Ограничения в инкрементной компиляции

В этом разделе содержится документация по ограничениям и условиям, с которыми приходится считаться во время использования инкрементной компиляции, включая итерации со средствами программы Quartus II. Некоторые ограничения накладываются на нисходящее и восходящее проектирование, а некоторые – только на восходящее.

Описаны следующие ограничения:

Сохранение точных временных характеристик

Временные характеристики могут слегка измениться в разделе с уровнем сохранения компоновки разводка и размещение, когда другие разделы объединяются или заново размещаются и разводятся. Временные изменения приводят к изменениям в паразитной загрузке или перекрёстным сообщением с другими (изменёнными) разделами. Эти временные изменения очень малы, обычно менее 30 пс на временной путь. Дополнительные ветвления по выходу или пути разводки при добавлении разделов, могут также ухудшить временные характеристики.

Чтобы гарантировать себе, что раздел будет достигать положенных временных ограничений при изменении других разделов, требуется предусмотреть небольшой запас. Компоновщик автоматически создаёт запас во время компиляции проекта, так что вам не потребуются какие-либо действия.

Когда размещение и разводка не могут быть в точности сохранены

Компоновщик может перекомпоновать задействованные узлы, если два узла имеют одно назначение локализации, путём установок импортированным спискам соединений или пустым разделам снова использовать предыдущие списки соединений пост-компоновки. Есть два случая, когда информация о разводке не может быть в точности сохранена. Первый, когда импортируются несколько разделов, которые могут иметь конфликты в разводке, потому что два блока нижнего уровня хотели бы использовать один провод, даже если назначения архитектуры этих низкоуровневых блоков не перекрываются. Эти конфликты разводки автоматически устраняются Компоновщиком Quartus II переразводкой в незадействованных сетях. Второй, когда импортированный регион LogicLock перемещён в головном проекте, взаимное расположение узлов сохраняется, но разводка не может быть сохранена, потому что соединения трассировки неоднородны в плоскости чипа.

Использование инкрементной компиляции с архивными файлами Quartus II

Информация списка соединений пост-синтеза и пост-компоновки для каждого раздела проекта сохраняется в базе данных проекта, директория *incremental_db*. Когда вы архивируете проект, информация базы данных не включается в архив, если вы не включите файлы базы данных в файл **.qar**.

Altera рекомендует вам включать файлы базы данных в диалоговом окне **Архивирование Проекта**, чтобы сохранить результаты компиляции. Крикните **Расширенные**, выберите файлы, которые содержат базу данных компиляции, или включите **Файлы базы данных компиляции** для создания набора Обычных файлов.

Когда вы включаете в состав архива базу данных, размер файла **.qar** значительно увеличивается, по сравнению с архивом без базы данных.

Информация списка соединений для импортированных разделов всегда сохраняется в соответствующем файле .**qxp**. Импортированные файлы .**qxp** автоматически сохраняются в подпапке *imported_partitions*, так что вам не требуется архивировать базу данных проекта для сохранения результатов импортируемых разделов. Когда вы восстанавливаете архив проекта, разделы автоматически заново импортируются из .**qxp** файла в этой подпапке, если она доступна.

Поддержка формальной верификации

Вы не сможете использовать разделы проекта, если вы создаете список соединений для инструмента формальной верификации.

Импорт зашифрованного IP ядра в восходящем проектировании

Предоставляемая в лицензии информация требуется для компиляции IP ядер. Если IP ядро импортировано в виде .**qxp** файла из другого проекта Quartus II в процессе восходящей компиляции, разработчик головного проекта должен иметь корректную лицензию. Это означает, что вам потребуется полная лицензия для генерации программного файла без ограничений. Если у вас нет лицензии, но IP в .**qxp** файле было скомпилировано с помощью оценочного оборудования OpenCore Plus, вы сможете генерировать оценочный программный файл без лицензии. Если IP поддерживает только OpenCore симуляцию, вы сможете скомпилировать проект и сгенерировать список соединений симуляции, но не сможете создать программные файлы, пока не получите полную лицензию.

Выводы SignalProbe и менеджер инженерных изменений в Планировщике Чипа

Когда вы создаете выводы SignalProbe или используете Редактор Характеристики Ресурсов для внесения изменений в порядке внесения изменений (ECO) после выполнения полной компиляции, перекомпиляция проекта не требуется. Эти изменения касаются только списка соединений без выполнения нового размещения и разводки. Вы можете сохранить эти изменения, используя список соединения пост-компоновки с уровнем размещение и разводка. Когда раздел перекомпилирован, выводы SignalProbe и изменения ECO в неизменённых разделах сохраняются.

За дополнительной информацией об использовании средств SignalProbe для отладки вашего проекта обратитесь к главе "Быстрая отладка проекта, используя SignalProbe" в томе 3 Настольной книги Quartus II. За дополнительной информацией об использовании Планировщика Чипа и Редактора Характеристик Ресурсов для производства ECO, обратитесь к главе "Менеджер инженерных изменений в Планировщика Чипа" в томе 2 Настольной книги Quartus II.

Для сохранения выводов SignalProbe или изменений ECO, тип списка соединений нужно установить как **пост-компоновка** с уровнем сохранения разводки как **размещение и разводка**. Если некоторые разделы с выводами SignalProbe или изменениями ECO установлены в **пост-компоновка** без разводки или **только список соединений**, программа выдаёт предупреждение и интеллектуально использует список соединений **пост-компоновка с размещением и разводкой**. Если в разделах установлены списки соединений **исходный код** или **пост-синтез**, программа выдаёт предупреждение и выводы пост-компоновка SignalProbe или ECO изменения не включаются в новую компиляцию. Поскольку разделы получаются связанными с выводами SignalProbe или ECO изменениями, как было описано ранее, в этом случае все связанные разделы наследуют тип списка соединений от связанного раздела с наивысшим уровнем сохранения.

Разделы, связанные с выводами SignalProbe или изменениями ECO

Если изменения ECO влияют более чем на один раздел или на соединения между другими разделами, разделы получаются связанными. Все высокоуровневые "родительские" разделы и ближайшие общие родительские также связаны. В этом случае, соединения между разделами правильно обозначить снаружи двух взаимно влияющих разделов, так чтобы все разделы компилировались вместе. Все связанные разделы используют тот же тип списка соединений и наследуют тип списка соединений от связанных разделов с наивысшим уровнем сохранения.

Когда вывод SignalProbe создан, он оказывает влияние на раздел, который содержит узел с существующим пробником. Выходной вывод SignalProbe назначается в головном разделе. Поэтому формируются новые соединения между головным разделом и низкоуровневым разделом, содержащим узел с пробником. Из-за этого соединения, низкоуровневый раздел, содержащий узел с пробником, и "родительские" разделы верхнего уровня получаются связанными. Все связанные разделы используют тот же тип списка соединений, который наследуется от типа списка соединений связанного раздела с наивысшим уровнем сохранения компоновки.

Когда разделы связаны, то можно выбрать, какой список соединений сохраняется во время перекомпиляции проекта, а именно:

■ Если все связанные разделы имеют тип списка соединений **исходный код** или **пост-синтез**, то разделы разводятся как обычно. В этом случае, выводы SignalProbe и изменения ECO не включаются в новый список соединений, таким образом, вам необходимо повторно применить изменения в **Менеджере Изменений**.

■ Если некоторым связанным разделам установлен тип списка соединений **пост-компоновка**, и не было изменения в исходном коде, программа выдает предупреждение и интеллектуально использует тип списка соединений **пост-компоновка** с размещением и разводкой для всех связанных разделов. Для сохранения соответствующего списка соединений **пост-компоновка**, программа сохраняет выводы SignalProbe или изменения ECO.

■ Если некоторым связанным разделам установлен тип списка соединений **пост-компоновка (строгий)**, программа выдает предупреждение и интеллектуально использует тип списка соединений **пост-компоновка с размещением и разводкой** для всех связанных разделов, несмотря на некоторые изменения в исходном коде. Для сохранения соответствующего списка соединений пост-компоновка, программа сохраняет выводы SignalProbe или изменения ECO. Обратите внимание, что в этом случае, изменения в исходном коде в некоторых связанных разделах не включаются в новый список соединений.

■ Если некоторые связанные разделы перекомпилируются с изменениями в исходном коде, программа выдает предупреждение и перекомпилирует также другие связанные разделы. Когда это происходит, выводы SignalProbe или изменения ECO не включаются в новый список соединений, таким образом, вам необходимо повторно применить изменения в **Менеджере Изменений**.

Экспортируемые разделы

В восходящей инкрементной компиляции, экспортируемый список соединений содержит все действующие выводы SignalProbe и изменения ECO. Это будет условием для выравнивания и комбинации низкоуровневых разделов в дочернем проекте для избежания нарушений границ разделов в головном проекте. После импорта этого списка соединений, изменения, сделанные в низкоуровневом разделе, не появляются в **Менеджере Изменений** в верхнем уровне.

Если вы делаете изменения ECO, которые влияют на интерфейс с низкоуровневым разделом, программа выдает предупреждение во время процесса экспорта, что этот список соединений не будет работать в головном проекте без модификации HDL кода в верхнем уровне, отображающем изменения в нижнем уровне.

Встроенный логический анализатор SignalTap II в процессе восходящей компиляции

Вы можете использовать встроенный логический анализатор SignalTap II в любом проекте, который вы компилируете и программируете в чипе Altera.

Вы не можете экспортировать низкоуровневый проект, который использует файл SignalTap II (**.stp**) для SignalTap II логического анализатора в процессе восходящей инкрементной компиляции. Вам необходимо убрать средство SignalTap II и перекомпилировать проект перед экспортом проекта в качестве раздела.

Вы можете реализовать мегафункцию SignalTap II напрямую в своем низкоуровневом проекте (взамен использования файла **.stp**) и экспортировать этот проект в верхний уровень в процессе восходящего проектирования.

Вы можете подключить некоторые узлы в проект Quartus II, в том числе узлы, импортируемые из других проектов. Используйте соответствующий фильтр в **Поиске Узлов** для поименного поиска узлов. Используйте SignalTap II: **пост-компоновка**, если тип списка соединений пост-компоновка, для инкрементного подключения имен узлов в базу данных списка соединений пост-компоновка. Используйте SignalTap II: **пре-синтез**, если тип списка соединений **исходный файл**, чтобы сделать соединения к узлам исходных файлов (**пре-синтез**), когда вы синтезируете раздел из исходного кода.

Детальное описание использования логического анализатора SignalTap II в процессе инкрементной компиляции в томе 3 Настольной книги Quartus II "Отладка проекта, используя встроенный логический анализатор SignalTap II."

Интерфейс логического анализатора в процессе восходящей компиляции

Вы можете использовать **интерфейс логического анализатора** в некоторых проектах, которые вы компилируете и программируете в чипе Altera. Вы не сможете экспортировать низкоуровневый проект, который использует интерфейс логического анализатора, в процессе восходящей инкрементной компиляции. Вам необходимо отменить средства интерфейса логического анализатора и перекомпилировать проект, прежде чем экспортировать проект в качестве раздела.

За дополнительной информацией об интерфейсе логического анализатора, обратитесь к тому 3 Настольной книги Quartus II "Отладка в системе с использованием внешних логических анализаторов".

Миграция проекта с разделами в различные чипы

Назначения разделов остаются действительными во время миграции в чипы различной плотности или семейства. Размер регионов LogicLock остается без изменений, если происходит миграция чипов внутри семейства, но направляющие не сохраняются. Определенные назначения в архитектуре не сохраняются для различных чипов или семейств, поскольку координаты локализации различаются между чипами.

Список соединений пост-синтез остается действительным во время миграции в различные по размеру чипы одного семейства. Список соединений пост-компоновка не действителен во время миграции в чипы различной плотности или семейств.

Компиляция HardCopy и процесс миграции

Чипы HardCopy APEX и HardCopy Stratix

Инкрементная компиляция в программе Quartus II не поддерживается для процессов разработки на HardCopy APEX и HardCopy Stratix.

Процесс миграции HardCopy ASIC

Нисходящая инкрементная компиляция поддерживается для основных семейств процесса миграции HardCopy – в основном для FPGA и HardCopy. Назначения разделов мигрируют в подходящий чип. Однако вы не можете вносить изменения после миграции, поскольку в проекте ещё не доступны результаты компиляции для основного семейства. Поэтому вы можете выполнить нисходящую инкрементную компиляцию для одного чипа семейства, но не сможете выполнить другие инкрементные компиляции после миграции.

Уровень сохранения компоновки **Только список соединений** не поддерживается для списка соединений пост-компоновка для компиляции чипов FPGA или HardCopy ASIC после того, как определены миграционные чипы (это означает, что для чипа компиляции HardCopy ASIC определен чип миграции FPGA, или для чипа компиляции FPGA определен чип миграции HardCopy ASIC).

Восходящая инкрементная компиляция для чипов FPGA или HardCopy ASIC не поддерживается, если в установках выбраны чипы миграции. Средство **Сравнение Ревизий** требует, чтобы списки соединений FPGA или HardCopy ASIC были одинаковыми. Поэтому, все операции, выполненные для одной ревизии должны совпадать и для других ревизий. Это совершается при записи всех операций и воспроизведении их затем в других ревизиях. Использование процесса восходящего проектирования и импорта разделов не поддерживается при этих условиях. Вам придется многократно использовать нисходящее проектирование с **Пустыми** разделами для имитации работы восходящего проектирования, так долго, пока вы не измените некоторые глобальные назначения между компиляциями. Все глобальные назначения должны быть одинаковыми для всех компилируемых разделов, так чтобы назначения воспроизводились в сопутствующем чипе после миграции.

Автономные компиляции HardCopy ASIC

Вы сможете использовать на выбор восходящую и нисходящую инкрементную компиляцию для автономных компиляций HardCopy ASIC.

Сохранение разводки не поддерживается для HardCopy ASIC. Поэтому, уровень сохранения Разводки и Размещения не доступен, а разводка не экспортируется в процессе восходящего проектирования.

Назначения, сделанные в исходном коде HDL в восходящем проектировании

Назначения, сделанные для I/O примитивов или атрибутов синтеза *altera_attribute HDL* в низкоуровневых разделах воспринимаются в верхнем уровне, но не применяются в головном **QSF** файле или **Редакторе назначений**. Эти назначения воспринимаются частью файлов исходных списков соединений. Вы можете управлять назначениями, сделанными в этих исходных файлах, изменением величины, с помощью назначений в головном проекте.

Ограничения в разделах мегафункций

Программа Quartus II не поддерживает разделы для блоков мегафункций. Если вы используете Менеджер Плагина MegaWizard™ для мегафункций различных вариаций и

генерируемый заголовочный файл блока мегафункции. Вы можете создать раздел для этого сгенерированного файла.

Программа Quartus II не поддерживает создание разделов для воображаемых мегафункций (это означает, когда программа подразумевает мегафункцию для размещения логики в вашем проекте). Если у вас есть модуль или блок для логики, которая будет подразумеваться, вам нужно создать раздел для этого иерархического уровня в проекте.

Программа Quartus II не поддерживает создание разделов для некоторой внутренней Quartus II иерархии, которая динамически генерируется во время компиляции для размещения содержимого мегафункций.

Упаковка регистров и границы разделов

Программа Quartus II выполняет автоматическую упаковку регистров во время компиляции. Однако когда разрешена инкрементная компиляция, логика в различных разделах не может быть упакована вместе, потому что границы разделов защищают от оптимизации между границами. Это ограничение относится ко всем типам упаковки регистров, включая I/O ячейки, блоки DSP, последовательной логики и несвязанной логики. Проще говоря, логика из двух разделов не может быть упакована в одном ALM.

Упаковка I/O регистров

Перекрестная упаковка между разделами регистров I/O допускается только в тех случаях, когда ваши входные и выходные выводы находятся в верхнем уровне иерархии (и головном разделе), а соответствующие регистры I/O находятся в других разделах.

Для перекрестной упаковки между разделами регистров для входного вывода требуются следующие обстоятельства:

■ входной вывод ведет в точности к одному регистру;

■ путь между входным выводом и регистром содержит только входные порты раздела, которые имеют только один выход ветвления.

Для перекрестной упаковки между разделами регистров для выходного вывода требуются следующие обстоятельства:

■ регистр имеет только один выходной вывод;

■ выходной вывод ведет единственный сигнал;

■ путь между регистром и выходным выводом содержит только выходные порты раздела, которые имеют только один выход ветвления.

Выходные выводы с сигналом разрешения не могут быть упакованы в I/O ячейку чипа, если логика разрешения выхода является частью выходного регистра другого раздела. Чтобы позволить упаковку регистров для выходных выводов с сигналом разрешения для выходного сигнала, структурируйте свой HDL код или назначения раздела проекта так, чтобы регистр и тристабильная логика была расположена в том же разделе.

Двунаправленные выводы должны быть выполнены так же, как и выходные выводы с сигналом разрешения. Если регистр и тристабильная логика находятся в одном разделе, то они могут быть упакованы.

Ограничения в применении тристабильной логики происходят из-за того, что I/O атомы (примитивы чипа) создаются в виде части раздела, которая содержит тристабильную логику. Если I/O регистр и его тристабильная логика находятся в одном разделе, то регистр вместе с тристабильной логикой будет упакован в I/O атом. Эти ограничения в применении между раздельной упаковки регистров также распространяются на I/O атомы для входных и выходных выводов. I/O атом должен соединяться с I/O выводом только одним сигналом. Путь между I/O атомом и I/O выводом должен состоять только из одного порта раздела, который имеет только

один выход ветвления. Обратитесь к главе "Лучшие примеры инкрементной компиляции разделов и назначения в архитектуре" в томе 1 "Настольной книги Quartus II" за дополнительной информацией и примерами упаковки I/O на межраздельных границах.

Ограничения в скриптах раздела восходящего проектирования

Программа Quartus II имеет некоторые ограничения в применении скриптов раздела восходящего проектирования.

Предупреждения о дополнительных тактах в отношении скриптов раздела восходящего проектирования

Сгенерированные скрипты содержат оперативную тактовую информацию обо всех тактовых сигналах в головном проекте. Большинство этих тактов могут не существовать в проектах нижнего уровня. Вы можете игнорировать эти предупреждения или ввести ваши ограничения, так чтобы эти сообщения не выводились.

Файл ограничения проекта Synopsys для временного анализатора TimeQuest в скриптах раздела восходящего проектирования

Как было описано в главе " Генерация скриптов раздела восходящего проектирования для менеджера проекта" на странице 2-40, скрипты раздела содержат только ограничения тактов и настройки минимальной и максимальной задержек для временного анализатора TimeQuest.

Настройки PLL и временные исключения не оставляются в скрипте разработчику нижнего уровня. Обратитесь к главе " Импорт SDC ограничений из низкоуровневого проекта" на странице 2-45 за советами по управлениями SDC ограничениями между головным и низкоуровневым проектами.

Поддержка дикой карты в скриптах раздела восходящего проектирования

Когда вы применяете ограничения с дикими картами, обратите внимание на то, чтобы дикие карты не анализировались вне иерархических границ. Например, назначения должны быть сделаны к этим узлам: *Top|A:inst|B:inst|*,* причём A и B – это низкоуровневые разделы, а B – дочерний раздел иерархии от A, так что B установлен в иерархии A. Это назначение применяется к модулям A и B, и ко всем дочерним блокам от B. Несмотря на это, назначение *Top|A:inst|B:inst** применяется к иерархии A, но не применяется к блокам иерархии B, поскольку описанный один уровень иерархии *B:inst** не раскрывает многих других уровней иерархии. Чтобы избежать этих проблем, следите за тем, чтобы дикая карта определялась к иерархическим границам, если вы хотите описать множество уровней иерархии. Когда вы используйте дикую карту для описания уровня иерархии, поддерживается только одна дикая карта. Это означает, что назначение *Top|A:inst|*|B:inst|** не поддерживается. Программа Quartus II в таких случаях выдаёт предупреждение.

Полученные такты и PLL в скриптах раздела восходящего проектирования

Если такты в головном разделе не подключены напрямую к выводу в разделе нижнего уровня, низкоуровневый раздел не принимает назначения и ограничения в скрипте раздела проекта от вывода в головном разделе.

Это особенно важно для тактовых выводов, которым требуются настройки временных ограничений и тактовых групп. Проблемы могут возникнуть, если ваш проект использует логику или инверсию для получения новых тактов от тактового вывода. Сделайте соответствующие временные назначения в вашем Quartus II проекте нижнего уровня, чтобы не было неограниченных тактов.

В дополнение, если вы используете PLL в вашем головном проекте и подключаете её к разделам нижнего уровня, то разделы нижнего уровня не владеют информацией об умножении или фазовом сдвиге внутри PLL. Сделайте соответствующие временные назначения в вашем Quartus II проекте нижнего уровня, чтобы не было неограниченных тактов или ограниченных некорректной частотой. С другой стороны, вы можете вручную продублировать логику получения тактов головного проекта или PLL в файле проекта нижнего уровня, чтобы иметь корректный множитель или величину фазового сдвига, компенсационные задержки и другие параметры PLL для полного и аккуратного временного анализа. Создайте раздел проекта для оставшейся логики раздела нижнего уровня, которая будет экспортироваться в головной проект. Когда низкоуровневый раздел закончен, экспортируйте только раздел, содержащий требуемую логику, с помощью средства, описанного в главе "Экспорт низкоуровневого блока внутри проекта" на странице 2-35.

Назначение выводов для блоков GXB и LVDS в скриптах раздела восходящего проектирования

Назначение выводов для высокоскоростных передатчиков GXB и блоков LVDS не описываются в скрипте. Вы должны вручную добавить назначения выводов для этих IP блоков в проекте нижнего уровня.

Временные назначения для виртуальных выводов в скриптах раздела восходящего проектирования

Скрипты раздела проекта используют назначения INPUT_MAX_DELAY и OUTPUT_MAX_DELAY для определения внешних задержек для разделов, ассоциированных с входными и выходными выводами, которые не могут быть видимы в проекте. Эти назначения требуют, чтобы программа определила тактовый домен для назначений и установила этому тактовому домену "*".

Это назначение тактового домена – это возможность ограничить некоторые пути и определить в отчёте временного анализа те, которые не требуются.

Чтобы ограничить, какие тактовые домены должны быть в этих назначениях, отредактируйте сгенерированный скрипт или измените назначения в вашем Quartus II проекте нижнего уровня. В дополнение, поскольку неизвестно ассоциация тактов с назначениями задержки, программа допускает наихудшую расфазировку, которая делает пути более критичными по времени, уменьшая значения задержки из скрипта. Если требуется, введите отрицательное число для входной и выходной задержек.

Порты головного проекта, ведущие к множеству выводов проектов нижнего уровня в скриптах раздела восходящего проектирования

Когда единственный I/O порт головного проекта ведёт к нескольким выводам в низкоуровневом модуле, это излишне ограничивает качество синтеза и размещения в нижнем уровне. Это происходит потому, что в проекте нижнего уровня программе нужно создать иерархические границы и не возможно использовать при этом информацию о выводах, имеющих логических эквивалент в верхнем уровне. Поскольку I/O ограничения переносятся от вывода в головном уровне каждому дочернему выводу, то получается большее количество выводов в

нижнем уровне, чем в верхнем. Эти выводы используют I/O ограничения и опции размещения из верхнего уровня, что делает невозможным размещение их в нижнем уровне. Программа обходит эту ситуацию насколько возможно, но лучше избегать такой практики проектирования, чем потенциально иметь такие проблемы. Реструктурируйте свой проект так, чтобы один порт I/O шёл к границе раздела, и разделялся на множество сигналов внутри низкоуровневого раздела.