Топологическая и временная оптимизация проектов на ПЛИС фирмы XILINX

В статье рассматриваются основные принципы топологической и временной оптимизации проектов на ПЛИС Xilinx, позволяющие существенно повысить быстродействие и компактность устройств на ПЛИС, приводятся основные конструкции временных и топологических ограничений, их базовый синтаксис.

В настоящее время основным средством проектирования приборов программируемой логики фирмы Xilinx является пакет проектирования Foundation, также производства фирмы Xilinx. Данный пакет является интегрированной САПР, включающей в себя набор программ (рис.1):

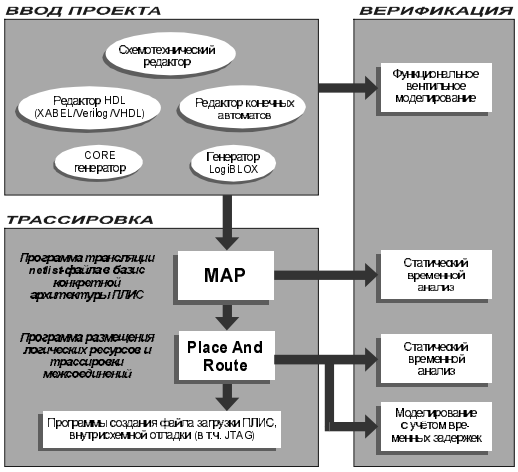


Рисунок 1. Маршрут проектирования FPGA Xilinx с использованием САПР Foundation

- ввода проекта: схемотехнический редактор, текстовый редактор HDL, редактор блочных диаграмм конечных автоматов,
- средств параметризированной генерации библиотек под различные приложения: COREGen и LogiBLOX
- моделятор функционального и временного моделирования на вентильном уровне
- встроенный синтезатор с языков Abel, VHDL и Verilog (два последних только в версиях пакета с поддержкой HDL)
- набор трансляторов и программ трассировки ПЛИС типа FPGA (MAP и Place And Route) и CPLD (Fitter)
- широкий набор очень полезных утилит оптимизации проектирования (Constraints Editor, FloorPlanner, ChipViewer, FPGA Editor и т.д.)
- утилиты загрузки ПЛИС и формирования файла программирования последовательных ПЗУ

В данной статье остановимся на более подробном рассмотрении именно программ трассировки ПЛИС и методик временной и топологической оптимизации, применимых к ним. Интегрированные средства проектирования ПЛИС фирмы Xilinx позволяют в полностью автоматическом режиме осуществить трансляцию исходного файла проекта (netlist в форматах EDIF или XNF) в файл прошивки ПЛИС, причём как для программы Map, так и для средств Place And Route существует возможность варьировать установками временной и топологической оптимизации для всего проекта в целом. На рис.2 приведён пример окна задания установок для программы Map. В то же время всем пользователям Xilinx настоятельно рекомендуется использовать методики топологической и временной оптимизации ещё на этапах создания netlist-файла в схемотехническом редакторе и программах-синтезаторах языка HDL (Hardware Description Language). Что обуславливает необходимость подобного подхода?

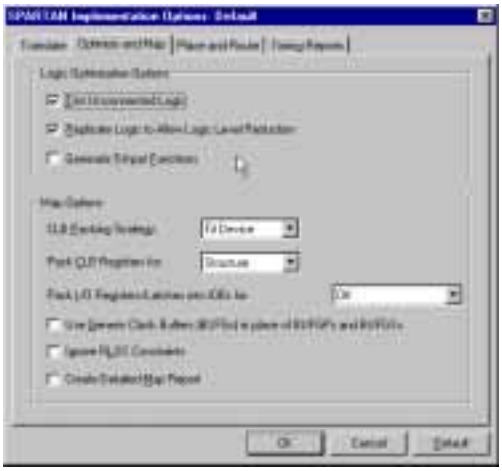


Рисунок 2. Окно задания опций трассировки программы Map

Во-первых, выборочное накладывание временных и топологических ограничений на наиболее критичную к быстродействию часть проекта пользователя позволяет дифференцировать усилие средств трассировки Xilinx по отношению ко всему проекту, и как, следствие, значительно снизить общее время трассировки.

Во-вторых, при правильном использовании методики задания временных ограничений на проект разработчик получает гарантированные значения временных задержек, что обуславливает правильность функционирования всего устройства на заданной частоте.

В-третьих, топологическая оптимизация позволяет уменьшить на 10 – 15%, а иногда и более, занимаемый проектом объем, сделать более прогнозируемыми задержки по цепям, а в ряде случаев и значительно повысить быстродействие, при этом исключается возможный разброс логики по кристаллу, т. е. логика, объединённая функциональными взаимосвязями после трассировки будет находиться рядом и на кристалле, что также благоприятно сказывается на быстродействии проекта.

В-четвёртых, занимаемый проектом объем не будет зависеть от версии трассировки, что весьма немаловажно при высоком проценте заполнения кристалла.

Таким образом, владение этими двумя методами позволяет разработчику получать компактные, быстродействующие проекты с практически неизменными временными и топологическими характеристиками.

Рассмотрим более подробно методики временной и топологической оптимизации проекта на ПЛИС Xilinx применительно к схемотехническому редактору пакета Foundation, однако это не исключает использование методик в схемотехнических редакторах третьих производителей, например, Viewlogic, OrCAD, Mentor Graphics и т.д., а также при работе на HDL, причём построение исходного кода на HDL, учитывающего топологическое размещение проекта - тема отдельной статьи.

В процедуре топологической оптимизации можно выделить три основных этапа:

- упаковка логики по функциональным генераторам;
- упаковка функциональных генераторов и триггеров в конфигурируемые логические блоки (CLB) с последующим размещением CLB между собой в пределах одного макро-символа (упаковка макросов). При этом подразумевается, что проект имеет иерархическую структуру;
- размещение упакованных макросов между собой.

Если проект сделан на верхнем уровне (не является иерархическим), то последний этап исключается.

Упаковка логики проекта по функциональным генераторам

Рассмотрим основную логическую структуру CLB ПЛИС серий XC4000 и Spartan (рис.3) без рассмотрения цепей логики ускоренного переноса /1, 2/.

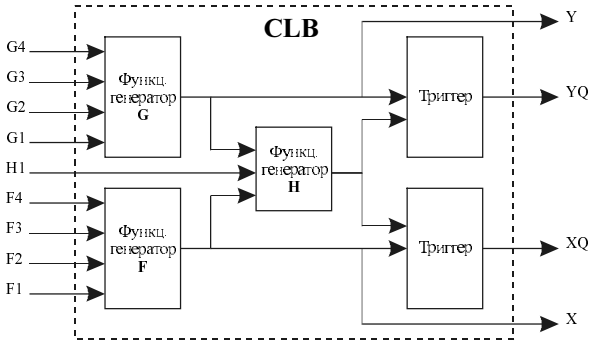


Рисунок 3. Структура CLB ПЛИС серий XC4000 и Spartan

Каждый CLB включает в себя три функциональных генератора – G, F, H, которые позволяют реализовывать любые логические функции четырех переменных (трёх для H-генератора) и два триггера. На первом этапе всю логику проекта необходимо упаковать в функциональные генераторы. Для этого в схемотехническом редакторе используются элементы FMAP и HMAP, причем FMAP ассоциируется с генераторами G и F, а HMAP – с генератором H. Вариант такой упаковки показан на рис. 4.

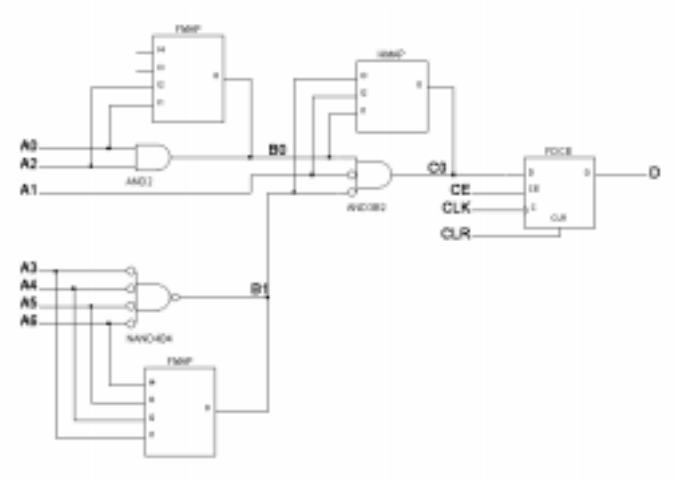


Рисунок 4. Первый этап топологической оптимизации

Упаковка функциональных генераторов и триггеров в конфигурируемые логические блоки (CLB) с последующим размещением CLB между собой в пределах одного макро-символа (упаковка макросов)

После того, как вся логика упакована по функциональным генераторам, необходимо произвести размещение функциональных генераторов и триггеров внутри CLB путем назначения атрибута RLOC (Relative Location Constraints) на элементы схемы.

Атрибут RLOC имеет следующий формат /3/:

RLOC=RxCy.[extension],

где **x, y** определяют положение упаковываемого CLB относительно других;

extension – расширение, которое может принимать значения: .G, .F для функциональных генераторов, RAM, ROM и др., .FFY, .FFX для регистров ПЛИС серий XC4000 и Spartan; .S0, .S1 для ПЛИС серии Virtex; .LC0, .LC1, .LC2, .LC3 для ПЛИС серии XC5200; 1, 2 для буферов с третьим состоянием.

Соответствие между расширением и физическим ресурсом CLB для ПЛИС серий XC4000 и Spartan приведено на рис. 5.

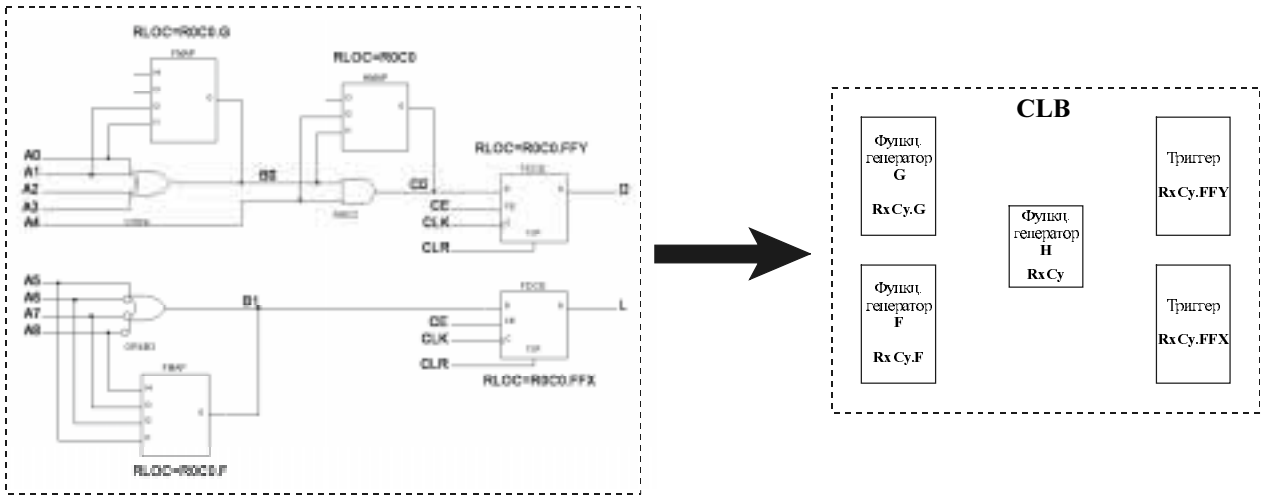


Рисунок 5. Соответствие между расширением и физическим ресурсом CLB

В общем случае атрибут RLOC может быть назначен на следующие библиотечные элементы:

- Регистры. Это элементы FDCE, FDPE, LDCE_1 и др.
- FMAP;
- HMAP;
- F5MAP;
- CY4;
- CY_MUX;
- ROM;

- RAM;
- RAMS, RAMD;
- BUFT;
- LUTs, F5MUX, F6MUX, MUXCY, XORCY, MULT_AND, SRL16, SRL16E;
- Макросы разработчика, внутри которых есть элементы, закреплённые с использованием RLOC.

Параллельно с упаковкой логики по CLB необходимо производить размещение CLB между собой. Это делается путем назначения переменным **x** и **y** конкретных числовых значений как показано на рис. 6 (для ПЛИС серий XC4000 и Spartan). Для ПЛИС других серий эта операция выполняется аналогично. Отличие состоит только в типе расширения, которое должно принимать значения .S0, .S1 для ПЛИС серии Virtex, .LC0, .LC1, .LC2, .LC3 для ПЛИС серии XC5200 /2/.

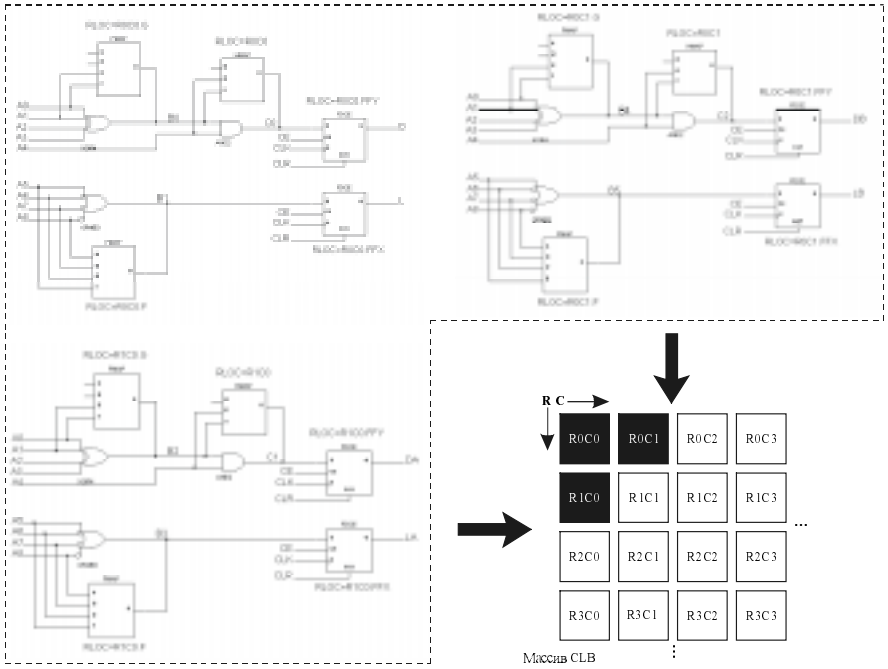


Рисунок 6. Второй этап топологической оптимизации

Для назначения атрибута RLOC на элемент, необходимо:

a) двойным щелчком мыши по выбранному элементу открыть окно Symbol Properties;
b) в разделе Parameters в поле Name ввести имя атрибута – RLOC;
c) в разделе Parameters в поле Description ввести значение атрибута, например R0C0.G;
d) справа от раздела Parameters нажать на кнопку ADD. В перечне назначенных атрибут должна появиться строка: RLOC=R0C0.G (рис. 7) /4/.

Рисунок 7. Пример окна задания атрибутов символа

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 макро-символы, а размещение выполняется без абсолютной привязки макросов к физическим ресурсам ПЛИС (рис. 8) – имеет место относительная ориентация макросов между собой. Необходимо обратить внимание, что при необходимости указания абсолютного местоположения CLB или макроса следует использовать атрибут RLOC_ORIGIN.

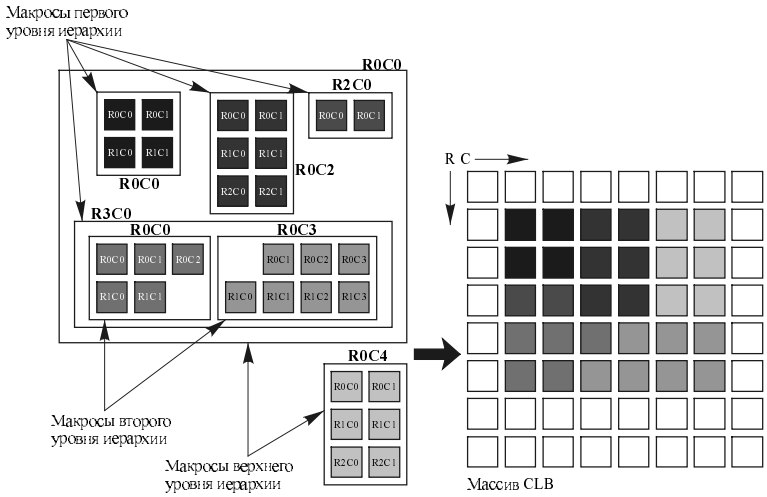


Рисунок 8. Третий этап топологической оптимизации проекта

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 приведён на рис. 9. 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 /5/.

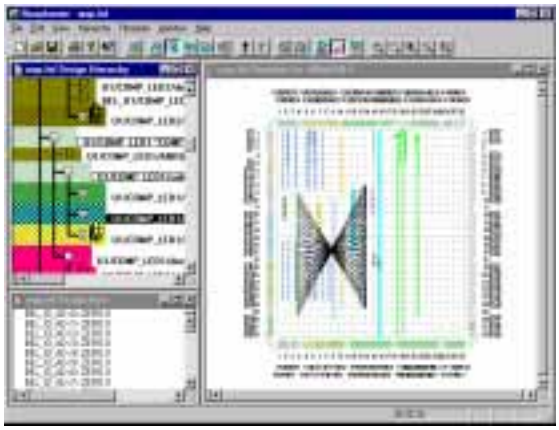


Рисунок 9. Пример окна программы Floorplanner

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 остаётся 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 объёма 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 определённого типа всего проекта, например, триггеры, ОЗУ и т.д.

Временные группы могут быть объявлены двумя способами /3/:

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.

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 (см. рис. 10).

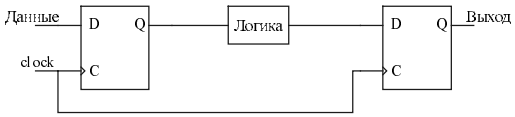


Рисунок 10. Ограничение задержек «от триггера до триггера»

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 /1/. 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-ов (см. рис. 11).

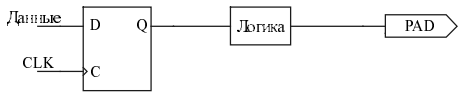


Рисунок 11. Оптимизация задержек от триггеров до PAD-ов

Конструкция “FROM:TO”. Данная конструкция анализирует задержки от выхода регистра до PAD-а, 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-ов до триггеров (см. рис. 12).

Рисунок 12. Оптимизация задержек от PAD-ов до триггеров

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.

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

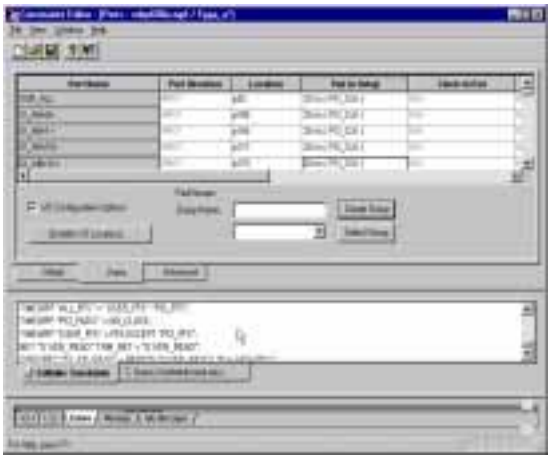
Иногда требуется, чтобы какой-либо путь был исключен из временной спецификации. Для этого необходимо воспользоваться ключевым словом “TIG” /3/. Например:

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 – иск



(Timing Analyzer пакета Foundation) и т.д., однако, мы надеемся охватить данные вопросы в последующих публикациях.

В качестве заключения хочется сказать, что использование вышеприведённых 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.

1.Программируемые логические ИМС на КМОП-структурах и их применение / П.П. Мальцев, Н.И. Гарбузов, А.П. Шарапов, Д.А. Кнышев – М.Ж Энергоатомиздат, 1998. – 160 с.
2. The Programmable Logic Data Book, Xilinx Inc., 1999.
3. Libraries Guide, Sector Attributes, Constraints and Carry Logic, CD Documentation Foundation Series 2.1i, Xilinx Inc., 1999.
4. Xilinx Foundation Help, Sector Schematic Editor, Xilinx Inc., 1999.
5. Xilinx Floorplanner Guide, Xilinx Inc., 1999.
6. Xilinx Constraints Editor Guide, Xilinx Inc., 1999.

Володин П.В.
Капитанов В.Д.
Scan Engineering Telecom, Воронеж
Тел.: (0732) 51-21-99, 72-71-01
E-mail: capt@scan.voronezh.su
http://www.xilinx.ru

SCAN, Москва