Рекомендованные процессы проектирования и примеры компиляции

В этой главе описываются процессы проектирования для большинства решающих временных закрытий и командных разработок, использующих инкрементную компиляцию. Каждый процесс описан ситуацией, при которой его нужно использовать, и даны пошаговые инструкции команд, применяемых для реализации процесса.

Следующие примеры процесса нисходящего проектирования уменьшают время компиляции при инкрементном изменении проекта. Эти примеры проектирования позволят вам быстро достичь временного завершения путем оптимизации или сохранения результатов для каждого вашего раздела:

■ "Уменьшение времени компиляции при изменении исходного файла для одного раздела"
■ "Сохранение результатов для выбранных разделов перед добавлением других разделов" на странице 2-51
■ "Оптимизация размещения для критичных ко времени разделов для достижения временного завершения" на странице 2-50
■ "Инкрементная отладка с помощью Логического Анализатора SignalTap II" на странице 2-52

Все примеры допускают возможность использования процесса полной компиляции, используя шаги, описанные в главе "Руководство по быстрому старту – общий подход к процессу инкрементной компиляции " на странице 2-7.

Следующие примеры процесса восходящего проектирования иллюстрируют метод командного проектирования и повторное использование проекта:

■ "Выполнение командного проекта в восходящем проектировании" на странице 2-53
■ "Выполнение итерации проекта в восходящем проектировании" на странице 2-56
■ "Создание макроса на аппаратном уровне (или предкомпилированных блоков проекта) для повторного использования IP" на странице 2-57
■ "Использование экспортированного раздела для передачи проекта без включения исходных файлов" на странице 2-59

Уменьшение времени компиляции при изменении исходного файла для одного раздела

Используйте этот процесс для обновления исходного файла в одном разделе без перекомпиляции других частей проекта. Для сокращения времени компиляции, установите список соединений пост-компоновка для всех неизменяемых разделов. Этим вы сохраните характеристики этих блоков, за счет уменьшения дополнительных попыток временного закрытия.

Подоплека примера. Вы только что завершили громоздкую полную компиляцию проекта, состоящего из нескольких разделов. Ошибка была найдена в HDL исходном файле для одного раздела, она зафиксирована. Поскольку проект не имеет текущих временных ограничения, а фиксированная ошибка не оказывает влияние на временные характеристики, имеет смысл компилировать только ошибочный раздел и сохранить остальной проект без изменений.

Выполните следующие шаги для обновления только одного исходного файла:

1. Решите проблему и сохраните исходный HDL файл.
2. В меню **Назначения** выберите **Окно Разделов Проекта**.
3. Для разделов, которые будут сохранены, выберите тип списка соединений **Пост-компоновка**. Вы можете установить уровень сохранения разводки как **Размещение** или **Размещение и Разводка**. Для раздела, в котором будет исправляться ошибка, установите тип списка соединений **Исходный Файл**. (Делать это не обязательно, потому что программа Quartus II автоматически перекомпилирует раздел при изменении исходного файла).
4. Выберите **Старт Компиляции** для инкрементной компиляции исправленного кода HDL. Эта компиляция займет меньше времени, чем первая полная компиляция.
5. Снова запустите симулятор, чтобы проверить правильность исправления ошибки, и используйте отчет Временного Анализатора для проверки того, что результаты не ухудшились.

Оптимизация размещения для критичных ко времени разделов для достижения временного завершения

Используйте этот процесс для оптимизации результатов одного раздела, когда другие разделы проекта уже достигли собственных ограничений.

Подоплека примера. Вы только что завершили громоздкую полную компиляцию проекта, состоящего из нескольких разделов. В отчете Временного Анализатора сообщается, что временные ограничения не достигнуты, и вы хотите оптимизировать один конкретный раздел. Вы хотите испытать технику оптимизации, такую как Попытка размещения умножителей, разрешение Физического синтеза и запуск Проводника параметров проекта. Все эти методики занимают значительное время компиляции, поэтому их применяют только к определённым разделам в задании.

Выполните следующие шаги для сохранения результатов для разделов с достигнутыми временными ограничениями, и перекомпилируйте критичный раздел с новыми настройками оптимизации:

1. В меню **Назначения** выберите **Окно Разделов Проекта**.
2. Для разделов в задании установить **тип списка соединений Пост-синтез**, если вы изменяете настройки Компоновщика, такие как Попытка размещения умножителей. Это означает, что раздел будет размещён и разведён с новыми настройками компоновщика (но не будет синтезирован заново) во время следующей компиляции. Установите тип списка соединений Исходный файл, если вы изменили настройки оптимизации, которые влияют на синтез, такие как Оптимизация физического синтеза.
3. Для остальных разделов (включая и головной блок) установите **Тип списка соединений Пост-компоновка**. Установите **уровень сохранения компоновки** как **Размещение**, для более гибкой разводки. Эти разделы сохранятся во время следующей компиляции.
4. Примените желаемые настройки оптимизации.

5. Кликните **Старт Компиляции** для выполнения инкрементной компиляции проекта с новыми настройками. Во время этой компиляции, на стадии Объединения Разделов автоматически объединятся списки соединений пост-синтеза для критичных разделов со списками соединений пост-компоновка для остальных разделов. Компоновщик разведет только требуемый раздел. Так как общая работа сократилась по сравнению с первоначальной полной компиляцией, время компиляции также сокращается.

Для использования Проводника Параметров Проекта, выполните следующие шаги:

1. Повторите п.п. 1 – 3 из вышестоящего списка.
2. Сохраните проект и запустите Проводник Параметров Проекта.

Сохранение результатов для выбранных разделов перед добавлением других разделов

Используйте этот процесс для отдельной компиляции одного набора разделов и закрепления её результатов, когда вы комплектуете оставшийся проект.

Подоплёка примера. Для уменьшения времени компиляции и помощи в достижении временного закрытия, вы решаете использовать один из следующих процессов компиляции:

В первом варианте, критичные разделы размешаются и разводятся сами по себе, когда отдельная оптимизация включена (вручную или с помощью Проводника Параметров Проекта). Когда достигнуто временное закрытие для этого раздела, его содержание и размещение сохраняются, а остальные разделы компонуются с нормальным или уменьшенным уровнями оптимизации, так чтобы уменьшить время компиляции. Например, вы можете компилировать IP блок, который получен с инструкцией, выполнить оптимизацию прежде, чем объединять его с прочей логикой.

Во втором варианте, только быстро компилируемые разделы размещаются и разводятся с нормальным или уменьшенным уровнем оптимизации, используя назначения локализации архитектуры для сохранения пространства в архитектуре для разделов, которые будут добавлены после. Эти быстро компилируемые разделы сохраняются и не изменяются, пока не добавляется последний раздел, с отдельно включенной оптимизацией (вручную или с помощью Проводника Параметров Проекта).

Для использования этого процесса проектирования, выполните следующие шаги:

1. Разделите проект и создайте назначения локализации в архитектуре.
2. Для разделов, которые компилируются первыми, в меню **Назначения** выберите **Окно Разделов Проекта** и установите **тип списка соединений Исходный Файл**.
3. Для остальных разделов (других, которые напрямую или косвенно являются родительскими для разделов в п. 2) установите **тип списка соединений Пустой**.
4. Для компиляции с требуемой оптимизацией, кликните **Старт Компиляции**.
5. Проверьте отчет Временного Анализатора, для определения, какие временные характеристики достигнуты. Если так, то продолжите с п. 6, иначе повторите п.п. 4 и 5, пока характеристики не будут достигнуты.
6. В **Окне Разделов Проекта** установите тип списка соединений **Пост-компоновка** для первых разделов. Установите уровень сохранения компоновки как **разводка и размещение** только, если требуется сохранить результаты для критичных по времени блоков, иначе, используйте **Размещение** для достижения большей гибкости во время разводки.
7. Измените тип списка соединений с **Пусто** на **Исходный файл** для остальных разделов.

8. Установите соответствующий уровень оптимизации и скомпилируйте проект. Измените оптимизацию для точек, не имеющих влияние на компонованные разделы, поскольку каждому разделу установлен тип списка соединений **Пост-компоновка**.
9. Проверьте отчет Временного Анализатора, для определения, какие временные характеристики достигнуты. Если нет, сделайте изменения в проекте или настройках, и повторите п.п. 8 и 9, пока характеристики не сойдутся.

Этот процесс подобен восходящему проектированию, при котором модули разрабатываются независимо, и объединяются в последствии с остальным проектом. Главное, оптимизация в этом процессе делается только, если каждый критический путь содержится внутри одного раздела. Это ещё одна причина, по которой и вход, и выход раздела должны быть регистрированы. Даже если некоторые определенные в проекте файлы отсутствуют, создавайте заголовочный файл для резервации места, который определяет интерфейс портов. Обратитесь к главе "Пустые разделы" на странице 2-22 за дополнительной информацией.

Инкрементная отладка с помощью Логического Анализатора SignalTap II

Инкрементная компиляция позволяет вам сохранять результаты синтеза и компоновки вашего оригинального проекта и добавлять Логический Анализатор SignalTap II без перекомпиляции оригинального исходного кода.

Используйте этот процесс для сокращения времени компиляции во время добавления логического анализатора для отладки вашего проекта, или когда вы хотите модифицировать конфигурацию файла SignalTap II без изменения логики проекта или её размещения.

Для использования модуля инкрементной компиляции SignalTap II не требуется создавать какие-нибудь разделы. Если в вашем проекте по умолчанию установлено использование полной инкрементной компиляции, Логический Анализатор SignalTap II действует как собственный отдельный раздел проекта.

Для использования логического анализатора SignalTap II в процессе инкрементной компиляции, выполните следующие шаги:

1. В меню **Назначения** выберите **Окно Разделов Проекта**.
2. Установите список соединений **Пост-компоновка** для сохранения размещения всех разделов. Тип списка соединений для головного раздела по умолчанию - **Исходный файл**, чтобы иметь возможность добавлять другие разделы проекта, которые вы создадите.
3. Если у вас еще выполнили компиляцию с текущим размещением разделов, выполните полную компиляцию. Если полная компиляция готова, то проект готов к добавлению Логического Анализатора SignalTap II.
4. Установите ваш файл SignalTap II, используя SignalTap II фильтр **пост-компоновка** в **Поиске Узлов,** для добавления сигналов логического анализа. Это позволит Компоновщику добавить логику SignalTap II в список соединений пост-компоновка без изменения результатов проекта. Для добавления сигналов из списка соединений пре-синтеза, настройте ваш файл SignalTap II, используя SignalTap II фильтр **пре-синтеза** в **Поиске Узлов**. Это позволит программе повторно синтезировать раздел и подключиться к узлам пре-синтеза, которые вы выбрали. В этом случае, раздел перекомпонуется, так что размещение будет отличаться от предыдущих результатов разводки.

Не используйте тип списка соединений **пост-синтез** вместе с логическим анализатором SignalTap II.

За подробной информацией о настройках логического анализатора SignalTap II, обратитесь к тому 3 "Настольной книги Quartus II" "Отладка проекта, используя встроенный логический анализатор SignalTap II".

Выполнение командного проекта в восходящем проектировании

Этот пример описывает использование инкрементной компиляции в восходящем проектировании.

Подоплека примера. Проект состоит из нескольких низкоуровневых подпроектов, которые разрабатывались различными разработчиками. Каждый блок головного проекта состоит только из одного такого подпроекта. Разработчики подпроектов хотят независимо оптимизировать свои проекты и предоставить руководителю проекта результаты оптимизации.

Руководителю проекта, в этом случае, необходимо выполнить следующие шаги для успешной подготовки проекта к методу восходящего проектирования:

1. Создайте новый проект Quartus II для сбора полной информацию о конечном проекте.
2. Для подготовки метода восходящего проектирования, создайте "каркас" проекта, в котором определена иерархия подпроектов, которые будут разрабатываться независимыми разработчиками. Головной проект состоит из головной модуля и блоков заголовочных файлов, которые представляют каждый, пока не реализованный, подпроект путём определения ему только интерфейсных портов.
3. Сделайте общие настройки в проекте. Выберите чип, сделайте глобальные назначения для тактовых цепей и I/O портов чипа, а также, сделайте глобальные ограничения сигналов, чтобы определить, каким сигналам будет дозволено использовать глобальные ресурсы трассировки.
4. Сделайте назначения раздела проекта для каждого подпроекта, а также установите тип списка соединений **Пусто** для каждого раздела, который будет импортирован, в **Окне Разделов Проекта**.
5. Создайте регионы LogicLock для каждого низкоуровневого раздела для создания архитектуры проекта. В этой архитектуре будут рассматриваться соединения между разделами, и рассчитываться размер каждого раздела на основании начальной количественной реализации и известной спецификации проекта.
6. В меню **Проект**, выберите **Генерация Скриптов разделов проекта в восходящем проектировании**, или запустите генератор скриптов Tcl или командную строку.
7. Установите опции скрипта по умолчанию. Altera рекомендует вам оставить все ограничения назначений по умолчанию, включая регионы LogicLock, для всех разделов и локализации виртуальных выводов. В дальнейшем, Altera рекомендует вам добавить максимальную задержку временных ограничений для виртуальных выводов I/O в каждом разделе, чтобы облегчить временное завершение во время интеграции в головной проект. Если проекты нижнего уровня не закончены другими разработчиками, используйте скрипты раздела для установки проектов, которые проще получить с помощью создателей файлов.
8. Передайте всем разработчикам проектов нижнего уровня файлы Tcl для создания проектов с оговоренными ограничениями. Если вы используете создатель файлов, подготовьте такой файл для каждого раздела.

Если вы - разработчик подпроекта нижнего уровня, выполните следующие шаги для успешного экспорта своего проекта, в зависимости от того, какой из двух способов использует ваша команда разработчиков: создатель файлов или ручную операцию экспорта/импорта.

Если вы используете **создатель файлов**, выполните следующие шаги:

1. Используйте команду **сделать** и файл, созданный руководителем проекта, чтобы создать проект Quartus II со всеми ограничениями для проекта, затем откомпилируйте проект.
2. Информация о том, какой исходный файл должен быть ассоциирован с которым разделом, не доступна программе по умолчанию, вам необходимо определить эту информацию в создателе файлов. Вы сможете определить зависимости перед тем, как программа соберет проект после начального вызова **создателя файла**.
3. После того, как вы достигли желаемых результатов компиляции, и проект готов к импорту в головной проект, руководитель проекта использует команду ***master_makefile*** для экспорта этого низкоуровневого раздела и создает файл **.qxp**, чтобы затем импортировать его в головной проект.

Если вы не используете **создатель файлов**, выполните следующие шаги:

1. Создайте новый Quartus II проект для подпроекта.
2. Сделайте назначения для регионов LogicLock и глобальные назначения (включая настройки тактовых сигналов), которые определены руководителем проекта.
3. Сделайте назначения для виртуальных выводов, которые описывают соединения с основной логикой вместо внешних выводов на чипе в головном модуле.
4. Сделайте назначения локализации в архитектуре для виртуальных выводов, так чтобы разместить их в определенных в головном модуле регионах. Это предоставляет Компоновщику больше информации о временных ограничениях между модулями. Альтернативно, вы можете применить временные ограничения I/O для путей, которые соединены с виртуальными выводами.
5. Возобновите компиляцию и оптимизацию проекта, если требуется.
6. После того, как вы достигли желаемых результатов компиляции, в меню **Проект**, выберите **Экспорт Раздела Проекта**. Появится вкладка **Экспорта Раздела Проекта**.
7. Под **Списком Соединений для экспорта**, выберите список соединений **Пост-компоновка** для сохранения размещения и рабочих характеристик подпроекта, и включите **Экспорт разводки**, для добавления информации о разводке, если требуется. С другой стороны, вы можете экспортировать список соединений **Пост-синтеза**, если не требуется сохранение размещения и рабочих характеристик.
8. Передайте файл .**qxp** руководителю проекта.

В завершении, руководитель проекта этого примера, выполняет следующие шаги для импорта файлов, присланных каждым разработчиком подпроекта.

Если вы используете **создатель файлов**, выполните следующие шаги:

1. Используйте команду ***master_makefile*** для экспорта каждого низкоуровневого раздела и создайте файлы **.qxp**, и затем импортируйте их в головной проект.
2. Программа не владеет информацией о том, какие исходные файлы должны быть ассоциированы с каким разделом, так что вам нужно учесть эту информацию в **создателе файлов**. Программа не может собрать проект, если исходные файлы каким-то образом изменяют определенные вами зависимости.

Если вы не используете **создатель файлов**, выполните следующие шаги:

1. После получения файла **.qxp** для каждого подпроекта от других разработчиков из команды, в меню **Проект**, выберите **Импорт Раздела Проекта** и определите, какие разделы в головном проекте описаны файлом подпроекта **.qxp**.
2. Повторите процесс импорта, описанный в первом пункте, для каждого раздела проекта. После каждого нового импорта каждого раздела, выбирайте все разделы и используйте опцию **Реимпорт, используя последние файлы импорта из прежнего места,** для импорта всех файлов одновременно из предыдущего их положения.

Разрешение конфликтов назначений во время импорта

Во время импорта подпроекта, руководитель проекта может получать извещения об определенных конфликтах назначений. Они встречаются, например, если разработчики подпроектов изменяют определенные регионы LogicLock за счет дополнительной логики или условия размещения, или если разработчики добавляют портам I/O временные условия, которые отличаются от условий, установленных руководителем проекта в головном проекте. Для определения источника конфликтов, руководитель проекта может использовать один из этих способов:

■ Разрешить импорт новых назначений
■ Разрешить переместить или изменить существующие назначения

Если встречаются конфликты назначений регионов LogicLock, руководитель проекта может выполнить одно из следующих действий:

■ Позволить импортируемому региону заменить существующий регион
■ Позволить импортируемому региону обновить существующий регион
■ Пропустить импорт назначений для конфликтных регионов

Руководитель проекта может найти все эти ситуации, используя **Расширенные Настройки Импорта**, как описано в главе "Импорт назначений и расширенный импорт настроек" на странице 2-38.

Если конфликтует размещение различных подпроектов, руководитель проекта может установить **Уровень Сохранения Компоновки** раздела **Только Список Соединений**, позволяя тем самым программе по новому выполнить размещение и разводку импортированного списка соединений.

Импортирование раздела для многократного использования

Это вариация примера, один из подпроектов неоднократно используется в головном проекте. Разработчик подпроекта желает компилировать и оптимизировать блок только как низкоуровневый проект, а затем импортировать результаты в виде нескольких разделов в головной проект.

В этом случае, решение конфликта размещения, как описано в главе "Разрешение конфликтов назначения во время импорта" на странице 2-55, обязательно, потому что разделы верхнего уровня используют импортированные списки соединений пост-компоновка. Если вы импортируете несколько блоков из подпроекта в головной проект, импортированным регионам LogicLock устанавливается автоматически статус **Плавающий**.

Если вы решаете конфликты вручную, вам необходимо использовать опции импорта и ручную настройку регионов LogicLock для определения размещения каждого блока в головном проекте.

Выполнение итерации проекта в восходящем проектировании

Используйте этот процесс, если вы заново оптимизируете низкоуровневые разделы в восходящей компиляции путём включения дополнительных ограничений из интегрируемого головного проекта.

Подоплека примера. Проект состоит из нескольких низкоуровневых подпроектов, которые были экспортированы из отдельных проектов Quartus II и были импортированы в головной проект в ходе восходящей компиляции. В этом примере, интеграция в верхний уровень не состоялась, потому что временные ограничения не достигнуты. Временные ограничения достигнуты в каждом индивидуальном низкоуровневом проекте, но критические пути между разделами в верхнем уровне дали сбой во временных ограничениях.

После изнурительных попыток оптимизировать головной проект, руководитель проекта определяет, что проект не может достигнуть временных ограничений, полученных после размещения текущего импортированного раздела. Руководитель проекта решает предоставить дополнительные ограничения низкоуровневому проекту для улучшения его размещения.

Для выполнения этого процесса проектирования, выполните следующие шаги:

1. В головном проекте, в меню **Проект** выберите **Генерация Скриптов разделов проекта восходящего проектирования**, или запустите генератор скрипта Tcl или командную строку.
2. В виду того, что низкоуровневый проект для каждого раздела уже создан, выключите опцию **Создать низкоуровневый проект, если он ещё отсутствует**.
3. Сделайте дополнительные изменения в опциях скрипта по умолчанию, если требуется. Altera рекомендует вам оставить все установки по умолчанию, включая регионы LogicLock, для всех разделов и назначения локализации виртуальных выводов. Сверх того, Altera рекомендует вам добавить ограничение максимальной временной задержки для виртуальных выводов I/O каждого раздела.
4. Программа Quartus II генерирует Tcl скрипты для всех разделов, но в этом примере, вы должны сосредоточиться на разделах, к которым подводятся критические пути. Следующие назначения импортируются в скрипте:
   ■ Назначения виртуальных выводов для выводов модуля, которые не подключены к I/O портам в головном
   ■ Ограничения локализации для виртуальных выводов, которые отображают начальное размещение в головном проекте исходных выводов или выводов назначений. Это помогает сделать "сознательное" место низкоуровневого раздела в окружающих соединениях головного проекта, давая больший шанс достигнуть временного закрытия во время интеграции в верхний уровень
   ■ Временные ограничения INPUT_MAX_DELAY и OUTPUT_MAX_DELAY для путей от и к I/O выводам раздела. Это ограничение выводов для оптимизации временных путей от и к выводам.
5. Низкоуровневые разработчики используют файл, разработанный руководителем проекта.
   ■ Для получения Tcl скрипта в GUI Quartus II, в меню **Инструментарий** выберите **Окно Утилиты** и откройте **консоль Tcl**. Для навигации по директории скриптов наберите команду:

```
source <filename> ↵
```

   ■ Для получения Tcl скрипта из системной командной строки наберите следующую команду:

```
quartus_cdb -t <filename>.tcl ↵
```

6. Разработчики проектов нижнего уровня компилируют свои проекты с новыми назначениями и используют внешние временные ограничения.
7. Разработчики проектов нижнего уровня экспортируют снова свои результаты.
8. Руководитель проекта снова импортирует эти результаты.
9. Вы можете снова анализировать проект для определения следующих временных ограничений. Поскольку низкоуровневые разделы компилировались с большей информацией о соединениях в головном проекте, очень вероятно, что пути между разделами размещены лучше, что поможет достигнуть временных ограничений.

Создание макроса на аппаратном уровне (или предкомпилированных блоков проекта) для повторного использования IP

Используйте этот процесс проектирования для создания макроса на аппаратном уровне или предкомпилированного блока IP, который будет использоваться в головном проекте. Этот процесс дает возможность экспорта блоков проекта с информацией пост-синтеза или размещения (и разводки) и импортировать любое количество копий этого предкомпилированного макроса в других проектах.

Подоплека примера. Поставщик IP хочет производить и продавать ядро IP для компонента, которое будет использоваться в системах высокого уровня. Поставщик IP хочет оптимизировать размещения этого блока для достижения максимальной характеристики на специальном чипе Altera и поэтому передает информацию о размещении конечному заказчику. Для сохранения этого IP, необходимо поддерживать посылку скомпилированного списка соединений, состоящего из поставляемого заказчику исходного HDL кода.

Заказчик сначала определяет тип микросхемы Altera, которую он будет использовать в этом проекте, и поставляет спецификацию проекта.

Поставщик IP в этом примере, должен выполнить следующие шаги для экспорта предразмещённого IP ядра (или аппаратного макроса):

1. Создать заголовочный файл, который определяет интерфейс портов для ядра IP и передать файл заказчику, чтобы он предусмотрел пустой раздел в головном проекте.
2. Создать проект Quartus II для ядра IP.
3. Создать регион LogicLock для иерархии проекта, которая будет экспортирована.

   Altera рекомендует создавать архитектуру, используя регионы LogicLock, несмотря на то, что это не требуется для генерации и использования .**qxp** файлов. Использование регионов LogicLock для IP ядер позволяет заказчику создать пустой охраняемый регион для сохранения места под IP в архитектуре проекта. Этим достигается отсутствие конфликтов с логикой в головном проекте, и ядро IP не будет влиять на временные характеристики другой логики в головном проекте.

   Регионы LogicLock – эффективное средство уменьшения конфликтов в использовании ресурсов, они позволяют сохранить рабочие характеристики. Дополнительно, без регионов LogicLock, размещение может быть определено только в окончательном виде. С регионами LogicLock, вы можете сохранить размещение постоянно или временно в границах ассоциированного региона. Это важно, когда файл .**qxp** импортируется для различных иерархий в одном проекте, в таком случае, размещение не менее одного блока в головном проекте не должно иметь такое же размещение, какое использует поставщик IP ядра.

4. Если требуется, добавьте другую логику (например, PLL или другую логику, которая определена в проекте заказчика) вокруг иерархии экспортируемого проекта. Если вы сделали так, создайте раздел проекта для иерархии проекта, чтобы она была экспортирована как ядро IP. За подробной информацией обратитесь к главе "Экспорт низкоуровневого блока внутри проекта" на странице 2-35.
5. Оптимизируйте проект и получите временное закрытие, определённое в спецификации проекта.
6. Экспортируйте соответствующий уровень иерархии в единый файл .**qxp**. После успешной компиляции проекта, вы сможете сгенерировать файл .**qxp** из **GUI**, командной строки или командой Tcl:
   ■ Если вы используете GUI, используйте команду **Экспорт Раздела Проекта**
   ■ Если вы используете командную строку, запустите *quartus_cdb* с опцией:
   ```
   --incremental_compilation_export
   ```
   ■ Если вы используете команды Tcl, выполните следующую команду:
   ```
   execute_flow -incremental_compilation_export
   ```
7. Передайте файл .**qxp** заказчику. Проследите за тем, чтобы вы не передали ему исходный код вашего проекта; список соединений проекта и информация о размещении и разводке находятся в одном этом файле.

Заказчик в этом примере вводит IP ядра в свой проект, выполняя следующие шаги:

1. Создать Quartus II проект для головного проекта, в котором определяется тип чипа и устанавливаются одна или несколько копий IP ядра. Используйте заголовочный файл чёрных ящиков для определения интерфейса портов с ядром IP.
2. В меню **Процессы**, выберите **Старт** и кликните **Выполнить Анализ и Разработку** для определения иерархии проекта.
3. Создайте разделы проекта для каждого блока IP ядра (обратитесь к главе "Создание разделов проекта" на странице 2-69) с типом списка соединений **Пусто** (обратитесь к главе "Установка типов списка соединений для разделов проекта" на странице 2-19).
4. Вы сможете продолжить работу над своей частью проекта и подключить IP ядро от поставщиков IP, когда оно будет готово.
5. Импортировать файл .**qxp** от IP поставщика для определенной иерархии разделов. Вы можете импортировать файл .**qxp** из **GUI**, из командной строки или с помощью команд Tcl.
   ■ Если вы используете GUI, используйте команду **Импорт Раздела Проекта**
   ■ Если вы используете командную строку, запустите *quartus_cdb* с опцией:
   ```
   --incremental_compilation_import
   ```
   ■ Если вы используете команды Tcl, выполните следующую команду:
   ```
   execute_flow -incremental_compilation_import
   ```
6. Вам нужно установить импортируемым регионам LogicLock **плавающую** локализацию или переместить их в новое место, с сохранением взаимного расположения содержимого регионов. Информация о разводке сохраняется всегда, когда это возможно.

Компоновщик игнорирует назначения взаимного размещения, если размещение локализации LogicLock регионов в головном проекте не соответствует локализации, экспортируемой в файле .**qxp**.

7. Вы должны контролировать для дальнейшего сохранения типа списка соединений только размещение, или только размещение и разводка (если эта информация содержится в файле .**qxp**) с помощью **Уровня Сохранения Разводки**.

По умолчанию, программа сохраняет полностью размещение и разводку для всех узлов в импортируемом списке соединений, если выбрано сохранение размещения и разводки. С другой стороны, если вы используете одинаковые файлы .**qxp** для нескольких разделов в одном проекте, программа сохраняет взаимное расположение для каждого импортируемого модуля (относительно направляющих региона LogicLock).

Если поставщик IP не определил регион LogicLock в экспортируемом разделе, программа сохраняет абсолютную локализацию размещения и это ведет к конфликтам размещения, если разделы импортируются больше чем для одного блока.

Использование экспортированного раздела для передачи проекта без включения исходных файлов

Используйте этот процесс для свертки целого проекта в один файл отправки конечному заказчику или в другое место разработки.

Подоплека примера: Разработчик хочет создать блок проекта, который требуется переслать в другой проект, но сохраняя его IP, которое предполагает отправку получателю синтезированного списка соединений из исходного HDL кода.

Отправитель в этом примере должен выполнить следующие шаги для экспорта блока проекта:

1. Предоставить отправителю название семейства устройства. Если вы посылаете информацию о размещении с помощью синтезированного списка соединений, предоставьте также точную информацию о выбранном чипе, чтобы выровнять установки проекта.
2. Создать заголовочный файл чёрного ящика, который определяет интерфейс портов для блока проекта, и предоставить её получателю, чтобы тот создал модуль из пустого раздела в головном проекте.
3. Создать проект Quartus II для блока проекта и завершить его разработку.
4. Экспортировать определенный уровень иерархии в один файл .**qxp**. Если вы используете Quartus II GUI, воспользуйтесь командой **Экспорт Раздела Проекта** (смотрите "Экспорт низкоуровневых разделов для использования в проекте верхнего уровня" на странице 2-34).
5. Выберите опцию для включения только **списка соединений пост-синтеза**, если вам не требуется пересылать информацию о размещении. Если получатель хочет воспроизвести ваши результаты в архитектуре, вы можете выбрать опцию **пост-компоновка** и опционально разрешить **экспорт разводки**.
6. Передайте файл .**qxp** получателю. Обратите внимание на то, чтобы не отправить кое-что из исходного кода проекта.

Получатель в этом примере, сначала создает проект Quartus II для вашего головного проекта и следит за тем, чтобы ваш проект был определён для одного чипа (или нескольких чипов внутри семейства, если .**qxp** файл не содержит информации о размещении), как условлено поставщиком IP, приславшим вам блок проекта. Установите блок проекта, используя предоставляемую информацию о портах. Затем, объедините блок проекта с головным проектом, выполняя одну из следующих процедур.

Для использования .**qxp** файла в качестве файла проекта в вашей разработке, просто добавьте .**qxp** файл поставщика IP в качестве исходного файла в вашем проекте Quartus II. Когда вы используете .**qxp** файл в качестве исходного файла, то в этом случае вы не можете импортировать другую информацию списка соединений пост-компоновки. Вам нужно выбрать, хотите ли вы иметь файл в виде раздела в головном проекте.

Для импорта модуля проекта из .**qxp** файла в качестве раздела проекта и дополнительно включая информацию пост-компоновки, выполните следующие шаги:

1. В меню **Процессы**, выбрать **Старт** и кликнуть **Выполнить Анализ и Разработку** для определения иерархии проекта.
2. Создать раздел проекта для блока проекта (глава "Создание назначений разделов проекта" на странице 2-16) с типом списка соединений **Пусто** (глава "Установка типа списка соединений для разделов проекта" на странице 2-19).
3. Импортировать файл .**qxp** от поставщика IP для соответствующей иерархии разделов. Если вы используете Quartus II GUI, используйте команду **Импорт Раздела Проекта** и укажите нахождение файла **.qxp** (глава "Использование .qxp файла в качестве исходного файла в головном проекте" на странице 2-36).
4. Если поставщик передает информацию Компоновщика, вы сможете проконтролировать её с помощью **Уровня Сохранения Компоновки**: только размещение, или только размещение и разводку.