Микроконтроллер для встроенного применения – NIOS. Конфигурация шины и периферии

Иосиф Каршенбойм

ik@lmail.loniis.ru

Введение

Приехал к нам как-то год назад в Питер лектор да и начал читать доклад. В названии доклада были слова «Altera», «микропроцессор», «применение» и т. д. И вот уже вскоре все мы, сидящие в зале узнали, что попытки разработки и реализации микроконтроллеров в FPGA не имеют смысла, так как реализация устройства, «похожего» на 386EX, довольно затруднительна и требует много ресурсов от микросхемы FPGA типа EPF10K20. И такую точку зрения можно услышать довольно часто. Действительно, если необходимо разработать довольно медленное и простое устройство, то тут и сомнений быть не может. Но что-же делать в остальных случаях?

Пришлось рассказать докладчику о тех проблемах, с которыми автор статьи встретился при разработке Интернет-шлюза, и о том, нужен ли в действительности микроконтроллер в FPGA. А дело было в следующем. Необходимо было разработать контроллер управления сетевым трансивером с функцией MAC-контроллера для сети Ethernet 10/100 МГц, да плюс к этому надо было разбираться с очередью в сотни пакетов на прием и передачу, распределять принятые пакеты на «свои» и «широковещательные», выбрасывать «битые» пакеты, вести статистику и т. д., а главное, при дуплексной передаче уметь быстро управлять многоканальным контроллером DMA, который и перегружал бы данные из Altera в хост для последующей обработки. Первое, что было сделано, — статический автомат. Однако через небольшое время он разросся в жуткого монстра, и от него отказались. Вот таким образом пришлось обратиться к процессорам. Далее выяснилось, что все быстрые «мелкие» процессоры работают асинхронно и не имеют аппаратного входа «готовность», а для программной привязки их к синхронному проекту в Altera нужно, во-первых, несколько команд процессора, что займет несколько тактов синхрочастоты, и, во-вторых, это также требует ресурса микросхемы FPGA и, в третьих, занимает довольно много места на плате. Быстрые «крупные» процессоры имеют возможность аппаратной синхронизации по входу «готовность», но дороги и занимают еще больше места на плате. Да и применение «крупного» процессора для таких задач нецелесообразно. Вот тогда-то и был разработан «самодельный» микроконтроллер, который можно условно назвать «Стрелочник». Описание этого контроллера приведено автором в [1]. Задачей такого контроллера было именно не обрабатывать, а переключать потоки информации. Контроллер занимает от 400 до 600 логических ячеек, что составляет до 15 % от 20K100, и программируется на ассемблере. Этот-то интерес ко встроенным микроконтроллерам и позволил написать данную статью.

«Ядра — чистый изумруд» и «Ядерные колхозы»

В конце 90-х годов эпоха расцвета специализированных микросхем завершилась. Сегодня вряд ли можно себе представить начинку сотового телефона, состоящую из отдельных микросхем процессора, сигнального процессора, отдельного контроллера дисплея и т. д. Развитие технологии по значительному увеличению ресурсов микросхем приводит к кардинальным изменениям в проектировании устройств. Сегодня «горячая тема» — разработка программной модели ядер устройств. Описания ядер на языках группы VHDL позволяет компоновать проекты из готовых и отлаженных частей. Далее все части собираются в один проект для создания «системы на кристалле».

Фирмы-производители микросхем предлагают различные ядра, оптимизированные под их продукцию, для встраивания в проекты пользователя, например http://www.altera.com/ipmegastore/index.html.

Фирма Altera предлагает микроконтроллеры на ядре an ARM9T (http://www.arm.com/) или на ядре MIPS32 4K (http://www.mips.com/).

Фирма Triscend предлагает 40-MHz 8051-ядро на кристалле с FPGA, доступной пользователю. Аналогичный продукт есть и у фирмы Atmel — FPSLIC.

Более подробное описание по предлагаемым продуктам приведено в [2, 3].

Части проекта, которые могут быть повторно использованы, становятся товаром. Появился рынок по перепродаже проектов и их частей — http://www.hellobrain.com/. Но вместе с тем существуют и развиваются некоммерческие центры, такие как http://www.fpgacpu.org/.

Далее о «ядерных колхозах». Как только появились ядра процессоров как товарные продукты, так тут же в игру вступили фирмы разработчики программного обеспечения. Это в первую очередь операционные системы реального времени, компиляторы языков высокого уровня, отладчики, симуляторы и т. д. Не остались в стороне и фирмы-разработчики вспомогательного оборудования. Здесь можно упомянуть о фирмах начиная от производителей «стартовых наборов» и до производителей измерительного и испытательного оборудования. Таким образом, весь комплекс средств для разработки, отладки и производства систем со встроенными микроконтроллерами становится также товарным продуктом, что позволяет значительно ускорить темпы разработки и сократить трудо-

емкость разрабатываемого проекта. Сюда можно отнести и работу фирмы Clear logic (http://www.clear-logic.com/) по разработке заказных аналогов загружаемых микросхем. Подробнее об этом можно узнать на сайте фирмы «ЭФО». Однократно запрограммированные микросхемы с «системой на кристалле» будут пригодны для применений, требующих долговременной работы устройства.

Что же касается российских разработчиков и разработчиков стран СНГ, то здесь несколько иная ситуация. Фирмы-производители микросхем имеют программу по поддержке партнеров-разработчиков ядер (мегафункций). Это и реклама, и сервис, и многое другое. Но для российских разработчиков попасть в партнеры пока еще практически невозможно, и это резко ограничивает их возможности. Но, тем не менее, и у нас есть интересные, на взгляд автора, разработки. Очень интересны разработки лаборатории НИП: IEESD-2000 — (WInter + HLCCAD) среда совместной разработки программного и аппаратного обеспечения встроенных систем http://nit.gsu.unibel.by/IEESD-2000, позволяющие проектировать на VHDL процессор и периферию, а затем отлаживать программное обеспечение на программном иммитаторе проектируемого устройства.

Также необходимо отметить и фирму «Технофорт» (http://www.technoforth.ru/), практически приступившую к коммерческому использованию «системы на кристалле» собственной разработки. В «систему на кристалле» входит ядро форт-процессора, развитая периферия, такая как UART, таймер, параллельный ввод — вывод (PIO), интерфейс SRAM, интерфейс FLASH-памяти, контроллер локальной сети 10/100 Ethernet. В процессор поддержан компилятором языка высокого уровня — форт с включением форт-ассемблера и отладчика.

NIOS как пример встроенного микроконтроллера

Целью данной статьи является не реклама конкретного представителя из группы встраиваемых микроконтроллеров, а описание методики применения и конфигурации ядра процессора под требования пользователя. Описание ориентировано на освещение аппаратных проблем, которые могут возникнуть как при применении данного микроконтроллера, так и при разработке для него собственных библиотечных компонентов. Некоторые технологические приемы, описанные в данной статье, например такие, как единообразная система названий сигналов, могут оказаться полезными и в других применениях. Описание «Системы Авалон» и «Шины Авалона» приводится в соответствии с документацией фирмы Altera™, поставляемой в комплекте со стартовым набором на процессор Nios™.

Особенности NIOS

Часть проекта фирмы Altera™ по встроенным решениям процессор Excalibur и его вариант реализации — программное ядро встроенного процессора Nios™.

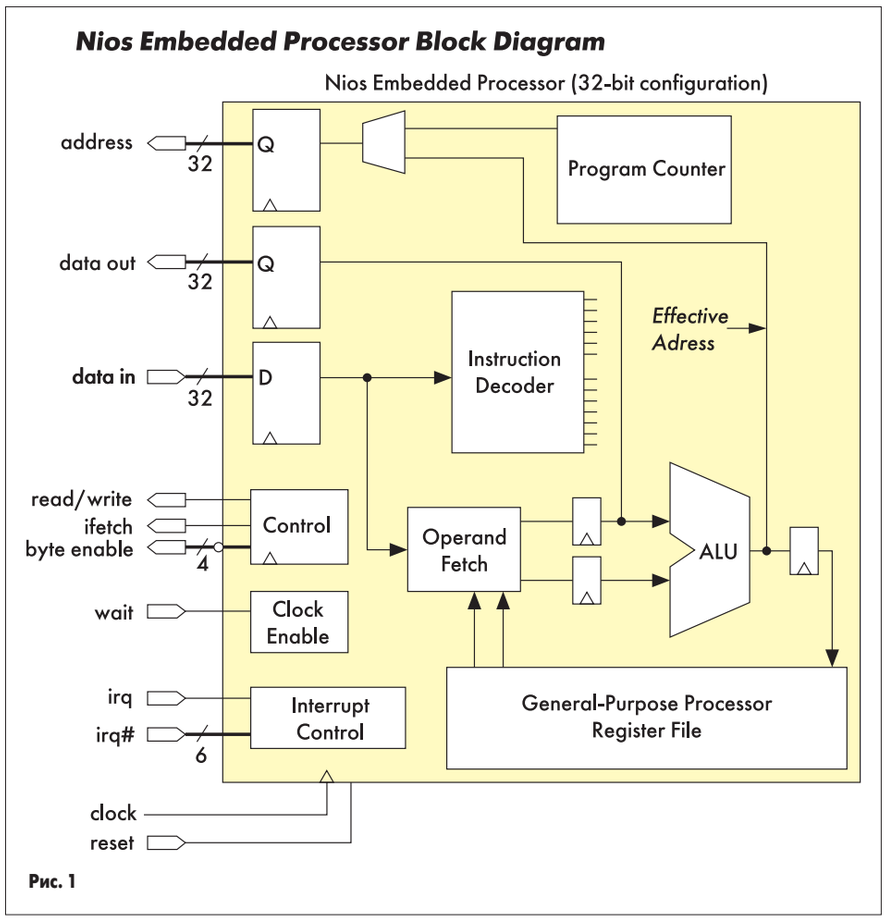

Рис. 1

Nios — это встраиваемый процессор общего назначения, с перестраиваемой конфигурацией, который легко вписывается в устройство Altera ® APEX ™, оставляя большинство логики, доступной для размещения там периферийных устройств и пользовательских функций. Встраиваемое ядро процессора Nios — конвейерный RISC-процессор, в котором команды выполняются за один цикл частоты синхронизации.

Рис. 1 показывает встроенный процессор Nios, который имеет следующие особенности:

Загружаемая RISC-архитектура с перестраиваемой конфигурацией

- Полностью синхронный адрес и интерфейс шины данных.
- Разрядность данных 16 или 32 бита.
- Адресное пространство 128 Кб и 8 Гб соответственно.
- 16-битовая система команд.
- Малые требования к памяти.
- Совместимость со стандартными FLASH-устройствами.
- Поддерживает как память на кристалле, так и внешнюю память.
- Архитектура конвейера с 5 стадиями.
- Одна команда выполняется за один цикл частоты.
- До 512 регистраторов общего назначения.
- Для ускорения обработки прерывания доступ к регистрам осуществляется через окно в 32 регистра.
- 64 векторизованных прерывания.

Периферийные устройства на кристалле, настраиваемые пользователем

- Универсальный Асинхронный Приемопередатчик (UART), таймер, параллельный ввод — вывод (PIO), SRAM, и интерфейс FLASH-памяти.
- Будущие периферийные устройства включают в себя: последовательный периферийный интерфейс (SPI), модулятор ширины импульса (PWM), IDE контроллер диска, контроллер Локальной сети 10/100 Ethernet, на основе протокола CSMA-CD (MAC) и SDRAM контроллер.

Оптимизированный для APEX

- Использует 13 % APEX EP20K200E устройство в конфигурации на 16 бит.
- Использует 20 % APEX EP20K200E устройство в конфигурации на 32 бита.
- До 50 MIPs и 50 МГц в APEX EP20K200E устройстве при конфигурации на 32 бита.

MegaWizard интерфейс, который конфигурирует ядро процессора, подключение шин и периферийные устройства

- Генерирует периферийный модуль шины (PBM).
- уровень IRQ и приоритеты.
- Назначает периферийные базовые адреса
- Осуществляет 8-, 16-, и параметры конфигурации ширины данных на 32 бита (динамическая шина, устанавливающая размеры).
- Конфигурирует периферийные состояния ожидания.

Интерфейс MegaWizard позволяет пользователю выбрать тип и производительность отдельных узлов процессора, чтобы иметь достаточно производительности при минимуме затраченных ресурсов. Подробное описание и последовательность действий по конфигурации процессора и периферии приведены в [6]. Далее остановимся только на основных

этапах конфигурирования, так как именно эти этапы связаны с файлом назначений параметров конфигурации.

Рис. 2–5 показывают конфигурацию процессора Nios при помощи MegaWizard.

На рис. 2 приведен внешний вид окна MegaWizard, определяющего выбор версии процессора 32 или 16 битов.

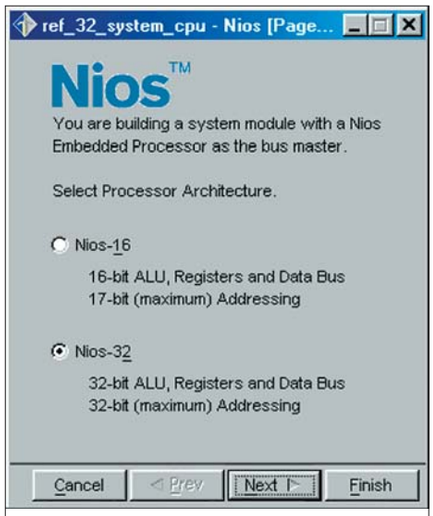

Рис. 2

На рис. 3 приведен внешний вид окна MegaWizard, предлагающего пользователю выбрать разрядность шины адресов.

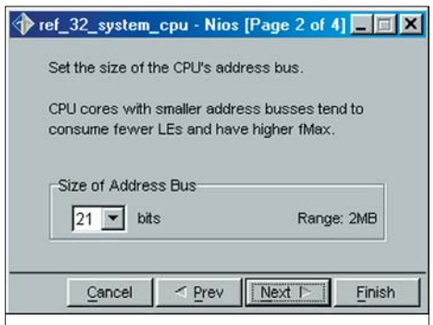

Рис. 3

На рис. 4 приведен внешний вид окна MegaWizard, определяющего число регисторв в регистровом файле и режим работы блока сдвигов.

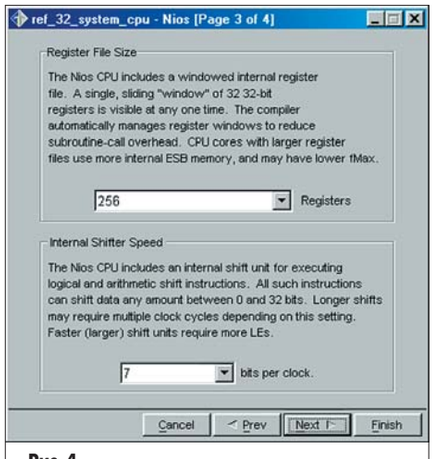

Рис. 4

На рис. 5 приведен внешний вид окна MegaWizard, позволяющего выбрать режим работы блока умножения.

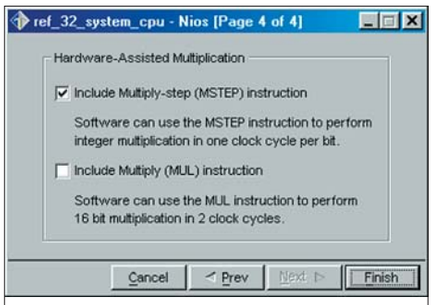

Рис. 5

Далее производится выбор и конфигурация периферии. Данный раздел является наиболее интересным и важным для пользователей — разработчиков устройств.

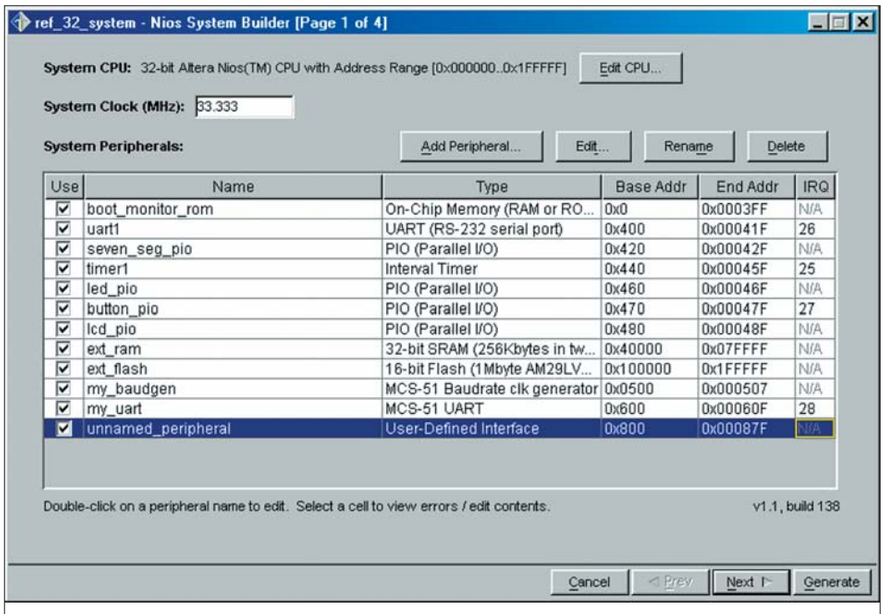

Рис. 6

Интерфейс MegaWizard позволяет пользователю определять подключения встроенного процессора к остальной части системы. Используя простой интерфейс, пользователь может проектировать карту адреса с различными типами подключения, разрядности, скоростями памяти и периферийных устройств. Определяются так же и векторы прерываний от периферии. Интерфейс MegaWizard генерирует логику интерфейса, которая подключает все периферийные устройства Nios, как определено пользователем. Первая страница MegaWizard, на которой показано, какая периферия должна быть реализована в данном проекте, изображена на рис. 6.

Периферийные устройства, которые могут быть реализованы, делятся на две большие группы: внутренние и внешние. Внутренние интегрируются в блочный элемент процессора: например, частью блока процессора будет являться функция MCS-51 UART, разработанная автором статьи. Данная функция была разработана как библиотечная для Nios и, после работы MegaWizard данная функция «видна» на блоке Nios только своими входами и выходами (см. рис. 12). Внешние устройства подключаются к блоку процессора через шины. Данный раздел подробно описан в разделе «AVALON: Спецификация Шины».

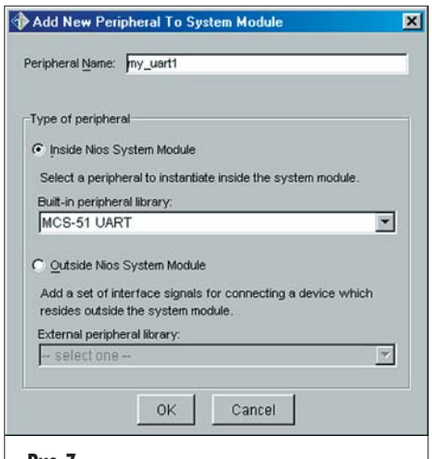

Рис. 7

Добавление устройств производится по нажатию кнопки «Add Peripheral…»

Устройства, соответственно, выбираются из библиотеки внутренних либо и внешних устройств, как показано на рис. 7.

Внутренние устройства жестко привязаны к временной диаграмме процессора, тогда как внешние устройста имеют достаточно параметров для настройки и сопряжения с процессором любых типов устройств и микросхем, которые требуются пользователю для реализации его проекта.

Внешние устройста, как показано на рис. 8, могут иметь раздельные шины данных для записи и для чтения или могут иметь общую трехстабильную шину.

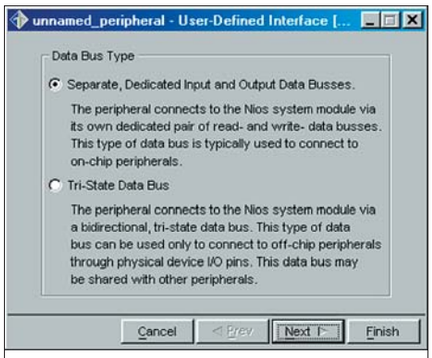

Рис. 8

На рис. 9 показано, что внешние устройста могут иметь дополнительные циклы преустановки данных при записи и при чтении.

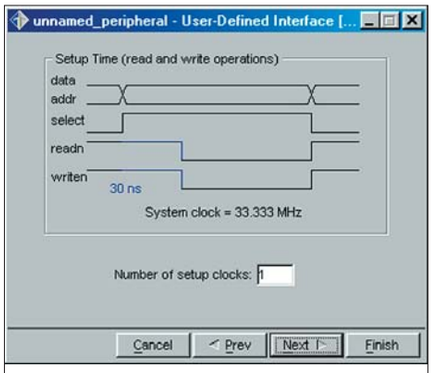

Рис. 9

На рис. 10 показано, что внешние устройства в операциях записи могут иметь дополнительные циклы для хранинния информации на шине. Так выглядит временная диаграмма записи с одним циклом предустановки и одним циклом хранения информации.

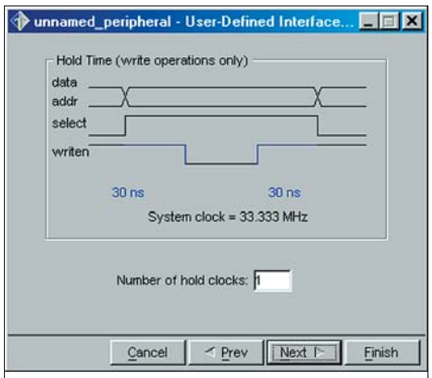

Рис. 10

На рис. 11 показано, что внешние устройста могут иметь дополнительные циклы ожидания для записи и для чтения. Эти циклы могут быть сгенерированы логикой шины или управляться от сигнала периферии.

Для каждого модуля выбирается имя, базовый адрес и уровень запроса прерывания, если он необходим. После того как периферийные устройства добавлены, интерфейс к каждому будет автоматически определен.

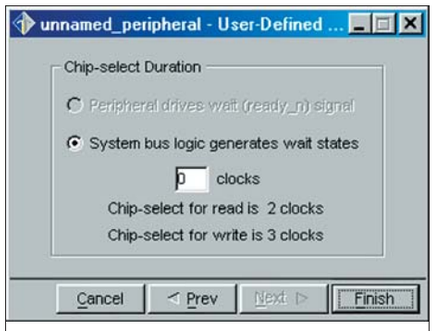

Рис. 11

Дополнение к программе MegaWizard создает, согласно указанной конфигурации, периферийный модуль шины (PBM).

Следующие особенности PBM полностью настраиваемы:

- Базовый адрес.
- Зона Адресов.
- Ширина Данных.
- Read-only/read-write/write-only.
- Состояния ожидания.
- Сигнал IRQ и его приоритет.

Типичные применения

Процессор Nios может быть конфигурирован для широкого диапазона прикладных программ. Применение ядра Nios на 16 бит, выполняющего маленькую программу из памяти, расположенной на кристалле, позволяет получить эффективный контроллер, который может быть установлен вместо тяжело кодируемого статического автомата. Применение ядра Nios на 32 бита с внешней памятью программы во FLASH-памяти и большой внешней главной памятью позволит получить мощный 32 битный встроенный процессор.

Типовая конфигурация встроенного процессора Nios при исполнении на 32 бита показана ниже.

- 256 регистраторов общего назначения.
- Один модуль умножителя.
- Один UART (скорость в бодах предварительно установлена).
- Один таймер на 32 бита.
- 512 байтов ROM на-кристалле.
- Одно периферийное устройство с 7-сегментным индикатором.

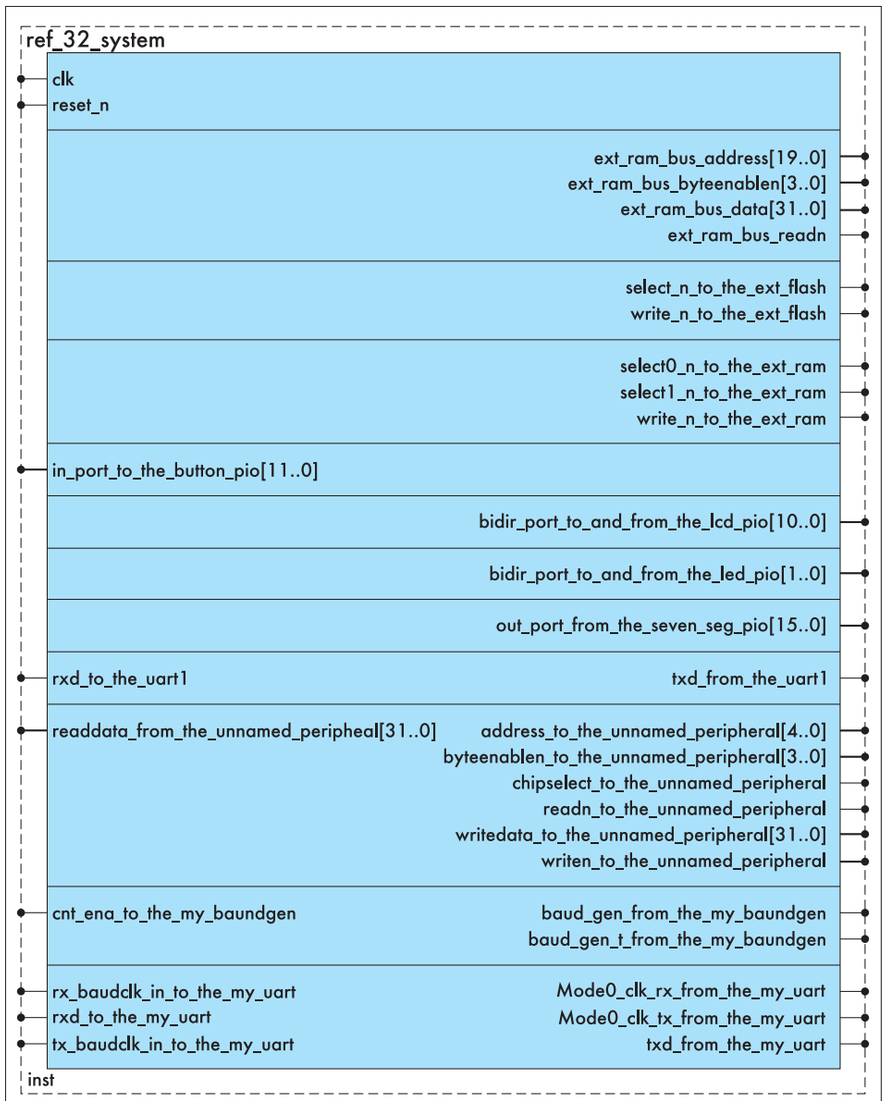

Рис. 12

- Периферийный модуль шины PBM (шина на чипе).
- SRAM-интерфейс (256 байтов).
- Интерфейс FLASH ROM (1 Мб).

Рис. 12 показывает блок встроенного процессора Nios для исполнения на 32 бита.

Программные средства разработки

Процессор Nios поддержан полным набором GNUPro встроенных системных инструментальных средств развития, созданных Cygnus ®, и Red Hat ®. Эти инструментальные средства включают C/C ++ транслятор, ассемблер и отладчик.

Система команд Nios ориентирована на встроенные прикладные программы и включает команды, которые являются особенно полезными во встроенных системах (например, за один такт выполняется команда «проверка бита и пропуск»). Ядро Nios включает поддержку для аппаратных контрольных точек и управления выполнения через отладчик GNUPro.

AVALON

Данный раздел будет наиболее интересен как для разработчиков библиотечных функций к Nios так и для разработчиков устройств. Раздел содержит описание шинного интерфейса, файлов конфигурации интерфейса и файлов сопряжения разрабатываемых библиотечных функций с шиной.

Спецификация Шины

1. Краткий обзор

Avalon — простая шина для построения микропроцессорной системы на кристалле.

Основные цели проекта шины Avalon:

- Простота
- Малое количество ячеек кристалла задействовано для логики шины.
- Полностью синхронные операции.

Поэтому спецификация Avalon преднамеренно не поддерживает особенности подобно передачам данных пакетами (burst-data transfers) и возможность работы в режиме multi-master.

Каждая шина Avalon подключает единственного мастера ко многим периферийным устройствам (slaves). Мастером шины Avalon может быть, например, ядро микропроцессора Nios. Периферийное устройство Avalon может быть, например, UART, таймер, или устройство памяти.

Все трансакции шины Avalon передают (перемещают) единственный байт, слово половинной разрядности или слово (8, 16, или 32 бита) между одним из периферийных устройств и мастером.

Avalon был адаптирован так, чтобы работать «на кристалле», где для взаимосвязи устройств вместо трехстабильных шин используются мультиплексоры.

Avalon также включает ряд особенностей и соглашений для автоматической генерации системы, шины и периферийных устройств посредством программы SOPC Builder software.

2. Определения Терминов

2.1. Система Avalon

«Система Avalon» является модулем, который содержит:

- Единственный мастер.
- Набор периферийных устройств.
- Шину Avalon (см. 2.2), которая объединяет их.
- Выводы ввода/вывода на модуле системы Avalon будут непосредственно соответствовать выводам устройств, входящих в состав блока, которые требуют внешних подключений. Модуль системы Avalon, модуль шины Avalon (см. 2.2) и все подмодули (периферийные устройства и хозяин), находящиеся в пределах системного модуля, сгенерированы SOPC Builder software. Программа SOPC Builder software является ответственной за создание и поставку hdl-файлов, которые осуществляют каждый из этих компонентов в каталоге проекта Quartus пользователей. Программное обеспечение может опционально генерировать синтезируемый .edf — список связей, который описывает модуль системы Avalon как единый модуль.

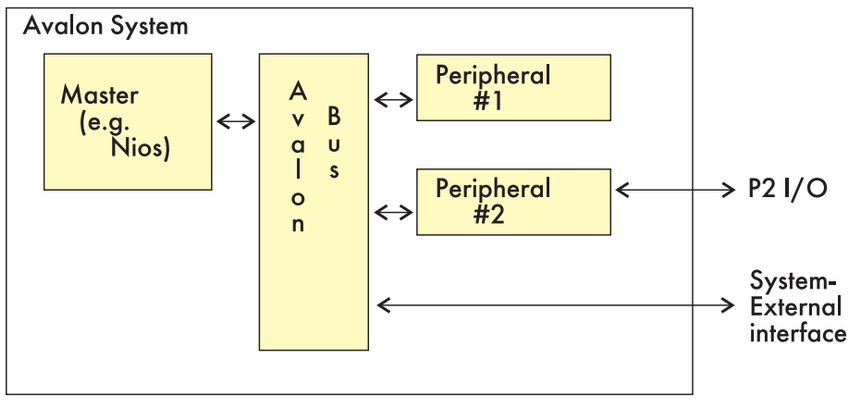

Рис. 13

2.2. Шины Avalon

Шина Avalon является модулем, как правило, с большим количеством выводов ввода — вывода. Все подмодули в пределах системы Avalon соединяются или с модулем шины Авалона, или, непосредственно с портами ввода/вывода уровня системы Avalon (рис. 2.2). Модуль шины Avalon будет иметь один периферийный-определенный порт интерфейса (PSIP, см. 2.3) для каждого периферийного устройства в системе, и один PSIP для хозяина. Шина Avalon, все ее порты, и вся логика, которая для этого необходима, генерированы программным способом при помощи SOPC Builder software.

Модуль шины Avalon будет содержать всю логику, необходимую для подключения сигналов интерфейса мастера ко всем периферийным устройствами, и наоборот.

Модуль шины Avalon содержит логику, которая исполняет функции:

- *Декодирование адреса.*
  Производит сигналы выбора кристалла для каждого периферийного устройства.
- *Мультиплексирование шины данных.*
  Передает данные от выбранного периферийного устройства к мастеру.
- *Генерация дополнительных циклов ожидания.*
  Добавляет дополнительные циклы, при чтении или записи, когда они требуются выбранному периферийному устройству. При этом учитывается добавление циклов для требуемой предустановки сигналов.
- *Динамическое изменение разрядности шины.*
  Чтобы читать или записывать данные большей разрядности, чем у периферийного устройства, автоматически выполняются многократные циклы шины.
- *Назначение номера прерывания*
  Устанавливает правильный, расположенный по приоритетам номер IRQ мастеру, когда одно или более периферийных устройств требуют запроса прерывания.

2.3. Специальный периферийный порт интерфейса (PSIP)

Модуль шины Avalon (см. 2.2) будет иметь специальный периферийный порт интерфейса (PSIP), предназначенный каждому периферийному устройству в системе, и еще один PSIP для хозяина шины.

PSIP — набор выводов ввода/вывода, сгенерированных SOPC Builder software. Этот набор выводов ввода/вывода приспособлен для того, чтобы точно состыковаться с портами интерфейса шины на целевом периферийном устройстве. Как правило, каждый PSIP содержит (например) выводы — address, входы/выходы данных, и вход запроса прерывания, если требуется.

SOPC Builder software «знает», какие сигналы включать в PSIP, на основании описания периферийных устройств в системном PTF файле (см. 2.5). Когда периферийные устройства добавляются или удаляются из системы, все связанное с ними в PSIP так же добавляется или удаляется из модуля шины Avalon.

2.4. SOPC Составитель программы Программное обеспечение.

Программа SOPC Builder software поставляется заказчику как встроенный в программу Quartus MegaWizard.

Это последовательность страниц MegaWizard, основанного на Java, которые позволяют пользователям создавать и редактировать систему Avalon (см. 2.1), используя графический интерфейс пользователя.

Программа SOPC Builder software автоматически обнаруживает компоненты библиотеки Avalon (которые должны содержаться в определенных каталогах и повиноваться некоторым соглашениям) и делает их доступными системному проектировщику.

После завершения редактирования, когда пользователь нажимает кнопку «Finish» программа SOPC Builder software выполнит следующие действия:

- Генерирует HDL-файлы, которые осуществляют всю логику в модуле системы Avalon, включая мастера, все «системно-внутренние» периферийные устройства, и Модуль шины Avalon, который объединяет их все вместе.
- Генерирует custom SDK — каталог, который содержит заголовочные файлы и другие программные библиотечные компоненты, предназначенные для данной системы Avalon.
- Синтезирует (опционально) все HDL-файлы, созданные в шаге (1), чтобы произвести список связей, предназначенный для размещения на кристалле.

Обычно конечным результатом работы «SOPC Builder software» — является:

- Список связей, который описывает модуль системы Avalon.
- Каталог Custom_SDK (см. выше).
- HDL-файл (или Verilog, VHDL или AHDL) который определяет интерфейс к модулю системы Avalon.
- bsf-(символ) файл, необходимый для включения модуля системы Avalon в схемном проекте пользователей.

2.5. PTF-файл

Программа SOPC Builder software использует PTF- файл (периферийный файл шаблона) как базу данных, чтобы сохранить информацию о системе Avalon.

Каждая система Avalon будет иметь собственное описание PTF-файлом, который содержит такую информацию, как:

- Список всех периферийных устройств в системе.
- Информация о каждом периферийном устройстве, включая:
- Его специфический набор сигналов интерфейса шины.
- Пользовательские параметры настройки, введенные в него мастером, если они были сделаны.
- Список hdl-файлов, требуемых для синтеза/симуляции периферийного устройства
- Любая другая информация, необходимая для программного обеспечения SOPC Builder software, чтобы генерировать определенный модуль системы Avalon.

Программа SOPC Builder software может создавать данную систему Avalon только на основе ptf-файла, который определяет эту систему (и, конечно, на основе библиотек, содержащих компоненты, которые используются данной системой). Программа SOPC Builder software при помощи MegaWizard изменяет содержание ptf-файла системы, чтобы соответствовать тем требованиям, которые ввел пользователь.

При завершении редактирования пользователем программы MegaWizard используют содержание ptf-файл как набор команд, чтобы генерировать всю требуемую логику и т. д.

Этот документ будет называться «system ptf file». «System ptf file» — файл, автоматически генерированный программой SOPC Builder software, который содержит полную спецификацию для Nios системного модуля.

3. Синтаксис ptf- файла

Ptf (периферийный файл шаблона) — файлы, доступные как для чтения, так и для редактирования, так как они основаны на текстовых базах данных SOPC Builder software библиотечных компонентах и системах. Программное обеспечение SOPC Builder software создает свой собственный ptf-файл, чтобы сохранять промежуточную информацию о разрабатываемой системе. В SOPC Builder software каждый библиотечный компонент также использует ptf-файлы, чтобы описать библиотечные компоненты.

3.1. ptf-элементы

Ptf-файл содержит два отличных типа синтаксических элементов — НАЗНАЧЕНИЯ и РАЗДЕЛЫ. Далее показаны примеры каждого:

Пример Назначения:

```
data_from_cpu = «16»;
Address_Alignment = «--unknown--»;
Has_IRQ=»1»;
```

Разделы Примера:

```
hdl_INFO
Simulation_hdl_Files = « — неизвестное — «;
Synthesis_hdl_Files = « — неизвестное — «;
USER_INTERFACE
USER_LABELS
name=»UART (RS-232 serial port)»;
```

3.2. ptf-назначения

Назначения — пары названий, одно из которых приписывается другому, и они имеют следующий синтаксис:

<Название (имя) назначения> = «<значение назначения>»;

Названия (имена) назначения могут иметь произвольную длину и могут содержать любой алфавитно-цифровой символ (верхние и строчные буквы), также как символ подчеркивания («_»). Названия(имена) назначения не могут содержать пробелы или другие символы пунктуации. Значения назначения всегда заключаются в двойные кавычки. Значения Назначения могут содержать любые цитаты ASCII-символы, кроме кавычек Все назначения заканчиваются точкой с запятой. Пробел (включая новые строки) игнорируется в ptf-файлах (кроме назначения в указанных значениях, конечно).

3.3.ptf-разделы

Разделы — другая группа ptf-элеменитов, и они имеют такой синтаксис:

<Тип раздела> <название (имя) раздела>
<Тело раздела>

Каждый тип раздела и поле названия (имени) раздела, повинуются тем же самым ограничениям, как и назначения названий (имена), описанное выше (только алфавитно-цифровой символы, никаких пробелов или пунктуации, и т. д.). Каждый раздел должен иметь пару фигурных скобок «{}», заключающих тела раздела. Тело раздела может быть пустое (не содержать никакие символы). Вообще тело раздела может содержать произвольное число назначений и произвольное число разделов. Поскольку разделы могут содержать другие разделы, то ptf-файлы по своей природе — иерархические. Табуляцию и сдвиг часто используют, чтобы делать их содержание более читаемым (но сдвиг, являющийся пробелом, формально игнорируется).

Когда документы относят к специфическому подразделу в ptf-файле, обычно принято использовать «/» (наклонную черту вправо) как разделитель иерархии (даже при том, что такая рекомендация не опознается и не требуется ptf-синтаксисом). Так, когда нужно, в этом документе обратиться, например, к разделу МОДУЛЯ, названный «my_module» в пределах СИСТЕМНОГО раздела, можно применить такую нотацию:

```
SYSTEM/Module my_module/...
```

Продолжение следует