

6. Перезагрузить проект в ПЛИС. Рядом с полем *SOF Manager*, находящемся в верхней правой части окна *Signal Tap*, нажать кнопку *Brows* и выбрать из директории проекта соответствующий файл *.sof. Непосредственная загрузка проекта в ПЛИС осуществляется нажатием кнопки *Program Device*.

7. Запустить логический анализатор в работу.

7.1. Задать условие срабатывания логического анализатора, для чего указать запускающий его сигнал во вкладке *Setup* и нажатием правой клавиши в колонке *Trigger Conditions* в открывшемся списке выбрать необходимое условие.

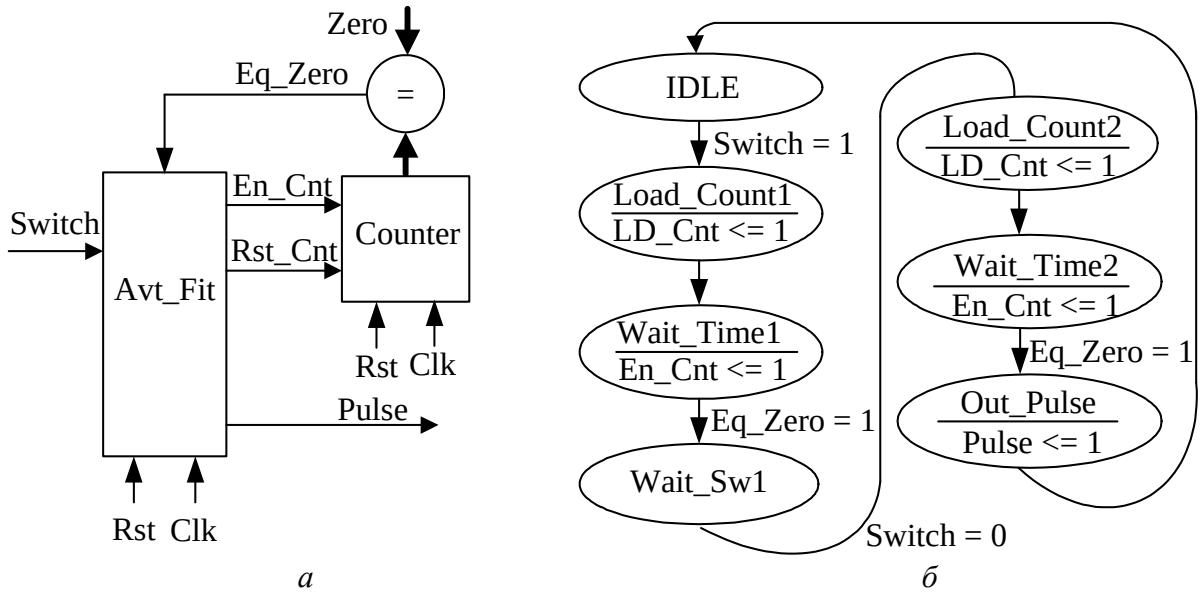
7.2. Перейти во вкладку *Data* и запустить логический анализатор, нажав на кнопку *Run analysis* – одиночный запуск анализатора или *Autorun Analysis*, – анализатор автоматически запускается каждый раз, когда выполняется установленное условие.

7.3. На учебной плате выполнить действие, соответствующее заданному условию (например, переключить соответствующий переключатель). В ответ на это в окне анализатора появится временная диаграмма зафиксированных сигналов (рис. 1.3).

В результате анализа линии переключателя на диаграмме выявлен дребезг (рис. 1.3), длительность которого соответствует 64 импульсам сигнала тактирования *Clk*. Дребезг на диаграмме может отсутствовать по причине ограниченного временного интервала работы логического анализатора. Для того чтобы дребезг проявился, следует ввести в схему счетчик 18–20 двоичных разрядов, на счетный вход которого подается сигнал тактирования логического анализатора *Clk*. Использование различных разрядов счетчика в качестве тактовых сигналов логического анализатора и анализ получаемых диаграмм позволит определить искомый диапазон дребезга.

Этап 6. Устранение дребезга контактов

Для устранения влияния дребезга в основной проект между входным контактом переключателя тактирования и разрешающим входом тактируемого компонента проекта необходимо вставить схему устранения дребезга (рис. 1.4, *а*) с соответствующим управляющим автоматом (рис. 1.4, *б*).



а *б*

Рис. 1.4. Схема устранения дребезга контакта и граф управляющего автомата

После внесения изменений в проект выполнить его компиляцию, загрузку в учебную плату и убедиться в правильности его работы с отладчиком и без него.

Содержание отчета

Отчет должен содержать следующую информацию:

1. Описание иерархической структуры проекта.
2. Тексты VHDL-программ, соответствующие модулям проекта.
3. Список и назначение внешних контактов.
4. Оценки варианта реализации (затраты оборудования, времена распространения сигналов вход-выход).
5. Временные диаграммы, полученные в ходе моделирования.
6. Описание логического анализатора.

Варианты заданий

1. Реализовать 10-разрядный счетчик кода Джонсона нажатий клавиши «счет» с выводом информации на светодиодные индикаторы.
2. Реализовать 16-разрядный двоичный счетчик нажатий клавиши «счет» с выводом выходной информации на семисегментные индикаторы.
3. Реализовать двухразрядный двоично-десятичный счетчик нажатий клавиши «счет» с выводом выходной информации на семисегментные индикаторы.
4. Реализовать 10-разрядный регистр сдвига содержимого вправо, реагирующий на каждое нажатие клавиши «сдвиг», с выводом выходной информации на светодиодные индикаторы. Значение вдвигаемого младшего разряда задается переключателем «данные».
5. Реализовать 10-разрядный регистр сдвига содержимого влево, реагирующий на каждое нажатие клавиши «сдвиг», с выводом выходной информации на светодиодные индикаторы. Значение вдвигаемого старшего разряда задается переключателем «данные».

Во всех заданиях информация в регистрах может обнуляться при включении переключателя «сброс» в единичное состояние.

Лабораторная работа 2.
РАЗРАБОТКА ПРОЕКТОВ С ИСПОЛЬЗОВАНИЕМ SOFT-ЯДРА ПРОЦЕССОРА NIOS II

Цель работы состоит в освоении процесса конфигурирования системы на кристалле (СнК) на базе процессора Nios II с использованием среды SOPC Builder и получения навыков разработки программного обеспечения в среде Nios II IDE.

Краткие теоретические сведения

Soft-ядро процессора Nios II встраивается в FPGA семейств Cyclone, Arria, Stratix и HardCopy компании «Altera» и представляет собой конвейерный RISC-процессор. Наряду с процессором Nios II в системе могут быть использованы периферийные модули (например, порты ввода-вывода, модули интерфейсов USB, SRAM, SPI, I^2C, таймер и др.), а также блоки памяти для хранения кода программы и данных. При этом применяются только те компоненты, которые необходимы для реализации функций проектируемой системы. Компоненты СнК объединяются с помощью специально разработанной шины Avalon, к ней же подключаются и внешние устройства.

Интеграция СнК на базе процессора Nios II выполняется с использованием программного пакета SOPC Builder. Он обеспечивает конфигурирование и подключение компонентов посредством шины Avalon в единую СнК, результатом проектирования является VHDL-проект СнК, пригодный для последующего использования в САПР Quartus II в качестве отдельного модуля.

Задание на работу

Разработать программно-аппаратную систему, состоящую из ядра процессора Nios II, памяти и блока ввода-вывода (рис. 2.1) и реализующую функцию в соответствии с индивидуальным заданием.

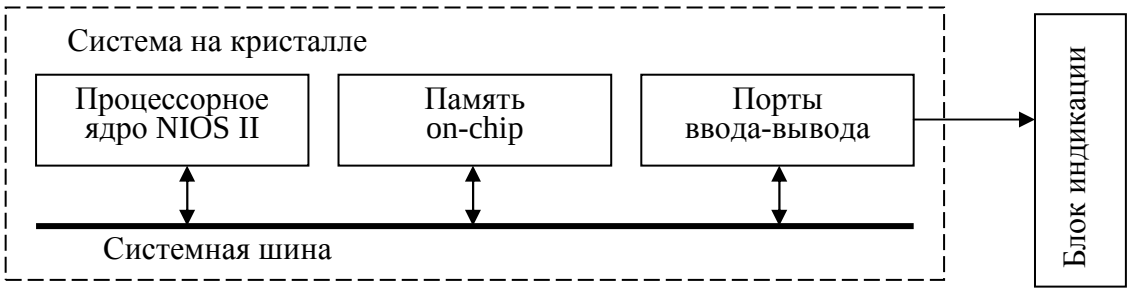


Рис. 2.1. Структура разрабатываемой системы на кристалле

Выполнить имплементацию проекта СнК в ПЛИС отладочной платы DE0 фирмы «Terasic». Для отображения результатов работы системы использовать светодиодные индикаторы, расположенные на плате.

Последовательность выполнения работы

Последовательность работы рассмотрим на примере создания системы, управляющей перемещением светящегося элемента по восьмиразрядной светодиодной линейке, при достижении крайнего правого или левого элемента направление перемещения меняется на противоположное.

Этап 1. Создание проекта системы

При создании проекта использовать последовательность действий, приведенную в пп. 1–3 этапа 1 лаб. раб. 1.

Этап 2. Конфигурирование аппаратных модулей СнК

1. Запустить пакет конфигурации NIOS II – SOPC Builder командой меню *Tools/SOPC Builder*. В окне *Create New System* задать имя системы и язык генерации кода проекта.

В появившемся окне (рис. 2.2) требуется указать параметры тактового сигнала (частота генератора платы DE0 50 МГц). На вкладке *System Contents* представлены доступные компоненты библиотеки *Component Library*, и в этом же окне в дальнейшем будет отражен состав СнК.

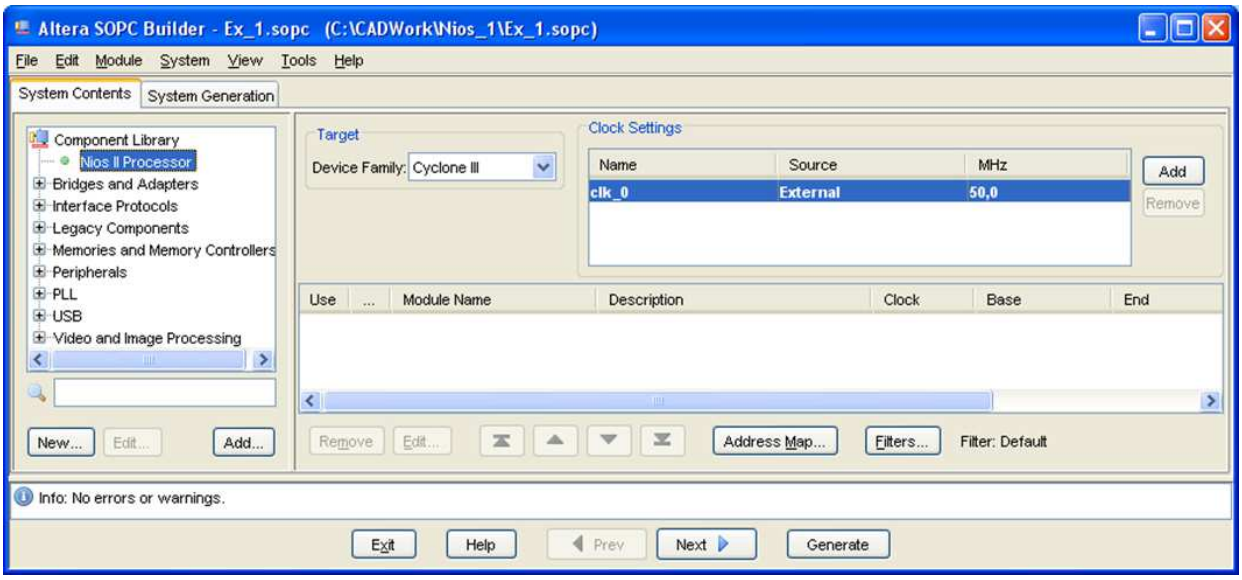


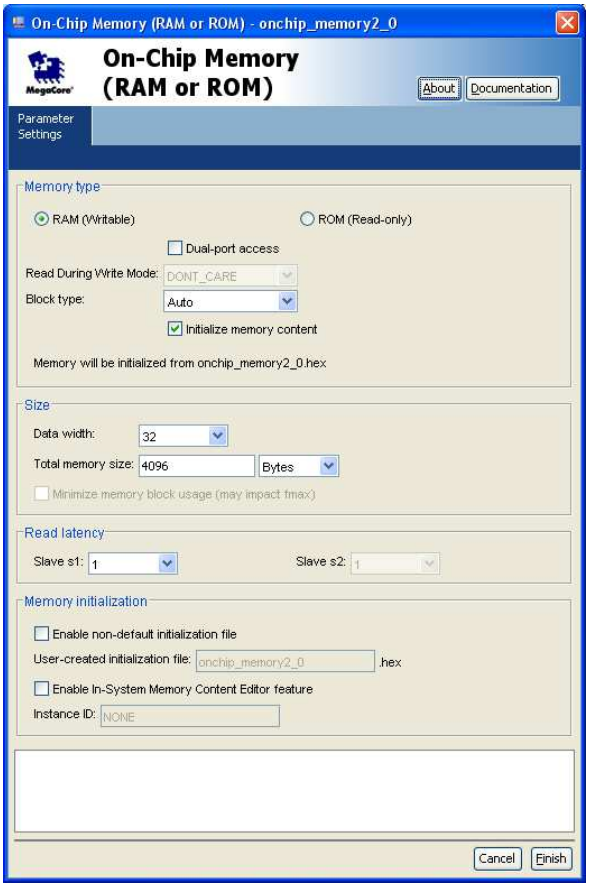
Рис. 2.2. Окно SOPC Builder

1.1. Сконфигурировать soft-ядро процессора Nios II, выбрав в библиотеке компонент «Nios II Processor» и указав в появившемся окне *Nios II Processor* требуемую модификацию ядра.

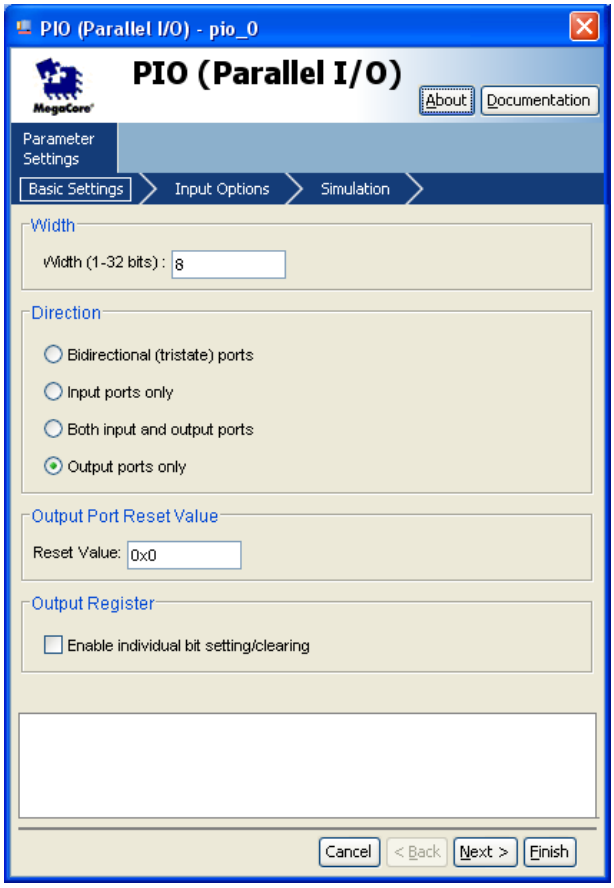
Предлагается три варианта ядер: Nios II/f (Fast) – оптимизированное по производительности; Nios II/e (Economy) – оптимизированное по площади, но ограниченное по функциональным возможностям; Nios II/s (Standart) – отличающееся минимальными требованиями к ресурсам и невысокой производительностью, при этом имеющее набор дополнительных аппаратных модулей. В этом же окне задаются адреса и смещения векторов прерываний и исключений, выполняются настройки кэш-памяти ядра, модуль отладки JTAG и другие элементы.

Для разрабатываемой системы достаточно установить ядро Nios II/e. Векторы прерываний и исключений на этом этапе могут оставаться незаполненными.

1.2. Сконфигурировать память, выбрав модуль памяти из библиотеки в разделе *Memories and Memory Controllers/On-Chip/On-Chip Memory (RAM or ROM)*. В окне конфигурации памяти настраиваются тип памяти RAM или ROM, общий объем и разрядность данных, файл инициализации памяти. Вариант настроек для текущего проекта приведен на рис. 2.3, *а*. После установки необходимых параметров и нажатия кнопки *Finish* система вернется на вкладку *System Contents*, а модуль памяти сразу появится в таблице доступных компонентов.



а



б

Рис. 2.3. Окна настроек модулей системы

1.3. Настроить порт ввода-вывода, через который будут выводиться данные на светоизлучающие диоды. Соответствующий элемент находится в разделе *Peripherals/Microcontroller Peripherals/PIO (Parallel I/O)*. В окне его конфигурации определяются разрядность, направление, значения после сброса и возможность побитового обращения к выходному регистру.

Поскольку в текущем проекте порт настраивается только на вывод информации, дополнительные опции на закладке *Input Options* не устанавливаются. Пример настроек порта ввода-вывода приведен на рис. 2.3, *б*.

Любой компонент после добавления в состав системы может быть переименован, что облегчит работу по созданию программного обеспечения, так как программы для NIOS II при обращении к конкретным элементам используют символьные константы. Для переименования можно воспользоваться контекстным меню, вызываемым на соответствующем элементе СнК.

1.4. Когда все необходимые аппаратные компоненты в систему добавлены, нужно определить форму взаимодействия этих компонентов в системе: назначить базовый адрес для каждого ведомого компонента и приоритеты прерываний.

SOPC Builder позволяет назначить базовые адреса автоматически с помощью команды *System/Auto-Assign a base addresses*. В большинстве случаев это допустимо, однако при необходимости базовые адреса могут быть скорректированы вручную с учетом следующих особенностей:

– ядра процессора NIOS II имеют диапазон адреса в 31 бит, следовательно, адреса должны быть ограничены значениями от 0x00000000 до 0x7FFFFFF;

– адреса разных компонентов, различающиеся только одним битом, способствуют созданию более эффективных устройств;

– адреса, укомплектованные в наименьший диапазон адресов, снижают эффективность устройств.

1.5. Определить адреса векторов прерываний и исключений, запустив окно настройки ядра Nios, в котором и указываются необходимые векторы *Reset Vector* и *Exception Vector* в соответствии с адресами памяти на кристалле. Система сама подставит физические адреса и их смещения для векторов. Результат конфигурирования СнК представлен на рис. 2.3.

1.6. Сгенерировать код Nios II нажатием кнопки *Generate*. После генерации в папке проекта появится директория «sopc_builder» с автоматически сгенерированным HDL-кодом (RTL-описание проекта Nios II).

Этап 3. Конфигурирование системы на кристалле ПЛИС

1. Добавить автоматически полученные файлы к проекту *Quartus II* командой меню *Project/Add/Remove Files in project*.

2. Установить сгенерированный файл системы файлом верхнего уровня, открыв его и выполнив команду *Project/Set as top-level entity*.

Замечание: если были заданы разные имена проекта и ядра, то эта опция недоступна, требуется создать еще один, например графический файл, с именем проекта который и будет файлом верхнего уровня, а в нем в качестве элемента использовать сгенерированное ядро, подсоединив к нему входные и выходные контакты.

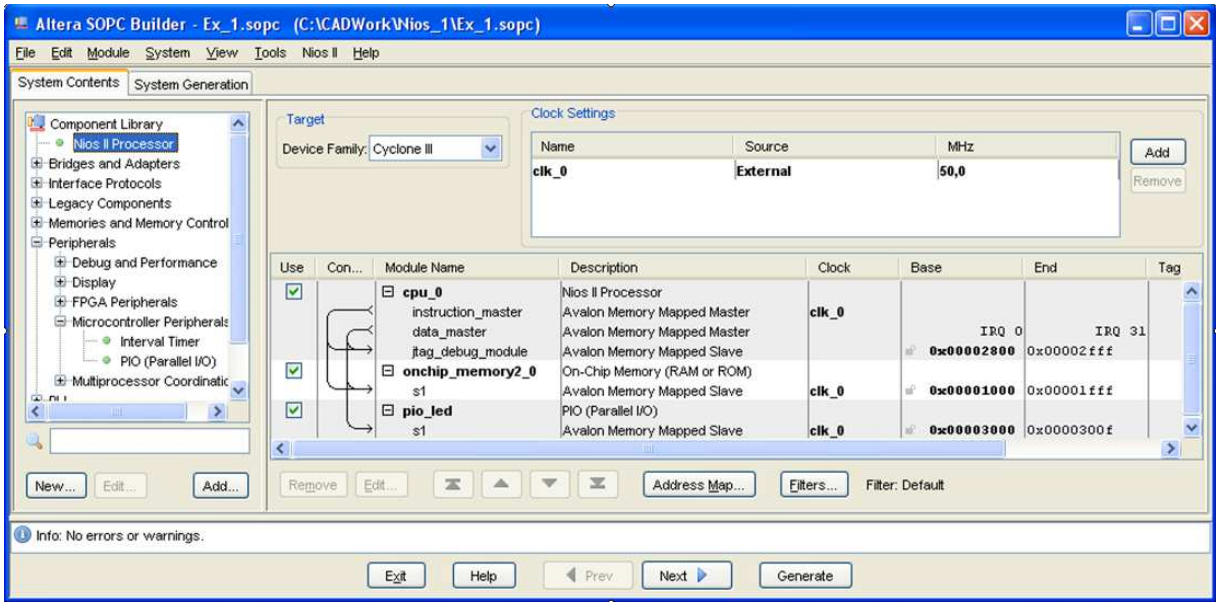


Рис. 2.3. Результат конфигурирования системы

3. Выполнить первую компиляцию, назначить внешние контакты схемы контактам ПЛИС, выполнить повторную компиляцию проекта в соответствии с рекомендациями пп. 5–7 этапа 1 лаб. раб. 1.

Этап 4. Разработка программного обеспечения

Программа должна управлять выводом сигнала на блок светодиодной индикации учебного стенда, создавая бегущий сигнал горящего индикатора справа налево, затем наоборот. Это реализуется выведением на внешние выводы СнК двоичного слова, содержащего одну единицу, и реализацией логического сдвига этого слова с задержкой порядка 1 с.

Алгоритм программы для встроенного процессора Nios II показан на рис. 2.4.

Разработка программ для Nios II осуществляется в среде *Nios IDE* с использованием диалекта языка Си++ с учетом некоторых особенностей структуры ядра Nios II.

1. Из программы SOPC Builder запустить интегрированную среду разработки программного обеспечения Nios II IDE командой меню *Nios II/Nios II IDE*. Через меню Nios II IDE создать новое приложение для *Nios II*, используя команду *File/New/Nios II C/C++ Application*.

Для создаваемого приложения необходимо указать систему на кристалле, для которой разрабатывается программа, и конкретный процессор (в том случае, если их несколько); в качестве шаблона выбрать базовый проект *Blank Project*.

Исходные файлы проекта создаются командой *File/New/Source File*. При этом файл будет добавлен в тот проект, который выделен в поле *Nios II C/C++ Projects*. Текст программы набирается в появившемся окне исходного файла.

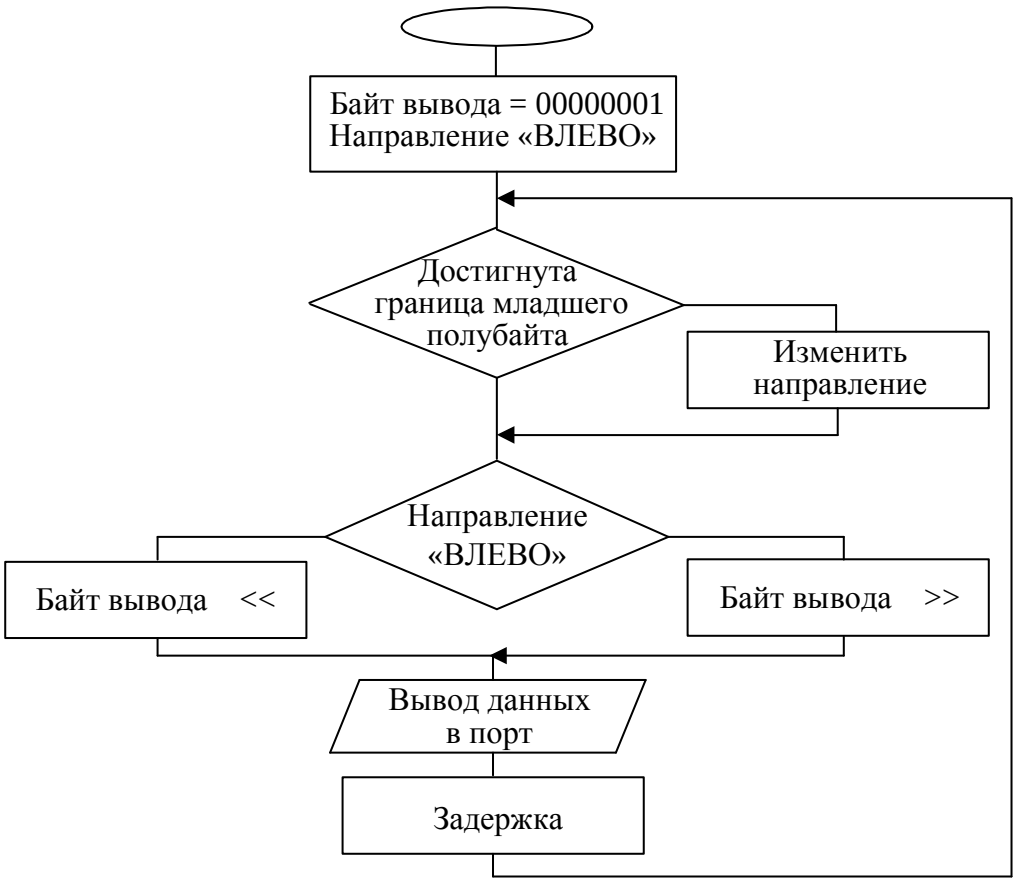


Рис. 2.4. Алгоритм работы программ для Nios II

Для работы приложения требуется подключение трех заголовочных файлов. Файл *system.h* содержит описание сконфигурированной аппаратной системы и ее периферии: аппаратную конфигурацию периферии, ее базовый адрес и символьное имя. Фрагмент описания представлен в листинге 2.1.

Листинг 2.1

```
#define PIO_LED_NAME "/dev/pio_led"
#define PIO_LED_TYPE "altera_avalon_pio"
#define PIO_LED_BASE 0x00003000
#define PIO_LED_DATA_WIDTH 8
#define PIO_LED_RESET_VALUE 0
```

Файл *altera_avalon_pio_regs.h* содержит макросы, обеспечивающие работу с портом ввода-вывода. В программе использован макрос записи в порт IOWR_ALTERA_AVALON_PIO_DATA (base, data), где base – адрес порта; data – выводимые данные. Адрес порта проекта 0x00003000 в файле *system.h* определен как PIO_LED_BASE, поэтому макрос записи данных в порт вызывается следующим образом:

IOWR_ALTERA_AVALON_PIO_DATA (PIO_LED_BASE, led)

Можно использовать и абсолютный адрес порта, для рассматриваемого примера макрос будет вызываться так:

IOWR_ALTERA_AVALON_PIO_DATA (0x00003000, led)

Файл *alt_types.h* содержит описание типов 8-, 16-, 32- и 64-разрядных знаковых и беззнаковых целых. В программе использован беззнаковый 8-разрядный тип alt_u8.

Программа, решающая задачу управления светодиодной линейкой, приведена в листинге 2.2 и может быть использована в качестве шаблона для реализации индивидуального задания. Особенности программы для NIOS II проявляются в том, что программа должна начинаться с вызова функции *alt_main()* и использовать набор макросов для обращения к сконфигурированным фрагментам, в остальном текст программы соответствует традиционным приемам программирования на языке Си.

Листинг 2.2

```
#include "system.h"
#include "altera_avalon_pio_regs.h"
#include "alt_types.h"

 int main (void) __attribute__ ((weak, alias ("alt_main")));
 int alt_main (void)
 {alt_u8 led = 0x2;    // Инициализация байта вывода
  alt_u8 dir = 0;      // Направление движения
  volatile int i;
  while (1)
    {if (led & 0x81)   // Проверка на достижение границы
                       // младшего полубайта
          { dir = (dir ^ 0x1); }
     if (dir)
          { led = led >> 1; }
     else
          { led = led << 1; }
   IOWR_ALTERA_AVALON_PIO_DATA(0x00003000, led);
                       // Вывод на блок индикации led
     i = 0;
     while (i<200000)  // Организация задержки
```

```
      i++;
    }
  return 0;
}
```

Дополнительно можно установить опции оптимизации исполняемого файла, выполнив команду *Project/Properties* и установив определенные опции в окне *System Library*.

2. Собрать проект, используя команду меню *Project/Build Project*.

Если ошибок в программе не обнаружено, то проект будет скомпилирован и готов к загрузке в СнК. Все файлы, необходимые для программного проекта, будут размещены в автоматически созданном каталоге *software* в рамках проекта Quartus II.

Этап 5. Отладка программного обеспечения в Nios II IDE

Отладка программы производится в два этапа: в режиме моделирования с *Instruction Set Simulator* и на реальной схеме с *Nios II Hardware Debugger*.

1. Запустить Instruction Set Simulator через контекстное меню *Debug as/Nios II Instruction Set Simulator* либо воспользовавшись пунктом меню *Run/Debug as/Nios II Instruction Set Simulator*. При этом откроется окно отладки Nios II IDE (рис. 2.5).

2. Выполнить отладку программы в пошаговом режиме с помощью команд *Step Into*, *Step Over*, а также с использованием точек останова. В процессе отладки рекомендуется контролировать значения переменных и содержимое регистров.

3. Провести отладку проекта на реальной схеме, для чего необходимо вернуться в рабочее окно Quartus II и выполнить компиляцию проекта. В результате дополнительной компиляции содержимое памяти программ с кодом программы будет интегрировано в СнК.

4. Подключить учебную плату к компьютеру через порт USB и включить ее. Запустить программатор, используя меню *Tools/Programmer*, и загрузить проект в ПЛИС в соответствии с последовательностью действий, приведенной в пп. 1–3 этапа 3 лаб. раб. 1.

5. После выполнения перечисленных настроек можно запускать отладку либо прямо из окна настроек Nios II IDE, нажав кнопку *Debug*, либо воспользоваться пунктами меню *Debug as/Nios II Hardware Debugger*.

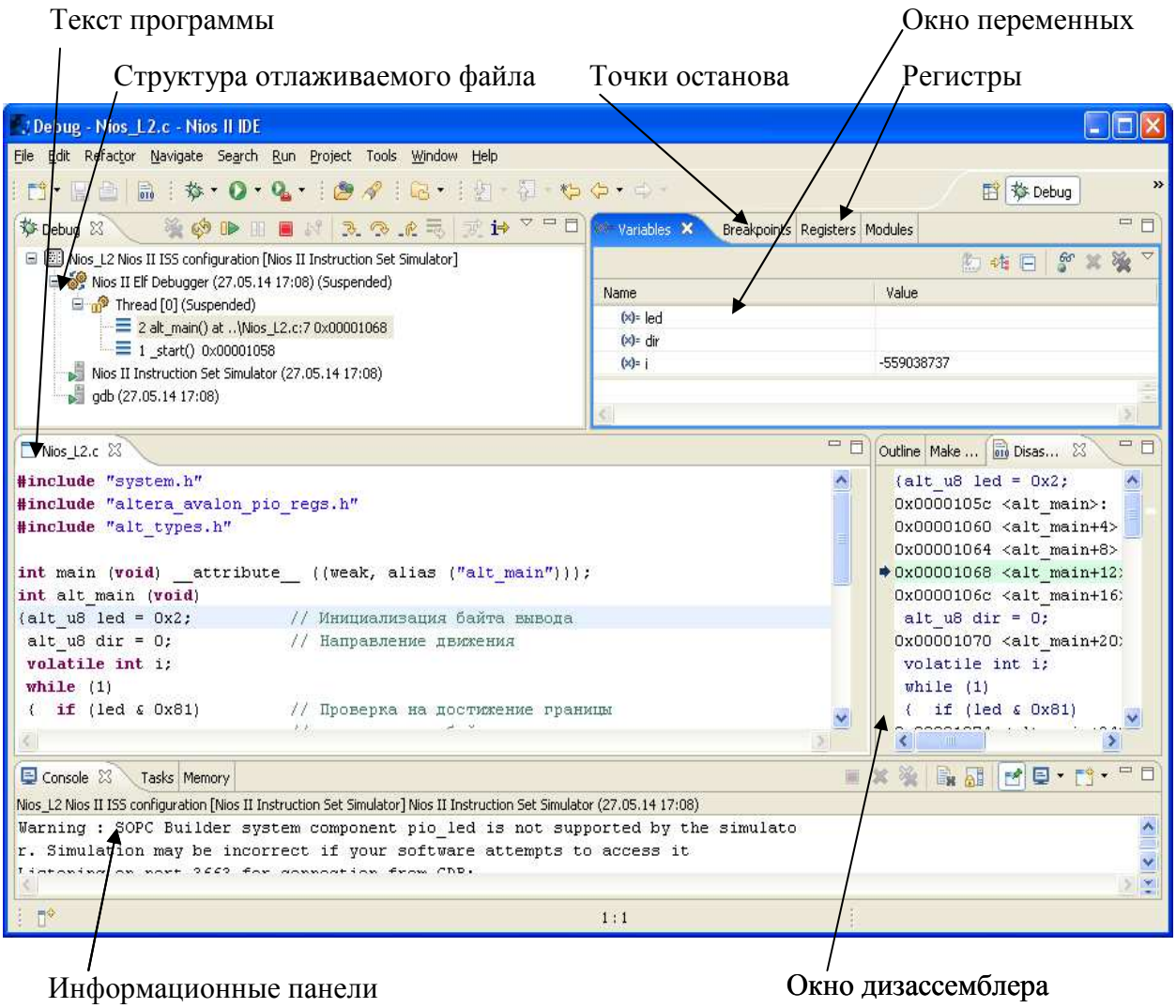


Рис. 2.5. Окно отладки Nios II IDE

Отладка выполняется аналогично отладке с ISS. В процессе отладки необходимо проследить за выполнением программы по светодиодам на учебном стенде.

Содержание отчета

Отчет должен содержать следующую информацию:

1. Структурную схему сгенерированного soft-процессора, его технические параметры (объем памяти программ и данных, частота ядра, количество прерываний и т. п.).

2. Блок-схему алгоритма оптимизированной программы для Nios II.

3. Текст оптимизированной программы с комментариями.

4. Выводы по проделанной работе и полученным навыкам построения смешанных программно-аппаратных систем.

Варианты заданий

1. Реализовать систему управления бегущей строкой, состоящей из четырех индикаторов. Данные для вывода задавать с помощью констант.

2. Реализовать последовательный вывод на индикатор чисел в шестнадцатеричной системе от 0 до F и далее по циклу.

3. Реализовать вывод на индикатор чисел в десятичной системе, числа для вывода на индикацию задавать с помощью переключателей.

4. Реализовать систему, управляющую перемещением светящегося элемента по десятиразрядной светодиодной линейке вкруговую.

5. Реализовать систему, управляющую перемещением светящегося сегмента по четырем семисегментным индикаторам.

Лабораторная работа 3.
СРАВНЕНИЕ ХАРАКТЕРИСТИК ПРОЕКТОВ НА ОСНОВЕ NIOS II В КРИСТАЛЛЕ CYCLONE II ПРИ ИХ АППАРАТНОЙ И ПРОГРАММНО-АППАРАТНОЙ РЕАЛИЗАЦИИ

Цель работы состоит в анализе особенностей проектирования и отладки и сравнении эффективности проектирования систем на базе soft-процессора Nios II и проектирования беспроцессорных аппаратных модулей при их реализации в условиях одной СнК.

Краткие теоретические сведения

Одно из основных достоинств СнК – гибкость решения различных задач. На одном и том же кристалле ПЛИС можно реализовать различные способы решения поставленной задачи, оценить результат и выбрать более подходящий в данном случае способ.

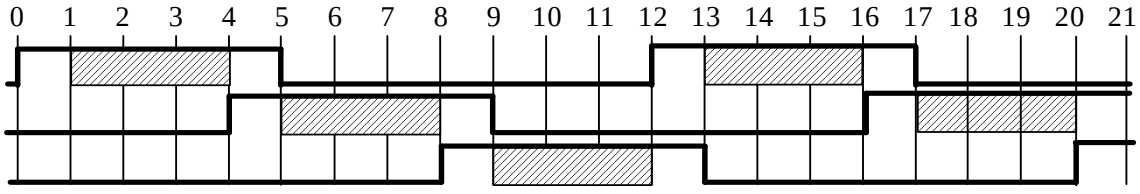


Рис. 3.1. Диаграмма сигналов трехфазного управления

В работе предлагается реализовать два независимых решения задачи формирования сигналов трехфазного управления, аппаратно выполненные на одном кристалле ПЛИС фирмы «Altera».

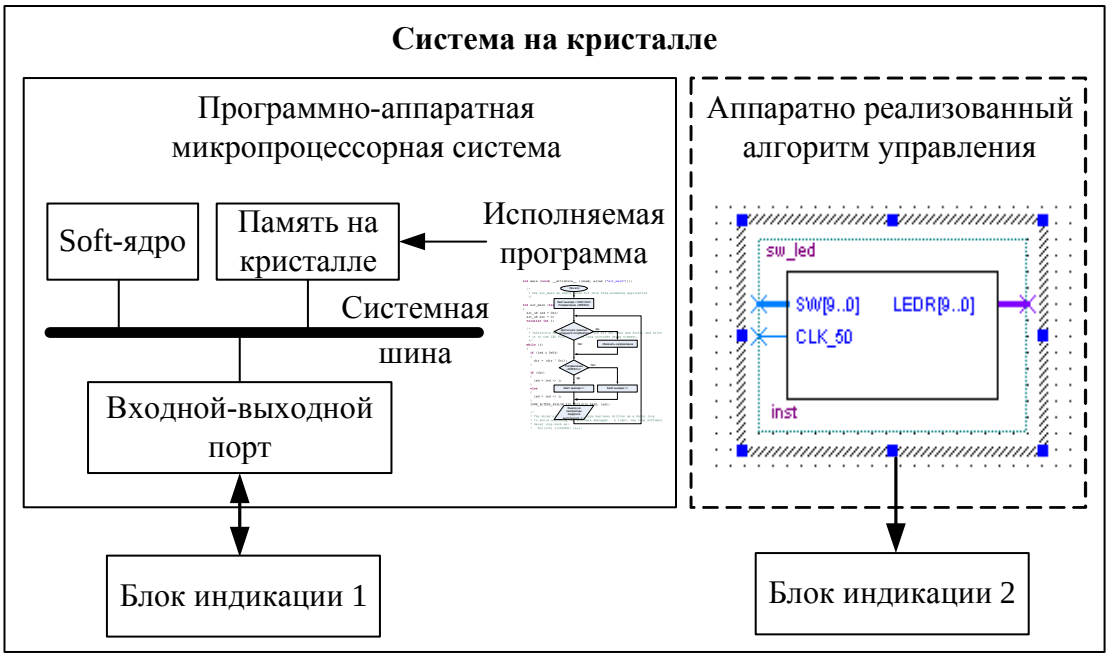


Рис. 3.2. Устройство трехфазного управления

Первая система предполагает чисто аппаратную реализацию алгоритма, а вторая должна быть выполнена как программно-аппаратная микропроцессорная система на основе процессорного ядра Nios II. Пример временной диаграммы трехфазного управления с перекрытием на один такт приведен на рис. 3.1, а структурная схема устройства – на рис. 3.2.

Задание на работу

Разработать СнК с двумя независимыми решениями задачи трехфазного управления: аппаратным и программно-аппаратным. Выполнить имплементацию проекта СнК в ПЛИС отладочной платы DE0 фирмы «Terasic». Визуальный контроль работоспособности проекта осуществить путем подключения светодиодных индикаторов.

Последовательность выполнения работы

Последовательность выполнения работы рассмотрим на примере создания системы, реализующей алгоритм трехфазного управления рис. 3.1.

Этап 1. Реализация программно-аппаратной системы

1. Выполнить конфигурирование процессорной части СнК, сгенерировать файлы описания soft-процессора NIOS II со встроенной в кристалл памятью программ и данных и блоком индикации. Выполнить компиляцию проекта, назначить контакты, выполнить загрузку процессорного ядра в ПЛИС.

2. Разработать и загрузить на выполнение программу трехфазного управления для Nios II, выполнить отладку программно-аппаратной системы, используя методы, описанные в лаб. раб. 2.

Текст программы, соответствующий временной диаграмме на рис 3.1, приведен в листинге 3.1. Работа программы построена на основе циклического формирования выходных сигналов. OUT0…OUT2 – выходные значения сигналов управления для каждой фазы, Counter – счетчик тактов для фаз, при этом период фаз равен 12 интервалам.

Листинг 3.1

```
#include "system.h"
#include "altera_avalon_pio_regs.h"
#include "alt_types.h"

int main (void) __attribute__ (weak, alias ("alt_main"));
int alt_main (void)
{
     alt_u8 led = 0;
     alt_u8 Counter = 0;
     alt_u8 OUT0 = 1;
     alt_u8 OUT1 = 0;
     alt_u8 OUT2 = 1;
 while (1)
 {
      switch(Counter)
      {
      case 1: OUT2=0;break;
      case 4: OUT1=1;break;
      case 5: OUT0=0;break;
      case 8: OUT2=1;break;
      case 9: OUT1=0;break;
      case 12: OUT0=1;Counter=0;break;
      }
      Counter++;
      led = OUT2*4 + OUT1*2 + OUT0;
      IOWR_ALTERA_AVALON_PIO_DATA(PIO_BASE, led);
      usleep(20000);
 }
 return 0;
}
```

3. Оценить временные и аппаратные затраты (без учета затрат на soft-процессор). Временные затраты оцениваются по результатам сеанса моделирования во временной области, а также с помощью осциллографа, подключенного к соответствующим выходам тестовой платы.

Этап 2. Аппаратная реализация алгоритма

Разработать на языке VHDL описание устройства, реализующего заданный алгоритм управления. Пример программы, реализующей временную диаграмму (рис 3.1), приведен в листинге 3.2.

Листинг 3.2

```
library ieee;
use ieee.std_logic_1164.all;
entity lab3_hw is
port (Led: out Std_logic _vector(2 downto 0);
      Clk_50: in Std_logic);
end lab3_hw;
architecture Behave of lab3_hw is       -- Поведение проекта
shared variable Counter  : integer := 0;
signal out0              : Std_logic:= '1';
signal out1              : Std_logic:= '0';
signal out2              : Std_logic:= '1';
begin
     Ct1: process (Clk_50)
     begin
     if ((Clk_50 = '1') and (Clk_50'event)) then
          Counter := Counter + 1;
          case (Counter) is
               when 10000000 => out2<='0';
               when 40000000 => out1<='1';
               when 50000000 => out0<='0';
               when 80000000 => out2<='1';
               when 90000000 => out1<='0';
               when 120000000 => out0<='1'; Counter :=0;
               when others => null;
          end case;
     end if;
     end process CT1;
     Leds: process (out0, out1, out2)  -- Вывод на светодиоды
     begin
          Led(0) <= out0; Led(1) <= out1; Led(2) <= out2;
     end process Leds;
end Behave;
```

Выполнить отладку полученной системы, используя методы, описанные в лаб. раб. 1. Проверить работу системы в реальных условиях, оценить затраты на реализацию проекта.

Содержание отчета

Отчет должен быть выполнен по требованиям ЕСПД и содержать следующую информацию:

1. Структурную схему сгенерированного soft-процессора, его технические параметры.
2. Блок-схему алгоритма оптимизированной программы для Nios II, текст оптимизированной программы с комментариями.
3. Схему аппаратного устройства для решения поставленной задачи, VHDL-код устройства с комментариями.
4. Результаты моделирования работы устройства.
5. Результаты расчетов быстродействия и аппаратных затрат для обоих способов решения задачи управления трехфазным двигателем.
6. Выводы по проделанной работе и полученным результатам сравнения аппаратных и смешанных программно-аппаратных систем.

Варианты заданий

1. Сигналы не имеют фазового сдвига, следуют друг за другом без пауз.
2. Сигналы имеют один такт паузы между собой.
3. Сигналы имеют два такта паузы между собой.
4. Сигналы перекрываются на один такт.
5. Сигналы имеют паузу в один такт на переднем фронте.

Лабораторная работа 4.
ПРЕРЫВАНИЯ В ПРОГРАММНО-АППАРАТНЫХ СИСТЕМАХ НА ОСНОВЕ КРИСТАЛЛА CYCLONE III

Цель работы состоит в изучении процесса проектирования систем с использованием аппаратных прерываний soft-процессора Nios II.

Краткие теоретические сведения

Процессор Nios II поддерживает работу с прерываниями двумя способами.

Первый способ основан на использовании внутреннего контроллера прерываний Nios II. Этот контроллер поддерживает 32 аппаратных прерывания, ядро процессора имеет 32 входа запроса прерываний (IRQ0 – IRQ31), для каждого источника прерываний используется уникальный вход. Приоритет прерываниям назначается программно, поддерживаются вложенные прерывания.

Настройка прерываний осуществляется установкой определенных битов в управляющих регистрах. Регистры, требующие настройки при организации прерываний, приведены в табл. 4.1.

Схема формирования запроса на прерывание soft-процессора на основании поступивших запросов и состояний регистров приведена на рис. 4.1.

Таблица 4.1

Регистры управления прерыванием

Регистр	Имя	b31 …b2	b1	b0
ctl 0	*status*	резерв	U	PIE
ctl 1	*estatus*	резерв	EU	EPIE
ctl 3	*ienable*	Биты разрешения прерывания		
ctl 4	*ipending*	Биты возникших прерываний		

Глобальные разрешение-запрет внешних прерываний осуществляется с помощью бита *PIE* в регистре *status*. Регистр *ctl1* хранит копию регистра состояния во время обработки прерываний. Запрет-разрешение отдельных прерываний процессорной системы осуществляется программно через контрольный регистр *ienable*, который содержит бит разрешения прерываний для каждого входа *IRQ*. Регистр содержит информацию о прерываниях, требующих обработки. Обработчик прерываний определяет, какие из них имеют наибольший приоритет, и передает управление соответствующей подпрограмме.

Для генерации прерываний необходимо выполнение следующих условий:

– бит PIE регистра *status* установлен в единичное состояние;

– один из входов запросов прерывания (IRQ_k) процессора активирован;

– соответствующий бит регистра разрешения прерываний *ienable* содержит единичное значение.

Второй способ реализации прерываний предполагает использование внешнего векторного контроллера прерываний (VIC, Vectored Interrupt Controller), например представленного в библиотеке периферийных модулей среды создания реконфигурируемых систем на кристалле SOPC Bulder.

Векторный контроллер позволяет существенно сократить время отклика процессора на прерывания, что актуально для систем реального времени. Один внешний контроллер позволяет обслуживать до 32 запросов прерываний, причем имеется возможность их каскадирования, что позволяет увеличить количество запросов. Контроллер подключается к процессору через интерфейс внешнего контроллера прерываний (EIC, External Interrupt Controller Interface). Если в систему включен внешний векторный контроллер прерыва-

ний, внутренний контроллер прерываний не реализуется, а SOPC Builder подключает прерывания к внешнему.

При поступлении нескольких запросов на прерывания внешний векторный контроллер на основании принятой системы приоритетов выбирает то, которое будет обрабатываться и передает ядру Nios II адрес соответствующего обработчика. Некоторые внешние прерывания могут быть сконфигурированы как немаскируемые (NMI, Nonmaskable Interrupt) – они не блокируются битом PIE регистра *status* и не имеют уровня.

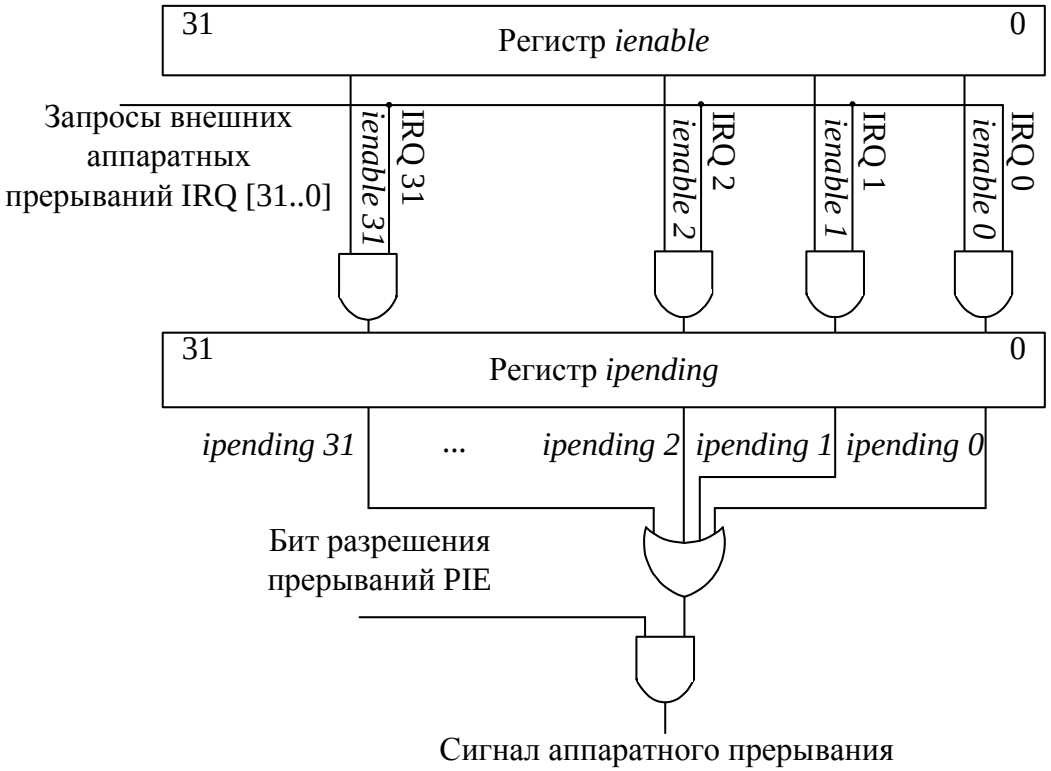


Рис. 4.1. Схема генерации прерываний soft-процессора NIOS II

При работе с прерываниями программное обеспечение состоит из двух частей: основной программы и функций, обеспечивающих работу в режиме прерываний. Функции должны реализовывать инициализацию прерывания, запреты прерываний, а также основные действия по обработке запросов IRQ.

При обработке прерывания необходимо учитывать асинхронность запроса. Ресурсы, используемые основной программой, не должны искажаться прерывающей функцией. Кроме создания собственно программ проектировщик должен задать места расположения фрагментов. При создании системы часто необходима определенная настройка приоритетов системы прерываний.

Задание на работу

Создать и отладить аппаратно-программную систему, которая выполняет две основные функции: выводит данные на сегментный индикатор и по прерыванию выводит информацию на индикаторы LED.

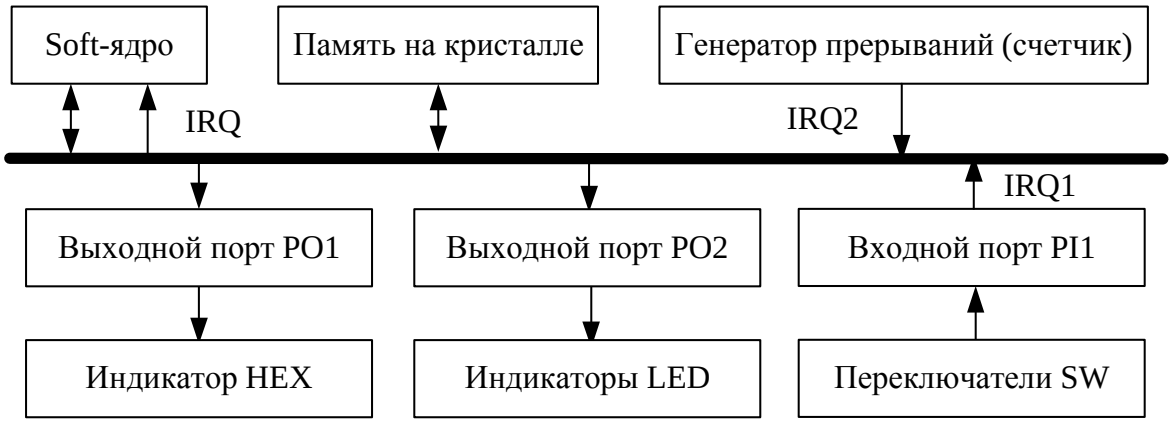


Рис. 4.2. Схема проектируемого устройства

Обобщенная функциональная схема основных аппаратных блоков микропроцессорной системы с прерываниями при реализации задач поставленных в данной работе поставленной задачи приведена на рис. 4.2, которая в зависимости от конкретного задания непрерывно выводит данные с переключателей на заданный тип индикатора, а по прерыванию выводит ту же или измененную информацию на заданный тип индикатора.

Последовательность выполнения работы

Последовательность выполнения работы рассмотрим на примере создания системы, осуществляющий вывод на светодиодную линейку двоичного кода числа, по прерыванию от внешнего переключателя выполняется инкремент этого числа.

Этап 1. Разработка системы

Разработать СнК, используя последовательность действий, приведенную на этапах 1 и 2 лаб. раб. 2.

Настройка прерывания по срабатыванию переключателя требует дополнительных настроек портов PIO. Для решения поставленной задачи должны быть предусмотрены порт для вывода информации на светодиодную линейку, а также порт для подключения кнопок генерации прерываний. Порт вывода информации настраивается только как *Output port only* – установкой соответствующей опции. Порт, к которому подключается кнопка, настраивает-

ся как *Input port only* и для него на закладке *Input Options* устанавливаются дополнительные опции.

Чтобы установить прерывание от линии порта, устанавливается опция *Generate IRQ*. Срабатывание прерываний возможно как по фронту, так и по уровню. В случае срабатывания по фронту, дополнительно устанавливаются *Synchronously capture* и указывается нарастание (*Rising*), спад (*Falling*) или по любому событию (*Either*).

После завершения конфигурирования системы дополнительно устанавливаются приоритеты прерываний. Для этого активизируется команда меню *System/Auto-Assign IRQs*.

Пример проекта с настроенными прерываниями приведен на рис. 4.3. В системе предусмотрено прерывание от одноразрядного порта pio_key, прерыванию присвоен нулевой номер.

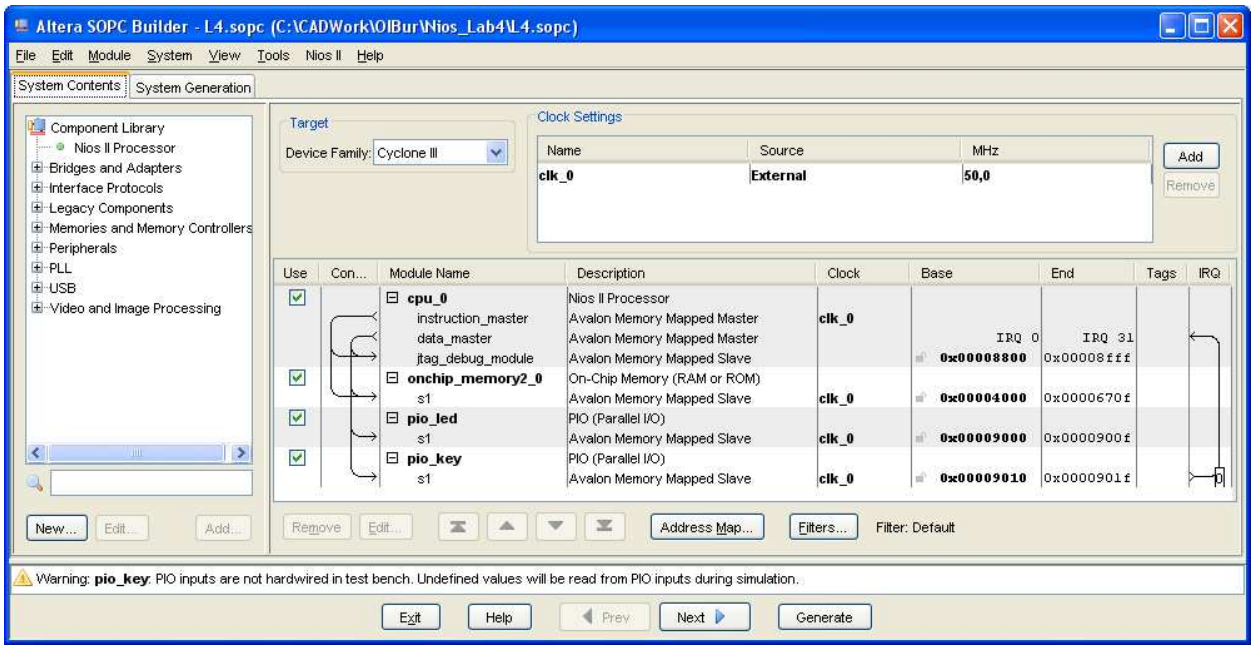


Рис. 4.3. Проект системы с использованием прерываний

По завершению настроек выполняется генерация системы, а также конфигурирование системы на кристалле ПЛИС в соответствии с последовательностью действий этапа 3 лаб. раб. 2.

Этап 2. Разработка программного обеспечения

Основная программа осуществляет вывод некоторого значения на линейку светодиодной индикации учебного стенда. В процедуре, вызываемой по прерыванию, будет выполняться инкремент этого значения.

Разрабатывать программу следует в соответствии с последовательностью действий этапа 4 лаб. раб. 2.

Написание программного обеспечения проекта, связанного с обращением к оборудованию, возможно как с помощью непосредственного обращения к соответствующим аппаратным средствам, так и с использованием уровня аппаратных абстракций.

Уровень аппаратных абстракций (HAL, Hardware Abstraction Layer) представляет собой библиотеку функций, предназначенных для работы с аппаратными средствами. Взаимодействие элементов системы, построенной с использованием уровня аппаратных абстракций, приведено на рис. 4.4.

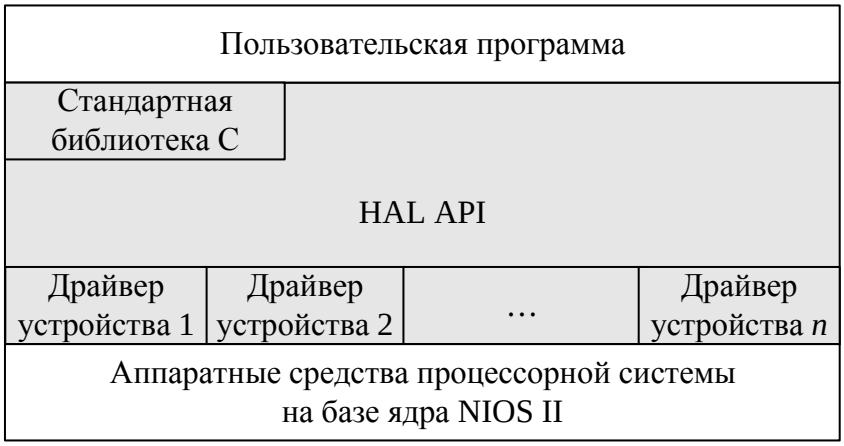


Рис. 4.4. Схема взаимодействия программных и аппаратных элементов системы

Сервис HAL для процессорного ядра Nios II предоставляет следующие возможности:

– интеграция со стандартной библиотекой *newlib* ANSI C с предоставлением возможности использования функций стандартной библиотеки;

– использование драйверов, обеспечивающих доступ к устройствам системы;

– предоставление интерфейса для написания прикладных программ (API) – интерфейса с HAL сервисами, такими как доступ к устройствам, обработка прерываний и сигнальные средства;

– инициализация системы для управления работой программы перед блоком main();

– инициализация устройств в системе перед запуском main().

Основные функции, предусмотренные для обеспечения работы системы прерываний Nios II.

Функция **int alt_ic_isr_register** (alt_u32 ic_id, alt_u32 irq, alt_isr_func isr, void *isr_context, void* flags) ставит в соответствие источнику прерывания функцию-обработчик и при этом имеет следующие параметры:

ic_id – идентификатор контроллера прерываний, определенный в файле *system.h*;

irq – идентификатор устройства, инициирующего прерывания, также определенный в *system.h*; при использовании внутреннего контроллера прерываний это значение соответствует приоритету, причем приоритетность устанавливается в следующем порядке IRQ_0 имеет высший приоритет, а IRQ_{31} – низший;

isr – функция-обработчик, вызываемая при срабатывании прерывания;

isr_context – указатель на структуру, используемую в качестве аргумента функции-обработчика;

flags – зарезервированный параметр.

В случае успешного завершения функция возвращает нулевое значение.

Функция **int alt_irq_enable** (alt_u32 id) разрешает указанное прерывание.

Функция **int alt_irq_disable** (alt_u32 id) запрещает указанное прерывание.

Функция **alt_irq_disable_all**() запрещает все маскируемые прерывания, немаскируемые прерывания функция не затрагивает.

Функция **void alt_irq_enable_all** (alt_irq_context context) разрешает все прерывания, запрещенные ранее. Входной аргумент – значение, возвращенной ранее при вызове запрещающей процедуры.

Текст программы приведен в листинге 4.1. Основная программа реализует вывод данных на светодиодную линейку, по нажатию кнопки запускается процедура прерывания, меняющая направление вывода.

В тексте программы можно выделить две процедуры, связанные с работой процедуры прерывания.

Во-первых, это процедура настройки порта void init_pio(). В этой процедуре настраиваются прерывания от линий порта: маскируются входы, события на которых инициируют прерывания, сбрасываются флаги прерываний. Соответствующие макросы

IOWR_ALTERA_AVALON_PIO_IRQ_ MASK(base, data),

IOWR_ALTERA_AVALON_PIO_EDGE_CAP(base, data)

находятся в файле *altera_avalon_pio_regs.h* .

Листинг 4.1

```
#include "system.h"
#include "altera_avalon_pio_regs.h"
#include "sys/alt_irq.h"
volatile int edge_capture;
void init_pio();
int main()
{   init_pio();
    nNumber = 0;   edge_capture = 0;
    while (1)
     {   if ( edge_capture != 0 )
          nNumber = edge_capture;
         IOWR_ALTERA_AVALON_PIO_DATA(0x00009000, nNumber);
         i = 0;
         while (i<200000)        // Организация задержки
          i++;
    }
 return 0;
 }
void handle_button_interrupts(void* context, alt_u32 id)
 { volatile int* edge_capture_ptr = (volatile int*) context;
    *edge_capture_ptr += 1;
    IOWR_ALTERA_AVALON_PIO_EDGE_CAP(0x00009010, 0);
    // Сброс флага прерывания
 }
 // Инициализация порта, настроенного на прерывания
 void init_pio()
 {   void* edge_capture_ptr = (void*) &edge_capture;
    // Разрешение прерывания
    IOWR_ALTERA_AVALON_PIO_IRQ_MASK(0x00009010, 0xf);
    // Сброс флага прерывания
    IOWR_ALTERA_AVALON_PIO_EDGE_CAP(0x00009010, 0x0);
    // Настройка прерывания для вызова процедуры обработчика
    #ifdef ALT_ENHANCED_INTERRUPT_API_PRESENT
    alt_ic_isr_register
              (BUTTON_PIO_IRQ_INTERRUPT_CONTROLLER_ID,
              PIO_KEY_IRQ, handle_button_interrupts,
              edge_capture_ptr, 0x0);
    #else     alt_irq_register  ( PIO_KEY_IRQ,
              edge_capture_ptr, handle_button_interrupts );
    #endif
 }
```

Этап 3. Проверка работы системы на отладочной плате

Выполнить имплементацию СнК в ПЛИС отладочной платы DE0 фирмы «Terasic» в соответствии с последовательностью действий, приведенной в лаб. раб. 1. Проверить работу системы в реальных условиях, оценить затраты на реализацию проекта.

Содержание отчета

Отчет должен содержать следующую информацию:

1. Описание СнК с описанием особенностей настроек ее компонентов.

2. Описание иерархической структуры программного проекта.

2. Тексты основной программы и функции, вызываемой по прерыванию, с комментариями.

3. Оценки результатов проектирования.

Варианты заданий

Варианты заданий приведены в табл. 4.2.

Таблица 4.2

Номер задания	Основная программа	Процедура прерывания	Прерывание
1	Выводит на индикацию номер включенного движкового переключателя SW0–SW5	Выводит бегущую единицу (с частотой порядка 1 с) на индикаторы светодиодной линейки	По переднему фронту
2	Выводит на индикацию цифры по возрастанию от 0 до 9	Меняет порядок смены цифр с возрастания на убывание, с убывания на возрастание	По обоим фронтам
3	Выводит на индикацию четырехразрядное число	Изменяет режим вывода: постоянное горение/моргание с частотой порядка 1 с	По переднему фронту
4	Перемещает светящийся элемент по светодиодной линейке	Изменяет направление перемещения на противоположное	По обоим фронтам
5	Выводит данные на четыре индикатора в режиме бегущей строки	Изменяет направление перемещения на противоположное	По заднему фронту

Для вывода буквенно-цифровой информации следует использовать семисегментные индикаторы, установленные на отладочной плате. При использовании светодиодной линейки задействовать все 10 ее элементов. Сигнал прерывания формировать от внешнего переключателя.

Лабораторная работа 5.
РЕЖИМ ПДП В ПРОГРАММНО-АППАРАТНЫХ СИСТЕМАХ НА ОСНОВЕ КРИСТАЛЛА CYCLONE III

Цель работы состоит в освоении процесса проектирования систем при использовании механизма прямого доступа к памяти (ПДП), (DMA, Direct Memory Access). Механизм ПДП применен для создания системы с проверкой правильности работы дублированием решения задачи при различных реализациях (аппаратной и программно-аппаратной) оборудования в условиях одной и той же системы на кристалле.

Краткие теоретические сведения

В процессе проектирования сложных систем возникают ситуации, при которых ввод-вывод в режиме ожидания или с использованием прерываний неэффективен, так как требует процессорного времени для организации пересылок между памятью и устройствами ввода-вывода с помощью циклического использования команд пересылки. Для перемещения данных может применяться более эффективный метод с прямым доступом к памяти. Метод реализуется с помощью специального контроллера. DMA-контроллер для каждого единичной пересылки данных может получать доступ к системной шине независимо от центрального процессора. Когда возникает необходимость пересылки блока данных, процессор посылает запрос контроллеру, который и инициализирует обмен данными. После окончания передачи заданного блока данных контроллер посылает процессору сигнал прерывания.

Каждая элементарная операция обмена выполняется занятием шины Avalon на два цикла – чтения и записи. Для этого же в RISC-процессоре пришлось бы выполнить три команды (каждая должна содержать два цикла занятия шины), и это не считая обрамления такой связки команд.

Контроллер содержит набор регистров, доступных центральному процессору для чтения и записи. Регистры контроллера задают порт (который должен быть использован), направление переноса данных (чтение-запись), единицу переноса (побайтно-пословно), число байтов в блоке, который следует перенести.

Для реализации ПДП в СнК на базе NIOS II разработчику предлагается специальное ядро контроллера DMA, ориентированное на работу на шине Avalon®. Структура ядра приведена на рис. 5.1.

Проекты с контроллером DMA создаются с использованием HAL-сервисов. В модели DMA уровня аппаратных абстракций предусмотрено два типа транзакций: прием и передача, и разработчику предоставлены два драйвера для реализации соответствующих каналов. Канал передачи передает данные из буфера передатчика в приемное устройство, а канал приема принимает данные от устройства и сохраняет их в буфере приемника.

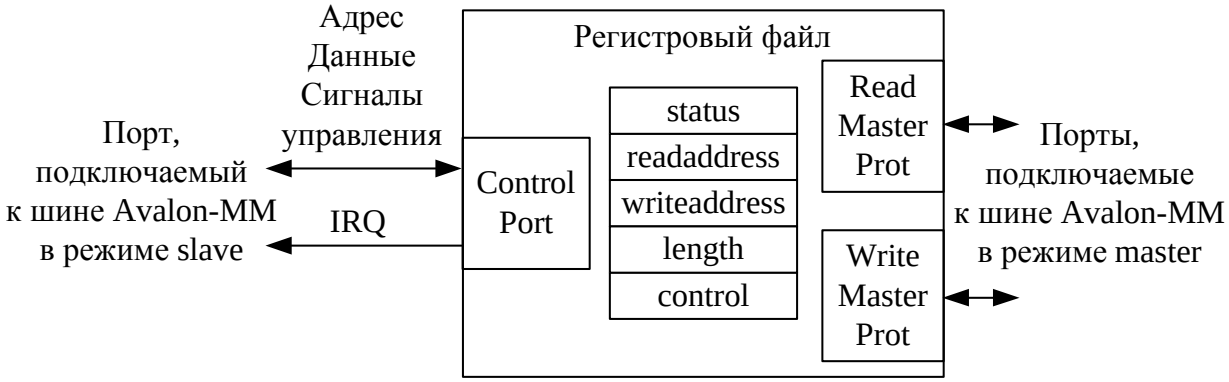


Рис. 5.1. Структура ядра DMA

Возможны три типа DMA-транзакций: из периферийного устройства в память, из памяти в периферийное устройство, из памяти в память. В первых двух случаях задействуют только каналы приемника или передатчика соответственно, копирование данных из памяти в память требует использования обоих DMA-каналов.

Основные принципы архитектуры Avalon:

– интерфейс к внешним и внутренним устройствам синхронизирован с синхрочастотой шины Avalon;

– все сигналы активны только по уровню, а не по фронту; мультиплексоры внутри шины Avalon определяют, какой из сигналов управляет переключением данных и от какого внешнего устройства;

– сигналы адреса, данных и управления внешних устройств подключаются к специальным блокам арбитража шины (индивидуальным для каждого устройства).

В системе Avalon есть также ряд особенностей и соглашений, которые дают возможность упростить и автоматизировать генерацию системы.

Внешнее устройство на шине Avalon понимается как логическое устройство, которое может быть расположено как внутри кристалла, так и вне его, устройство выполняет некоторую задачу системного уровня, связываясь с другими системными компонентами через шину.

Внешними устройствами Avalon могут быть память, процессоры, а также традиционные периферийные компоненты (асинхронные приемо-передатчики, порты ввода-вывода, таймеры или мосты шины и т. п.). Внешнее устройство соединяется с определенными портами на модуле шины Avalon, выделенными для этого внешнего устройства.

Созданные разработчиком модули, подключаемые к шине Avalon, должны соответствовать требованиям спецификации шины. Существует зарезервированный набор сигналов и правила для этих сигналов с помощью которых осуществляется взаимодействие с шиной. Большинство этих наборов настраиваемые, что обеспечивает гибкость реализации. Настройка правил выполняется через конфигурационные регистры процессора NiosII, а подключение этих правил осуществляется подключением библиотек в исходном коде для soft-процессора.

Листинг 5.1

```
entity module_name is
port (clk, reset    : in std_logic;
      chipselect    : in std_logic;
      read          : in std_logic;
      adress        : in std_logic_vector (31 downto 0);
      readdata      : out std_logic_vector (31 downto 0);
      write         : in std_logic ;
      writedata     : in std_logic_vector(31 downto 0) );
end module_name;
architecture struct of module_name is
signal
data_buffer: std_logic_vector(31 downto 0):=(others => '0');
begin
process(clk, reset, data_buffer,adress)
   begin
     if reset = '1' then readdata <= (others => '0');
                         data_buffer <= (others => '0');
     elsif rising_edge(clk) then
          if (chipselect = '1' and write = '1') then
               data_buffer <= writedata;
          elsif chipselect = '1' and write = '1' then
               if adress= X"2" then readdata <= data_buffer;
               else    readdata <= (others => '0');
               end if;
          end if;
     end if;
   end process;
--
--   Описание разрабатываемого модуля
--
end struct;
```

В листинге 5.1 приведен пример описания простого интерфейса модуля шины для аппаратного блока, представленного на языке VHDL.

Имена сигналов, перечисленные в разделе Port, зарезервированы. Для данного интерфейса использованы правила, устанавливаемые по умолчанию:

– одна тактовая частота (название должно совпадать с название на шине, по умолчанию clk0);

– один сигнал сброса reset (имя зарезервировано);

– два отдельных независимых канала readdata и writedata (имена зарезервированы);

– размерность каналов readdata и writedata – 32 разряда.

Задание на работу

Разработать два разных решения задачи трехфазного управления в соответствии с рис. 5.2. Для первого следует решения использовать аппаратную реализацию алгоритма, для второго программную на основе встроенного софт-процессора Nios II.

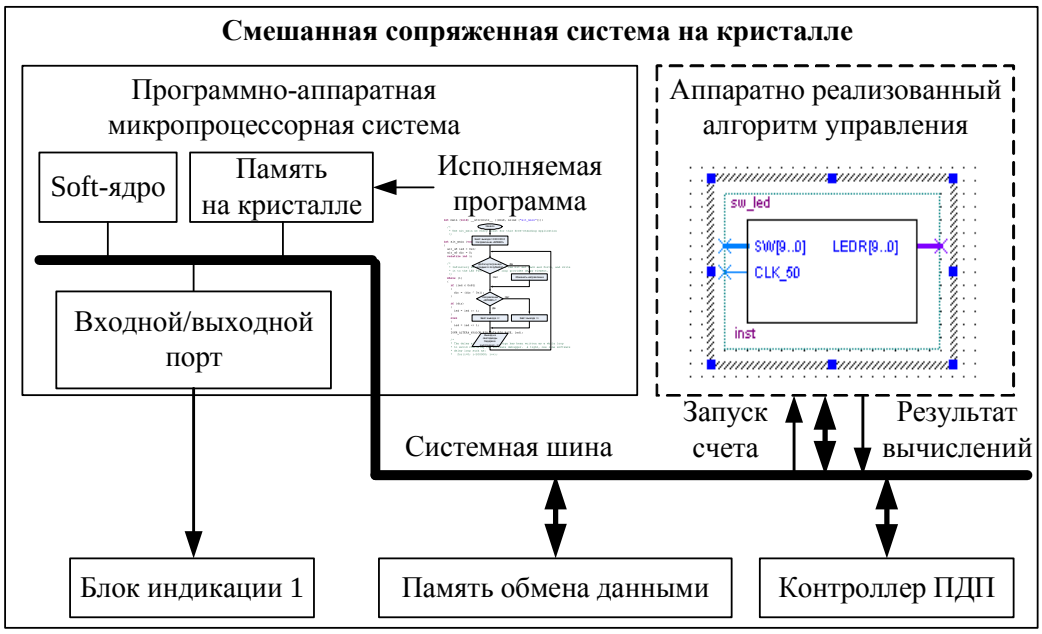


Рис. 5.2. Смешанная сопряженная система на кристалле

Аппаратная реализация представляет собой аппаратное решение алгоритма трехфазного управления, созданное в лаб. раб. 3. В процессе работы аппаратное решение осуществляет быстрое вычисление значения выходного регистра на текущем такте (в данном примере это – состояние выходов трех фаз управления) и кроме передачи для внешнего управления отправляет результат в программную часть системы, основанную на soft-процессоре Nios II.

Программно-аппаратная микропроцессорная система представляет собой систему на основе soft-процессора, созданную в лаб. раб. 2. Задачей программной части системы является формирование контрольной последовательности выходных сигналов и сравнение этих данных с данными, формируемыми аппаратной частью. Программа должна проверять правильность исполнения задачи аппаратной частью и выдавать внешний сигнал об отсутствии или наличии разногласий.

Существенное различие скоростей работы отдельных частей системы заставляет использовать программную часть для выборочного контроля формируемой последовательности. Получение данных в МП, отражающих последовательность работы аппаратной части, осуществляется на основе механизма ПДП (прямой доступ к памяти). Контролер ПДП накапливает получаемые данные в памяти, подчиненной процессору. Накопив заданную порцию данных, процессор сравнивает полученные данные с вычисленными им необходимыми значениями сигналов.

Программное обеспечение (исполняемая программа и программа прерываний) из предыдущих работ должно быть доработано и изменено в соответствии с указанным заданием.

В рамках данной работы будут рассмотрены особенности организации процессора Nios II с модулем DMA, также будет предложен вариант реализации аппаратного модуля VHDL с учетом того, что этот модуль (как и остальные модули ввода-вывода системы) в дальнейшем будет подключен к процессору Nios II посредством шины Avalon-MM.

Последовательность выполнения работы

Этап 1. Разработка системы

1. Разработать СнК, используя последовательность действий, приведенную на этапах 1 и 2 лаб. раб. 2. При этом в состав СнК должны входить ядро процессора, память, порт ввода-вывода, модуль прямого доступа к памяти.

Замечание: если задание предполагает работу с определенными областями памяти, то следует аккуратно использовать адресное пространство. Необходимо заранее учитывать корректное расположение данных в нем, так как при работе с модулем ПДП может происходить его переполнение или указание заведомо неверного адреса, при обращении к которому произойдет сбой. Отследить такие ошибки довольно трудно.

Настройка модуля DMA выполняется следующим образом. В проект СнК добавляется DMA Controller, который находится в разделе *Memories and Memories Controller/DMA Controller*. В окне его конфигурации следует определить размер рабочей посылки *Transfer size*, а также параметры пакетной транзакции *Burst transaction*, необходимой при работе в режиме передач «память–память», например при передачах SDRAM ↔ DDR3.

Порты модуля DMA: подключение портов Read_master, Write_master зависит от задачи выполняемой модуля DMA: если стоит задача считать данные из внешнего модуля в память, то соединить порты можно, как показано на рис. 5.3.

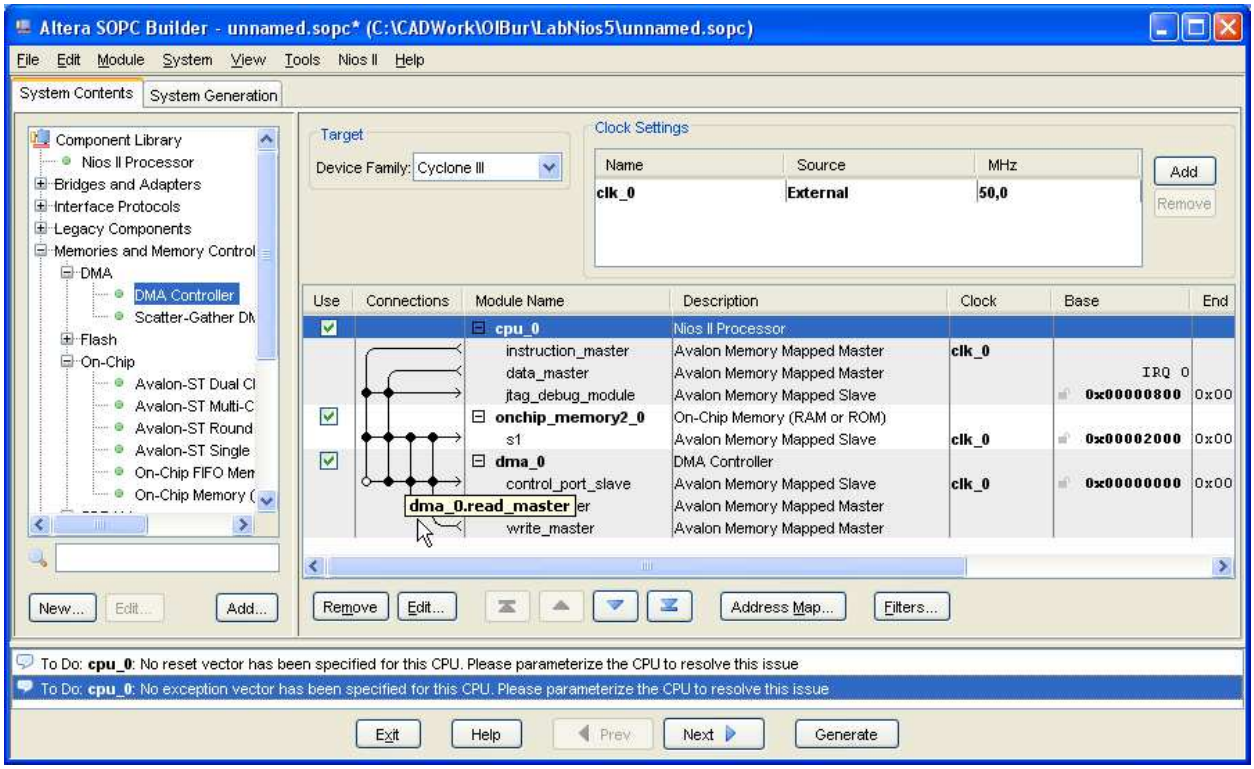


Рис. 5.3. Смешанная сопряженная система на кристалле

2. Ввести в состав системы аппаратный модуль, который при поступлении сигнала «запуск счета» выполняет формирование выходных сигналов в соответствии с временной диаграммой трехфазного управления и выдает сигнал готовности данных.

VHDL код модуля из лаб. раб. 3 необходимо доработать в части вывода полученных данных. Вывод данных на индикацию следует заменить на их сохранение в соответствующем регистре и приостановить работу до получения сигнала квитирования. Так путем сохранения данных до их прочтения будет выполнена синхронизация с программной частью системы.

3. Настроить интерфейс взаимодействия разработанного модуля и сконфигурированной СнК. Эта задача решается с использованием интерфейса SOPC Builder активизацией кнопки *New* на закладке *System Contents*. В результате будет запущен *Component Editor*. На вкладке *HDL Files* необходимо указать файл с описанием нового аппаратного модуля, при необходимости можно откорректировать настройки портов ввода-вывода модуля на вкладке *Signals*, проконтролировать параметры интерфейса на вкладке *Interfaces*. Конфигурирование завершается нажатием кнопки *Finish*, после чего новое устройство будет добавлено в список компонентов.

4. Выполнить генерацию системы, а также конфигурирование системы на кристалле ПЛИС в соответствии с последовательностью действий этапа 3 лаб. раб. 2. В процессе генерации интерфейс обмена информацией между частями системы сгенерируется автоматически.

Этап 2. Разработка программного обеспечения

Алгоритм программы для встроенного процессора Nios II показан на рис. 5.4. В алгоритме использована переменная *St* – счетчик тактов.

Разработку программы следует производить в соответствии с последовательностью действий этапа 4 лаб. раб. 2. При разработке подпрограммы формирования значений выходного сигнала на текущем такте воспользоваться результатами лаб. раб. 3.

Кроме уже использованных ранее системных библиотек *system.h*, *stdio.h*, *altera_avalon_pio_regs.h* требуется подключить библиотеки, описывающие типы переменных, и зарезервированные наборы сигналов, функций (описание регистров процессора NIOS II) *alt_types.h*, *stdlib.h*, *stddef.h*.

Дополнительно для модуля DMA требуется подключить библиотеку, описывающую зарезервированный набор сигналов и функций *sys/alt_dma.h*.

Для данных, поступающих от модуля DMA, необходимо предусмотреть буфер, и флаг проверки окончания работы модуля DMA. Этот флаг будет устанавливаться функцией dma_complete после завершения приема данных. Соответствующие фрагменты приведены в листинге 5.2.

Замечание: при работе в циклическом режиме после начала приема данных может возникнуть ситуация, при которой флаг проверки окончания работы модуля DMA никогда не установится. Чтобы избежать такой ситуации, необходимо реализовать сторожевой таймер, который

будет сбрасывать систему. Для этой цели можно разработать отдельный модуль на VHDL, а можно воспользоваться IP-ядром Timer, входящим в состав SOPS-builder.

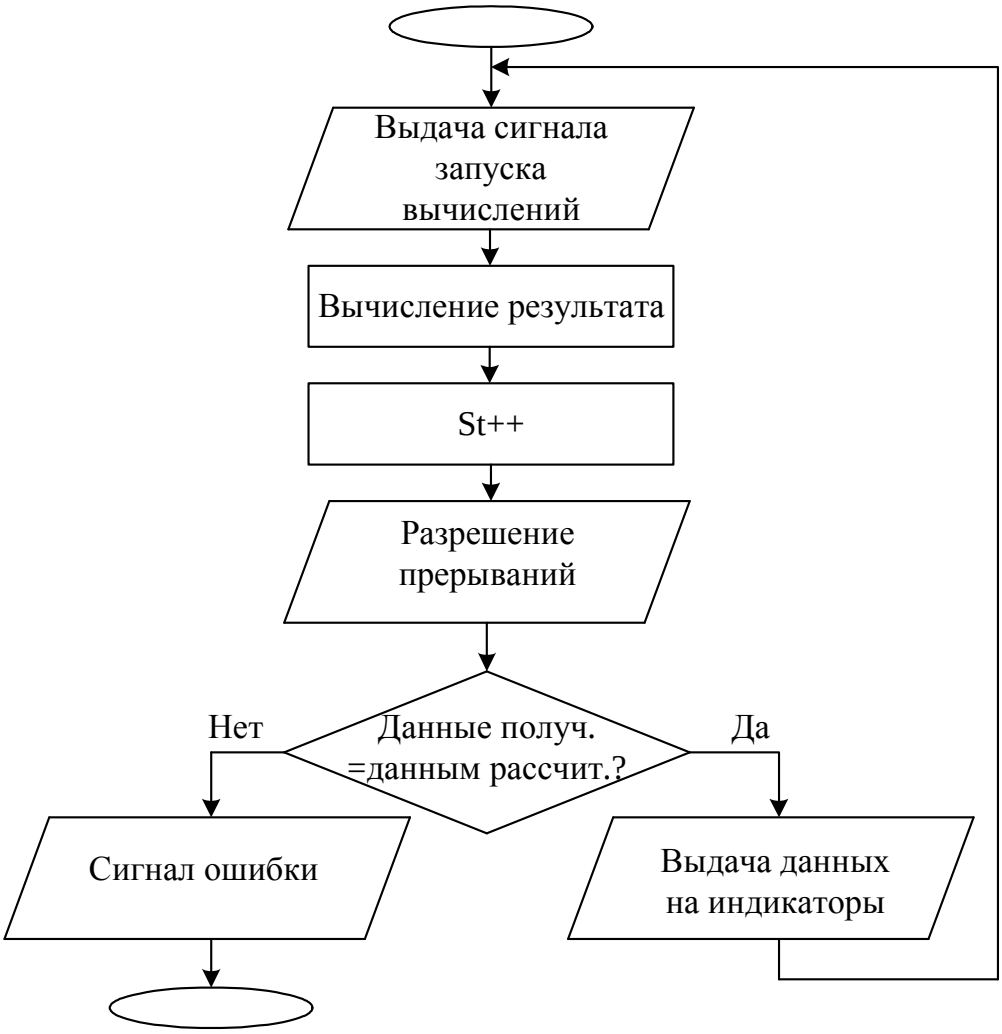


Рис. 5.4. Алгоритм программы управления процессором

Листинг 5.2

```
volatile int dma_complete = 0;     // Переменная-флаг
volatile alt_u32 buffer[1024];     // Буфер данных
….
void dma_done (void* handle, void* data)
{ dma_complete = 1; }
```

Процедура инициализации канала приема модуля DMA приведена в листинге 5.3.

Для настройки канала приема Rxchan использованы следующие функции из библиотеки *sys/alt_dma.h* :

alt_dma_rxchan_open (const char * name) – функция, получающая дескриптор DMA канала, аргумент – имя физического устройства;

alt_dma_rxchan_ioctl (alt_dma_rxchan dma, int req, void* arg) – функция настройки канала на работу в определенном режиме, например:

ALT_DMA_RX_ONLY_ON – DMA работает только на прием данных от внешнего устройства.

Для канала передачи используются аналогичные функции, фрагмент соответствующей программы приведен в листинге 5.4.

Листинг 5.3

```
void init_dma()
{
     if ((rxchan = alt_dma_rxchan_open("/dev/dma")) = = NULL)
            {// Ошибка канала}
     if ((rc = alt_dma_rxchan_ioctl
           (rxchan, ALT_DMA_RX_ONLY_ON,
           (void *)(TEST_COMPONENT_BASE))) < 0)
            {// Ошибка канала приема }
     if ((rc = alt_dma_rxchan_ioctl
           (rxchan, ALT_DMA_SET_MODE_32, NULL)) < 0)
            {// Ошибка DMA }
}
```

Листинг 5.4

```
void init_dma()
{
     if ((txchan = alt_dma_txchan_open("/dev/dma")) = = NULL)
          {// Ошибка канала}
     else
          {// Канал передачи открыт}
return 0;
 }
```

Листинг 5.5

```
int main()
{
     unsigned long data[7];
     void* buffer = data;

     init_dma();
     if ((rc = alt_dma_rxchan_prepare
(rxchan,buffer,1024,done,NULL)) < 0)
     // Проверка успешности передачи
     {
     // Ошибка канала
     exit (1);
     }
     while (!dma_complete);
     {// Передача успешна}
     return 0;
}
```

Пример основной программы, предусматривающей проведение однократного контроля, приведен в листинге 5.5, в реальных приложениях контроль правильности работы должен осуществляться с определенной частотой (например, задаваемой таймером).

Этап 3. Совместная отладка программной и аппаратной частей системы

Система, которая содержит в себе две асинхронные части, требует особых методов отладки. В предыдущих лабораторных работах были изучены и освоены методы отладки аппаратуры и программного обеспечения. Эти методы позволяли отлаживать лишь раздельную работу аппаратной и аппаратно-программной систем, поэтому могут использоваться для предварительной отладки каждой из составных частей СнК: части, реализованной аппаратно, и части с программно-аппаратной реализацией. Полноценная отладка системы обязательно должна содержать этап совместной отладки, когда проверятся совместимость двух разработанных систем и отлаживается их взаимодействие друг с другом.

Для совместной отладки спроектированной СнК предлагается совместно использовать средства отладки Nios II IDE и SignalTap II. При этом будем считать, что аппаратная и программно-аппаратная системы были отлажены на моделях, независимо друг от друга.

Для совместной отладки СнК с использованием средств Nios II IDE и SignalTap II необходимо выполнить следующие действия.

1. Настроить SiganlTap II в соответствии с рекомендациями этапа 5 лаб. раб. 1. При этом указать анализируемые сигналы и настроить необходимое условие защелкивания.

В качестве тактового сигнала необходимо выбрать сигнал *clk*. Для наблюдения за совместной работой аппаратной и программной частей рекомендуется добавить в список наблюдаемых сигналов следующие сигналы:

– разрешения записи данных в регистр аппаратной части;

– разрешения начала вычисления аппаратной частью.

Номер текущего такта можно отслеживать либо по данным, приходящим от программной части, либо через внутренний вектор аппаратной части, предназначенный для хранения текущего номера такта.

Результат, полученный аппаратной частью, можно контролировать либо через внутренний вектор аппаратной части, предназначенный для хранения

полученного результата, либо через выходной вектор аппаратной части, из которого данные попадают на шину Avalon.

В качестве условия защелкивания удобно использовать сигнал квитирования.

После настройки следует загрузить отлаживаемую систему вместе со встроенным логическим анализатором в ПЛИС. Запустить встроенный логический анализатор в автоматическом режиме.

2. Запустить *Nios II Hardware Debugger* в Nios II IDE и выполнять программу по шагам в соответствии с последовательностью действий этапа 5 лаб. раб. 2.

В определенные моменты выполнения программы по шагам встроенный логический анализатор будет фиксировать значения выбранных для анализа сигналов и отображать полученные результаты на экране компьютера.

В итоге при выполнении программы в пошаговом режиме имеется возможность наблюдать за внутренними сигналами системы, переменными программного обеспечения, содержимым регистров процессора. То есть совместное использование средств SignalTap II и Nios II IDE позволяет построить полную картину поведения системы, а главное, помогает проследить за взаимодействием двух ее составных частей: аппаратной и программно-аппаратной.

На этом этапе требуется проанализировать полученные результаты, устранить ошибки, если они есть.

Этап 4. Формирование искусственной неисправности в аппаратной части

1. Для проверки возникновения аппаратной неисправности необходимо внести ошибку в код аппаратной части (например, сместить одну из фаз во времени). После перекомпиляции и записи в ПЛИС система должна выдать сигнал неисправности (красный светодиод) и остановить выполнение программы. Таким образом, демонстрируется повышенная надежность соответствия проекта заданным спецификациям. При ошибке разработчика в программной или аппаратной части на этапе описания алгоритмов либо на этапе написания программы или же при ошибке записи в кристалл ПЛИС система выдаст сигнал о неисправности.

2. Процесс взаимодействия систем в состоянии неисправности следует также пронаблюдать в режиме совместной отладки и на реальном примере

убедиться в работоспособности данного метода. Сделать выводы о его достоинствах и недостатках.

Для увеличения ремонтопригодности и верифицируемости проекта можно использовать более гибкие возможности микропроцессорной части, что не входит в цели данной работы.

По завершении работы необходимо теоретически оценить быстродействие частей системы и построить диаграммы обмена данными между ее частями, провести анализ полученных и рассчитанных результатов, сделать выводы по результатам лабораторной работы.

Содержание отчета

Отчет должен быть выполнен по требованиям ЕСПД и включать в себя:

1. Структурную схему сгенерированного soft-процессора.
2. Блок-схему алгоритма оптимизированной программы для Nios II.
3. Текст оптимизированной программы с комментариями.
4. Схему аппаратного устройства для решения поставленной задачи.
5. VHDL-код устройства с комментариями.
6. Результаты моделирования работы системы.
7. Результаты расчетов быстродействия и аппаратных затрат.
8. Временную диаграмму работы системы с комментариями по этапам.
9. Выводы по проделанной работе и полученным результатам совместной работы аппаратуры и программируемой части.

Список литературы

1. Грушвицкий Р. И., Мурсаев А. Х., Угрюмов Е. П. Проектирование систем на микросхемах с программируемой структурой. СПб.: БХВ-Петербург, 2006.

2. Угрюмов Е. П. Цифровая схемотехника: учеб. пособие для вузов. 3-е изд. СПб.: БХВ-Петербург, 2010.

Ресурс Интернета

http://www.altera.com

ПРИЛОЖЕНИЕ

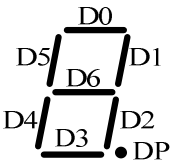
Рекомендованные номера контактов ПЛИС учебной платы DE0

Обозначение переключателя на плате	Номер контакта переключателя*
SW0	J6
SW1	H5
SW2	H6
SW3	G4
SW4	G5
SW5	J7
SW6	H7
SW7	E3
SW8	E4
SW9	D2

* «0» – внизу, «1» – вверху

Обозначение светодиода на плате	Номер контакта светодиода*
LED G0	J1
LED G1	J2
LED G2	J3
LED G3	H1
LED G4	F2
LED G5	E1
LED G6	C1
LED G7	C2
LED G8	B2
LED G9	B1

* «0» – погашен, «1» – горит

Обозначение индикатора на плате	Контакты сегментов индикаторов *								Нумерация сегментов
	D0	D1	D2	D3	D4	D5	D6	DP	
HEX0	E11	F11	H12	H13	G12	F12	F13	D13	
HEX1	A13	B13	C13	A14	B14	E14	A15	B15	
HEX2	D15	A16	B16	E15	A17	B17	F14	A18	
HEX3	B18	F15	A19	B19	C19	D19	G15	G16	

* «1» – сегмент погашен, «0» – сегмент горит

Номер контакта тактового сигнала – G21.

Номера контактов кнопок: But0 – H2, But1 – G3, But2 – F1; «0» – нажата, «1» – отпущена.

Содержание

Редактор Н. В. Лукина

Подписано в печать 09.12.14. Формат 60×84 1/16.
Бумага офсетная. Печать цифровая. Печ. л. 3,0.
Гарнитура «Times New Roman». Тираж 61 экз. Заказ 173.

Издательство СПбГЭТУ «ЛЭТИ»
197376, С.-Петербург, ул. Проф. Попова, 5