Продолжение, начало в № 2'2002

Микроконтроллер для встроенного применения – NIOS.

Конфигурация шины и периферии

Иосиф Каршенбойм

ik@lmail.loniis.ru

4. Соглашения в разделах и назначениях ptf

Синтаксис ptf, определенный в разделе 3, имеет общий иерархический формат базы данных и в принципе может хранить любой вид информации.

Конечно, программное обеспечение SOPC Builder software использует некоторые соглашения для обозначения и использования элементов ptf, которые определяют системный модуль. Поэтому в системе ptf-файла одни разделы и назначения обязательны, другие опциональны, а третьи игнорируются.

4.1. Раздел SYSTEM

Вся система ptf-файлов должна иметь раздел типа SYSTEM. Этот раздел должен быть назван так же, как и модуль верхнего уровня, который генерируется SOPC Builder software (см. блок «Avalon System» на рис. 13). Все элементы ptf вне системного раздела игнорируются программой SOPC Builder software. Системный раздел должен иметь имя, которое должно соответствовать принятым именам в Verilog, VHDL и AHDL.

4.1.1. Содержимое раздела SYSTEM

Раздел SYSTEM может содержать любое количество элементов ptf, но программное обеспечение SOPC Builder software опознает только следующие:

```
WIZARD_SCRIPT_ARGUMENTS
MODULE <module_name1>
MODULE <module_name2>
...
```

Любой другой тип раздела и любые другие назначения вне этих разделов игнорируются.

Раздел SYSTEM может иметь только один подраздел типа WIZARD_SCRIPT_ARGUMENTS и произвольное число разделов MODULE. Каждый раздел MO-DULE должен иметь имя, соответствующее принятым именам в Verilog, VHDL, и AHDL. Имя каждого раздела MODULE будет использоваться программой SOPC Builder software как формальное HDL-имя модуля. Назначения, указанные пользователем и сделанные в разделе WIZARD_SCRIPT_ ARGUMENTS с помощью системного составителя программы MegaWizard, передаются в параметры настройки системного уровня.

Каждый раздел SYSTEM/MODULE формально связан с hdl-модулем в конечной системе. Для примера: на рис. 13 раздел SYSTEM имел бы раздел MODULE для Peripheral #1, Peripheral #2, Master и System-External Interface. Обратите внимание, что модуль шины Avalon не имеет раздела MODULE в ptf-файле, так как он полностью определен объединением всех элементов в разделе SYSTEM. Описание всех разделов и назначений, принятых в пределах раздела MODULE, приводится в следующем разделе этого документа (п. 4.2).

4.2. Разделы MODULE

Каждое устройство — и slave, и master — в системе Avalon будет иметь раздел MODULE в пределах подраздела SYSTEM ptf-файла. Этот раздел MODULE будет содержать больше всего необходимой для генерации логики этого устройства информации. Эта глава описывает разделы и назначения в пределах раздела SYSTEM/MODULE:

class	(assignment)
PORT_WIRING	(section, name ignored)
SYSTEM_BUILDER_INFO	(section, name ignored)
WIZARD_SCRIPT_ARGUMENTS	(conventional, but not required)

Раздел MODULE может содержать или не содержать другие разделы и назначения. Дополнительные разделы и назначения могли бы использоваться, например, программой-генератором для ее собственных целей или мастером, связанным с MODULE. Любые другие назначения или разделы, отличные от перечисленных выше (или описанных ниже), игнорируются.

4.2.1. Назначение MODULE/CLASS

Каждый модуль является членом класса модулей. Каждый раздел MODULE должен иметь назначение — «класс», и оно должно быть библиотечным компонентом для Excalibur. Имя класса всегда совпадает с именем каталога, который содержит MODULE файл Class.ptf. Программа SOPC Builder software пользуется назначением MODULE/CLASS, чтобы найти все библиотечные файлы, программы генератора и т. д., требуемые для построения MODULE.

4.2.2. Раздел MODULE/PORT_WIRING

Каждый раздел MODULE должен «сообщить» программе SOPC Builder software обо всех его интерфейсных сигналах (о портах). Информация обо всех портах ввода-вывода модуля содержатся в разделе MODULE/PORT_WIRING. Содержание этого раздела подробно описано в разделе 5.

Пока модуль не будет окончательно сгенерирован, точный список его портов не всегда известен, и SOPC Builder software не использует раздел PORT_WIRING до тех пор, пока не будет выполнена генерация модулей.

4.2.3. Раздел MODULE/SYSTEM_BUILDER_INFO

В дополнение к портам, программа SOPC Builder software должна иметь и множество другой информации о модуле, чтобы должным образом подключить его к сигналам шины. Эта дополнительная информация дается в виде назначений в разделе MODULE/SYSTEM_BUILDER_INFO. Этот раздел далее сокращенно называется SBI. Список всех признанных назначений SBI следует ниже.

4.2.4. Кто устанавливает назначения SYSTEM_BUILDER_INFO?

Большинство назначений в разделе MODULE/SBI представляет собой атрибуты того или иного модуля (разрядность шины данных и т. п.). Эти атрибуты становятся известны прежде, чем программой SOPC Builder software модуль добавляется к таблицам карты памяти. Такие назначения были подобны вопросам «Has_IRQ», «Is_Bus_Master» и т. д. Некоторые из назначений точно не известны до завершения построения таблиц карты памяти. Примеры назначений, которые являются свойствами всей системы (в противоположность свойствам непосредственно модуля) — «Base_Address» и «IRQ_NUMBER».

Для каждого назначения, перечисленного ниже, строка «Set by SWB?: <Yes/No>» указывает, является ли это назначение установленным программой SOPC Builder software в таблице карты памяти.

4.2.5. Назначения в SYSTEM_BUILDER_INFO о применении модуля

Is_Enabled

type:	boolean
required?:	No.
default:	1
Set by SBW?:	Yes.

В таблице адреса программы системного составителя MegaWizard, в строке каждого модуля имеется флажок, разрешающий применение данного модуля. Если модуль не разрешен (флажок не установлен), строка затемняется и модуль «временно удаляется» из системы. Программа SOPC Builder software не будет обрабатывать модуль, если он не разрешен. Значение назначения «Is_Enabled» определяет, допускается ли модуль (1) или нет (0).

Instantiate_In_System_Module

type:	boolean
required?:	No.
default:	1
Set by SBW?:	No.

Если это назначение — 1, программа SOPC Builder software будет включать этот модуль в систему Avalon. Если эта установка — 0, то к системе Avalon ничего не будет добавлено. Система Avalon выдает набор требуемых портов интерфейса шины, чтобы соединиться с модулем этого типа. На рис. 13 раздела 2.2 оба периферийных устройства #1 и #2 имеют набор «Instantiate_In_System_Module = 1». Любая периферия с набором «Instantiate_In_System_Module = 0», будет расположена за пределами блока Avalon System, обращаясь через выводы по System-External Interface.

Is_Bus_Master

type:	boolean
required?:	No.
default:	0
Set by SBW?:	No.

Если это назначение — 1, то родительский раздел MODULE описывается как мастер шины Avalon. Если 0, то раздел MODULE описывается как slave. Это назначение определяет, как модуль будет обработан при производстве логики шины Avalon программой SOPC Builder software. Конечно, любой мастер шины Avalon должен иметь часть требуемого набора портов, который, вообще говоря, может быть несовместим с устройствами slave Avalon.

Если назначение модуля SBI/IS_BUS_MASTER — 1, то такой модуль должен также иметь правильный и полный набор портов Мастера шины Avalon, так, как это описано в разделе 5.

Base_Address

type:	Numeric (positive integer, multiple of device span)
required?:	Yes.
Set by SBW?:	Yes.

Назначение «Base_Address» для модулей slave определяет, где они будут находиться в карте памяти мастера. Назначение «Base_Address» установлено в таблице карты памяти MegaWizard программы «SOPC Builder software». «Base_Address» для периферийного устройства должен быть кратным адресному промежутку устройств. Алгоритм для вычисления зоны адреса устройства в зависимости от его других SBI-назначений описаны в разделе 4.2.6.

Uses_Tri_State_Data_Bus

type:	boolean
required?:	No.
default:	0
Set by SBW?:	No.

Одни устройства используют трехстабильную, двунаправленную шину данных, тогда как другие используют раздельные, однонаправленные шины ввода-вывода данных. Это назначение указывает программе SOPC Builder software, какую шину данных использует родительский модуль. Только внешние модули могут устанавливать «Uses_Tri_ State_Data_Bus» в 1. Внешний модуль — любой с «Instantiate_In_System_Module» = 1.

Это назначение используется вместе с назначением «Tri_State_Data_Bus», которое описано ниже.

Tri_State_Data_Bus

type:	string (signal-name, valid HDL identifier).
required:	Yes, if «Uses_Tri_State_Data_Bus» = «1».
Set by SBW?:	Yes.

«Общедоступные» порты (см. раздел 5.4) могут быть подключены к трехстабильной шине данных. Несколько периферийных устройств Avalon могут быть соединены с одним набором выводов трехстабильной шины на системном модуле. По этой причине каждый модуль, который использует трехстабильную шину данных («Uses_Tri_State_Data_Bus = 1») должен иметь определение шины как трехстабильной. Все трехстабильные шины данных поименованы. Это назначение указывает имя шины, которая соединяется с выводами «данных» этого модуля.

Has_IRQ

type:	boolean
required?:	No.
default:	0
Set by SBW?:	No.

Кроме того, модуль может генерировать сигнал запроса прерывания мастеру. Любые модули с выводом типа «irq» (см. таблицу 2) должны установить назначение «Has_irq» в 1. Это указывает программе SOPC Builder software на необходимость построить соответствующую логику в шине Avalon, чтобы расставить приоритеты и подключить сигнал запроса прерывания. Это назначение используется вместе с назначением «IRQ_Number», как описано ниже.

IRQ_Number

type:	Numeric (range: [16..62], inclusive.
required?:	Yes, if «Has_IRQ=1».
Set by SBW?:	Yes.

Любой модуль с «Has_IRQ = 1» должен определить номер запроса прерывания, на который он назначен. Это делается путем установки «IRQ_Number». Разрешенный диапазон — между от 16 до 62 включительно. Прерывания ниже 16 сохранены для внутренних исключений центрального процессора и системного программного обеспечения. Прерывание 63 не используется, и поэтому запрещено.

Address_Width

type:	Numeric (range: [1..<master-address-width>], inclusive.
required:	Yes.
Set by SBW?:	No.

Для всех модулей необходимо определить, сколько выводов адреса они потребуют. Этот число должно соответствовать разрядности портов модуля address (см. табл. 2). Обратите внимание, что это число входных проводов адреса непосредственно на модуле. Не следует путать его со старшим битом адреса, используемым модулем. Отображение между сигналами адреса мастера и выводами модуля адреса генератором шин Avalon выполняется автоматически, как установлено в «Address_Alignment» и «Data_Width» модуля (см. ниже).

Модуль Avalon slave не может иметь «Address_Width» больше, чем «Address_Width» хозяина.

Data_Width

type:	Numeric (range: [1..<master-data-width>], inclusive.
required:	Yes.
Set by SBW?:	No.

Для всех модулей необходимо определить, сколько выводов данных они потребуют. Этот число должно соответствовать разрядности портов модуля — «readdata» и «writedata» (см. таблицу 2), если они существуют. Подключения шины данных Avalon к устройствам, разрядность шины данных которых та же, что и у мастера, просты. Но, когда периферийная шина имеет меньшее количество битов данных, чем мастер, логика шины Avalon должна произвести определенные действия со старшими битами данных. Логика шины Avalon реализует две схемы: «native» и «dynamic».

Address Alignment. См. раздел описания «Address Alignment».

Address_Alignment

type:	Enumerated (legal values: «native», «dynamic»)
required:	Yes.
Set by SBW?:	No.

Address_Alignment для модуля управляет путем, которым «узкие» периферийные устройства связаны с шиной Avalon («узкое» периферийное устройство — то, чьи «Data_Width» меньше, чем «Data_Width» мастера). Есть два имеющих силу параметра настройки для Address_Alignment: «native»

и «dynamic». Раздел 7.0 описывает каждую опцию в подробности. Но вот три вещи, которые действительно нужно помнить:

1) Address_Alignment не имеет значения, если модуль не «узкий».

2) «dynamic» применяют для устройств памяти, которые содержат программы и переменные.

3) «native» — для периферийных устройств, управляемых через регистры.

Read_Wait_States

type:	Numeric (integer) or string «peripheral_controlled».
required:	No
default:	0
Set by SBW?:	No.

Назначение «Read_Wait_States» устанавливает продолжительность операций чтения slave-модуля. По умолчанию, Read_Wait_States = 0, и операции чтения длятся один цикл системной частоты. Раздел 6.1 детально описывает синхронизацию шины для операций чтения. Если «Read_Wait_States» назначено «волшебное» значение «peripheral_controlled» (вместо номера), то продолжительность цикла шины управляется периферийными устройствами по состоянию вывода «Запрос такта ожидания» (см. таблицу 2). Если «Read_ Wait_States» назначено числовое значение, то продолжительность цикла шины управляется автоматически сгенерированным счетчиком внутри модуля шины Avalon. Обратите внимание, что «Read_ Wait_States», вообще говоря, примет значение «1» или «Больше» для синхронных slave (например, slave, осуществленные в PLD-логике) и что значение «0» обычно является подходящим только для асинхронных устройств. Назначение «Read_ Wait_States» используется вместе с назначением «Setup_Time» (описанным ниже и в разделе 6.1). Для назначения на «Read_Wait_States» не существует верхнего предела числового значения.

Write_Wait_States

type:	Numeric (integer) or string «peripheral_controlled».
required:	No
default:	0
Set by SBW?:	No.

Назначение «Write_Wait_States» устанавливает продолжительность операции записи к slave-модулю. По умолчанию, «Write_ Wait_States» — ноль, и операции записи длятся один цикл системной частоты. Раздел 6.2 детально описывает синхронизацию шины для операции записи. Если для «Write_ Wait_States» назначено «волшебное» значение «peripheral_controlled» (вместо номера), то продолжительность цикла шины управляется периферийными устройствами по состоянию вывода «Запрос такта ожидания» (см. таблицу 2). Если назначению «Write_ Wait_States» приписано числовое значение, то продолжительность цикла шины управляется автоматически сгенерированным счетчиком внутри модуля шины Avalon. «Read_Wait_States» используется вместе с назначениями «Setup_Time» и «Hold_Time» (описанными ниже, а также в разделе 6.2). На верхний предел ограничений нет.

Setup_Time

type:	Numeric (integer)
required:	No
default:	0
Set by SBW?:	No.

Назначение «Setup_Time» влияет на синхронизацию операций чтения и записи slave-модуля. По умолчанию «Setup_Time» — ноль, и «адрес», «данные», «выбор устройства» и стробы для чтения-записи устанавливаются для slave-устройства одновременно. Назначая в «Setup_Time» значение, отличное от нуля, начало строба чтения и записи для slave будет задержано относительно установления других управляющих сигналов, адреса и сигналов данных. Разделы 6.1.5 и 6.2.3 подробно описывают синхронизацию шины для циклов с «Setup_Time». Назначение «Setup_Time» часто полезно для создания «безопасных» циклов шины для асинхронных устройств, где быстродействие несущественно (например, интерфейсы IDE, Flash-memory и т. д.).

Hold_Time

type:	Numeric (integer) or magic value «half_clock»
required:	No
default:	0
Set by SBW?:	No.

Назначение «Hold_Time» влияет на синхронизацию операций чтения и записи к slave-модулю. Обратите внимание, что на операции чтения синхронизация «Hold_Time» не воздействует. По умолчанию, «Hold_Time» — ноль, и стробы адреса, данных, чтения и записи устанавливаются slave-устройству одновременно. Если в «Setup_Time» содержится значение, отличное от нуля, начало строба записи для slave задерживается относительно других управляющих сигналов и сигналов данных. В разделе 6.2.3 подробно описывается синхронизация шины для циклов с «Hold_Time». Назначение «Hold_Time» часто полезно для создания «безопасных» циклов шины для асинхронных устройств, там, где быстродействие несущественно (например, интерфейсы IDE, Flash-memory и т. д.).

Если для Hold_Time назначено «волшебное» значение «half_clock», шина Avalon генерирует узкий (продолжительностью ~$^1/_2$ такта) импульс, подходящий для использования при записи в асинхронных устройствах памяти с нулевым временем ожидания. Детальная синхронизация для назначения «half_clock» описывается в разделе 6.2.4.

Uses_Registered_Select_Signal

type:	boolean
required:	No
default:	0
Set by SBW?:	No

«Has_Registered_Select_Signals» должно быть установлено в 1 для любого slave, который имеет порт типа «registeredselectn» (см. табл. 2). Во всех остальных случаях это назначение должно быть установлено в «0» (по умолчанию это «0»).

4.2.6. Алгоритм для определения промежутка адреса в SBI

Зона адресов для любого модуля (число последовательных адресов байта, которые устройство занимает) однозначно определено тремя назначениями в разделе SYSTEM _BUILDER_INFO: «Address_Width», «Data_ Width», и «Address_Alignment». Обратите внимание, что устройство может быть отображено только в «Base_Address», который является целочисленным множителем его зоны адресов.

5. Порты Avalon

Ptf-файл, описывающий любую Nios-систему, должен называться System.ptf. System.ptf содержит один раздел MODULE для каждого модуля, интегрируемого в систему, включая мастера шины Avalon (например, раздел MODULE для CPU Nios и разделы MODULE для каждого периферийного slave-устройства). Каждый раздел MODULE содержит раздел PORT_WIRING, который описывает все сигналы, которые появляются на границе интегрируемого модуля. Другими словами, раздел PORT_WIRING описывает все порты ввода-вывода модулей. То есть разделы PORT_WIRING должны содержать один подраздел PORT для каждого из сигналов ввода-вывода каждого модуля. Каждый подраздел PORT имеет то же самое название, что и сигнал, который это описывает. Назначения в пределах каждого раздела PORT описывают по крайней мере ширину и направление сигналов порта совместно с некоторой другой информацией о том, как порт используется в системе Avalon.

5.0.1. Пример — фрагмент ptf, раздел PORT_WIRING

```
PORT_WIRING
PORT main_clock
        direction = «input»;
        width = «1»;
        avalon_role = «clk»;
PORT register_select_address
        direction = «input»;
        width = «3»;
        avalon_role = «address»;
PORT data_out
        direction = «output»;
        width = «16»;
        avalon_role = «readdata»;
PORT led_drive_out
        direction = «output»;
        width = «1»;
```

5.1. Раздел PORT

Каждый порт модуля Avalon (соответствующий каждому выводу на схемном символе этого модуля) имеет собственный раздел PORT. Название раздела PORT (слово, следующее непосредственно за словом «PORT») — формальное название порта ввода-вывода —то же, что пишется в hdl-файле или на схеме, но без спецификаторов ширины (например, «[7:0]» или «7 DOWNTO 0»). Назначения в пределах раздела PORT сообщают программе SOPC Builder software разрядность порта, каким типом сигнала он является и как он подключается к шине Avalon (если это необходимо).

Для любого конкретного модуля в системе программа SOPC Builder software должна будет подключить часть его портов ввода-вывода к шине Avalon, то есть к шине адресов,

управления или данных. Другие порты могут не быть частью интерфейса модуля шины Avalon и должны быть «продвинуты» так, чтобы они были выставлены как выводы ввода-вывода на верхнем, конечном блоке, который обрабатывается на уровне системного модуля. Раздел PORT (в пределах раздела PORT_WIRING, входящий, в свою очередь, в раздел MODULE), описывает, какова роль портов модуля при соединении с шиной Avalon.

В следующей таблице описаны назначения, принимаемые программой SOPC Builder software в пределах раздела PORT. Любые другие назначения игнорируются.

Каждый раздел PORT (в пределах PORT_WIRING раздела MODULE) должен содержать назначение — «WIDTH». Это назначение сообщает программе SOPC Builder software ширину (число битов) порта. Значение может быть любым положительным целым числом, отличным от нуля.

5.3. Назначение «PORT/DIRECTION»

Каждый раздел PORT (в пределах PORT_WIRING раздела MODULE должен содержать назначение «DIRECTION». «DIRECTION» сообщает программе SOPC Builder software, является ли сигнал этого порта вводом, выводом или двунаправленным.

Для назначения направления есть три разрешенных значения:

* input
* output
* inout

Любое другое значение неправильно.

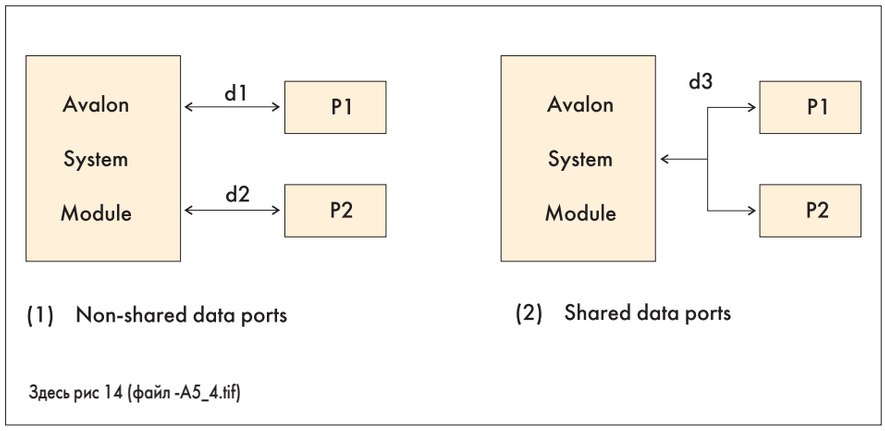
5.4. Назначение «PORT/IS_SHARED»

Каждый раздел PORT (в пределах PORT_WIRING раздела MODULE) может произвольно содержать назначение «is_shared» в битовом виде (1/0). Если «is_shared» явно не назначен, его значение по умолчанию равно нулю.

Все порты, для которых не указано «avalon_role», связаны с выводами ввода-вывода на уровне модуля Avalon System. Если PORT общедоступный «is_shared = 1», то, значит, он связан с выводом модуля Avalon System, который может также соединяться с портами другого модуля. Назначение «is_shared» может быть установлено в «1» только для портов, которые не имеют «avalon_role».

Порты часто используются совместно, если они — часть группы трехстабильной шины. Общедоступные порты обычно подключаются к трехстабильным шинам и используются на внешних периферийных устройствах.

Следующий пример показывает два внешних периферийных устройства и подключение их к модулю Avalon System (все другие подключения опущены для ясности) — см. рис. 14.



Здесь рис 14 (файл -A5_4.tif)

В первом случае порты данных периферийных устройств P1 и P2 определены как «is_shared = 0».

Во втором случае порты данных периферийных устройств P1 и P2 определены как «is_shared = 1». Некоторым типам («avalon_ role») портов разрешено быть разделенными, а некоторым — нет. Эти требования описаны в табл. 2.

5.5. Назначение «PORT/AVALON_ROLE»

Поле AVALON_ROLE сообщает программе SOPC Builder software, как порт должен быть связан с шиной Avalon. Если поле AVALON_ROLE для порта опущено, программа SOPC Builder software не подключает никакие сигналы шины Avalon к этим портам.

Есть перечень разрешенных значений AVALON_ROLE. В табл. 2 приведен перечень названий AVALON_ROLE, распознаваемых версией комплекта Nios1.1.

5.6. Имена портов модуля Avalon System

На рис. 13 есть два вида портов, которые могут быть в интерфейсе верхнего модуля Avalon System:

1. Выводы, которые являются внешними вводами-выводами, связанными со специфическими модулями, интегрированными внутри Avalon System;
2. Адрес шины Avalon, данные или сигналы управления для соединения со slave-устройствами, расположенными вне модуля Avalon System.

В любом случае есть четкая схема названий портов модуля Avalon System. (Помните, что Avalon System является модулем, и поэтому все его порты генерируются автоматически.) Мы рассмотрим эти два случая отдельно.

5.6.1. Названия для выводов, связанных с внутренними модулями

Любой вывод внутреннего модуля (Instantiate_In_System_MODULE = «1») который не имеет определенного «avalon_role», становится внешним портом так, чтобы быть портом в верхнем уровне модуля Avalon System.

Этот вывод обязательно должен иметь название, отличное от соответствующего внутреннего вывода модуля.

Эта становится понятно, если рассмотреть случай, когда два одинаковых устройства (с теми же самыми именами портов) интегрируются в пределах системного модуля.

Внешнему порту дают уникальное имя, основанное на имени внутреннего модуля и имени порта:

<promoted-port-name> = <port-name>_<tolfrom|to_and_from>_<instance-name>

В вышеупомянутом выражении <promoted-port-name> является формальным hdl-именем вывода ввода-вывода модуля Avalon System, <port-name> — формальное hdl-имя порта во внутреннем модуле, а <instance-name> — формальное hdl-имя внутреннего модуля Avalon System. Если иначе не определено, генератор шины Avalon создает названия, совпадающие с названием раздела MODULE для внутреннего модуля с добавлением знака «_».

Кроме того, имена имеют «_to_» или «_from_» или «_to_and_from_» в середине, в зависимости от того, ввод это, вывод или двунаправленный выход на модуле.

В результате «_to_and_from_» и «_» соглашения, возможны почти комические названия, подобные следующим:

bidir_port_to_and_from_the_lcd_pio

txd_from_the_printf_uart

Тем не менее эти названия не оставляют сомнений относительно их цели, назначении и роли в системе.

5.6.2. Названия для сигналов Avalon, используемые для соединения с внешними slave-устройствами

Все сигналы интерфейса Avalon (адрес, данные и управление) для внешних по отношению к системе slave-устройств обязательно должны быть определены как порты ввода-вывода верхнего уровня в Avalon System. Так как шина Avalon и Avalon System — модульные, то все соответствующие порты генерируются автоматически и программа SOPC Builder software должна иметь четкую схему обозначения каждого порта.

Есть две различные схемы обозначения для интерфейсных сигналов внешних модулей Avalon: одна схема для общедоступных портов, другая — для необщедоступных (см. раздел 5.4 для описания общедоступных портов).

Таблица 1. Назначения раздела PORT

Назначение	Требуемый?	Значение по умолчанию	Краткое описание
width	ДА	-	Векторная ширина (число битов)
direction	ДА	-	Должно быть: input, output, или inout
ws_shared	НЕТ	0	Использует ли системный вывод совместно с другими модулями?
Avalon_role	НЕТ	" "	Тип сигнала шины Avalon, если присутствует

Таблица 2. Значения поля AVALON_ROLE

AVALON_ROLE	Связь порта
clk (m)	Частота синхронизации системы, ввод — разрядность должна быть 1 (скаляр). Всесистемные импульсы (clk) обеспечивают все устройства Avalon тактовыми сигналами типа «clk». Это вход верхнего уровня по отношению к системному модулю.
resetn (m)	Всесистемный импульс сброса, ввод – ширина должна быть 1 (скаляр). Всесистемный сброс обеспечивает все устройства Avalon сигналом типа «resetn». Это вход верхнего уровня по отношению к системному модулю.
address (ms)	Мастер Avalon управляет шиной address, которая подключена ко всем slave-модулям. Все порты «address» связаны с частью или со всей шиной address мастера.
writedata (m)	Шина данных, ввод (для мастера — вывод) – разрядность не должна быть больше разрядности шины данных мастера. Все порты типа «writedata» связаны с частью или со всей шиной вывода данных мастера. Логика, динамически устанавливающая разрядность шины, может управлять вводом записываемых в процессор данных от периферии с меньшей разрядностью (см. ниже).
readdata (m)	Шина данных, вывод (для мастера — ввод) — разрядность не должна быть больше разрядности шины данных мастера. Каждое slave-устройство в системе Avalon имеет собственную выходную шину данных для чтения. Генерированная PBM логика включает мультиплексор, который выбирает для мастера только один из сигналов readdata.
data (S)	Шина данных, двунаправленный (inout) — разрядность не должна быть больше разрядности шины данных мастера. Внешние модули могут произвольно включать порт типа «data» (вместо портов типа «readdata» и «writedata»). Порты типа «data» должны быть двунаправленными (inout) и должны быть объявлены «shared». Модули, находящиеся внутри системы, не могут иметь портов типа «shared».
chipselect	Сигнал device-select, ввод — разрядность должна быть 1 (скаляр). Автоматически генерируемый Avalon PBM включает декодирование логики адреса. Индивидуальные сигналы выбора декодированы для каждого подчиненного модуля в системе. Модуль должен игнорировать все другие управляющие сигналы шины Avalon, если на его порт «chiselect» не подается логическая единица (1).
byteenablen (ms)	Шина byte-enable, ввод (для мастера — вывод) — разрядность не должна превышать разрядность шины выводов byteenablen мастера. Некоторые подчиненные устройства могут поддерживают индивидуальное побайтное чтение и запись. Такие устройства используют вводы be_n, чтобы выбрать, какие байты должны использоваться для текущей операции шины. Мастер Avalon управляет выходящей из него шиной byteenablen, которая подключается ко всем подчиненным модулям с портами ввода типа byteenablen. Сигналы шины byteenablen имеют отрицательную логику работы.
writen (ms)	Write-strobe, ввод (для мастера — вывод) — разрядность должна быть 1 (скаляр). Мастер Avalon управляет выходящим из него сигналом управления — стробом записи, который подключается ко всем подчиненным модулям. Мастер переключает этот сигнал в низкий уровень, чтобы указать, что текущая операция шины — запись данных в выбранный slave.
readn (ms)	Read-strobe, ввод (для мастера — вывод) — разрядность должна быть 1 (скаляр). Мастер Avalon управляет выходящим из него сигналом управления — стробом чтения, который подключается ко всем подчиненным модулям.Мастер переключает этот сигнал в низкий уровень, чтобы указать, что текущая операция шины — чтение данных из выбранного slave.
irq (m)	Interrupt-request, вывод (для мастера — ввод) – разрядность должна быть 1 (скаляр). Каждое slave-устройство Avalon может иметь единственный вывод прерывания. Автоматически генерируемый PBM будет объединять их по «или» и передаст результат к выводу типа «irq» мастера. PBM также включает логику, чтобы одновременно представить номер самого высокого приоритета выработанного прерывания (с самым меньшим номером) на вывод мастера типа «master_irq_number».
irqnumber (M)	nterrupt-number, ввод — разрядность не должна быть больше 6. Автоматически генерируемый PBM представляет мастеру Avalon номер приоритета самого высокого обрабатываемого прерывания на порте типа «irqnumber» мастера. Этот порт может быть только на устройствах мастера Avalon.
waitrequest (m)	waitrequest, вывод (для мастера — ввод) — разрядность должна быть 1 (скаляр). Каждое slave-устройство Avalon может иметь единственный «wait_request» порт. Если периферийное устройство, будучи выбранным, установит логическую единицу на его порту «wait_request», то операция шины Avalon будет продлена до тех пор, пока «wait_request» не будет сброшен. Автоматически генерируемый Avalon PBM включает всю логику, необходимую для соединения сигналов типа «wait_request» и передачи их мастеру.
registeredselectn	Выбор чипа, вывод — разрядность должна быть 1 (скаляр). Автоматически генерируемый Avalon PBM включает декодирование логики адреса. Некоторые внешние модули могут также иметь порт типа registeredselectn вместо обычного порта типа chipselect. Тогда для этого модуля сигнал chip-select будет произведен регистром «fast_output» в Apex. См. объяснения ниже.
ifetch (M)	Операция выборки команды, вывод. Разрядность должна быть 1 (скаляр). Мастер Avalon должен иметь единственный порт типа ifetch, который он устанавливает в 1 при выборке команды, и в 0 при чтении или записи данных. Только мастера Avalon могут иметь этот тип порта.
memis32bits (M)	индикатор Memory-width, вводимые данные — 32 бита. Мастера Avalon должны иметь отдельный порт ввода типа memis32bits. Автоматически генерируемый PBM устанавливает этот сигнал в 1, когда адресованное периферийное устройство выдает полные 32 бита данных на центральный процессор.PBM устанавливает этот сигнал в 0, когда к процессор обращается к периферийным устройствам с меньшей разрядностью.
always0	Статический сигнал 0, ввод. Автоматически генерируемый PMB сигнал будет непрерывно управлять любым портом ввода типа always0 на любом устройстве с подключением логических сигналов в 0.
always1	Статический сигнал 1, ввод. Автоматически генерируемый PMB сигнал будет непрерывно управлять любым портом ввода типа always1 на любом устройстве с подключением логических сигналов в 1.

Примечание:

Части AVALON_ROLE в таблице сопровождаются примечаниями «m» и «s». Вот их значение:

m: этот тип порта может быть как у мастера шины Avalon, так и у slave.

M: этот тип порта может быть только у мастера шины Avalon и не может быть у slave.

s: этот тип порта может быть объявлен общедоступным.

S: этот тип порта может быть только общедоступным.

Для необщедоступных портов схема названий такова:

а) <system-port-name> = <avalon-role>_<tolfromlto_and_from>_<instance-name>

Для общедоступных портов схема названий такова:

б) <system-port-name> = <tri-state-bus-group-name>_<avalon-role>

Для назначения порта в случае *а* поле <avalon-role> имеет значение «AVALON_ROLE». Другие поля — так, как описано в секции 5.6.1. В случае *б*, если <tri-state-bus-group-name> — имя, которое дано трехстабильной шине, к которой подключен модуль (см. раздел 5.4, так как все общедоступные порты должны быть связаны с названиями трехстабильных шин).

Продолжение следует