Из опыта работы с САПР Quartus фирмы Altera

Дмитрий ИОФФЕ
dsioffe@yandex.ru

В предлагаемой вниманию читателей статье рассматриваются два частных вопроса работы с САПР Quartus. Фирма Altera наделила свой пакет огромным количеством разнообразных возможностей. Оборотной стороной этого является некоторая трудоемкость освоения на начальном этапе. Статья предназначена для инженеров, недавно начавших работать с пакетом Quartus. Автор будет рад, если приведенные здесь материалы помогут им сэкономить немного времени.

Работа с сообщениями временного анализатора

Временной анализатор (Timing Analyzer) — очень полезный инструмент. Он позволяет находить в синтаксически правильном проекте места, которые будут работать со сбоями или вообще не будут работать. Но для работы с временным анализатором проект должен быть специальным образом подготовлен. В противном случае Timing Analyzer будет генерировать множество сообщений, в которых трудно найти нужную информацию. Например, в большом количестве будут появляться сообщения вида:

```
Warning: Can't achieve timing requirement Clock Setup: 'InputMainClock' along 5420 path(s). See Report window for details.
```

Это означает, что, например, в 5420 случаях тактовая частота на входе InputMainClock слишком велика, и соответствующие сигналы не успеют установиться за один ее период. На самом деле Timing Analyzer просто не смог правильно оценить проект из-за недостатка информации.

Рассмотрим подготовку проекта на несложном примере.

Создадим новый проект. Напишем на языке AHDL такой текст:

```
TITLE « Работа с временны́м анализатором»;
INCLUDE «lpm_counter.inc»;
SUBDESIGN TestTAN
(
   InClock: input;
   Signal: output;
)
VARIABLE
   Counter1: lpm_counter with (lpm_width = 8);
   Counter2: lpm_counter with (lpm_width = 8);
BEGIN
   Counter1.clock = InClock;
   Counter2.clock = Counter1.q[7];
   Signal = Counter2.q[7];
END;
```

В файле описаны два счетчика, Counter1 и Counter2, созданные на основе стандартной мегафункции счетчика LPM_COUNTER. Первый из них тактируется входным сигналом InClock, а второй — выходным сигналом старшего разряда первого счетчика. На выход схемы подается сигнал старшего разряда второго счетчика.

Сохраним этот файл под именем TestTAN.tdf в каталоге нашего проекта и сделаем его файлом верхнего уровня проекта (меню Project/Set as Top-Level Entity). Далее по тексту будет подразумеваться работа с микросхемой семейства Cyclone в Quartus II v.5.0. Для других семейств и версий САПР тексты сообщений могут несколько отличаться.

Запустим компилятор (нажмем комбинацию клавиш Ctrl+L). Убедимся, что проект откомпилировался без ошибок. Теперь можно начинать подготовку проекта для работы с временным анализатором.

Этап первый

После окончания компиляции Quartus покажет нам окно отчета о компиляции (Compilation Report), в котором смотрим секцию временного анализатора (Timing Analyzer), раздел сообщений (Messages). Вид окна отчета показан на рис. 1. В левой части окна находится панель навигации, а в правой — список сообщений, которые Quartus генерировал в ходе компиляции.

Синим цветом отмечены предупреждения (Warning). Находим сообщение вида:

```
Warning: Found pins functioning as undefined clocks and/or memory enables
```

Слева от сообщения есть небольшой знак +, обведенный квадратом. Если щелкнуть по нему мышью, то под сообщением развернется список не определенных (пользователем) тактовых сигналов (undefined clocks). Появление этого сообщения недопустимо: оно говорит о том, что временной анализатор не знает, как ему контролировать распространение таких сигналов.

Для правильной работы временного анализатора надо описать все тактовые частоты из этого списка. В нашем списке находится один пункт:

```
Info: Assuming node «InClock» is an undefined clock
```

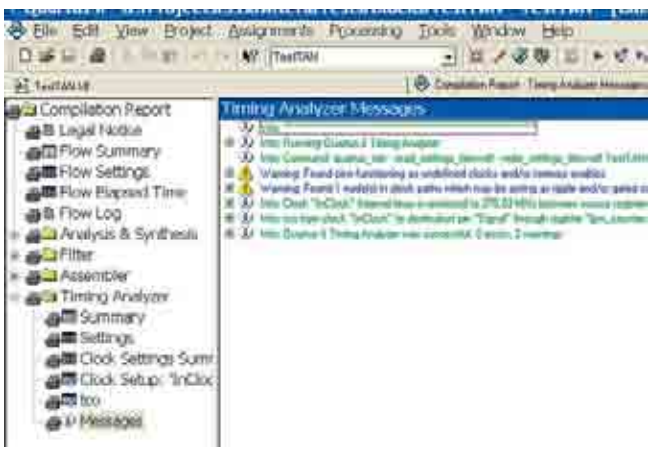

Рис. 1. Фрагмент окна Compilation Report с развернутой секцией Timing Analyzer — Messages

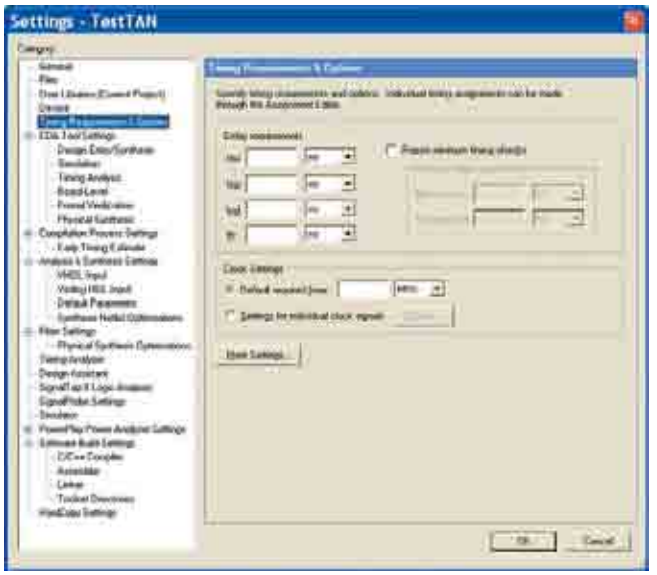

Рис. 2. Окно Settings с выбранной секцией Timing Requirements & Options

Это означает, что мы не определили тактовую частоту, поступающую на вход InClock. Определим ее.

Порядок действий:

1. Вызываем пункт меню Assignments/Settings..., или нажимаем Ctrl+Shift+E либо выбираем на панели инструментов кнопку с изображением карандаша.
2. В появившемся окне в поле Category (оно слева, белое) щелкаем мышью по Timing Requirements & Options (рис. 2).
3. В правой части окна в группе Clock Settings выбираем Settings for individual clock signals.
4. Нажимаем на кнопку Clocks... Появляется окно, в котором будут собраны сведения обо всех назначенных тактовых частотах. Пока оно пустое.
5. Жмем на кнопку New... Появляется окно New Clock Settings.
6. В верхнем поле этого окна вводим любое имя. Мне нравится, когда оно понятное, поэтому я наберу «150MHz».
7. В группе Relationship to other clock settings выбираем пункт Independent to other clock settings.
8. Выбираем сигнал, для которого будет действовать тактовая частота, описанная под именем 150MHz. Для этого поле Applies to node заполняем через Node Finder (рис. 3), вызвав его кнопкой с многоточием. В окне Node Finder из выпадающего списка Filter выбираем Pins: Input, нажимаем на кнопку List и в поле Nodes Found делаем двойной щелчок мышью по единственному в нашем проекте входу InClock. Закрываем Node Finder.
9. Описываем тактовую частоту, для чего вводим нужное число в группе Relationship to other clock settings. Там все понятно. Результат показан на рис. 4.

Рис. 3. Окно Node Finder

Рис. 4. Определение входного тактового сигнала

10. Последовательно нажимаем на кнопки OK до закрытия окна Settings.

Компилируем проект (Ctrl+L). Снова смотрим на сообщения временного анализатора и убеждаемся, что первое предупреждение отсутствует. Теперь Quartus «знает», какой частотой тактируется наш проект.

Этап второй

Определили все тактовые частоты, откомпилировали проект. Теперь надо устранить сообщения вида:

```
Warning: Found 1 node(s) in clock paths which may be acting as ripple and/or gated clocks -- node(s) analyzed as buffer(s) resulting in clock skew
```

Оно означает, что в качестве тактового используется сигнал, который может работать как не определенный вторичный, то есть сформированный из других сигналов (возможно, пульсирующий — ripple или стробируемый — gated). Система Quartus требует корректного описания этого вторичного сигнала (или сигналов, если их несколько).

В нашем проекте Quartus нашел один такой сигнал — старший разряд первого счетчика, выход которого тактирует второй счетчик:

```
Info: Detected ripple clock «lpm_counter:Counter1|cntr_155:auto_generated|safe_q[7]» as buffer
```

Определим его, выполнив пункты 1–5 из списка действий на первом этапе. Далее.

1. Присваиваем имя новому определению. Пусть оно называется, например, Clk2.
2. Указываем сигнал, для которого будет действовать новое определение. Вызовем Node Finder, нажав на кнопку с многоточием. Из списка Filter выберем Registers: post-fitting. Так как наш проект совсем небольшой, усложнять фильтр через поле Named не будем. Нажмем кнопку List и выберем из появившегося списка lpm_counter: Counter1|cntr_155:auto_generated|safe_q[7], о котором говорилось в предупреждении.
3. В группе Relationship to other clock settings выбираем вариант Based on. Выбираем из выпадающего списка нужную частоту, ту, из которой формируется наш вторичный сигнал. (В нашем примере будет предложен для выбора только один вариант — 150MHz.)
4. Нажимаем на кнопку Derived Clock Requirements. В появившемся одноименном окне предлагается большое разнообразие средств для описания вторичной тактовой частоты. Введем в поле Divide base absolute clock число 256, так как мы знаем, что описываем частоту последнего триггера 8-разрядного счетчика. Обратим внимание: Quartus тут же рассчитал получившиеся частоту и период (рис. 5).

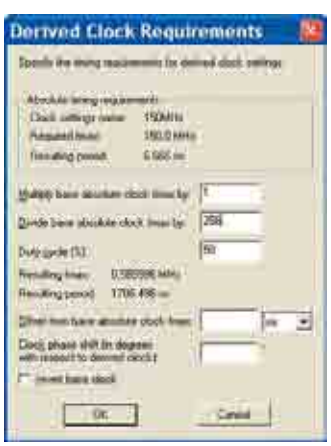

Рис. 5. Описание вторичного тактового сигнала

5. Последовательно нажимаем кнопки OK до закрытия окна Settings.

Если теперь откомпилировать проект, то получим только информационные сообщения с заголовками Info, окрашенные в зеленый цвет. Синих строк Warning больше не будет.

Может сложиться ситуация, когда с помощью окна Derived Clock Requirements не удастся описать сигнал, который формируется по сложному алгоритму. Тогда можно описать его как независимый (Independent to other clock settings) таким образом, каким это было сделано на первом этапе, зная максимальную частоту, с которой он может формироваться.

Файлы текстов на языке AHDL *.TDF и функция «Восстановление системы» Windows XP

Описание проблемы

Я прихожу утром на работу, включаю компьютер и вижу сообщение о том, что система восстановлена после серьезного сбоя. Накануне вечером все было хорошо. Вчера я весь день работал с Quartus, запускаю его и сегодня и получаю что-то непонятное. После получасового расследования обнаруживаю, что в файлах ***.tdf** пропали последние сделанные мною изменения.

Происхождение проблемы

Известно, что в операционной системе Windows XP есть замечательная функция восстановления системы (System Restore). Она по просьбе пользователя, а также в соответствии со своими замыслами, создает так называемые контрольные точки, то есть в какой-то момент резервирует все файлы, от которых зависит работоспособность системы. Если, например, после установки новых драйверов или после серьезного сбоя систему не удается нормально запустить, то можно запустить утилиту «Восстановление системы», и она восстановит сохраненные ею старые версии всех жизненно важных файлов для той контрольной точки, которую вы ей укажете. Кроме того, эта утилита может запуститься и сама, обнаружив серьезный сбой. Тогда она сообщает, что «система восстановлена после серьезного сбоя» и еще что-то.

Принято считать, что при восстановлении никогда не затрагиваются пользовательские документы, восстанавливаются только системные файлы. К сожалению, «Восстановление системы» причисляет к лику системных файлы ***.tdf**. И стало быть, при восстановлении системы замещает их последние версии старыми. Я проверил: за пределами каталогов с рабочими файлами Quartus и MAX+Plus у меня на компьютере файлы с этим расширением встречаются только в каталоге программы Microsoft Firewall Client. После удаления этих файлов ничего плохого не произошло, впоследствии они снова появились. Поиск в Yandex привел меня к выводу, что расширение TDF используется в экзотических (для меня) программах, и было бы очень хорошо «уговорить» «Восстановление системы» не восстанавливать эти файлы.

Решение проблемы

Подсказал Tosha, модератор форума на **for um.winall .ru**, за что огромное ему спасибо.

Есть такой файл: **%systemroot%\system32\restore\filelist.xml**. (Для большинства компьютеров это **c:\windows\system32\restore\filelist.xml.**) Он скрытый и только для чтения. Снимаем атрибут «Только для чтения» и открываем файл любым текстовым редактором, который может сохранять «обычный» текст — например, «Блокнотом». Удаляем следующую строчку:

```
<REC>TDF</REC>
```

Вот и все. Снятый атрибут на всякий случай лучше восстановить.

Совет: если вы хотите открыть **filelist.xml** «Блокнотом» из «Проводника», то выбирайте из контекстного меню не «Открыть», а «Изменить». В противном случае система попытается открыть этот файл при помощи Internet Explorer, и начнутся неожиданные события. Меня спасла только кнопка Reset.

Исходный текст примера к этой статье можно взять в Интернете по адресу **ht tp:/ /ww w.dsioffe.narod.ru/myquartus/sample1.zip**.