Synopsys Design Constraint — язык задания временных ограничений на примере Altera TimeQuest. Часть 1

Денис ШЕХАЛЕВ
shdv@micran.ru

Разработка устройств на базе ПЛИС — всегда борьба двух противоположных сторон: функциональности, заложенной разработчиком, и возможностей ПЛИС по ресурсу/производительности. Мерой оценки готовности устройства является его работоспособность во всем диапазоне условий, определенных техническим заданием на разработку.

Зачем нужно задание временных ограничений?

Есть два способа, которые используют разработчики для оценки работоспособности устройства на ПЛИС: натурные испытания и статический временной анализ (Static Timing Analysis, STA).

Эти два способа отличаются кардинально. При натурных испытаниях каждое изделие должно быть проверено на работоспособность во всех наихудших случаях эксплуатации. При использовании STA можно без всяких испытаний и даже при отсутствии самого изделия проверить его работоспособность. Это возможно благодаря тому, что производители ПЛИС предоставляют библиотеки со всеми временными параметрами логических элементов в различных условиях. Поэтому оценить все временные ограничения можно аналитически, проверив все цепи в результирующем списке соединений (нетлисте) ПЛИС.

Производители ПЛИС пошли еще дальше: все чаще они заявляют о поддержке технологии Timing Driven Synthesis для тех или иных ПЛИС. В этом случае при синтезе и разводке устройства на ПЛИС накладываемые на него временные ограничения влияют на результат синтеза, но не в ущерб временным ограничениям.

Выбор способа оценки работоспособности зависит от разработчика. Кто-то не задает никаких ограничений и рассчитывает на авось и осциллограф, но автор предпочитает задать все временные ограничения и убедиться в работоспособности устройства, получив отчет об отсутствии нарушений заданных ограничений.

Задание временных ограничений в Quartus

В свое время фирма Altera, чье программное обеспечение очень удобно в работе, создала инструмент под названием Timing Analyzer. Все было просто: задаем в Setting –> Timing Analyzer Settings тактовые частоты проекта и нажимаем кнопку Run для анализа. Но проекты становились все сложнее, частоты все выше, появились высокочастотные интерфейсы, многочастотные проекты, и возможностей Timing Analyzer стало не хватать.

Нужен был более гибкий инструмент задания ограничений и их учета при синтезе, разводке и временном анализе проекта. Altera не стала изобретать велосипед и выбрала для этого Synopsys Design Constraint (SDC). Этот формат задания различных ограничений широко используется при разработке ASIC и представляет собой пакеты специальных команд на языке Tool Command Language (TCL). Этот язык, несмотря на все его недостатки, широко распространен среди производителей различных САПР, в том числе и в фирме Altera. Поэтому интеграция поддержки в Quartus формата SDC не составила труда. В результате этой интеграции и появилось программное обеспечение под названием TimeQuest.

С этого момента в программном обеспечении фирмы Altera произошло разделение Timing Analyzer на Classic Timing Analyzer и TimeQuest Timing Analyzer. На текущий момент фирма Altera не рекомендует использовать Classic Timing Analyzer, потому как качество временного анализа и синтеза с ним хуже. Более того, для новых семейств ПЛИС (Aria II, Stratix IV/V) возможности выбора Classic Timing Analyzer просто нет.

Для разработчиков, привыкших работать в Classic Timing Analyzer, переход на TimeQuest труден. Нужно понимать последовательность и необходимость задания тех или иных ограничений, помнить форматы SDC-команд, уметь пользоваться оболочкой TimeQuest для эффективной работы. Часто начинающие пользователи TimeQuest делают типовые ошибки, которые приводят их к ложной уверенности, что все временные ограничения выполняются. В результате попытка освоения TimeQuest заканчивается неудачей, и его дальнейшее использование, совершенно напрасно, откладывается на неопределенный срок.

Этот цикл статей предназначен для тех, кто только начинает работать с TimeQuest. Автор не ставит перед собой цели перевода документации TimeQuest, вместо этого он предлагает осваивать его на практических примерах. Мы рассмотрим примеры задания временных ограничений для типовых проектов с комментариями и пояснениями. Каждый пример содержит тестовый код на языке SystemVerilog (при проверке примеров не забывайте включать его поддержку) и соответствующий ему SDC-файл. Все примеры выполнены в Quartus 9.0sp2. В качестве целевой платформы использовалась ПЛИС EP3C5E144C8 семейства Cyclone III. Итак, начнем с азов.

Задание временных ограничений на примере проекта счетчика

Временные ограничения для TimeQuest пишутся в файле с расширением *.sdc. Этот файл представляет собой TCL-скрипт, в котором временные ограничения задаются с помощью специальных команд. Рассмотрим аналог проекта HelloWorld для ПЛИС:

```
module hello (input clk, output led) ;

   logic [31 : 0] cnt;

   always_ff @(posedge clk) begin
      cnt <= cnt + 1'b1;
      led <= cnt[31];
   end

endmodule
```

Как мы видим, в этом проекте всего 2 порта: порт тактовой частоты (***clk***) и порт све-

тодиода (***led***). Для того чтобы задать все временные ограничения для этого проекта, нужно задать значение тактовой частоты и описать, как анализировать вывод светодиода. Создаем файл ***hello.sdc***.

Тактовые частоты описываются с помощью команды ***create_clock***, полный формат которой можно посмотреть в документации на Quartus. Предположим, что у нас используется генератор с частотой 10 МГц. Тогда тактовая частота будет задаваться следующим образом:

```
create_clock -period 10MHz -name {clk} {get_ports {clk}}
```

В команде мы указали значение тактовой частоты, ее логическое имя и физический источник этой частоты — порт ПЛИС. Логическое имя тактовой частоты используется для задания всех временных ограничений относительно этой частоты.

Теперь давайте ответим на такой вопрос: «Так ли важно в этом примере, какова задержка вывода сигнала на светодиод?» Естественно нет, поэтому нужно описать этот путь как путь, анализировать который не нужно. Сделать это можно с помощью команды ***set_false_path***, указав в ней в качестве приемника пути порт ПЛИС ***led*** несколькими способами:

1. Указав в качестве источника регистр ***led***:

```
set_false_path -from [get_registers {led*}] -to [get_ports {led}]
```

2. Указав в качестве источника все регистры, тактируемые частотой ***clk***:

```
set_false_path -from [get_clocks {clk}] -to [get_ports {led}]
```

3. Указав в качестве источника все регистры, тактируемые любыми частотами проекта:

```
set_false_path -from [all_clocks] -to [get_ports {led}]
```

Выбор способа описания определяется предпочтениями разработчика. Автору нравится способ задания через указание тактовых частот, так как этот метод более универсальный и легко поддается методу ***Copy-Paste***. На этом создание файла временных ограничений одночастотного проекта для мигания светодиодом закончено. Как видите, никаких сложностей это не представляет, все просто и понятно.

Введение в основы временного анализа с помощью TimeQuest

TimeQuest — это программа для проверки выполнения временных ограничений, описанных в файле *.sdc. Тут следует упомянуть первое правило TimeQuest, которое нужно учитывать при работе: если TimeQuest рапортует вам об отсутствии ошибок, то не надо заранее полагать, что все хорошо. Может быть, вы просто не задали часть временных ограничений.

Таким образом, главное отличие от Classic Timing Analyzer в том, что TimeQuest проверяет только те ограничения, которые вы задали. Поэтому при разработке файла *.sdc внимательно читайте предупреждения, которые он вам выдает, и обязательно проверьте, все ли временные ограничения вы задали.

С TimeQuest можно работать в двух режимах: графическом и консольном. В консольном режиме все команды временного анализа и задания временных ограничений вводятся в консоли, а результаты могут быть просмотрены как в консоли, так и в специальных окнах. Но фирма Altera позаботилась о пользователях и снабдила TimeQuest неплохим графическим интерфейсом (GUI), с помощью которого можно, не зная полного синтаксиса sdc-команд, работать с той же эффективностью. Это относится не только к командам анализа, но и к командам задания самих временных ограничений.

Итак, собираем наш проект, состоящий из одного счетчика. Через меню ***Tools –> TimeQuest Timing Analyzer*** запускаем TimeQuest. Видим несколько окон (рис. 1). В окне ***Getting Started*** есть подсказка:

1. ***Tasks Pane*** — окно команд. В этом окне перечислены основные команды временного анализа.
2. ***Report Pane*** — окно отчетов. В этом окне отображаются закладки отчетов временного анализа.
3. ***View Pane*** — окно отображения путей. В этом окне будут перечислены пути, которые использовались при анализе, а также отчеты об этих путях.
4. Консоль.

Временной анализ возможен только по уже существующему списку соединений ПЛИС (нетлисту). Для анализа нужно загрузить нетлист и соответствующий ему sdc-файл. С помощью правой кнопки мыши загружаем их (рис. 2). Выполним анализ нашего проекта на пути, по которому не выполняются заданные временные ограничения (рис. 3).

В консоли мы видим, какие команды выполнялись, и их результат:

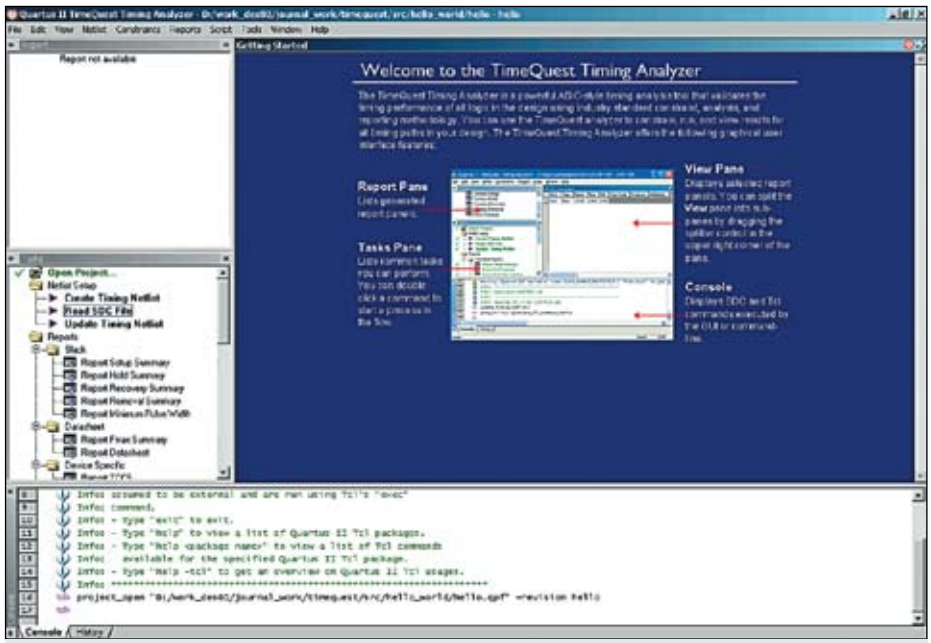

Рис. 1. Состояние TimeQuest Timing Analyzer после запуска

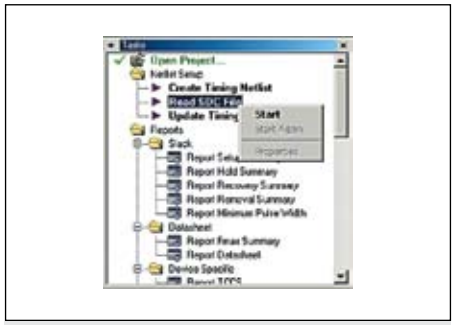

Рис. 2. Загрузка списка соединений и соответствующего ему sdc-файла

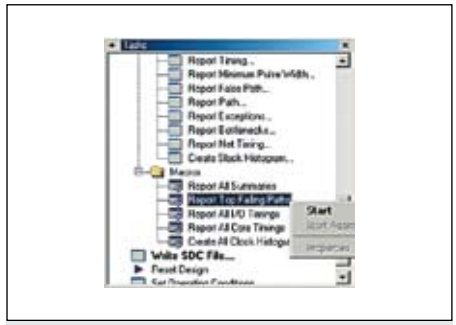

Рис. 3. Запуск анализа всех цепей на выполнение временных ограничений

```
Info: Reading SDC File: 'hello.sdc'
update_timing_netlist
Critical Warning: The following clock transfers have no clock uncertainty assignment. For more accurate results, apply clock uncertainty assignments or use the derive_clock_uncertainty command.
Critical Warning: From clk (Rise) to clk (Rise) (setup and hold)
qsta_utility::generate_top_failures_per_clock "Top Failing Paths" 200
Info: No fmax paths to report
Info: No fmax paths to report
Info: No failing paths found
```

Рассмотрим подробнее, что происходило при временном анализе. Сначала TimeQuest применил загруженные в него временные ограничения к нетлисту (***update_timing_netlist***). Потом он обнаружил, что временные ограничения, указанные в sdc-файле, не позволяют ему достоверно проанализировать нетлист, и выдал соответствующее предупреждение. Затем выполнил анализ путей на выполнение заданных временных ограничений. Как мы видим, таких путей, по которым ограничения не выполняются, нет (еще бы, в таком-то проекте на частоте 10 МГц!), и с этой точки зрения все хорошо.

Но для более корректного анализа следует устранить предупреждение. Для этого нужно добавить в sdc-файл команду, задающую нестабильность тактовых частот (***clock uncertainty***). После добавления файл ***hello.sdc*** выглядит так:

```
derive_clock_uncertainty
create_clock -period 10MHz -name {clk} [get_ports {clk}]
set_false_path -from [get_clocks {clk}] -to [get_ports {led}]
```

Теперь нужно перезапустить временной анализ. Можно заново выполнить команду ***Read SDC File***, но в этом случае поверх уже существующих временных ограничений будут загружены новые. Это может привести к их переопределению и дальнейшей путанице. Поэтому автор рекомендует перезапускать временной анализ через полный сброс результатов анализа и его повторный запуск. Делаем сброс (рис. 4). Запускаем анализ снова. В итоге нет ни предупреждения, ни ошибок:

```
Info: Reading SDC File: 'hello.sdc'
Info: Clock uncertainty calculation is delayed until the next update_timing_netlist call.
update_timing_netlist
Info: Deriving Clock Uncertainty
    Info: set_clock_uncertainty -rise_from [get_clocks {clk}] -rise_to [get_clocks {clk}] -setup 0.020
    Info: set_clock_uncertainty -rise_from [get_clocks {clk}] -fall_to [get_clocks {clk}] -setup 0.020
    Info: set_clock_uncertainty -fall_from [get_clocks {clk}] -rise_to [get_clocks {clk}] -setup 0.020
    Info: set_clock_uncertainty -fall_from [get_clocks {clk}] -fall_to [get_clocks {clk}] -setup 0.020
    Info: set_clock_uncertainty -rise_from [get_clocks {clk}] -rise_to [get_clocks {clk}] -hold 0.020
    Info: set_clock_uncertainty -rise_from [get_clocks {clk}] -fall_to [get_clocks {clk}] -hold 0.020
    Info: set_clock_uncertainty -fall_from [get_clocks {clk}] -rise_to [get_clocks {clk}] -hold 0.020
    Info: set_clock_uncertainty -fall_from [get_clocks {clk}] -fall_to [get_clocks {clk}] -hold 0.020
qsta_utility::generate_top_failures_per_clock "Top Failing Paths" 200
Info: No fmax paths to report
Info: No fmax paths to report
Info: No failing paths found
```

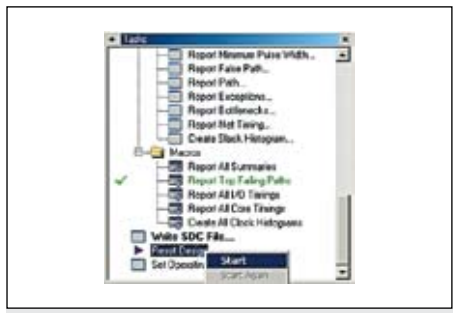

Рис. 4. Сброс результатов временного анализа

Как видите, по умолчанию TimeQuest считает нестабильность (джиттер) тактовой частоты равным 20 пс. В случае использования генератора тактового сигнала с другим значением нестабильности для корректного

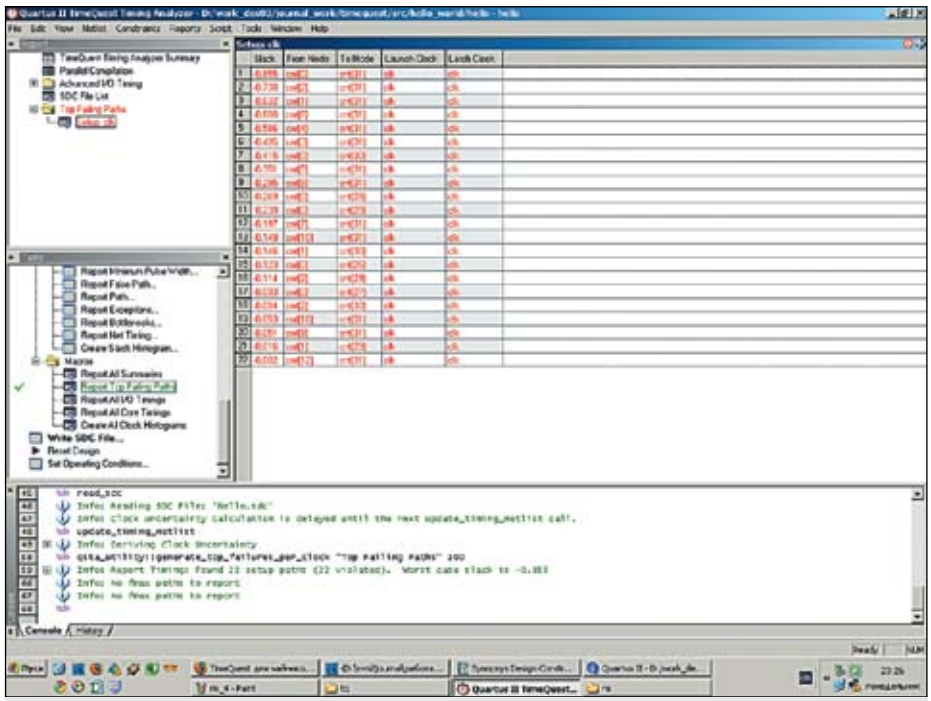

Рис. 5. Результат временного анализа проекта HelloWorld для тактовой частоты 300 МГц

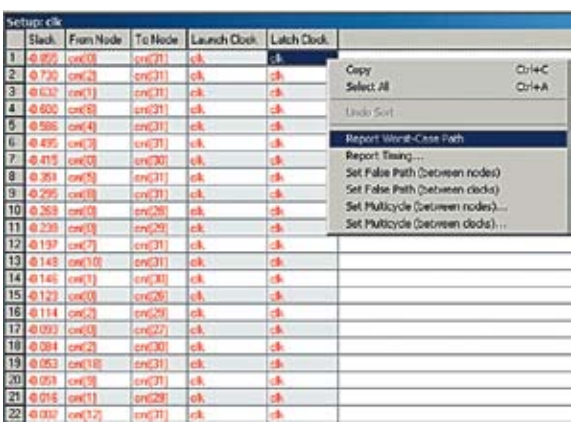

Рис. 6. Запуск генерации подробного отчета об анализе пути

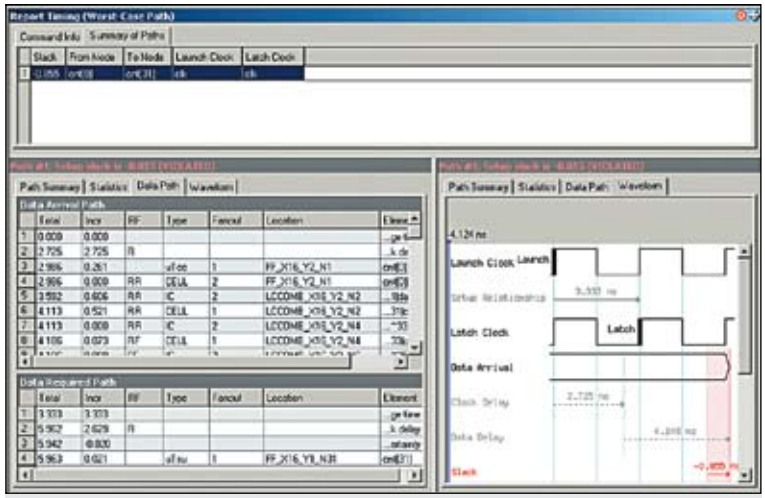

Рис. 7. Результат генерации подробного отчета об анализе пути данных между регистрами cnt[0] и cnt[31]

временного анализа его нужно обязательно указать (командой ***set_clock_uncertainty***).

Для перезапуска временного анализа с новыми ограничениями сбрасывать проект необязательно, наложить нужное временное ограничение можно, набрав его в консоли или через меню **constrains**. Но для начинающих автор рекомендует описанный выше способ. Также помните, что все временные ограничения, вводимые через меню или консоль, не сохраняются в sdc-файле до тех пор, пока вы не выполните команду ***Write SDC File***.

Теперь давайте сломаем наш проект. Укажем в sdc-файле намного более высокую тактовую частоту:

```
create_clock -period 300MHz -name {clk} [get_ports {clk}]
```

Результат анализа в этом случае неутешительный (рис. 5). Появляются многочисленные нарушения временных ограничений. На сленге это называется что-то вроде «все в слаках»/«сплошные слаки». Это пошло от слова ***slack***, обозначающего запас по тому или иному временному параметру. Положительный ***slack*** означает, что запас для работы на нужной частоте есть, отрицательный же говорит о том, что запас (в данном случае по ***tsu***) выбран и его не хватает для работы на нужной частоте. В зависимости от того, что является источником отрицательного ***slack***, его можно улучшить изменением настроек синтеза и разводки, изменением временных ограничений или изменением дизайна. Для определения «узкого места» нужно посмотреть подробный отчет об анализе выполнения условий по ***tsu*** (рис. 6). Появилось окно подробного отчета о пути ***cnt[0]/cnt[31]*** (рис. 7).

Введение в основы временного анализа синхронных схем

Для того чтобы понять, как правильно читать отчеты TimeQuest, нужно знать, как проводится временной анализ и какими временными параметрами описываются триггеры. У триггера есть три основных параметра, определяющие его временные характеристики:

1. ***Tsetup (tsu)*** — время предустановки данных (время, в течение которого данные на входе триггера должны быть неизменны до фронта тактовой частоты).
2. ***Thold (th)*** — время удержания данных (время, в течение которого данные на входе триггера должны быть неизменны после фронта тактовой частоты).
3. ***Tclock-to-out (tco)*** — время появления данных на выходе триггера после фронта тактовой частоты.

При временном анализе вычисляется путь данных между триггером-источником, работающим на тактовой частоте источника (***Source Clock***), и триггером-приемником, работающим на тактовой частоте приемника (***Destination Clock***) сигнала. Анализ проводится на выполнение условий по ***tsu/th*** триггера-приемника, и для него вводится понятие запускающего фронта (***Launch Edge***) и защелкивающего фронта (***Latch Edge***) тактового сигнала. На рис. 8 изображены все соотношения фронтов тактовых частот для временного анализа одноцикловых путей. Многоцикловые пути и их временной анализ будут рассмотрены позже.

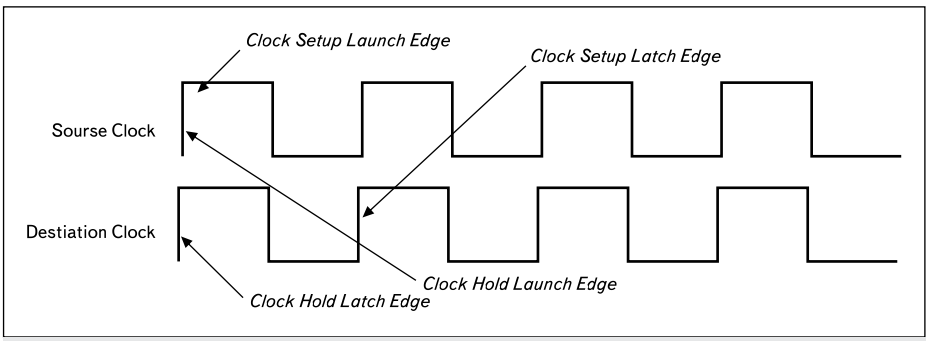

Рис. 8. Фронты тактового сигнала, используемые для временного анализа одноцикловых синхронных путей

Рассмотрим, как выполняется временной анализ по ***tsu*** (рис. 9) и по ***th*** (рис. 10). ***Launch Clock*** — это тактовая частота триггера-источника ***cnt[0]***, ***Latch Clock*** — тактовая частота триггера-приемника ***cnt[31]***. Они выровнены по переднему фронту, потому что идут из одного физического источника (порта ПЛИС). Жирными линиями выделены фронты тактового сигнала, относительно которых проводится анализ.

Для анализа по ***tsu*** TimeQuest рассчитывает два параметра:

1. ***Data Arrival*** — это реальное время прибытия данных от триггера-источника до триггера-приемника. На рис. 9 видно, что это время состоит из двух компонентов: ***Clock Delay*** = = 2,725 нс и ***Data Delay*** = 4,108 нс и рассчитывается относительно ***Setup Launch Edge***. ***Clock Delay*** — это задержка сигнала тактовой частоты от порта ПЛИС до тактового входа триггера ***cnt[0]***. ***Data Delay*** — это задержка сигнала с выхода триггера ***cnt[0]*** до входа триггера ***cnt[31]***, в эту задержку входит ***tco*** триггера-источника и задержка комбинационной и трассировочной логики.
2. ***Data Required*** — это требуемое время прибытия данных от триггера-источника до триггера-приемника. На рис. 10 видно, что это время состоит из ***Clock Delay*** = = 2,629 нс, ***Clock Uncertainty*** = 0,020 нс и ***tsu*** = = 0,021 нс триггера-приемника и считается относительно ***Setup Latch Edge*** (со сдвигом на период тактовой частоты для одноцикловых путей). При этом обратите внимание на знаки, с которыми входят те или иные значения в расчет требуемого времени.

Для анализа по ***th*** TimeQuest рассчитывает те же два параметра:

1. ***Data Arrival*** считается относительно ***Hold Launch Edge*** и состоит из ***Clock Delay*** = = 2,628 нс и ***Data Delay*** = 3,621 нс.

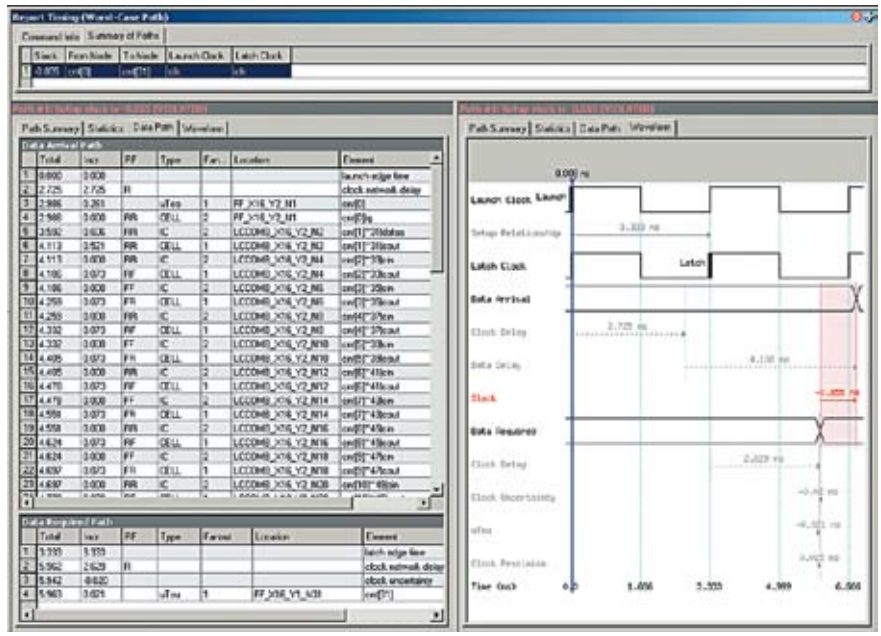

Рис. 9. Отчет об анализе пути данных между регистрами cnt[0] и cnt[31] на выполнение условий по tsu

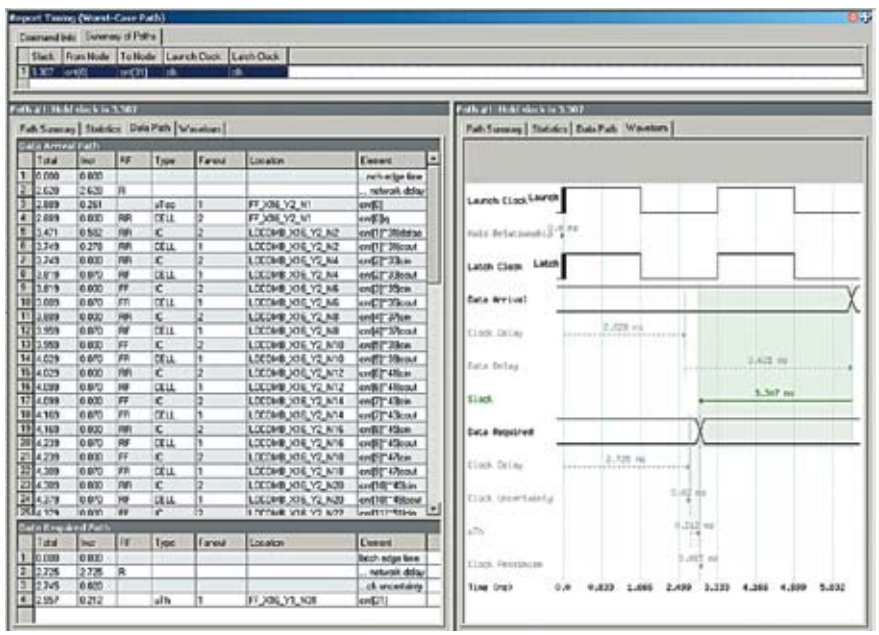

Рис. 10. Отчет об анализе пути данных между регистрами cnt [0] и cnt [31] на выполнение условий по th

2. ***Data Required*** считается относительно ***Hold Latch Edge*** и состоит из ***Clock Delay*** = = 2,725 нс, ***Clock Uncertainty*** = 0,020 нс и ***th*** = 0,212 нс триггера-приемника.

Data Slack рассчитывается как ***Data Requaired – Data Arrival***. Когда он положительный, значит, запас есть, и все хорошо, когда отрицательный, это значит, что на требуемой тактовой частоте схема работать не будет.

Внимательный читатель обязательно задаст вопрос, почему цифры задержек для одних и тех же путей разные. Ответ прост: при оценке запаса по ***tsu/th*** TimeQuest берет такие значения задержек ***Clock Delay/Data Delay***, которые соответствуют наихудшему случаю. Очевидно, что наихудшим случаем для ***tsu*** будет наибольшее ***Data Delay*** и наименьшее ***Clock Delay***, а для ***th*** наоборот. Выполнение временных ограничений в наихудшем случае гарантирует работоспособность любых микросхем выбранного типа в условиях, заданных в ***Project Settings –> Operating Settings and Conditions***.

Автор хочет отметить тот факт, что временные ограничения по ***tsu/th*** связаны между собой. На рис. 9 и 10 видно, что чем меньше задержка по данным, тем больше запас по ***tsu***, но тем меньше запас по ***th***. Поэтому при разводке проекта Quartus решает не только вопросы размещения логики, но и вопросы балансировки запаса по ***tsu/th***.

Выполнение временных ограничений проекта со счетчиком

Вернемся к нашему проекту со счетчиком. Во вкладке ***Data Path*** (рис. 9) отчета о временном анализе видно большое количество слоев логики между триггерами (если судить по вкладке ***Statistic***, это 78% всей задержки). В этом случае причина нарушения легко определяется. Это задержка цепей последовательного переноса на счетчике. Другими словами, счетчик не успевает считать.

Есть два варианта обеспечить выполнение требуемых временных ограничений:

1. Уменьшить время ***Data Arrival***. Это можно сделать, взяв более быструю ПЛИС или разбив 32-битный счетчик, например, на два 16-битных с конвейеризированным переносом между ними.
2. Увеличить время ***Data Required*** для некоторых триггеров. Это можно сделать с помощью дополнительного тактового сигнала, сдвинутого по фазе с помощью PLL, или непосредственно задержав сигнал тактовой частоты на логике. В этом случае будет больший перекос между битами счетчика, но для проекта ***hello*** это не критично.

Уменьшение времени Data Arrival

Разобьем 32-битный счетчик на два 16-битных счетчика с конвейеризированным переносом между ними:

```
module hello (input clk, output led) ;

  logic [15 : 0] cnt_low, cnt_high;
  logic          cnt_low_done;

  always_ff @(posedge clk) begin
     cnt_low      <= cnt_low + 1'b1;
     cnt_low_done <= (cnt_low == 16'hFFFE);

  if (cnt_low_done)
        cnt_high  <= cnt_high + 1'b1;
     led  <= cnt_high[15];
  end

endmodule
```

Временных ошибок нет (рис. 11). Такая техника увеличения производительности модулей называется конвейеризацией.

Увеличение времени Data Required

Разобьем триггеры 32-битного счетчика на два 16-битных регистра. Один из регистров будем тактировать от сигнала тактовой частоты, а второй — от этого же сигнала, но пропущенного через дополнительный LUT:

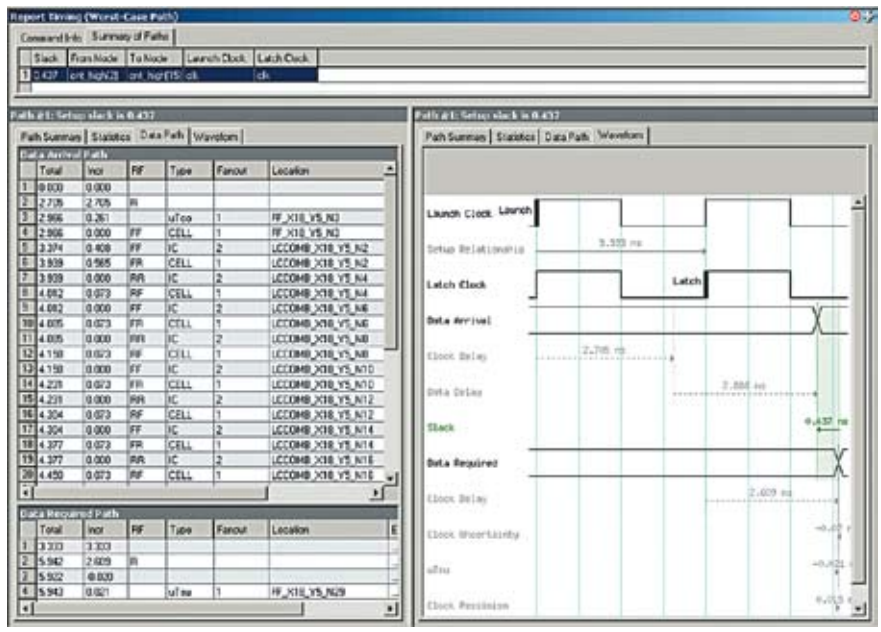

Рис. 11. Отчет об анализе пути данных между регистрами cnt[0] и cnt[31] на выполнение условий по tsu после конвейеризации модуля

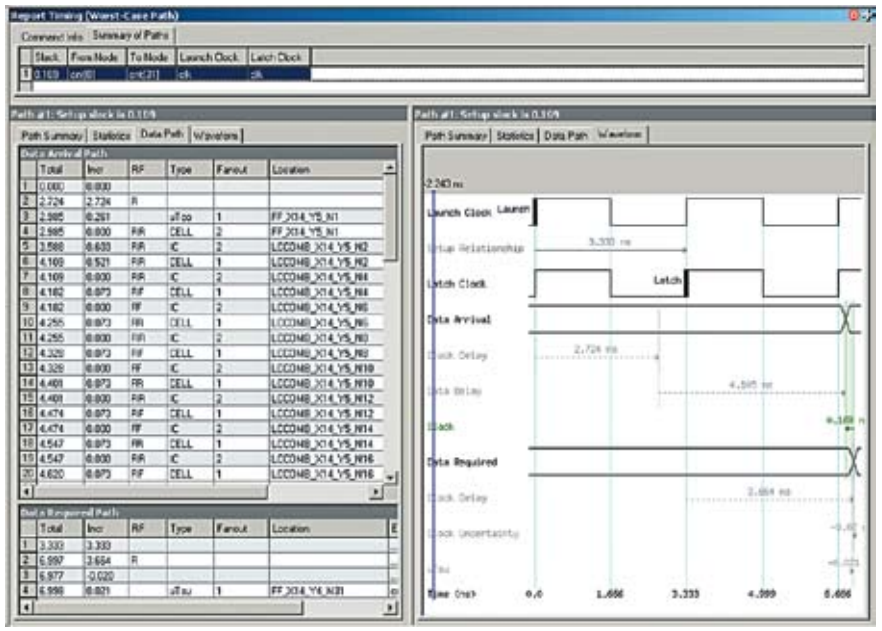

Рис. 12. Отчет об анализе пути данных между регистрами cnt[0] и cnt[31] на выполнение условий по tsu после введения дополнительной фазы тактовой частоты

```
module hello (input clk, output led);

   logic clk_delay;
   lcell lcell (clk, clk_delay);
   logic [31 : 0] cnt, cnt_next;

   assign cnt_next = cnt + 1'b1;

   always_ff @(posedge clk) begin
      cnt[15 : 0]  <= cnt_next[15 : 0];
   end

   always_ff @(posedge clk_delay) begin
      cnt[31 : 16]  <= cnt_next[31:16];
      led  <= cnt[31];
   end

endmodule
```

Ошибки также ушли (рис. 12). Такая техника увеличения производительности модулей называется многофазной синхронизацией. За исключением внешних интерфейсов для ПЛИС, эта техника не рекомендуется к использованию.

Заключение

Мы рассмотрели самый простой проект для ПЛИС и работу с временными ограничениями для этого проекта. Причина нарушения временных ограничений в этом проекте была очевидна, но в более сложных случаях искать причину возникновения «узкого места» можно с помощью контекстно-вызываемых RTL/Techology Mapper Viewer (рис. 13). Благодаря возможности контекстного вызова различных Viewer, TimeQuest является простым, но в то же время мощным и удобным инструментом для анализа проекта.

Важно то, что удовлетворение заданных временных ограничений требует точного знания источника их возникновения. Без этого невозможно эффективно, а самое главное — правильно устранить эти нарушения. Эта часть статьи представляет собой основу для дальнейшего изложения материала. В следующей части мы рассмотрим типовые многочастотные проекты и задание для них временных ограничений.

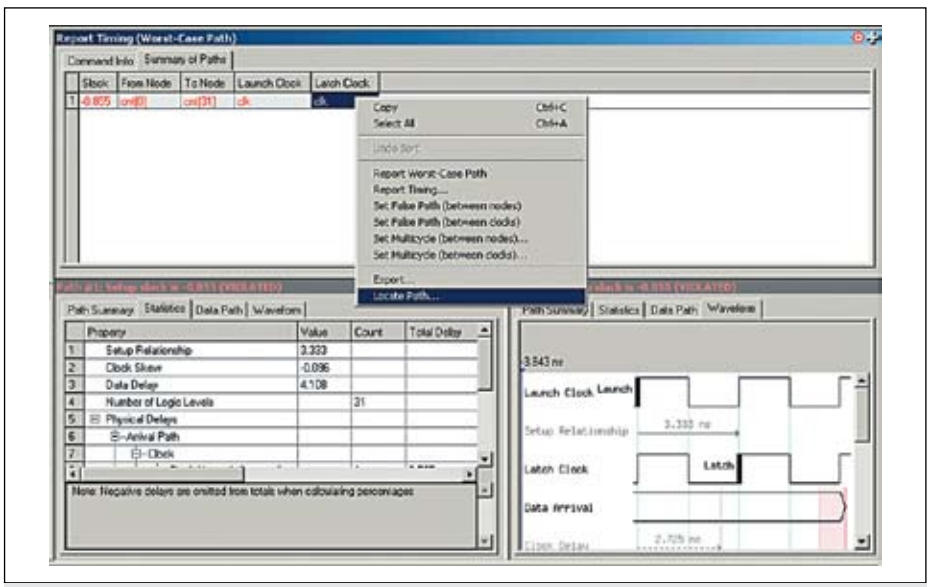

Рис. 13. Вызов различных Viewer для просмотра информации о конкретном пути

Литература

1. Quartus II Handbook Version 9.0 — http://www.altera.com/literature/lit-qts.jsp
2. SDC and TimeQuest API Reference Manual — http://www.altera.com/literature/manual/mnl_sdctmq.pdf
3. Quartus II TimeQuest Timing Analyzer Cookbook — http://www.altera.com/literature/manual/mnl_timequest_cookbook.pdf
4. http://embedders.org/blog/des00