Топологическая и временная оптимизация проектов на ПЛИС XILINX

(окончание, начало в КиТ №1)

Владимир Капитанов

capt@scan.voronezh.su

Павел Володин

Размещение упакованных макросов между собой

Размещение упакованных макросов между собой производится аналогично размещению упакованных CLB в макросах. Отличие состоит лишь в том, что роль CLB здесь выполняют предварительно упакованные макросимволы, а размещение выполняется без абсолютной привязки макросов к физическим ресурсам ПЛИС (рис. 1) — имеет место относительная ориентация макросов между собой. Необходимо обратить внимание, что при необходимости указания абсолютного местоположения CLB или макроса следует использовать атрибут RLOC_ORIGIN.

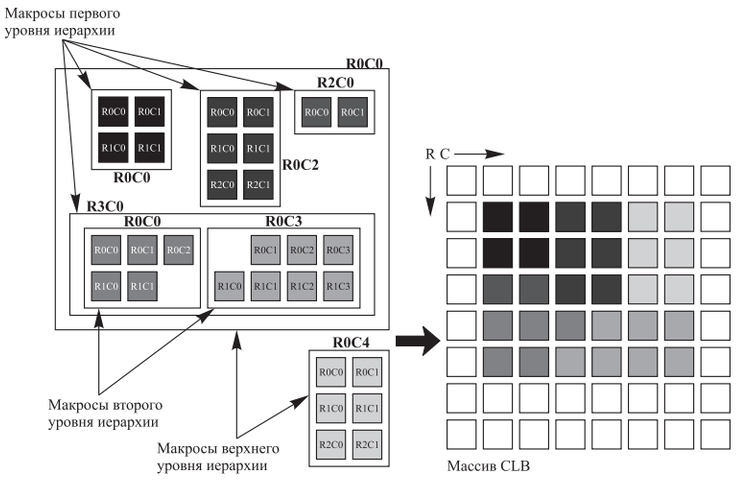

Рис. 1. Третий этап топологической оптимизации проекта

Здесь необходимо сделать несколько пояснений:

1. В пределах одного макроса размещение производится относительно элемента с атрибутом RLOC=R0C0, который будет размещен в верхнем левом углу — точке привязки.
2. Элемент с атрибутом RLOC=R0C0 может отсутствовать. В этом случае размещение производится относительно верхнего левого угла — точки привязки.

Топологическую оптимизацию проекта на этапе размещения CLB по кристаллу непосредственно после программы Map достаточно быстро и легко можно выполнять с помощью очень удобной программы Floorplanner пакета Foundation. Пример окна программы приведен на рис. 2. Всем начинающим разработчикам на ПЛИС мы настоятельно рекомендуем использовать Floorplanner в работе над высокоскоростными проектами. Полное описание методики проектирования с использованием Floorplanner приведено в сопутствующей пакету Foundation документации.

Как уже было сказано выше, очень хорошие результаты при проектировании дает временная оптимизация. Поэтому следует более подробно остановиться именно на ней. Задание временных ограничений на проект можно осуществить несколькими путями: непосредственно в схемотехническом редакторе (через символы TIMESPEC и TIMEGRP и атрибут TNM), в UCF-файле (Users Constraints File) или посредством редактора ограничений пакета Foundation (Constrains Editor), который в свою очередь модифицирует UCF-файл. Поскольку во всех перечисленных подходах синтаксис конструкций временной оптимизации остается неизменным, а изменяется лишь их графическая интерпретация, то дальнейшее рассмотрение основных моментов методики оптимизации не будет строго привязано к какому-либо из подходов. Ввиду ограниченности объема статьи подробно описать все варианты синтаксиса временных ограничений не представляется возможным, поэтому остановимся на самых основных и наиболее наглядных.

Все временные ограничения накладываются на так называемые временные группы, включающие в свой состав как выборочные элементы схемы, в простейшем случае отдельные цепи, так и все элементы определенного типа всего проекта, например, триггеры, ОЗУ и т. д.

Временные группы могут быть объявлены двумя способами:

1 способ: привязка осуществляется к имени цепи: вся логика, подключенная к цепи '*my_net*', будет объединена во временную группу 'logic_grp'. Например:

NET *my_net* TNM_NET = logic_grp.

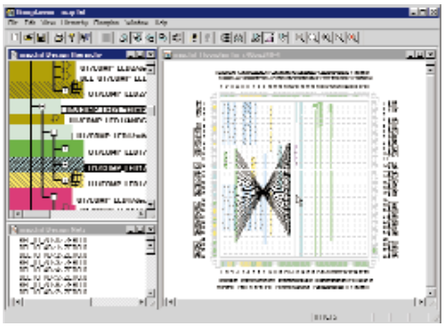

Рис. 2. Пример окна программы Floorplanner

2 способ: использование ключевого слова 'TIMEGRP'. Например, чтобы создать временную группу 'group_name', которая включает в себя все триггеры внутри иерархического символа U1, необходимо использовать следующую конструкцию:

TIMEGRP group_name = FFS («U1/*»).

Создание временных групп может быть очень полезным, если необходимо получить проект определенного быстродействия, например, в случае использования конвейерной обработки. При этом более удобным является второй способ объявления временных групп.

Временные ограничения на внутренние задержки

В этом случае ограничения накладываются на распространение сигнала с выхода одного регистра, тактируемого сигналом *clock*, до выхода другого регистра, также тактируемого сигналом *clock*, с учетом задержек на логике (рис. 3).

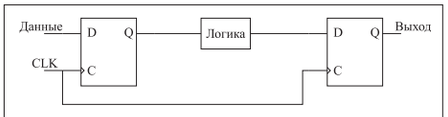

Рис. 3. Ограничение задержек «от триггера до триггера»

Конструкция «PERIOD». Она позволяет оптимизировать все пути, начинающиеся и заканчивающиеся на регистрах-триггерах, синхронном ОЗУ или регистрах-защелках и тактируемых эталонным сигналом *clock*. При этом учитывается необходимое время установления (Setup-time) всех сигналов. Например:

NET *clock* PERIOD = 40ns.

Конструкция «FROM:TO». Данная конструкция может использоваться для оптимизации задержек между элементами ранее объявленных временных групп. Существуют предопределенные временные группы: RAMS — синхронное ОЗУ, FFS — триггеры, LATCHES — регистры-защелки, PADS — контакты. Например:

TIMESPEC TSF2F = FROM:FFS:TO:FFS = 25ns.

Рассмотрим более подробный пример использования конструкции FROM:TO.

TIMEGRP RFFS = RISING FFS («*») — создание временной группы триггеров, переключающихся по переднему фронту тактового сигнала;

TIMEGRP FFFS = FALLING FFS («*») — создание временной группы триггеров, переключающихся по заднему фронту тактового сигнала;

TIMESPEC TS001 = FROM:RFFS:TO:FFFS = 45ns — временная оптимизация задержек от триггеров, переключающихся по переднему фронту, до триггеров, переключающихся по заднему фронту;

TIMESPEC TS002 = FROM:FFFS:TO:RFFS = 45ns — временная оптимизация задержек от триггеров, переключающихся по заднему фронту до триггеров, переключающихся по переднему фронту;

TIMESPEC TS003 = FROM:FFS:TO:FFS = 45ns — временная оптимизация задержек от триггеров до триггеров, переключающихся по одинаковому фронту.

Если в проекте используется несколько тактовых сигналов, то необходимо воспользоваться комбинацией конструкций «PERIOD» и «FROM:TO»:

NET *clock1* TNM_NET = clk1_grp;
NET *clock2* TNM_NET = clk2_grp;
TIMESPEC TS_clk1 = PERIOD:clk1_grp:50ns;
TIMESPEC TS_clk2 = PERIOD:clk2_grp:30ns;
TIMESPEC TS_clk1_clk2 FROM:clk1_grp:TO: clk2_grp:50ns;
TIMESPEC TS_clk2_clk1 FROM:clk2_grp:TO: clk1_grp:30ns.

Ограничение выходных задержек

В данном случае производится оптимизация задержек от триггеров до PAD (рис. 4).

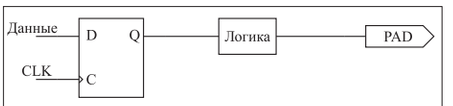

Рис. 4. Оптимизация задержек от триггеров до PAD

Конструкция «FROM:TO». Данная конструкция анализирует задержки от выхода регистра до PAD, но не учитывает задержку распространения тактового сигнала. Поэтому рекомендуется пользоваться конструкцией «OFFSET», которая в этом случае является более корректной.

TIMESPEC TSF2P = FROM:FFS:TO:PADS:25ns.

Конструкция «OFFSET». Данная конструкция автоматически учитывает задержку тактового сигнала на тактовом буфере и цепях распространения. Например:

NET *out_net_name* OFFSET = OUT 25 AFTER *clock_net_name* — ограничение задержки на уровне 25 нс.

Ограничение входных задержек

В данном случае производится оптимизация задержек от PAD до триггеров (рис. 5).

Если требуется учитывать задержку распространения тактового сигнала, то необходимо воспользоваться конструкцией OFFSET:

NET *in_net_name* OFFSET = IN 25 BEFORE *clock_net_name* — максимальное время установления — 25 нс.

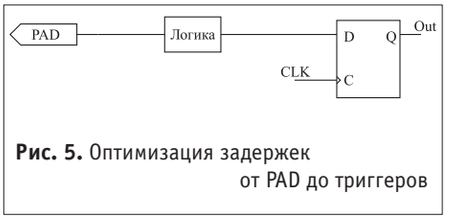

Рис. 5. Оптимизация задержек от PAD до триггеров

Конструкция «FROM:TO».

TIMESPEC FROM:PADS:TO:FFS:25ns — ограничение задержки на уровне 25 нс.

Здесь не учитываются задержки распространения тактового сигнала, поэтому рекомендуется использовать конструкцию «OFFSET», которая является более корректной.

Если в проекте отсутствует тактовый сигнал, но есть необходимость в ограничении задержек распространения сигналов, можно воспользоваться следующей конструкцией:

TIMESPEC TSP2P = FROM:PADS:TO:PADS: 150ns — ограничение задержек на уровне 150 нс.

Исключение путей из временной спецификации

Иногда требуется, чтобы какой-либо путь был исключен из временной спецификации. Для этого необходимо воспользоваться ключевым словом «TIG». Например:

NET : *reset_n* : TIG — исключение цепи *reset_n*;

NET : *mux_mem/data_reg : TIG** — исключение шины *data_reg[7:0]* в макросе mux_mem;

NET : *mux_mem/data_reg : TIG = TS01** — исключение шины data_reg[7:0] в макросе mux_mem из спецификации с именем TS01.

NET : *data?_sig* : TIG — исключение цепей *data1_sig* и *data2_sig*.

Временные ограничения для цепей разного быстродействия

Если в проекте некоторые выходные цепи являются менее быстродействующими по сравнению с другими, то их можно включить в отдельную временную группу и назначить на нее TIMESPEC. Например:

TIMEGRP slow_outs = PADS (*out_data : *irq_n*)** — создание временной группы slow_outs и включение в нее шины *out_data[7:0]* и цепи *irq_n*;

TIMEGRP fast_outs = PADS : EXCEPT : slow_outs — создание временной группы fast_outs и включение в нее остальных выходных цепей;

TIMESPEC TS08 = FROM:FFS:TO:fast_outs: 22ns;

TIMESPEC TS09 = FROM:FFS:TO:slow_outs: 65ns — задание временных ограничений.

Если в проекте есть пути «от триггера до триггера» различного быстродействия, то для них можно создать временную группу, используя конструкции TIMEGRP или TNM. Например:

TIMEGRP slowffs = FFS (*mux_mem/ff_q_ out_net1 : mux_reg/ff_q_out_net2**)** или

INST *mux_mem/ff_q_out_net1 TNM = slowffs;**

INST *mux_reg/ff_q_out_net2 TNM = slowffs.**

В том случае, когда все триггеры одного быстродействия имеют общую цепь (например, clock_enable), то можно воспользоваться следующей конструкцией:

NET *ff_clock_enable_net* TNM = slowffs.

После этого объявленные временные группы используются следующим образом:

TIMESPEC TS10 = FROM : slowffs : TO : FFS : 100ns — ограничение задержек от триггеров группы slowffs до остальных триггеров на уровне 100 нс.

Ограничение задержек по цепям

Для этого используются две конструкции:

NET *any_net_name* MAXDELAY = 20ns — ограничение задержки по цепи *any_net_name* на уровне 20 нс;

NET *any_net_name* MAXSKEW = 5ns — ограничение максимального разброса времени распространения сигнала по цепи *any_net_name* значением 5 нс.

Как уже подчеркивалось ранее, задать временные ограничения на проект возможно тремя путями: первые два являются классическими с самого начала выпуска ПО проектирования ПЛИС под Windows, а вот новый программный продукт Xilinx Constraints Editor появился совсем недавно и в настоящее время поставляется в комплекте пакета Foundation версии 2.1i. Пример окна задания временных ограничений Constraints Editor приведен на рис. 6. Данное средство достаточно прогрессивно и позволяет значительно упростить временную оптимизацию сложных проектов с большим количеством декларируемых временных групп. Более подробное описание работы с программой Constraints Editor можно найти в сопутствующей пакету Foundation документации.

В данной статье мы не упомянули еще многого, что следует знать высококвалифицированному разработчику на ПЛИС, а именно использование логики ускоренного переноса, использование Guide-файла и Floorplanner, правила работы с анализатором временных задержек распространения сигналов по кристаллу (Timing Analyzer пакета Foundation)

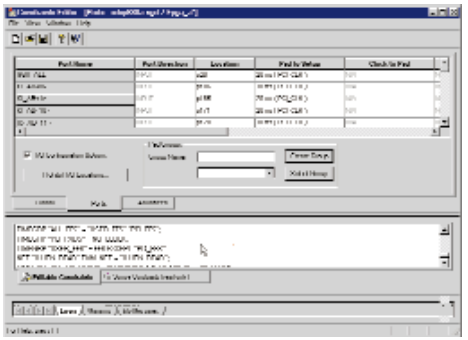

Рис. 6. Пример окна программы Constraints Editor

и т. д., однако, мы надеемся охватить данные вопросы в последующих публикациях.

В качестве заключения хочется сказать, что использование вышеприведенных методик топологической и временной оптимизации практически необходимо для построения высокоскоростных, плотно упакованных проектов на ПЛИС Xilinx и позволяет разработчику получать гарантированные временные характеристики создаваемых устройств в кратчайшие сроки.