Пользовательские модули и специальные инструкции Nios II

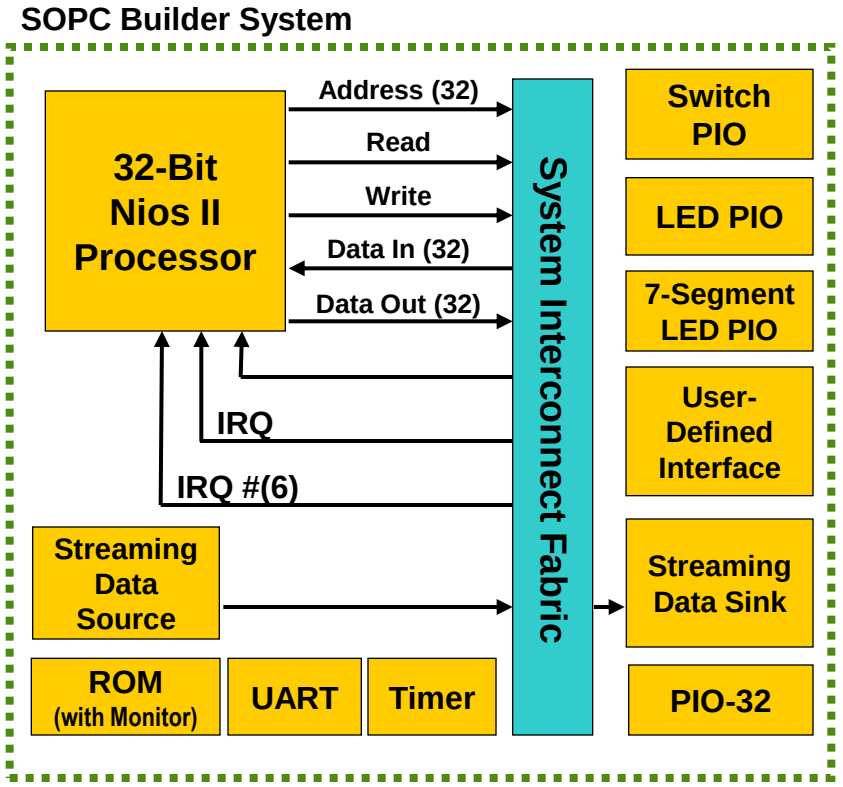
Программа курса

- Шина Avalon
- Пользовательские модули
- Специальные инструкции

Системная шина

Межсоединения QSys

- Архитектура Network-on-Chip (NoC)
- Характеристики
  - Малый объем логики
  - Простота интерфейса
  - Синхронная работа
  - Высокая производительность
- Типы передач
  - Slave Transfers
  - Master Transfers
  - Latency-Aware Transfers
  - Burst Transfers
  - Transfers with Flow Control
  - Streaming Transfers



Соединения

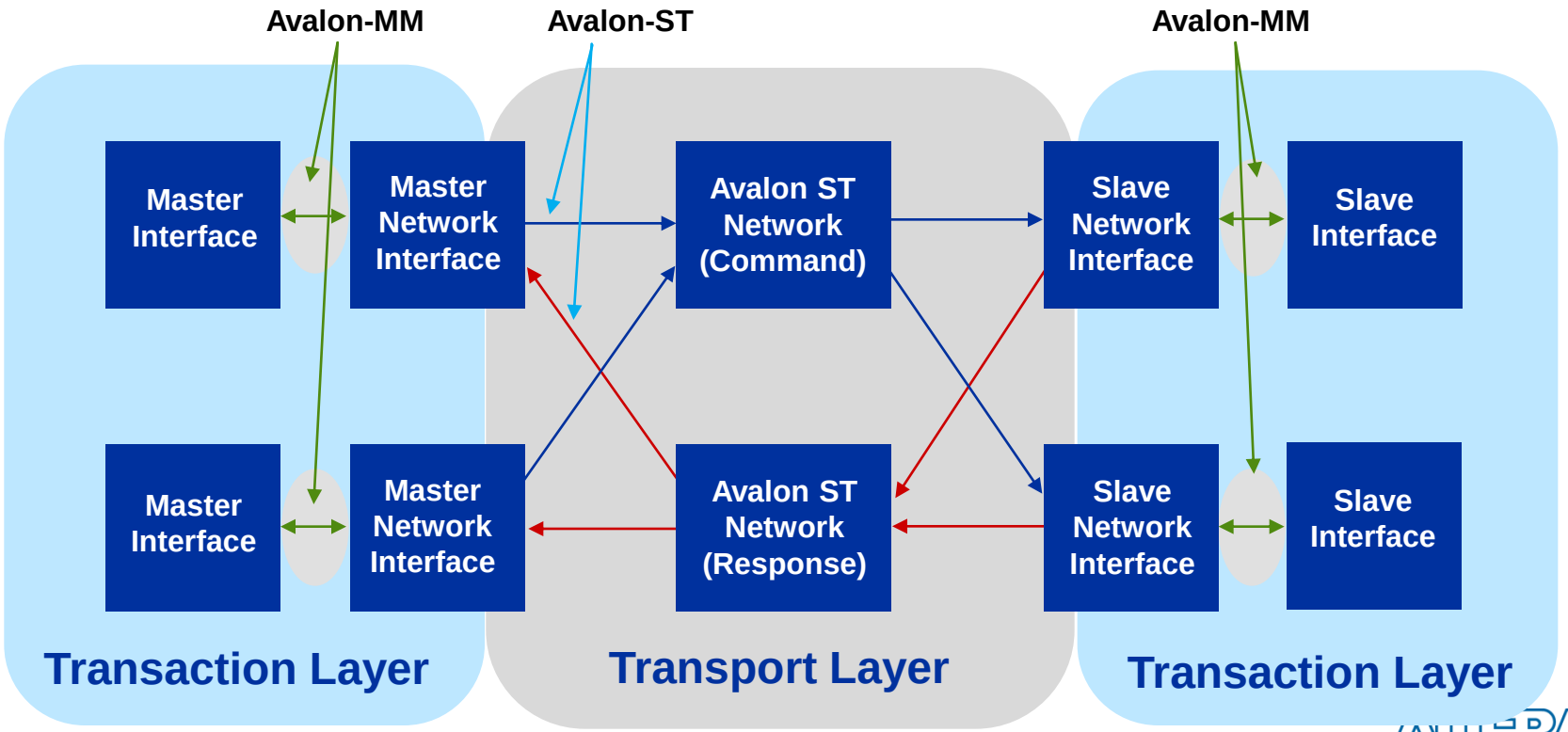
- Генерируются автоматически в QSys
- Формируются отдельно для каждого периферийного модуля
  - Используемые сигналы определяются типом передач
  - Ненужные сигналы шины не используются и не усложняют проектирование и реализацию
- QSys автоматически реализует
  - Арбитраж
  - Декодирование адреса
  - Мультиплексирование данных
  - Управление разрядностью шин
  - Генерацию циклов ожидания
  - Обработку сигналов прерывания

Интерфейсы компонент

- **Две спецификации:**
  - **Avalon Memory Mapped Interface *(Avalon-MM)***
    - Определяет двунаправленные соединения между периферийными модулями
      - *peripheral* ←→ *interconnect* ←→ *peripheral*
    - Периферийные модули должны соответствовать спецификации для используемого ими типа передач
    - Поддерживает одновременный доступ к шине нескольких главных устройств (*multi-mastering*)
    - Предназначена для реализации модулей на общей шине
  - **Avalon Streaming Interface *(Avalon-ST)***
    - Определяет стандартный, гибкий, модульный протокол для однонаправленной синхронной передачи данных от источника к приемнику
      - Мультиплексируемые потоки данных
      - Соединения точка-точка
    - Предназначена для высокоскоростной однонаправленной пересылки данных с малой задержкой

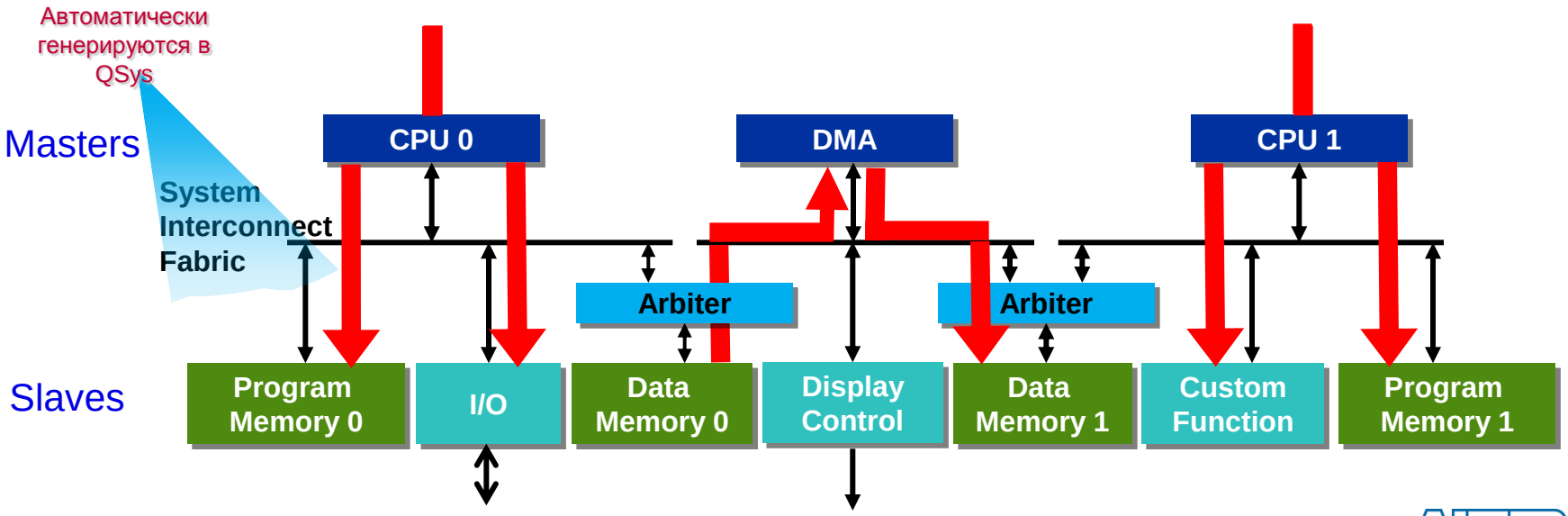
NoC Architecture

- Пакетные транзакции и передача информации
  - Преобразование интерфейса Memory-mapped
    - Каждая передача/запрос упаковывается в пакет и посылается подчиненному устройству
    - Ответ подчиненного устройства также передается в пакете
  - Потоковые передачи соответствуют формату пакета NoC

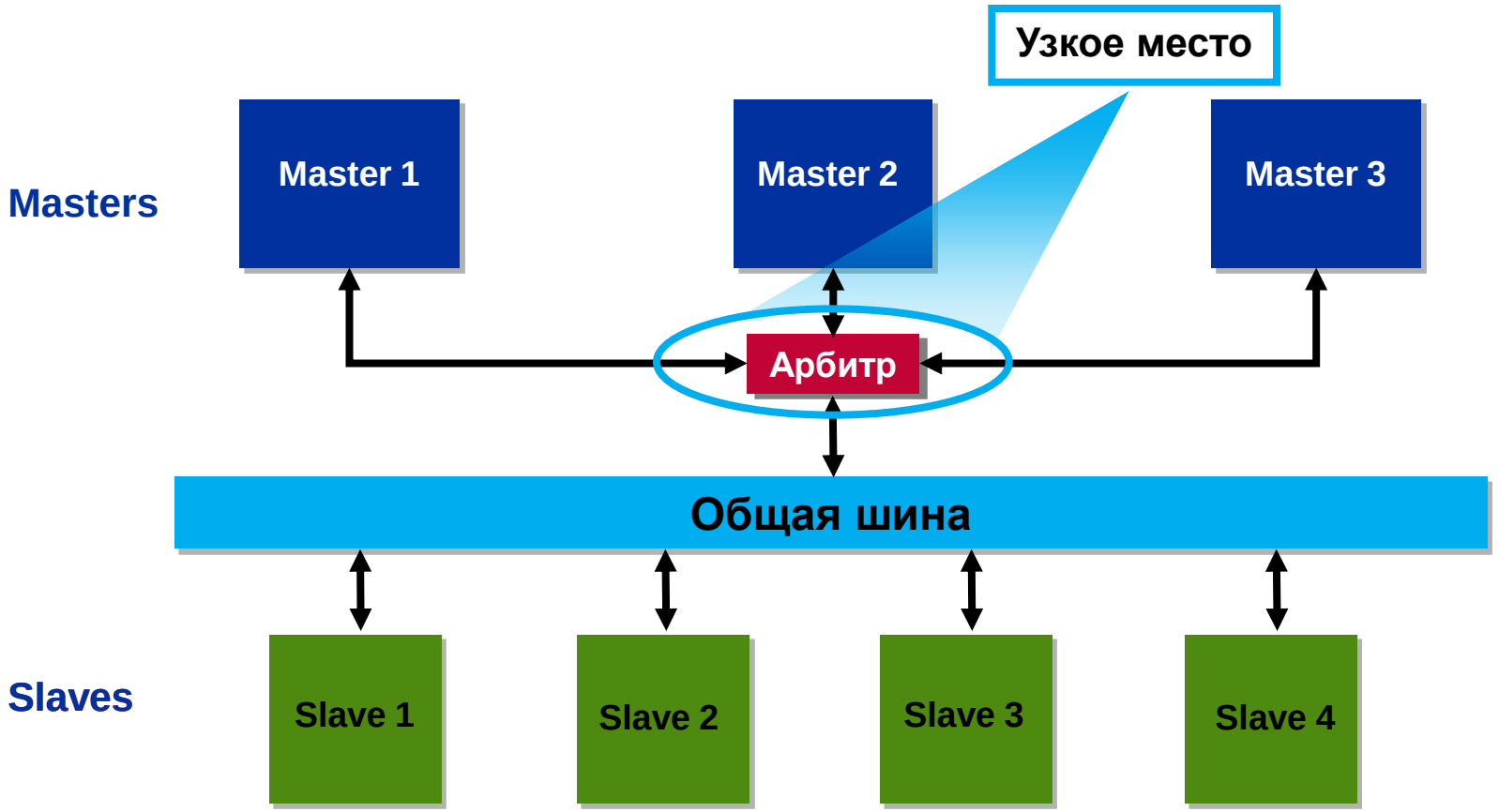


Multi-Mastering

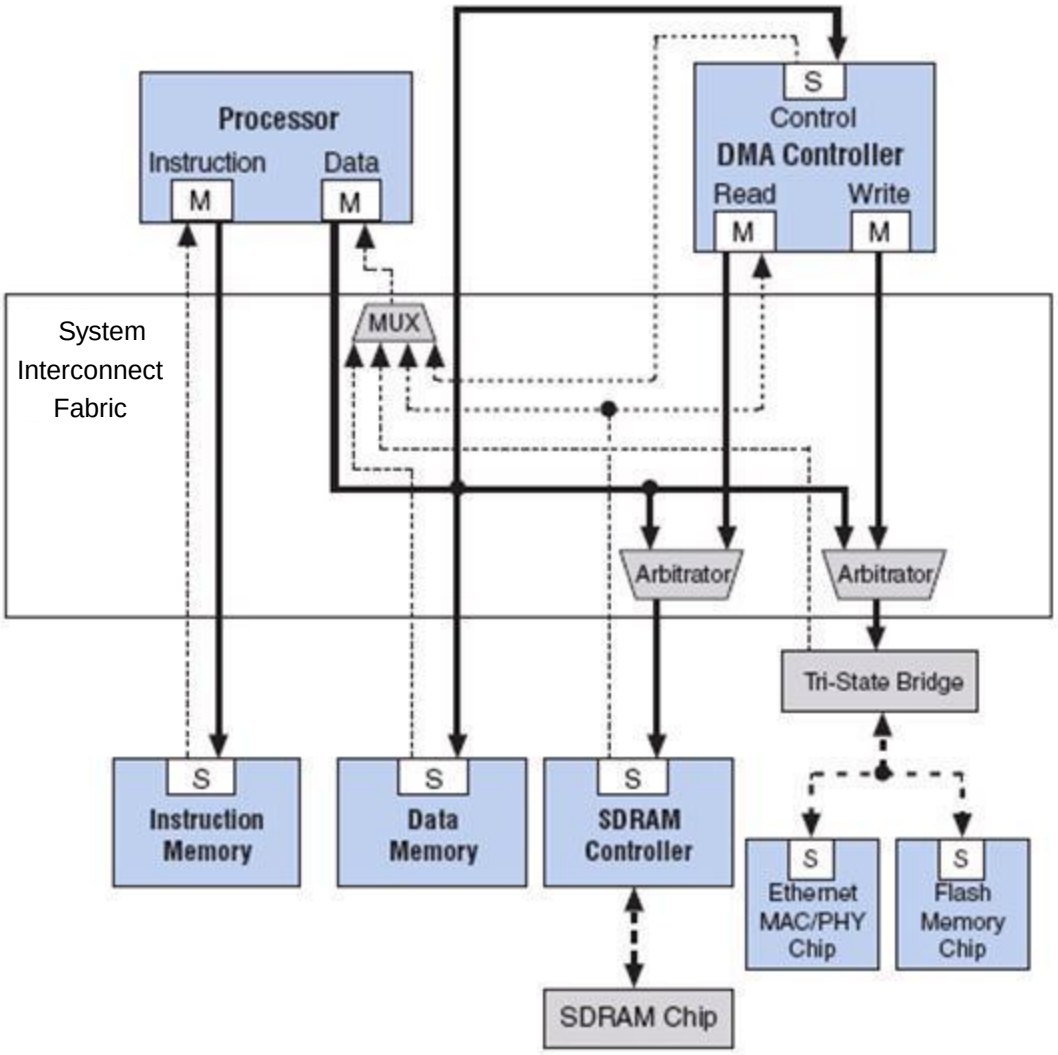
- QSys использует межсоединения в FPGA
  - В кристалле реализуются связи точка-точка
  - Арбитраж и мультиплексирование на стороне подчиненного устройства
    - Несколько транзакций могут осуществляться одновременно при условии, что доступ осуществляется к разным подчиненным устройствам
  - Может осуществляться группировка периферийных модулей по шинам исходя из требований к производительности
- Недостатки
  - Может увеличиваться требуемый объем логики



Сравнение с “Традиционным” подходом: Арбитраж на стороне главного устройства



Структурная схема интерфейса Avalon-MM



Write Data & Control Signals

Read Data

Interface to Off-Chip Device

Шина состоит из связей точка-точка между главными (**master**) и подчиненными (**slave**) устройствами.

Общая шина в ПЛИС физически нереализуема ввиду отсутствия линий с третьим состоянием и несколькими источниками

M Avalon-MM Master

S Avalon-MM Slave

Спецификация интерфейса Avalon

- Полностью определяет стандарт Avalon
- Информация о различных режимах
  - Примеры применения
  - Временные диаграммы

www.altera.com/literature/manual/mnl_avalon_spec.pdf

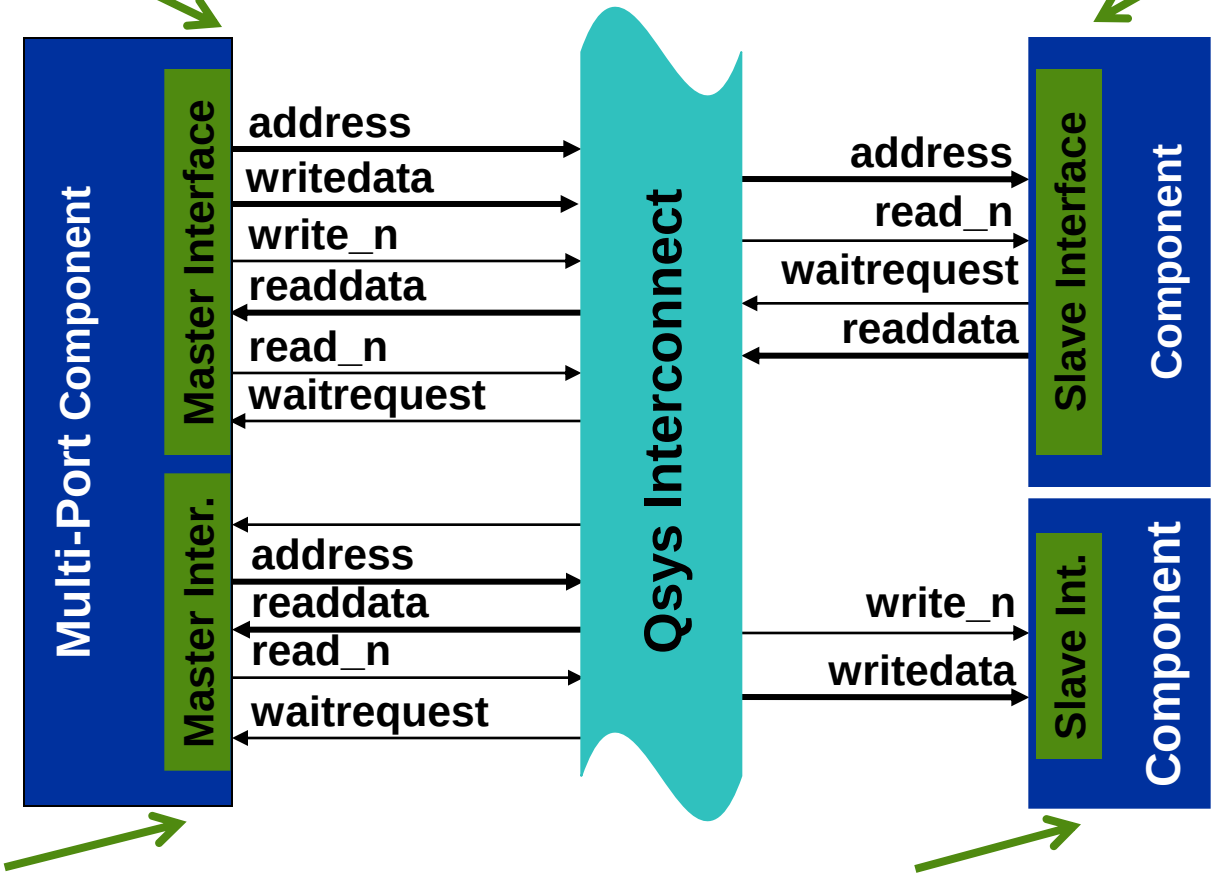
Терминология QSys

- Интерфейс
  - Группа сигналов Avalon-MM или Avalon-ST, определяющая связи между компонентами в системе QSys.
- Компонент (периферийное устройство)
  - Логическое устройство с одним или более интерфейсом Avalon
- Главное устройство (Master)
  - Интерфейс, инициирующий передачи по Avalon-MM
- Подчиненное устройство (Slave)
  - Интерфейс, отвечающий на передачи Avalon-MM
- Источник(Source) / Приемник (Sink)
  - Интерфейсы, которые посылают/получают потоковые данные в системе QSys
- Передача
  - Чтение или запись блока данных

Пример связи компонент Avalon-MM

Интерфейс главного устройства (Master) с поддержкой чтения и записи

Интерфейс подчиненного устройства (Slave) с переменной задержкой



Интерфейс главного устройства (Master) с поддержкой чтения

Интерфейс подчиненного устройства (Slave) с фиксированной или нулевой задержкой

Порты Avalon-MM Master

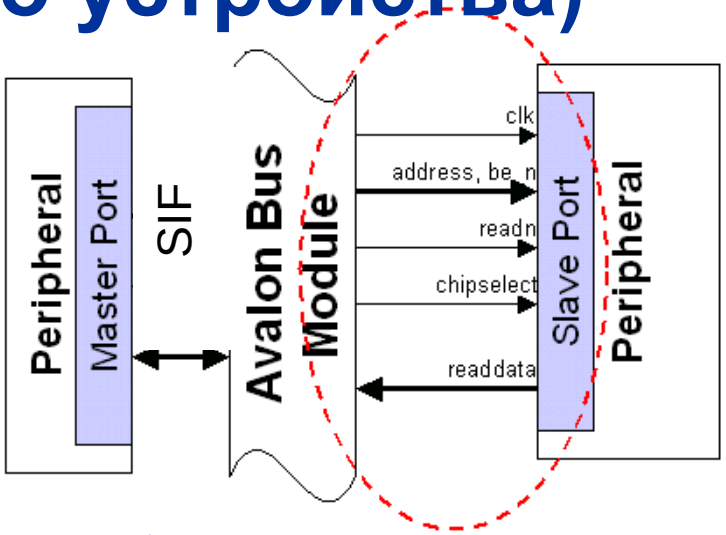
- Инициализируют передачи по системной шине
- Типы передач
  - Fundamental Read
  - Fundamental Write
- Avalon-MM master должен корректно обрабатывать сигнал **waitrequest**
- Характеристики передач
  - Поддержка задержки
  - Пакетные передачи
  - Управление потоком передачи

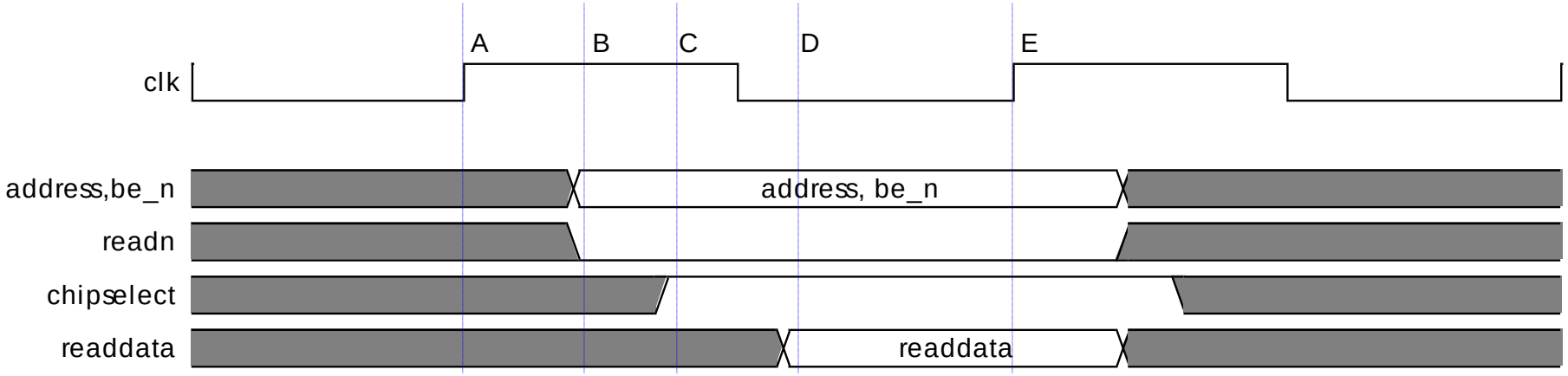
Порты Avalon-MM Slave

- Отвечают на запросы передач от системной шины
- Типы передач
  - Fundamental Read
  - Fundamental Write
- Характеристики передач
  - Циклы ожидания
  - Задержка
  - Пакетные передачи
  - Управление потоком передачи

Avalon-MM Slave Read Transfer (чтение из подчиненного устройства)

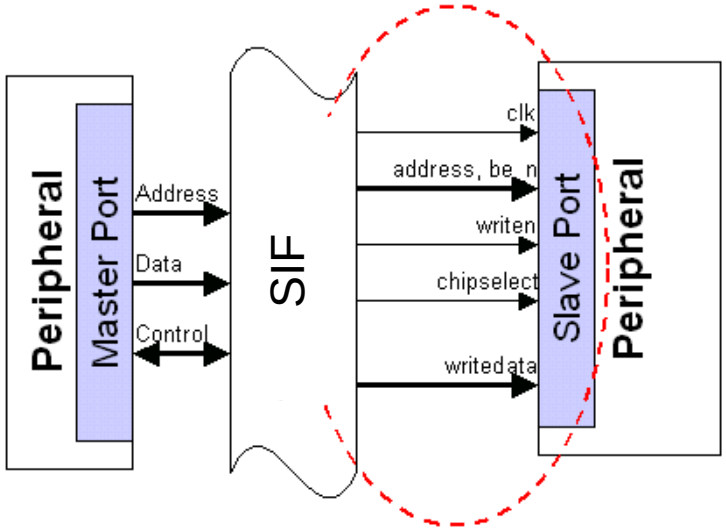
- 0 Setup Cycles
- 0 Wait Cycles

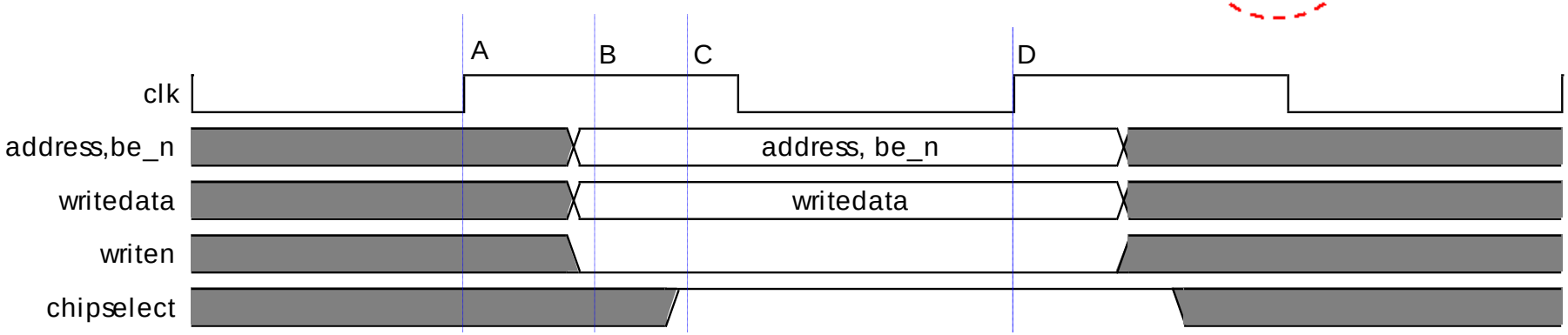




Avalon-MM Slave Write Transfer (запись в подчиненное устройство)

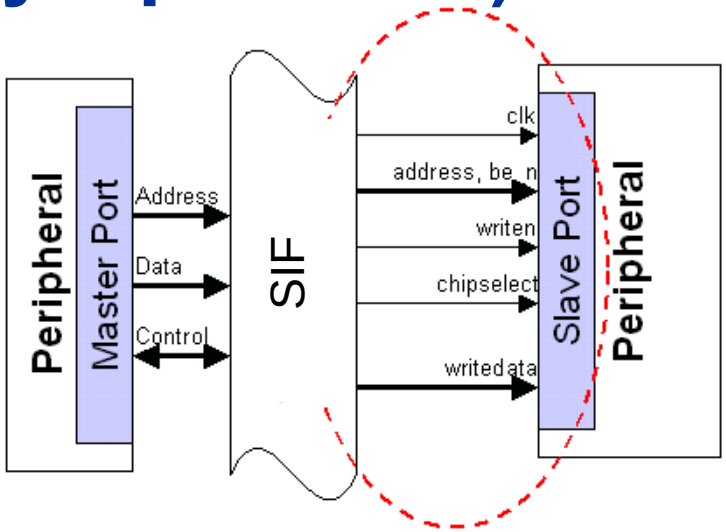
- 0 Setup Cycles
- 0 Wait Cycles
- 0 Hold Cycles

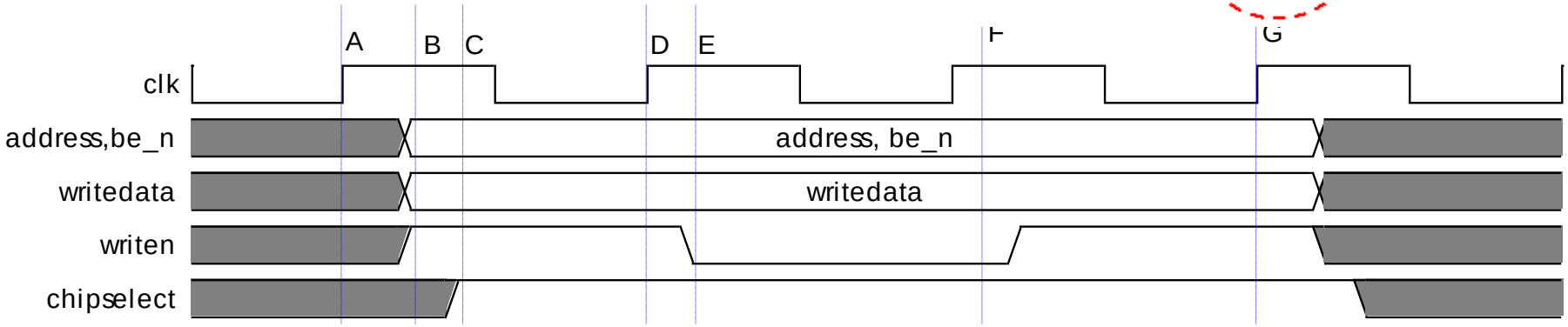




Avalon-MM Slave Write Transfer (запись в подчиненное устройство)

- 1 Setup Cycle
- 0 Wait Cycles
- 1 Hold Cycle

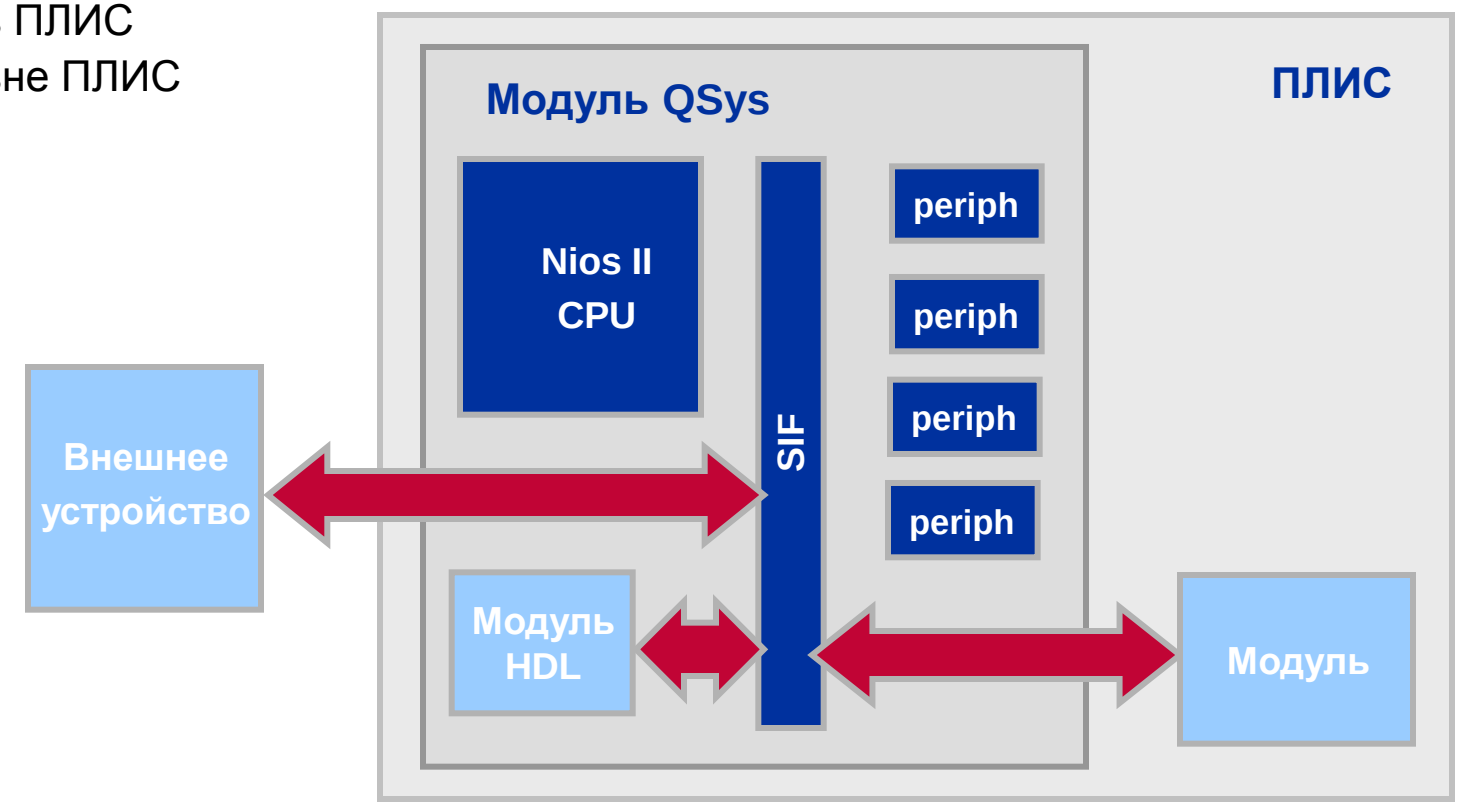




Пользовательские модули

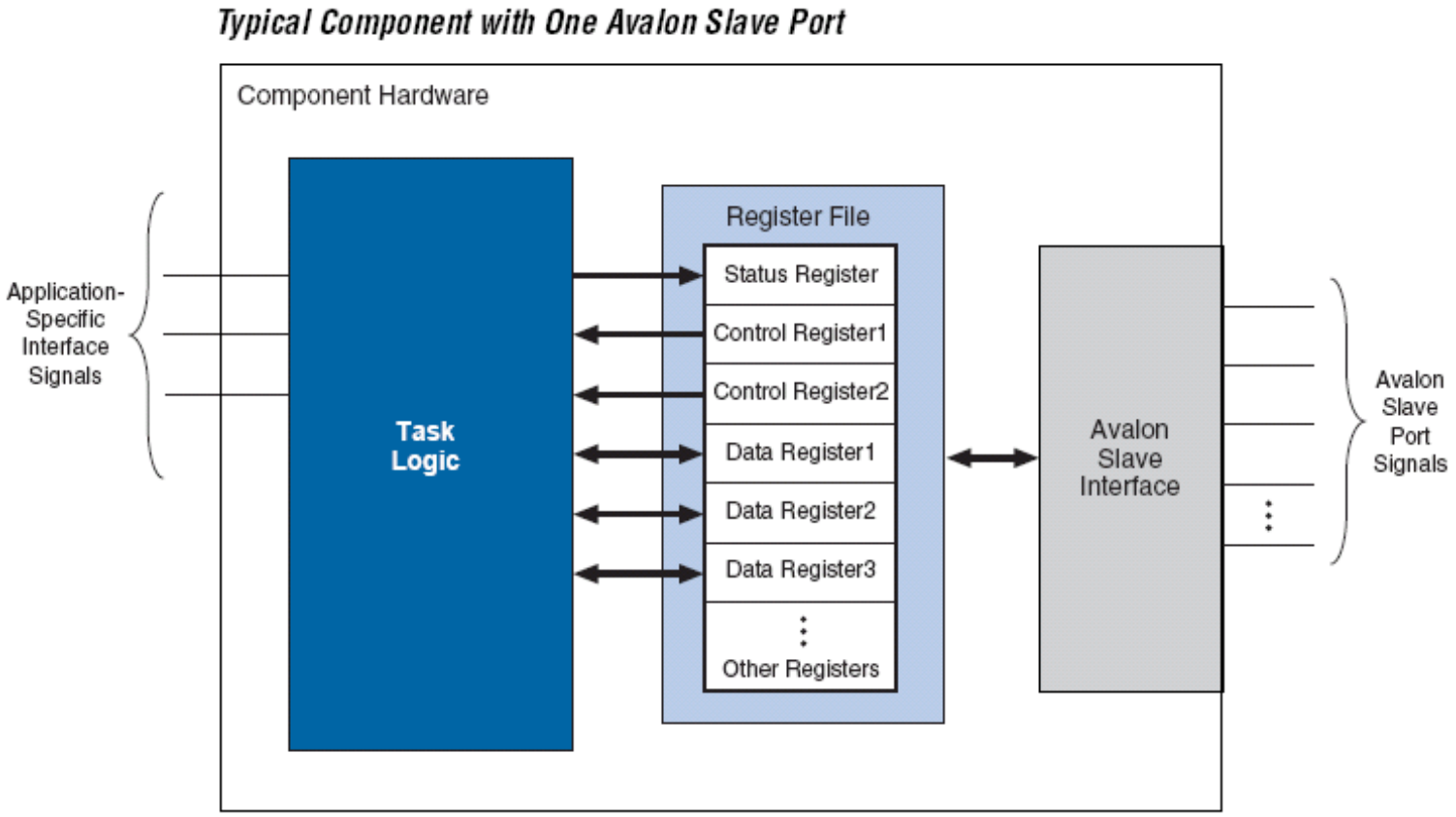
Пользовательские модули

- Отображаются в карту памяти процесора Nios II
- Могут реализовываться
  - в системе на кристалле
  - вне системы на кристалле
    - в ПЛИС
    - вне ПЛИС

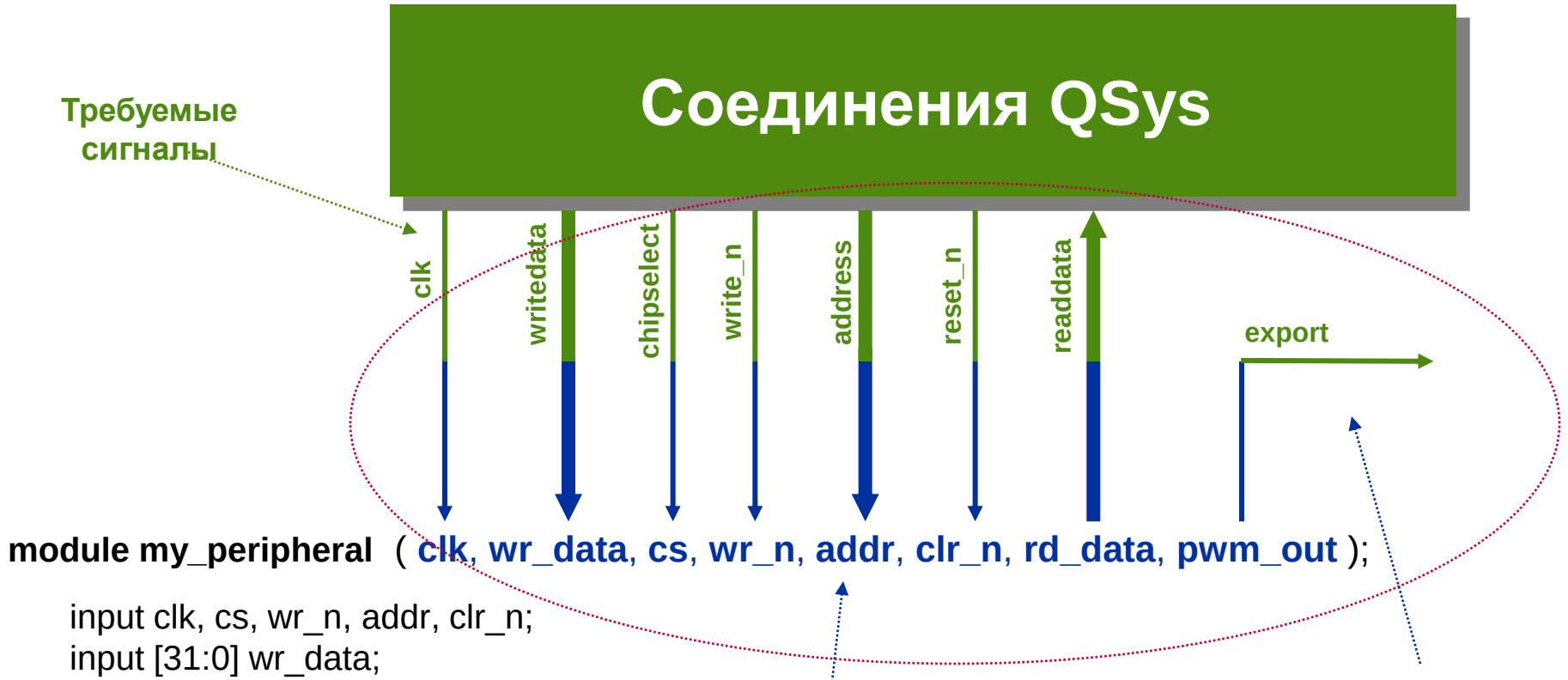


Пример регистрового Avalon-MM Slave

- Интерфейс системной шины реализуется в пользовательском модуле
- Временные характеристики интерфейса задаются при конфигурации модуля в QSys и реализуются шиной автоматически

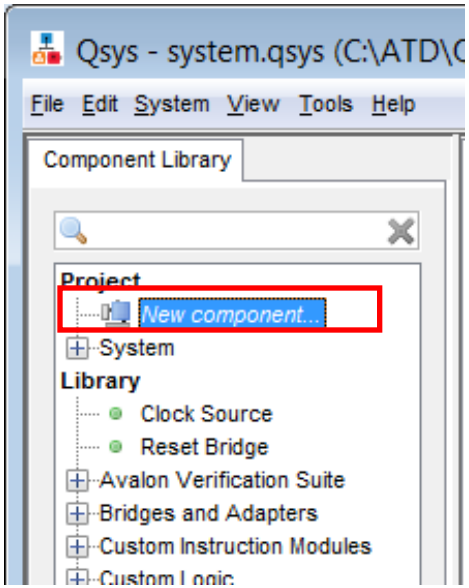


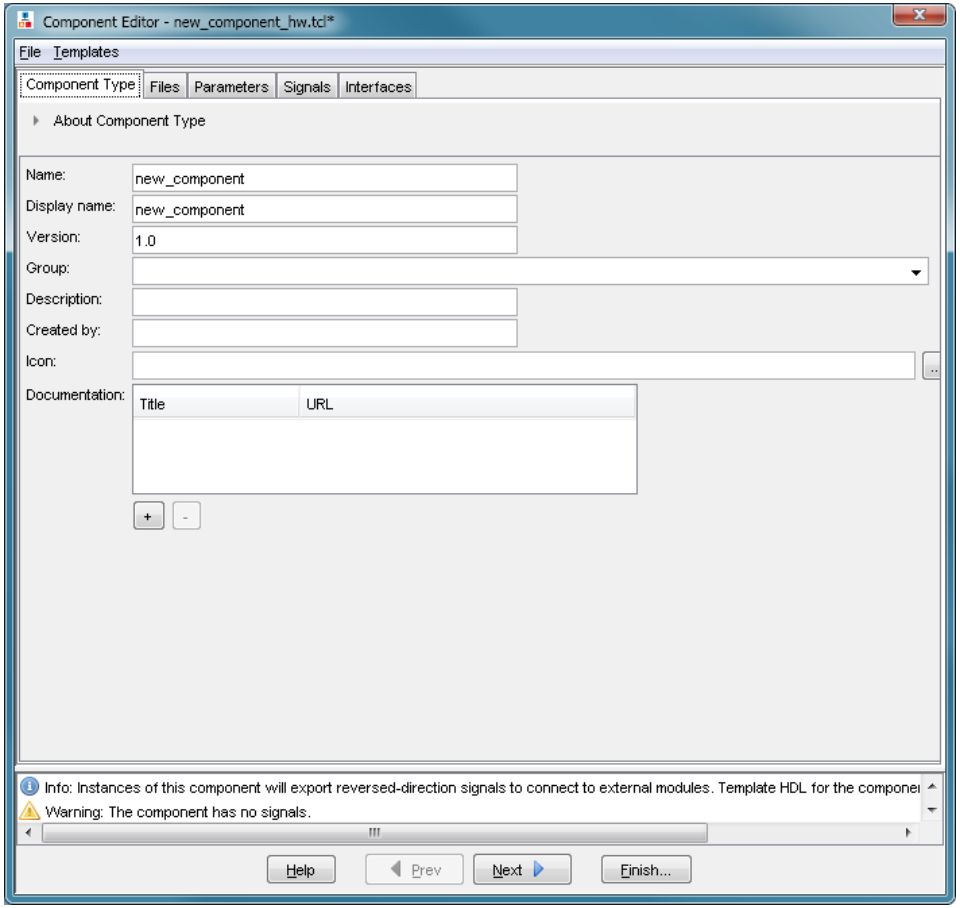
Связывание портов с сигналами шины



Редактор компонент

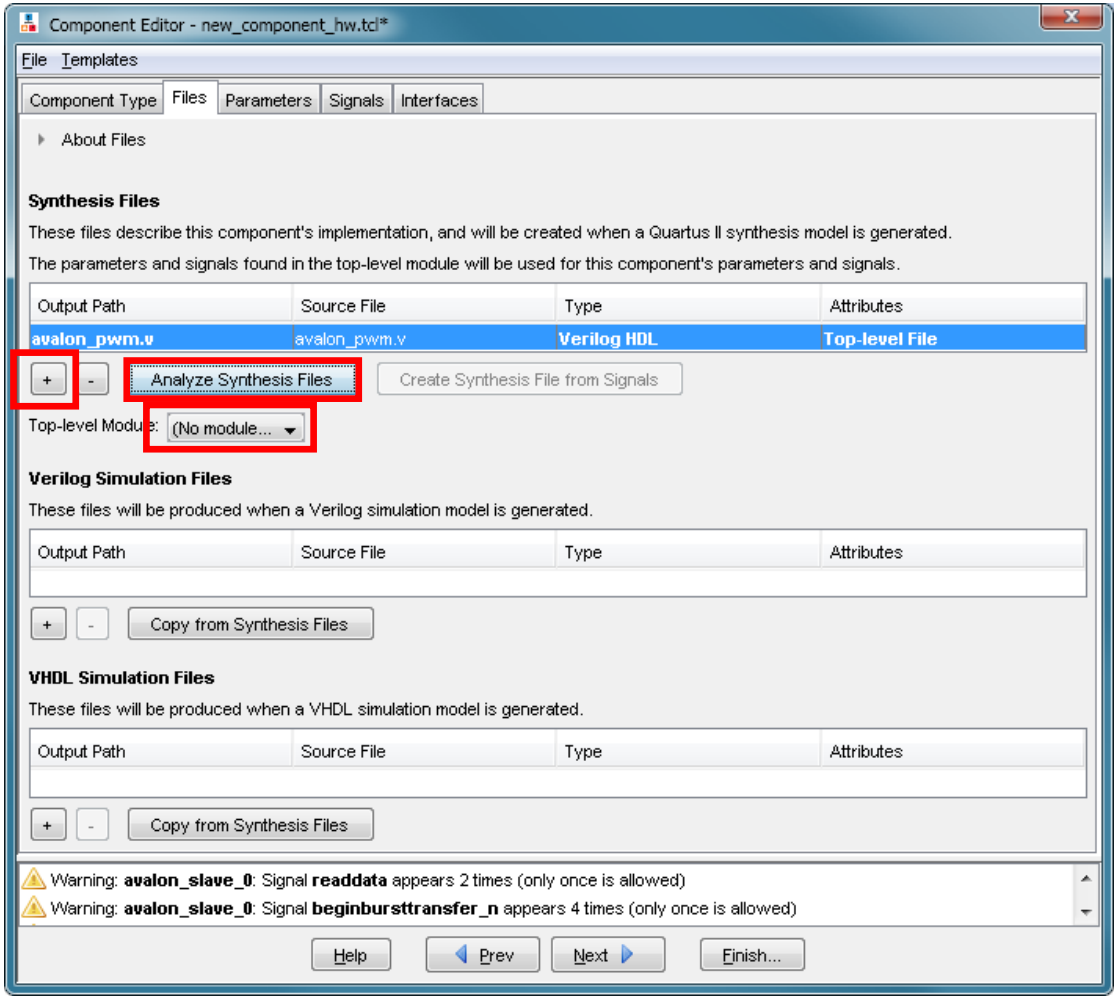
Используется для импорта модулей в состав библиотеки компонент системы на кристалле.





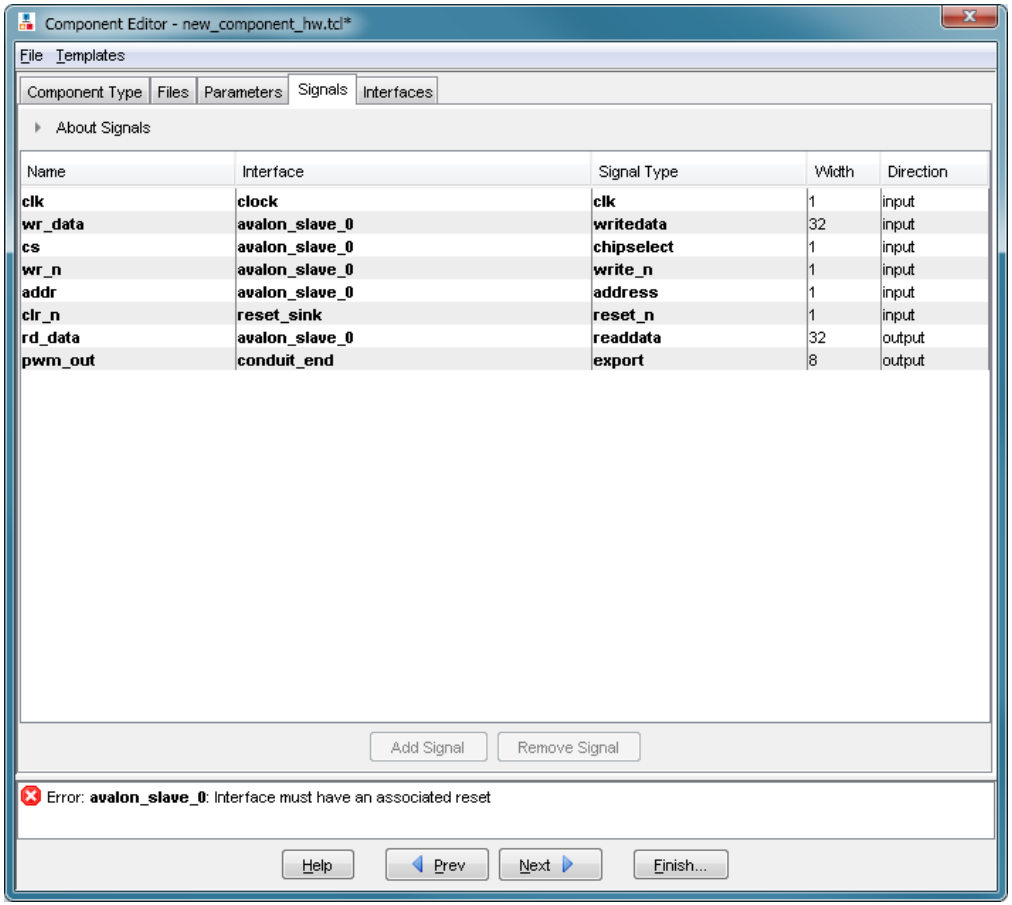
Закладка HDL Files

- Добавляет файлы на HDL в состав системы на кристалле
  - Нажмите кнопку +
  - Добавьте файлы
  - Нажмите Analyze…
  - Определите модуль верхнего уровня
- Использует Quartus II для анализа модуля и определения его интерфейса



Закладка Signals

- Определяет связывание сигналов с системной шиной
  - Столбец name заполняется автоматически при анализе исходного файла
  - Выберите тип интерфейса interface и укажите тип сигнала signal types в межсоединениях



Использование предопределенных префиксов упрощает связывание

В документации приведены префиксы и названия сигналов в интерфейсах

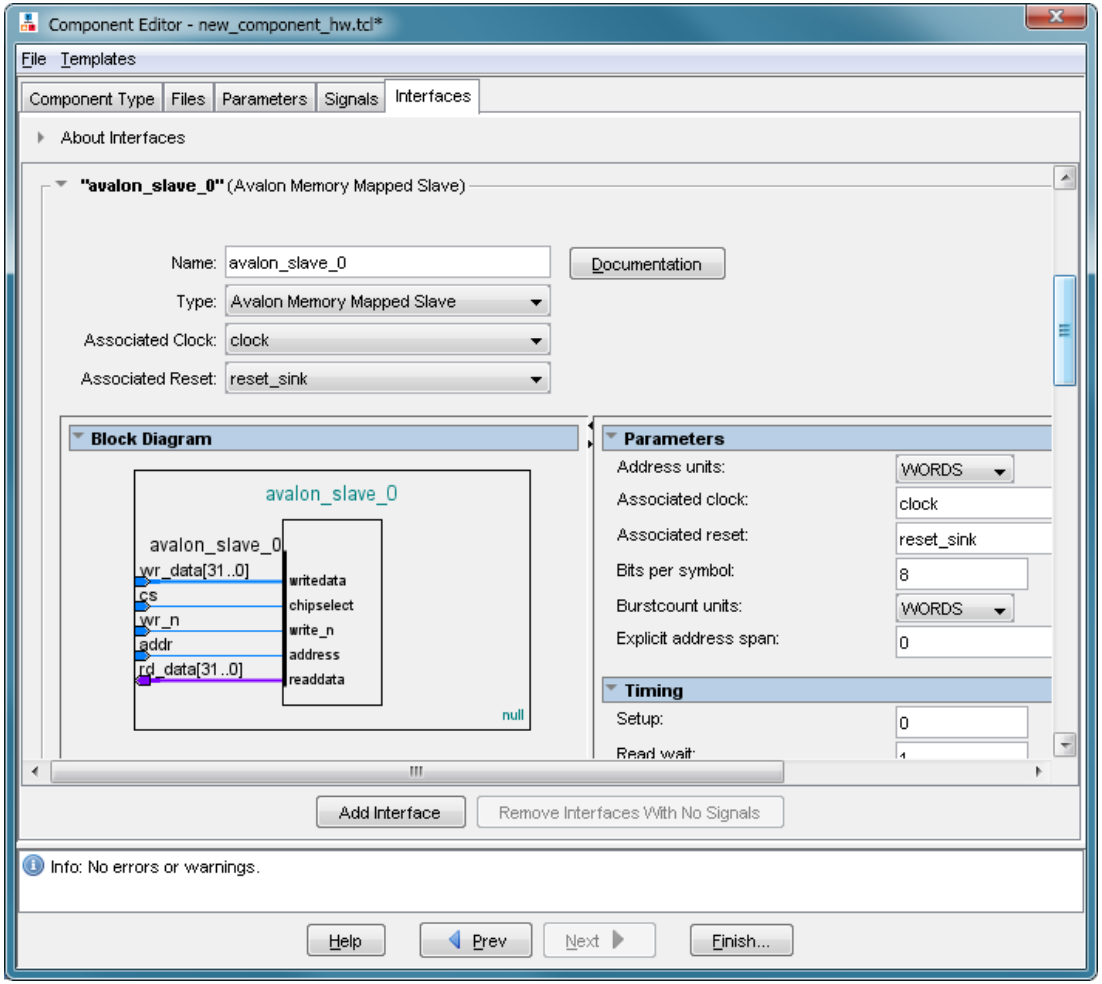
Пример:

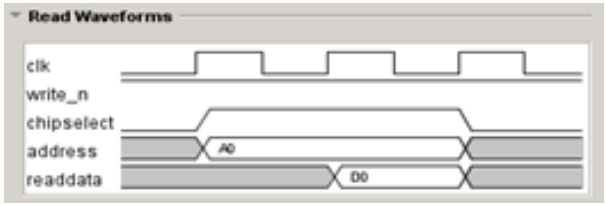
avs_s0_readdata отображается в сигнал *readdata* интерфейса Avalon-MM slave (*avs*) с именем *s0*

Value	Meaning
avs	Avalon-MM slave
avm	Avalon-MM master
aso	Avalon-ST source
asi	Avalon-ST sink
cso	Clock output
csi	Clock input
coe	Conduit
inr	Interrupt receiver
Ins	Interript sender
ncm	Nios II custom instruction master
ncs	Nios II custom instruction slave
rsi	Reset sink
rso	Reset source
tcm	Avalon-TC master
tcs	Avalon-TC slave

Закладка Interfaces

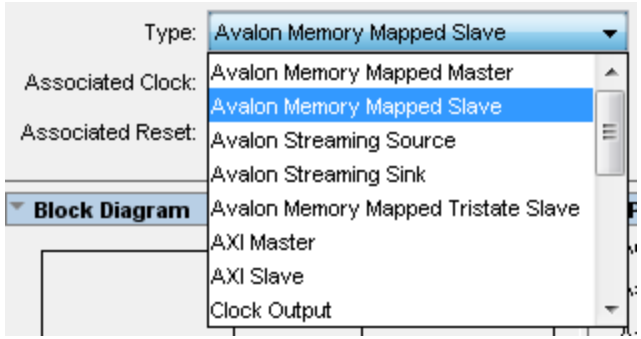
- Определяет характеристики всех интерфейсов компоненты
- Настраиваются связанные интерфейсы
- Задаются временные характеристики интерфейсов и отображаются временные диаграммы





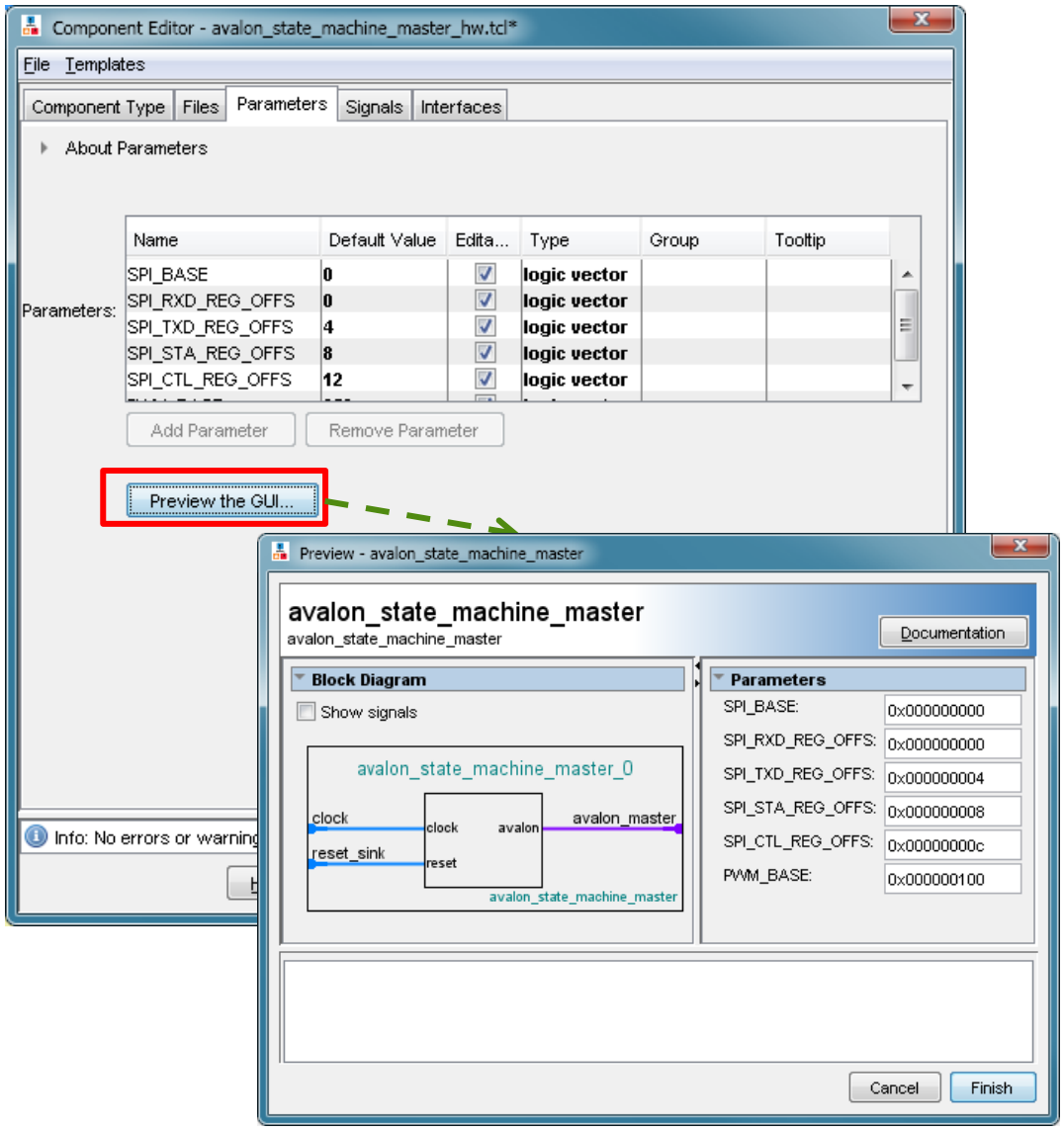
Интерфейсы компонент

- Clock (input и output)
  - Источники тактовых импульсов для других интерфейсов
- Reset (input и output)
  - Источники сброса для других интерфейсов
- Memory mapped master и slave
  - Порты Avalon-MM (требуют наличия интерфейсов clock input и reset input)
- Streaming source и sink
  - Порты Avalon-ST (требуют наличия интерфейса clock input)
- Conduit
  - Экспорт входных и выходных сигналов
- Interrupts (sender и receiver)
  - Поддержка сигналов прерывания в модулях
- Custom instruction
  - Интерфейсы специальных инструкций
- Также реализована пробная поддержка протокола ARM AMBA AXI



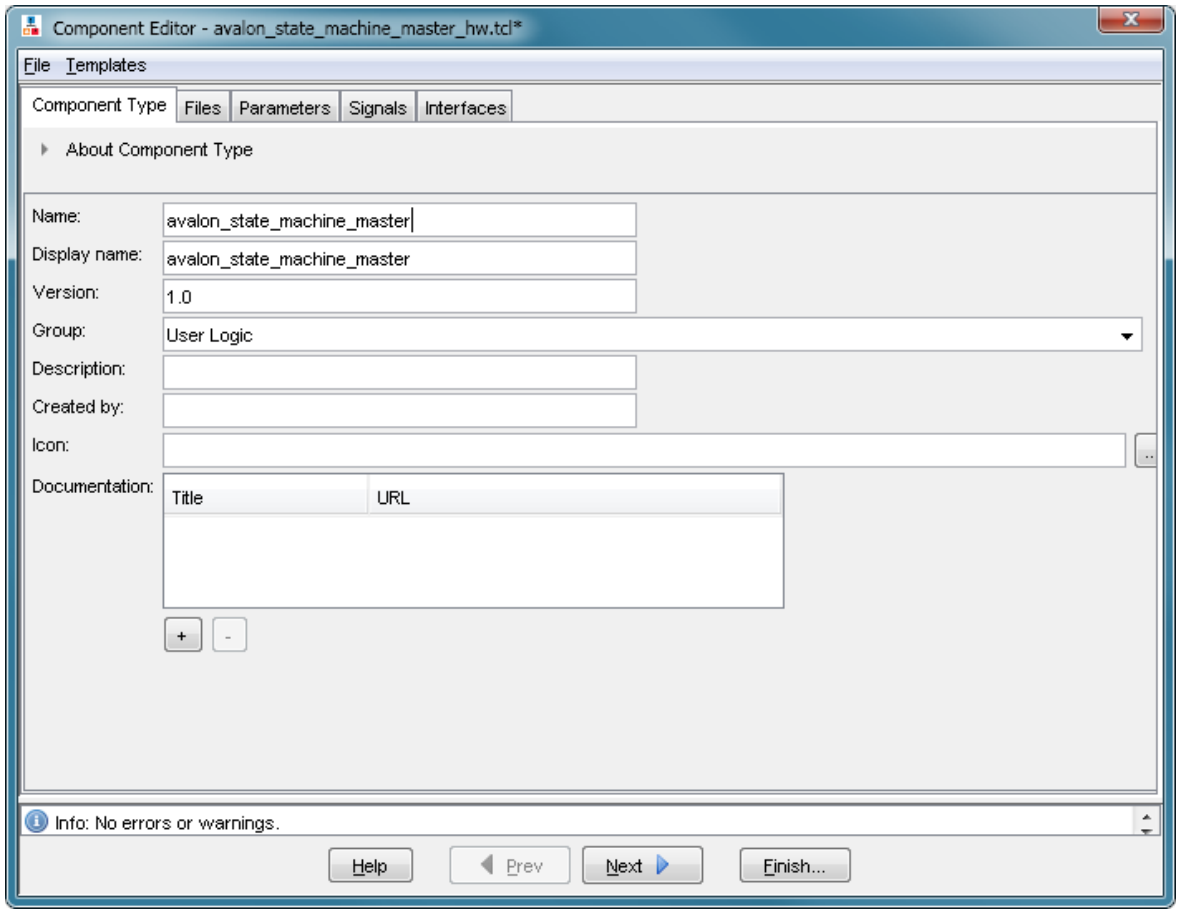
Закладка HDL Parameters

- Отображаются параметры компонента на HDL (если они определены в тексте модуля)
- Можно указать, какие параметры могут настраиваться при создании экземпляра в диалоге настройки



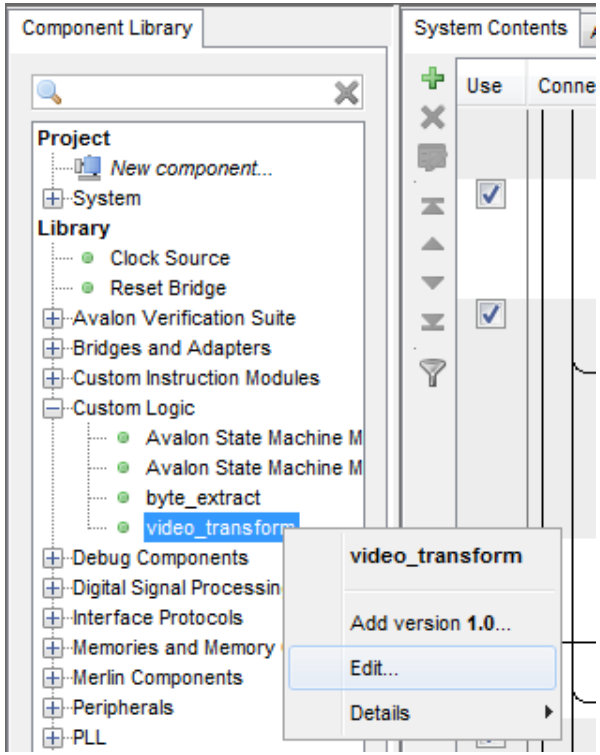
Закладка Library Info

- Задается имя компонента в библиотеке
- Определяется, к какой группе компонент будет приписан
  - Можно создать новую группу
- Можно добавить краткое описание , иконку и ссылку на документацию по компоненту



Для редактирования компонент

- Вызовите контекстное меню и выберите **Edit…**
- Или вызовите диалог создания компонента и вызовите в нем меню **File->Open**



- **После редактирования вызовите меню File > Refresh System для обновления списка компонент и соединений в системе**

Выходной файл редактора компонент

- Файл на Tcl
  - ***<top_level_module>_hw.tcl***
  - Находится в одном каталоге с исходными файлами компонент
- Может редактироваться вручную для изменения компонент
- Для удаления компоненты:
  - Удалите все экземпляры компоненты из проекта
  - Удалите файл <***top_level_module***>*_hw.tcl* в папке проекта
  - Выполните File->Refresh System для обновления системы и списка компонент

Создание компонент на Tcl

- Описание компоненты в виде скрипта Tcl предоставляет возможность автоматического редактирования и создания компонент без графического интерфейса пользователя
- Создание компонент на Tcl:
  - Создайте файл описания компоненты

    <top_level> _hw.tcl
  - Сохраните его в каталоге с кодом на HDL
  - Подробное описание в “Hardware Developer’s Handbook”, глава 6

Пример скрипта: (“avalon_pwm_hw.tcl”)

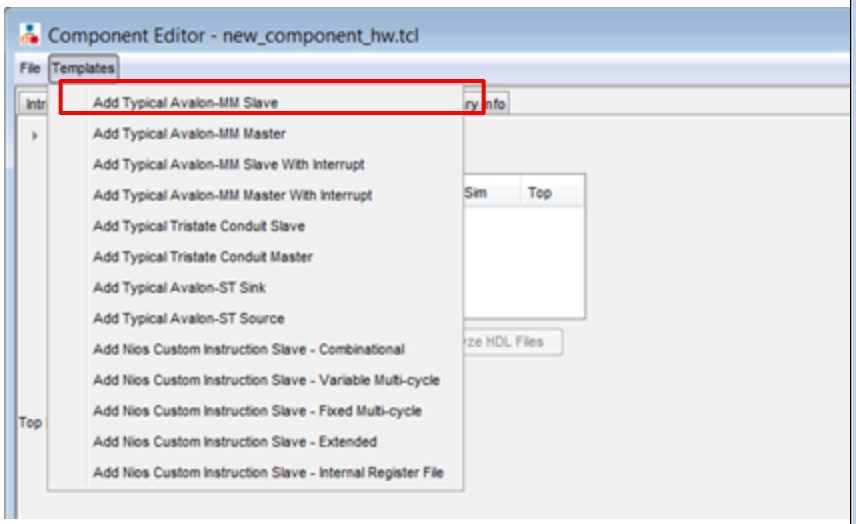
```
add_file                avalon_pwm.vhd {SYNTHESIS SIMULATION}
set_module_property     NAME           avalon_pwm
set_module_property     VERSION        1.0
set_module_property     GROUP          "User Logic"
set_module_property     DISPLAY_NAME   avalon_pwm
                              ⋮
```

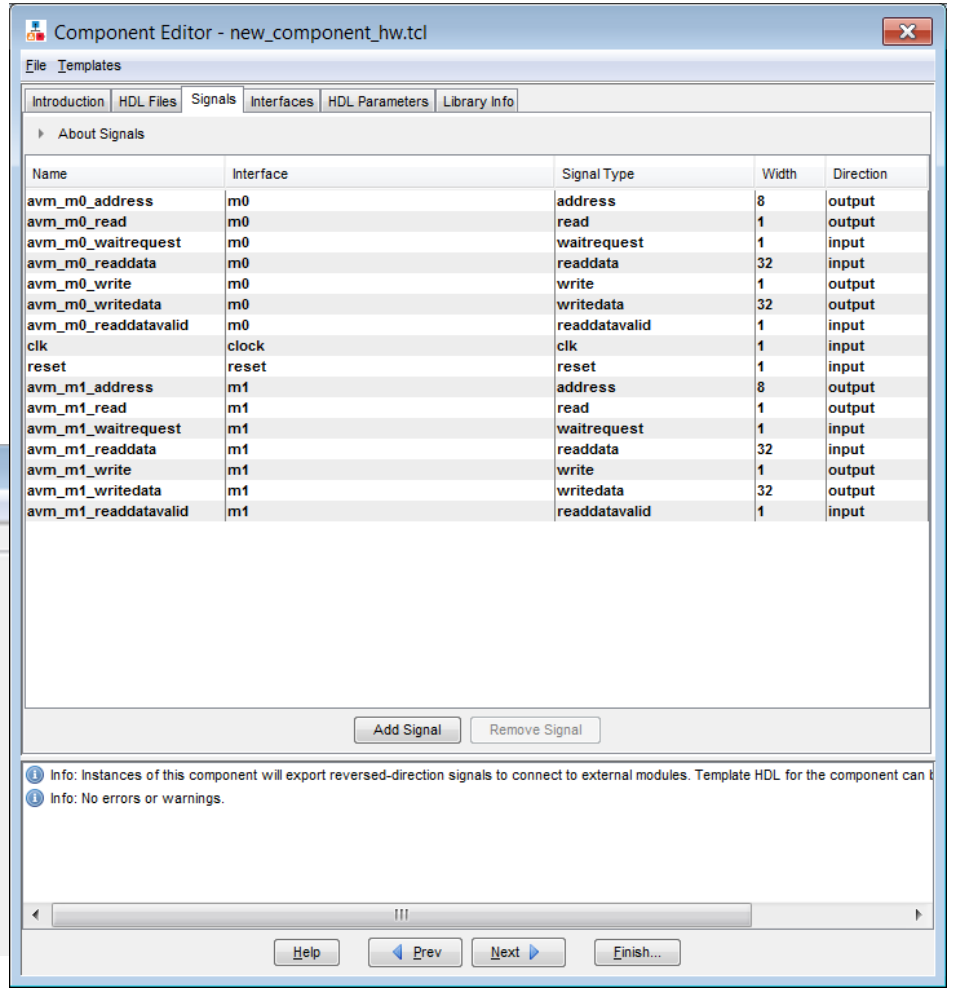
Пример описания интерфейса на Tcl

```
# | connection point avalon_slave_0
add_interface           avalon_slave_0  avalon end
set_interface_property  avalon_slave_0  holdTime 0
set_interface_property  avalon_slave_0  linewrapBursts false
set_interface_property  avalon_slave_0  minimumUninterruptedRunLength 1
set_interface_property  avalon_slave_0  bridgesToMaster ""
set_interface_property  avalon_slave_0  isMemoryDevice false
set_interface_property  avalon_slave_0  burstOnBurstBoundariesOnly false
set_interface_property  avalon_slave_0  addressSpan 4
set_interface_property  avalon_slave_0  timingUnits Cycles
set_interface_property  avalon_slave_0  setupTime 0
set_interface_property  avalon_slave_0  writeWaitTime 0
set_interface_property  avalon_slave_0  readWaitStates 0
set_interface_property  avalon_slave_0  maximumPendingReadTransactions 0
set_interface_property  avalon_slave_0  readWaitTime 0
set_interface_property  avalon_slave_0  readLatency 0
set_interface_property  avalon_slave_0  ASSOCIATED_CLOCK clock_reset
add_interface_port      avalon_slave_0  wr_data writedata Input 32
add_interface_port      avalon_slave_0  cs chipselect Input 1
add_interface_port      avalon_slave_0  wr_n write_n Input 1
add_interface_port      avalon_slave_0  addr address Input 1
add_interface_port      avalon_slave_0  rd_data readdata Output 32
                                .
                                .
                                .
```

Шаблоны интерфейсов

- Добавление шаблона интерфейса в список сигналов
- Реализованы шаблоны для разных интерфейсов
- **Пример**
  - **Avalon MM Slave**





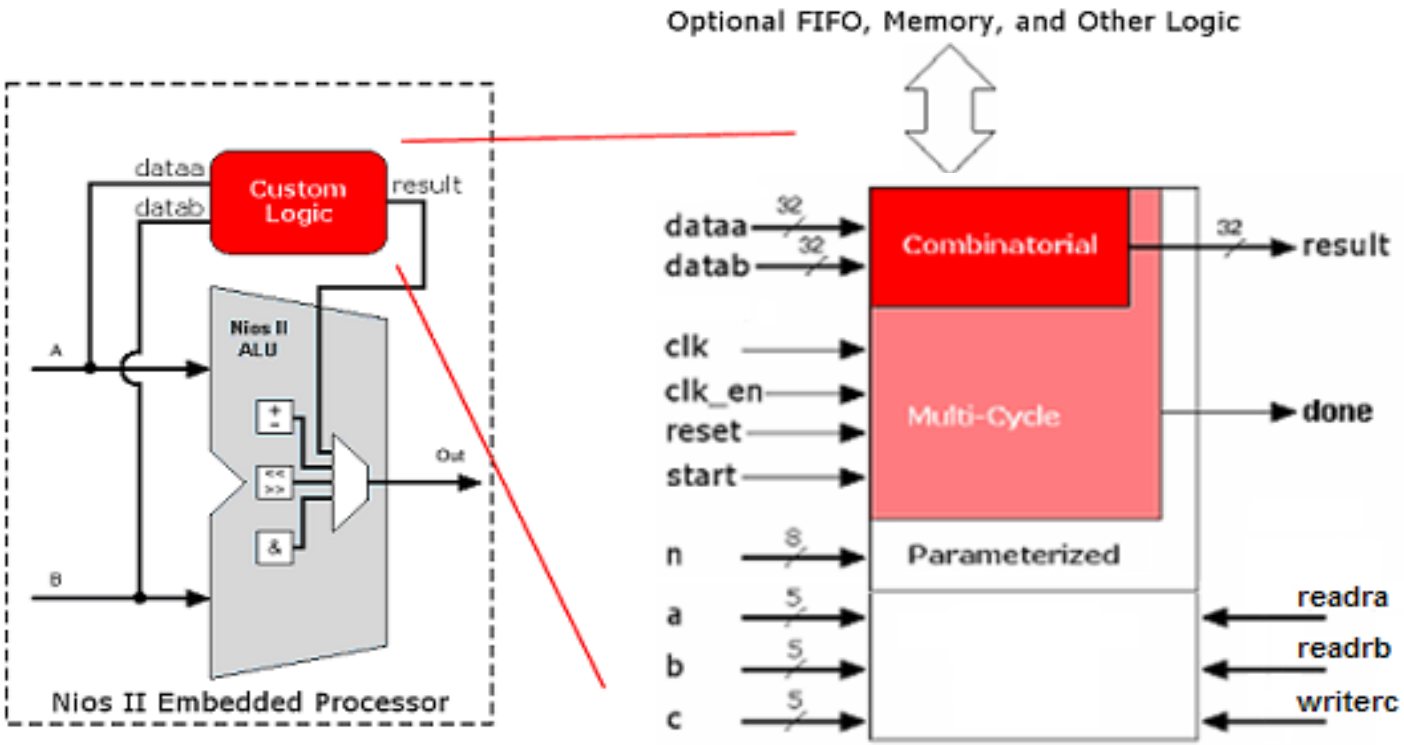
Специальные инструкции

Custom Instructions

- Специальные инструкции
- Добавляют функциональные возможности в проект на основе Nios II
  - Используется возможность модифицировать и синтезировать ядро процессора для конкретного проекта на ПЛИС
- Могут значительно повышать производительность
- Примеры применения
  - Операции надо потоком данных (например, сетевые приложения)
  - Специализированные процессоры (например, декодирование MP3)
  - Оптимизация ядра циклов

Специальные инструкции

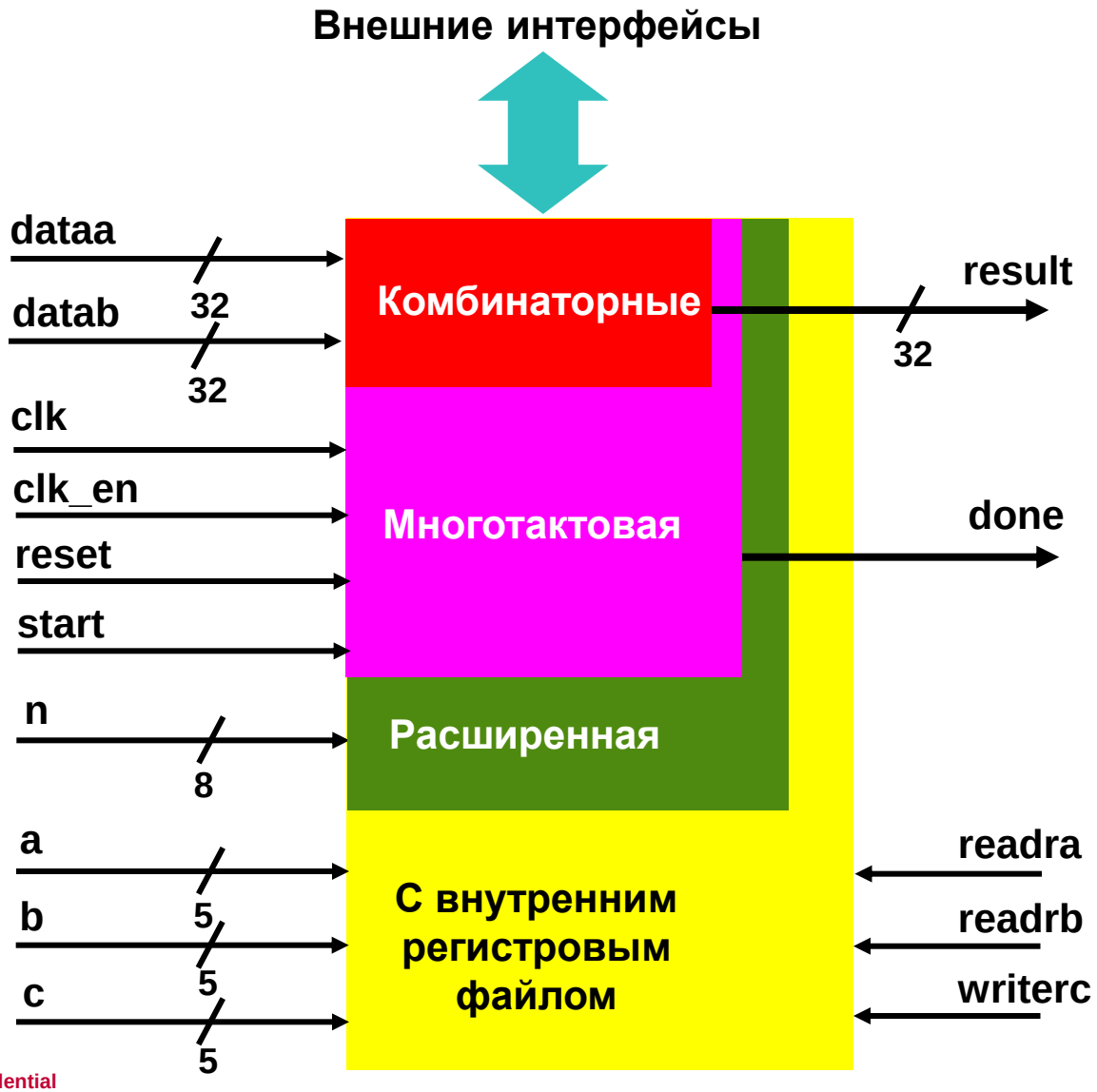
- Расширяют систему команд процессора Nios II
  - Добавляют логику в АЛУ



Специальные инструкции

- Поддержка интегрирована в маршрут проектирования систем на кристалле на основе Nios II
  - QSys автоматически назначает код команды
  - Генерируются макросы для вызова на C и ассемблере
  - До 256 различных инструкций (кодов команд)
  - Инструкции могут выполняться несколько тактов
    - Многотактовые инструкции останавливают конвейер процессора и могут снижать производительность
  - Фиксированная и переменная длительность выполнения

Типы специальных инструкций



Комбинаторные специальные инструкции

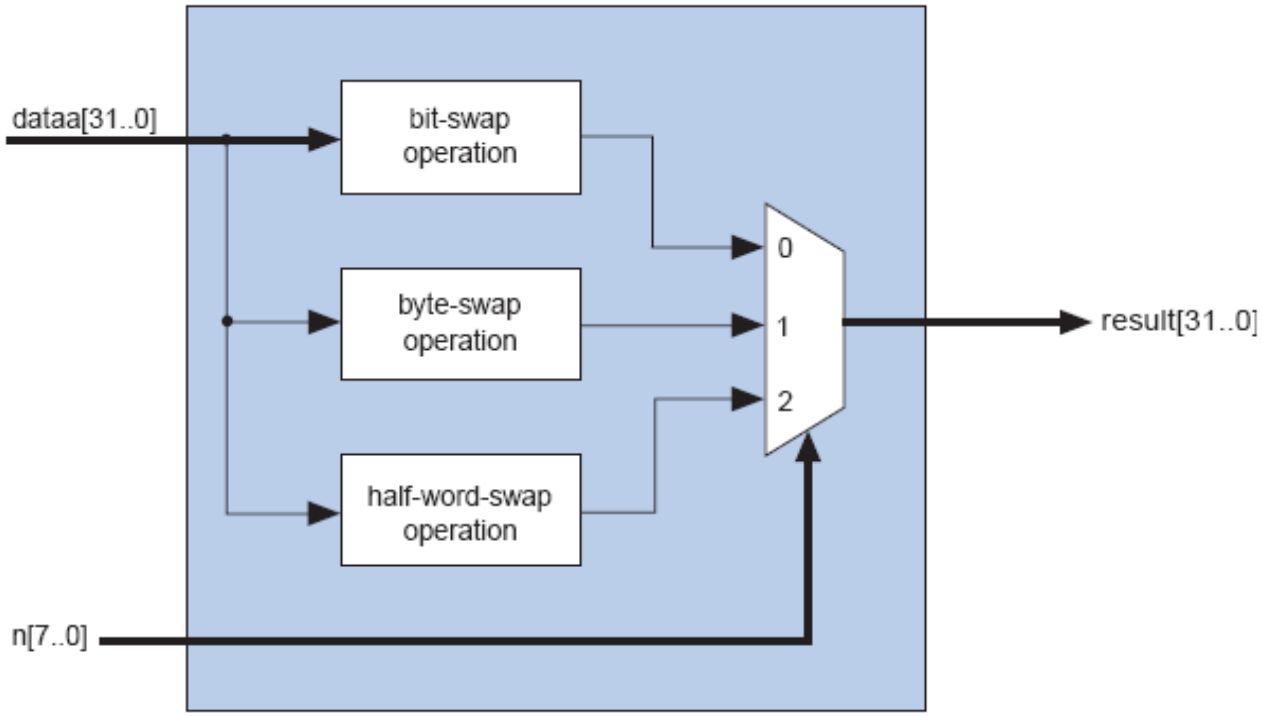
- Список портов
  - Входы dataa и datab, выход result

Table 1–2. Combinatorial Custom Instruction Signals

Signal Name	Direction	Required	Purpose
dataa[31..0]	Input	No	Input Operand to custom instruction
datab[31..0]	Input	No	Input Operand to custom instruction
result[31..0]	Output	Yes	Result from custom instruction

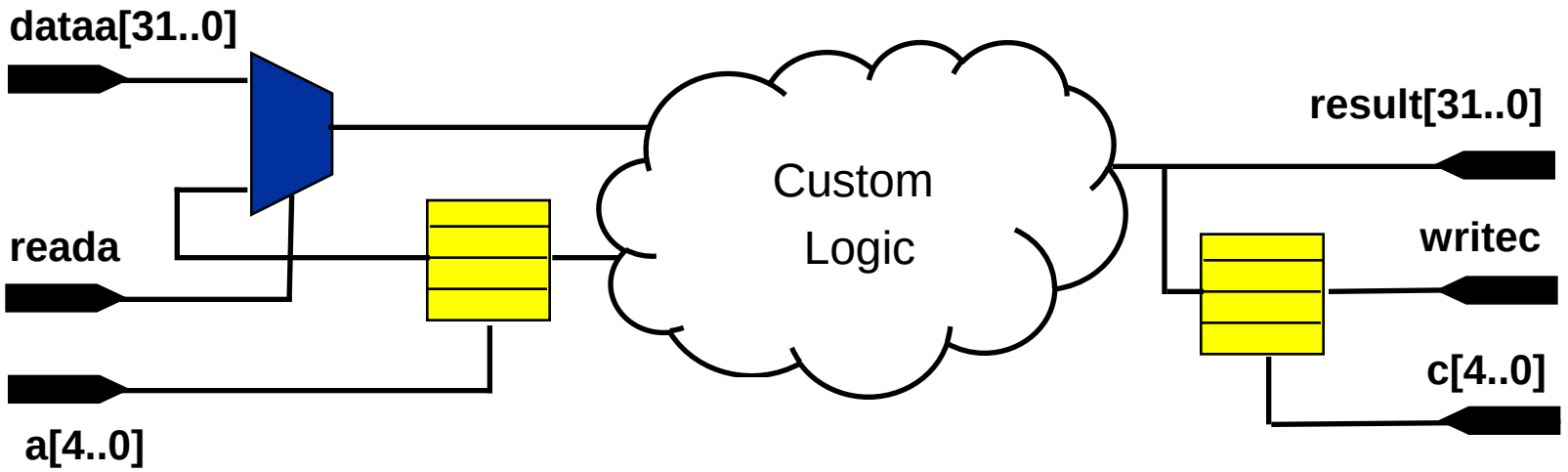
Расширенные специальные инструкции

- Порт n[7..0] определяет тип выполняемой операции
- Один модуль может реализовывать подмножество из 256 доступных кодов специальных инструкций



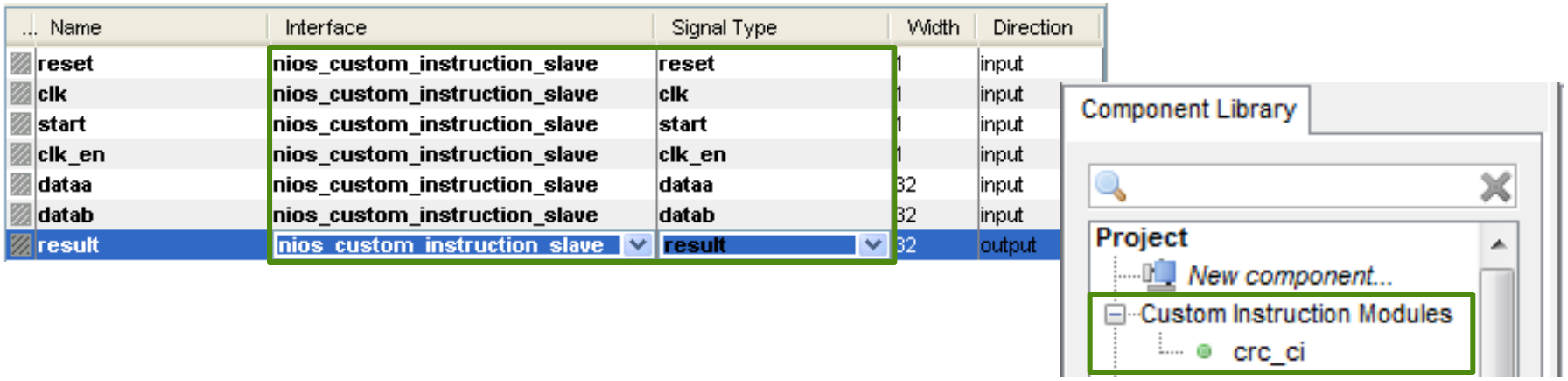
Инструкции с регистровым файлом

- В качестве операндов выбираются входы данных dataa/datab или внутренние регистры a[]]/b[]. Выбор осуществляется сигналами reada и readb.
- Если сигнал writec активен, результат записывается во внутренний регистр c[]



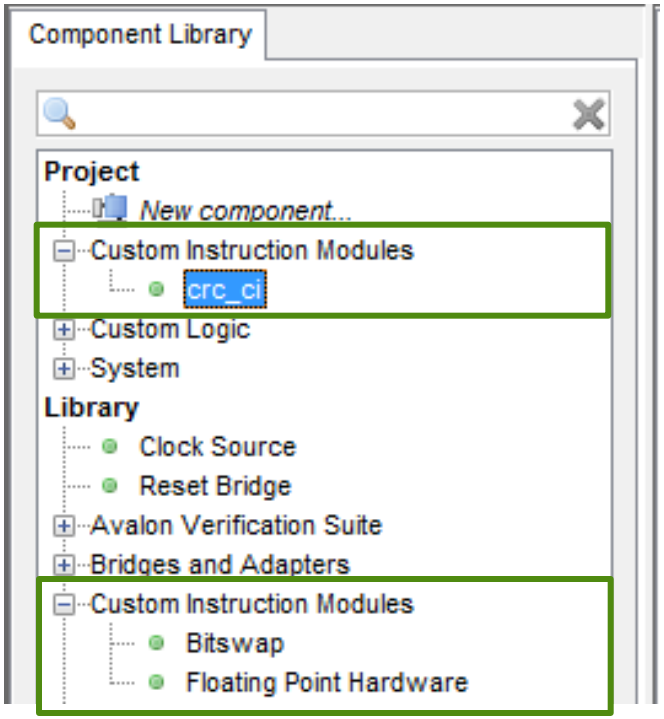
Импорт специальных инструкций

- В редакторе компонент
  - Импортируйте HDL код
  - Свяжите сигналы с интерфейсом Custom Instruction Slave
  - Все сигналы, включая ТИ и сброс, относятся к этому интерфейсу
  - Для сохранения выберите папку Custom Instruction Modules в списке Library
- Выполните File->Refresh System для обновления системы и списка компонент

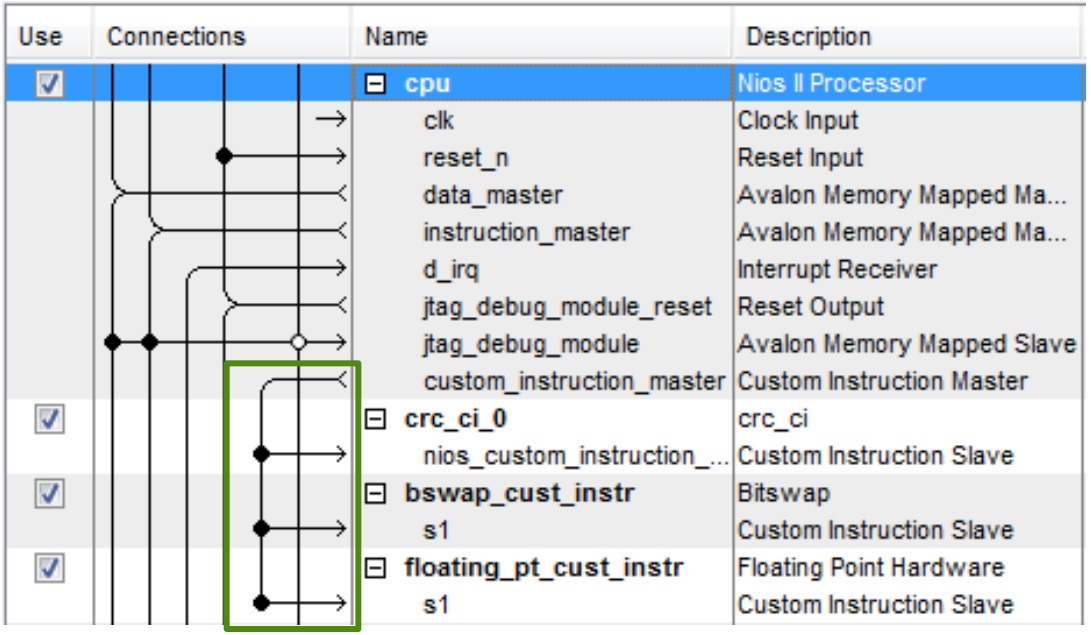


Добавление специальных инструкций

- Добавьте нужные специальные инструкции из библиотеки компонент
- Соедините их с портом процессора custom_instruction_master



Can rename them like any other component



Удаление специальных инструкций

- Удалите инструкцию из списка задействованных компонент на странице System Contents page
- Удалите файл “`<custom_instruction>_hw.tcl`” из папки проекта

Интерфейс C

- NIOS II SBT автоматически генерирует макросы при построении пакета поддержки платы

- Макросы определяются в файле **system.h**

  #define ALT_CI_<your instruction_name>(instruction arguments)

- Пример кода, использующего специальную инструкцию:

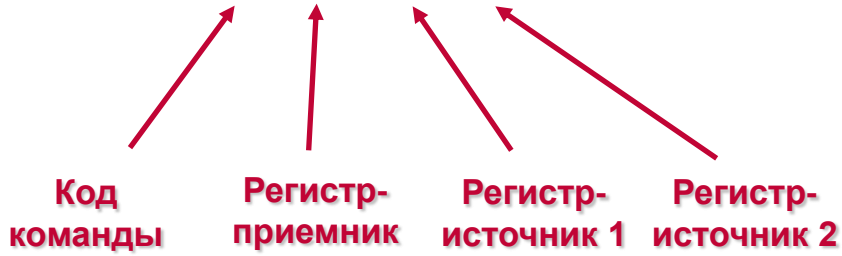
```
#include "system.h"
int main (void)
{
    int a = 0x12345678;
    int a_swap = 0;

    a_swap = ALT_CI_BSWAP(a);
    return 0;
}
```

Интерфейс ассемблера

- Синтаксис для специальной инструкции:

custom N, rC, rA, rB



- Примеры:

custom 0, r6, r7, r8
custom 3, c1, r2, c4

r = Обращение к регистру ядра Nios II

c = Обращение к внутреннему регистровому файлу

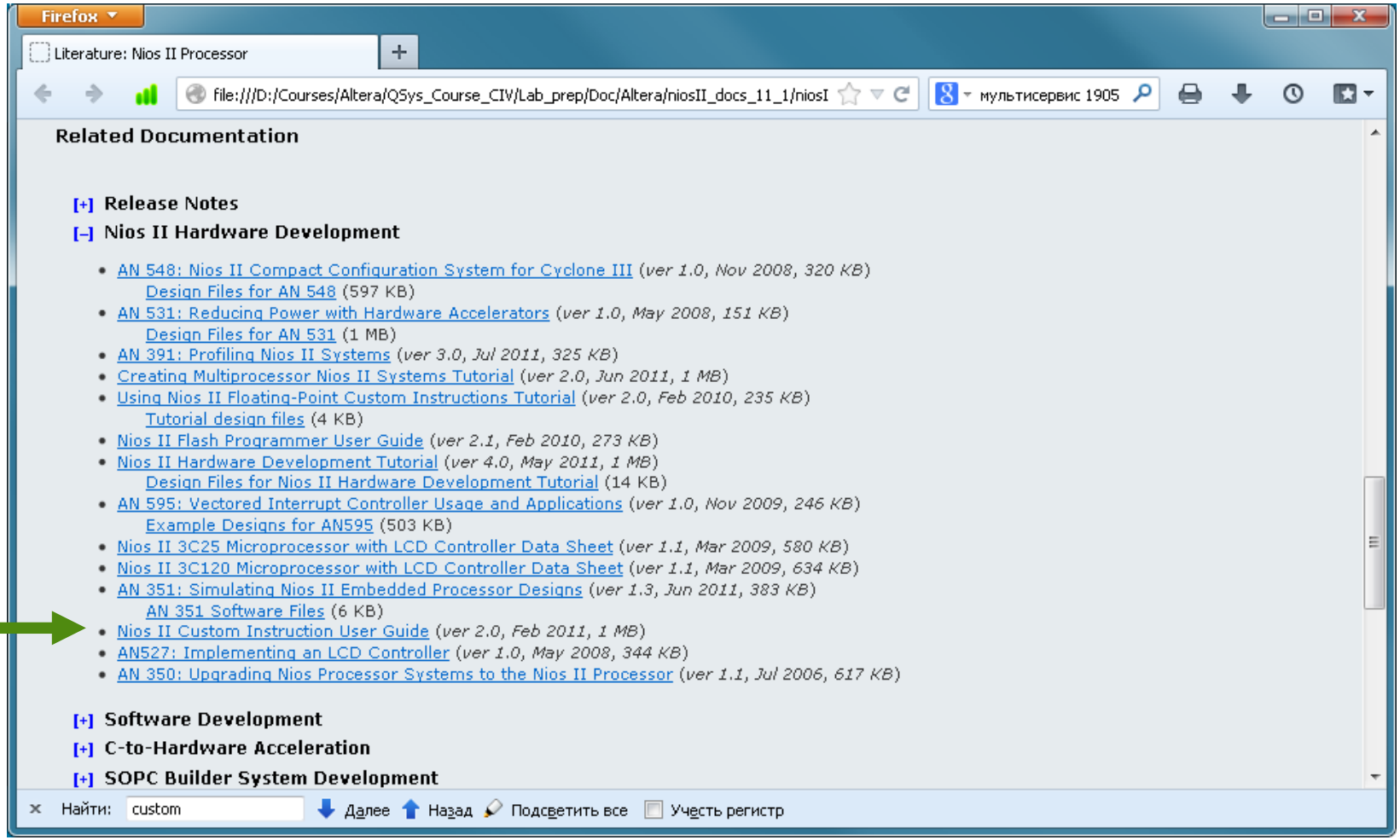
Применение специальных инструкций

- Сократить сложную последовательность инструкций до одной или нескольких инструкций
- Пример:
  - Умножение с плавающей запятой (6 тактов)
  - Примерно в 30 раз быстрее программной реализации
- Рекомендуемый маршрут
  - Осуществите профилирование кода
  - Идентифицируйте критичные по быстродействию циклы
  - Реализуйте логику специальных инструкций, полностью или частично заменяющую логику цикла
  - Импортируйте инструкцию в проект
  - Реализуйте вызов инструкций из C или ассемблера

Floating Point Custom Instructions

- Инструкции с плавающей запятой
- Реализуют операции в формате single
- Доступны в любой версии ядра Nios II
  - Включают сложение, вычитание, умножение и деление
  - Расширяют стандартный набор команд
- Автоматически задействуются компилятором для операций +, -, *, / над переменными типа single
- Для отключения автоматического задействования можно использовать следующие макросы
  - **+** #pragma no_custom_fadds
  - **-** #pragma no_custom_fsubs
  - ***** #pragma no_custom_fmuls
  - **/** #pragma no_custom_fdivs

Nios II Custom Instruction User Guide



http://www.altera.com/literature/ug/ug_nios2_custom_instruction.pdf