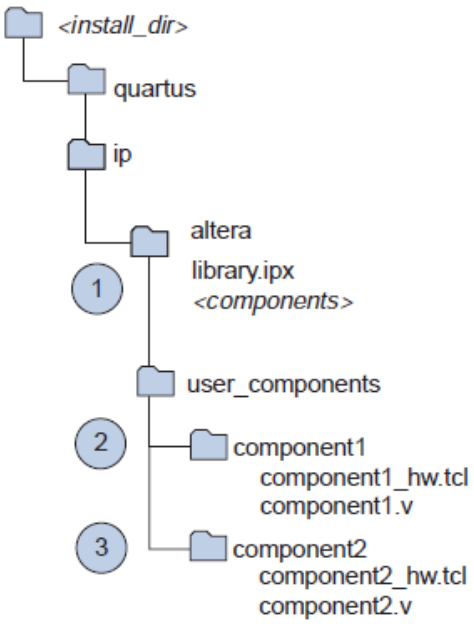
Инсталляция дополнительного компонента

Существует несколько способов сделать ваш компонент доступным в проектах SOPC Builder. В следующих секциях описаны эти способы.

Копирование в IP корневую директорию

Простейший способ – это копирование ваших компонентов в стандартную IP директорию, предлагаемую Altera. На рис. 4-2 показан этот метод.

Figure 4–2. User Library Included In Subdirectory of $IP_ROOTDIR



На рис.4-2 номера в кружках показывают три шага алгоритма, которому следует SOPC во время инициализации. Эти шаги описаны в следующих параграфах:

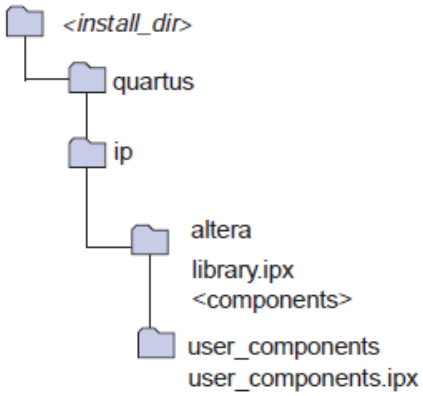
1. SOPC Builder рекурсивно ищет по умолчанию директорию ***<install_dir>*/ip/**. Он находит файлы в поддиректории altera, где хранятся все компоненты Altera. Файл **library.ipx** содержит список всех компонентов, найденных в поддиректориях. Рекурсивный поиск останавливается, как только SOPC Builder находит этот **.ipx** файл.
2. Частью рекурсивного поиска SOPC Builder является определение соседней с altera директории **user_components**. Одним уровнем ниже SOPC Builder находит директорию **component1**, в которой находится **component1_hw.tcl**. Когда SOPC Builder находит этот компонент, рекурсивный анализ синтаксиса останавливается, так что ни один компонент в поддиректории **component1** не ищется.
3. Затем SOPC Builder находит соседнюю директорию **component2**, в которой находится компонент **component2_hw.tcl**. Если SOPC Builder находит этот компонент, рекурсивный анализ синтаксиса останавливается.

Если вы сохраните ваш **_hw.tcl** файл в директории ***<install_dir>*/ip/**, SOPC Builder найдёт ваш **_hw.tcl** файл и остановится. SOPC Builder не будет вести дальнейших поисков, описанных выше.

Связанные компоненты в .ipx файле

Второй метод – это задание вашей IP директории в файле **user_components.ipx**, расположенном в ***<install_dir>*/ip/**. На рис. 4-3 показан этот метод.

Figure 4–3. Specifying A User .ipx directory



Файл **user_components.ipx** содержит одну линию кода, направляющую SOPC Builder к месту расположения библиотеки пользователя. В примере 4-1 показан направляющий код.

Example 4–1. Redirect to User Library

```
<library>
  <path path="c:/<user_install_dir>/user_ip/**/*" />
/<library>
```

Для обоих этих методов, если вы установите новую версию программы Quartus II, вам потребуется обновить установку, чтобы включить в неё ваши библиотеки.

Вы можете проверить, какие компоненты доступны и, заодно, уменьшить время запуска SOPC Builder, используя две утилиты: ip-catalog и ip-make-ipx. В следующих секциях описываются эти утилиты.

ip–catalog

Показывает каталог компонентов либо в открытом тексте, либо в формате XML.

Использование

```
ip-catalog --project-dir[=<directory>] --name[=<value>]
--verbose[=<true|false>] --xml[=<true|false>] --help
```

Опции

- --project-dir[=<*directory*>]. Опционально. Компоненты могут быть найдены в определённом месте, по отношению к проекту. По умолчанию, используется ".". Чтобы указать другую директорию проекта, используйте "".
- --name[=<*value*>]. Опционально. Этот аргумент содержит последовательность для фильтрации имён, во время поиска компонентов. Чтобы показать все компоненты, используйте * или " ". По умолчанию показываются все компоненты. Этот аргумент не чувствителен к регистру.

- --verbose[=<*true*|*false*>]. Опционально. Когда истина, отчитывается об успешном исполнении команды.
- --help. Показывает помощь для команды ip-catalog.

ip-make-ipx

Эта команда создаёт файл индекса для заданной директории. Она возвращает 0 при успешном выполнении и ненулевое значение при сбое.

Использование

```
ip-make-ipx --source-directory[=<directory>] --output[=<file>]
--relative-vars[=<value>] --thorough-descent
--message-before[=<value>] --message-after[=<value>] --help
```

Опции

- --source-directory=<*directory*>. Опционально. Директория индекса. По умолчанию для директории – ".". Вы можете задать список директорий, разделённых запятой.
- --output[=<*file*>]. Опционально. Имя генерируемого файла. По умолчанию имя файла - **./components.ipx**.
- --relative-vars[=<*value*>]. Опционально. Включает в выходной файл родственную ссылку на определённую переменную или на возможные переменные. Вы можете задать множество переменных списком, разделяя их запятыми.
- --thorough-descent[=<*true*|*false*>]. Опционально. Если установлено, то компонент или **.ipx** файл в директории не мешает использовать в поиске поддиректории.
- --message-before[=<*value*>]. Опционально. Сообщение выводится в stdout, когда начинается индексирование.
- --message-after[=<*value*>]. Опционально. Сообщение выводится в stdout, когда завершено индексирование.
- --help. Показывает помощь для этой команды.

Понимание синтаксиса IPX файла

Файл **.ipx** – это XML файл, в котором элемент верхнего уровня – это <библиотека> с подэлементами <путём>, состоящим из <путей> и <компонентов> (<*path*> and <*component*>).

Элемент <*path*> имеет единственный атрибут, также называемый путём, который связывается с директорией символом дикой карты (*) или связывается с одним файлом. Две звёздочки (**) указывают любое количество поддиректорий. Одна звёздочка (*) указывает на один файл или директорию. При отыскании в указанных путях, идентифицируются три типа файлов:

- **.ipx—** дополнительный файлы индекса
- **_hw.tcl—** обозначение компонента SOPC Builder
- **_sw.tcl—** обозначение программного компонента поддержки пакета платы (BSP) Nios II.

Элемент <*component*> содержит несколько атрибутов для определения компонента. Если вы предоставите все необходимые детали для каждого компонента в **.ipx** файле, время запуска SOPC Builder будет меньше чем, если SOPC Builder будет пытаться обнаружить файлы в директории.

В примере 4-2 показаны два элемента <*component*>. Обратите внимание, что пути для имён файлов заданны относительно **.ipx** файла.

Example 4–2. Component Elements

```
<library>
  <component
    name="An SOPC Component"
    displayName="SOPC Component"
    version="2.1"
    file="./components/sopc_component/sc_hw.tcl"
   />
  <component
    name="legacy_component"
    displayName="Legacy Component (Classic Edition!)"
    version="0.9"
    file="./components/legacy/old_component/class.ptf"
   />
</library>
```

Обновление ранних версий

Если вы задали собственный путь поиска в SOPC Builder версии ранее 8.1, используйте опцию **IP Search Path** или добавьте её в $SOPC_BUILDER_PATH, SOPC Builder автоматически добавит эти директории в **user_components.ipx** файл в вашей домашней директории. Этот файл сохраняется в <*home_dir*>**/altera.quartus/ip/8.1/ip_search_path/user_components.ipx.** Перейдите к опции **IP Search Path** в диалоге **Options** , чтобы посмотреть представленные директории.

Структура компонента

Большинство компонентов определяются с помощью **_hw.tcl** файла, текстового файла, написанного на языке Tcl скриптирования, который описывает компоненты в SOPC Builder. Вы можете добавить компонент в SOPC Builder либо написав Tcl описание, либо используя редактор компонента для генерирования его автоматического Tcl описания. В этой секции описывается структура Tcl компонентов и то, как они сохраняются.

Подробнее о редакторе компонентов, обратитесь к главе "Редактор компонентов" в томе 4 Настольной книги Quartus II. Подробнее о Tcl командах в SOPC Builder, обратитесь к главе "Tcl связь с интерфейсом компонента" в томе 4 Настольной книги Quartus II.

Файл описания компонента (_hw.tcl)

Tcl компонент состоит из:

- Файла описания компонента, который является Tcl файл с именем вида <*название объекта*>**_hw.tcl.**
- Файлы Verilog HDL или VHDL, которые определяют модуль верхнего уровня собственного компонента (опционально).

Файлы **_hw.tcl** определяют всё, что нужно знать SOPC Builder об имени и размещении файлов проекта компонента.

Редактор компонента SOPC Builder сохраняет компоненты в формате **_hw.tcl**. Вы можете использовать эти Tcl файлы в качестве шаблонов при редактировании компонентов вручную. Когда вы редактируете сохранённый ранее **_hw.tcl** файл, SOPC Builder автоматически сохраняет предыдущую версию в виде **_hw.tcl~**.

За подробной информацией о том, какую информацию вы можете включить в **_hw.tcl** файл, обратитесь к главе "Tcl связь с интерфейсом компонента" в томе 4 Настольной книги Quartus II.

Организация файла компонента

Обычный компонент использует следующую структуру директорий. Точные названия директорий не важны.

- <*component_directory*>**/**
- <*hdl*>**/**— директория, содержащая HDL файлы проекта компонента и **_hw.tcl**
- <*component name*>**_hw.tcl**— файл описания компонента
- <*component name*>**.v or .vhd**— HDL файл модуля верхнего уровня
- <*component_name*>**_sw.tcl**— файл конфигурации программного драйвера. Этот файл задаёт путь для **.c** и **.h** файлов, ассоциированных с этим компонентом.
- <*software*>**/**— директория, содержащая программные драйверы и/или библиотеки, связанные с компонентом. Altera рекомендует, чтобы программная директория была поддиректорией директории, содержащей **_hw.tcl** файл.

За подробной информацией о написании драйвера устройства или соответствующей системы программного обеспечения для использования в процессе проектирования Nios® II IDE, обратитесь к секции "Слой аппаратной абстракции" в Настольной книге программиста Nios II. Глава "Справочное пособие по инструменту создания Nios II программ" в Настольной книге программиста Nios II описывает команды, используемые в Tcl скрипте.

Контроль версий компонента

Вы можете создать и использовать различные версии одного компонента с помощью следующих опциональных средств:

- Определить версии свойств модуля в вашем **_hw.tcl** файле.

За подробной информацией обратитесь к главе "Tcl связь с интерфейсом компонента" в томе 4 Настольной книги Quartus II.

- Если несколько версий компонента определены в вашей библиотеке компонентов, вы можете применить различные версии компонента, правым кликом на компонент, затем выбрать **Add version** <*version_number*>.
- Вы можете создать **.ipx** файл в той же директории, что и проект SOPC Builder, чтобы контролировать путь поиска в вашем проекте.

Классические компоненты в SOPC Builder

Если вы используете классические компоненты, созданные в версии SOPC Builder 7.2 и раньше, прочитайте эту секцию, чтобы понять отличия. Этот документ использует термин *классические компоненты* для связи с компонентами, сделанными в ранних версиях программы Quartus II на основе **class.ptf**. Если вы не используете классические компоненты, пропустите эту секцию.

Классические компоненты совместимы с новыми версиями SOPC Builder, но со следующими условиями:

- Классические компоненты, конфигурируемые на вкладке **More Options** в SOPC Builder, например, сложные IP компоненты сторонних IP разработчиков, не поддерживаются в программе Quartus II версии 7.1 и старше. Если ваш компонент связан с программой, вы не сможете использовать его без повторного создания в редакторе компонентов или без Tcl скриптирования.
- Чтобы внести изменения в классический компонент в редакторе компонентов, вы должны сначала обновить компонент, редактируя классический компонент, а потом сохранить его в формате компонента **_hw.tcl** в редакторе компонента.