**Основы статического временного анализа.**

**Часть 6: Multicycle Path Constraint.**

**Введение.**

Данная работа завершает серию статей по временным ограничениям в FPGA. Продемонстрирована возможность изменения положения запускающих и защёлкивающих фронтов при передачи данных между синхронными доменами. Показаны несколько практических примеров использования команды *set\_multicycle\_path*.

1. **Пересечение синхронных тактовых доменов.**

В предыдущей статье [1] был продемонстрирован временной анализ передачи данных при пересечении тактовых доменов (Clock Domain Cross). Подробно рассмотрен случай асинхронных тактовых доменов. Так как временные соотношения между фронтами асинхронных тактовых сигналов неизвестные, для пути между доменами невозможно корректно провести временной анализ. Поэтому единственный способ избежать проблем, связанных с метастабильностью – это исключение пути из временного анализа и добавление между доменами синхронизатора

В случае синхронных доменов временной анализатор может точно провести расчеты и, если все временные ограничения выполнены, то можно гарантировать корректность передачи данных. При этом бывают ситуации, когда при определенных временных соотношениях между фронтами тактовых сигналов, ограничение оказываются трудновыполнимыми, что приводит к их нарушению.

В этом случае можно, как и для асинхронных доменов, добавить синхронизатор и исключить путь из временного анализа. Однако, этот способ требует дополнительных ресурсов, которые будут потрачены на реализацию синхронизатора. Другим решением является изменение правил выполнения временного анализа. Можно модифицировать правила сопоставления запускающих и защелкивающих фронтов между собой, тем самым ослабляя временные ограничения. Для синхронных доменов это возможно, так как положения фронтом тактовых сигналов друг относительно друга точно известны. Рассмотрим, как это можно сделать.

На рисунке 1 показан путь, на который нанесены задержки для данных и тактовых сигналов. Напомним, что Vivado считает тактовые сигналы синхронными, если они формируется внутри FPGA с помощью PLL или MMCM. Передающий триггер FF1 тактируется сигналом clk\_1, а приёмный триггер FF2 – сигналом clk\_2.

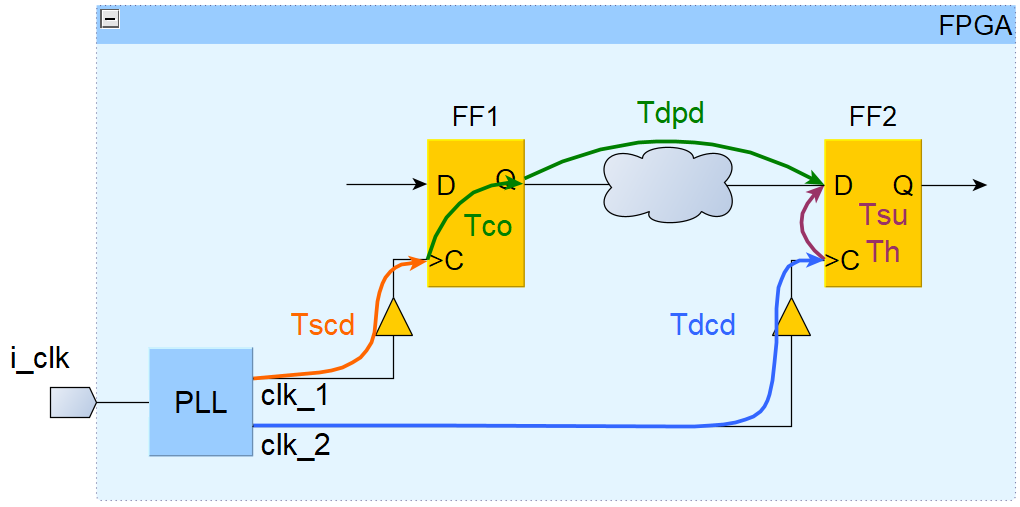


Рисунок 1. Путь с задержками для данных и тактовых сигналов.

Ниже даны определения задержек, представленных на рисунке 1.

* *Tscd* (***S****ource* ***C****lock* ***D****elay*) – задержка запускающего тактового сигнала от выхода clk\_1 PLL до тактового входа триггера FF1;
* *Tdcd* (***D****estination* ***C****lock* ***D****elay*) – задержка защелкивающего тактового сигнала от выхода clk\_2 PLL до тактового входа триггера FF2;
* *Tco* (***C****lock to* ***O****utput*) – интервал времени между приходом фронта на тактовый вход триггера и появлением данных на его выходе Q;
* *Tdpd* (***D****ata* ***P****ropagation* ***D****elay*) – задержка распространения данных по соединениям и через комбинационную логику;
* *Tsu* (***S****et****U****p time*) – время установки защелкивающего триггера;
* *Th* (***H****old time*) – время удержания защелкивающего триггера.

Временной анализ для пути на рисунке 1 проводится тем же способом, что показан в [1], поэтому сразу запишем уравнения для *Slack*:

где *ΔTsu* и *ΔTh* – интервалы времени между появлением запускающего фронта и защелкивающего фронта выходах clk\_1 и clk\_2 PLL при анализе по *Setup* и *Hold* соответственно:

В уравнениях для *Slack* величины слагаемых *ΔTsu* и *ΔTh* не известны временному анализатору и должны рассчитываться исходя из взаимного расположения фронтов тактовых сигналов clk\_1 и clk\_2. В качестве примера будем считать, что период сигнала clk\_1 равен 3 нс, а   
период clk\_2 – 4 нс. Также пусть в нулевой момент времени формируются фронты обоих тактовых сигналов (см. рисунок 2).

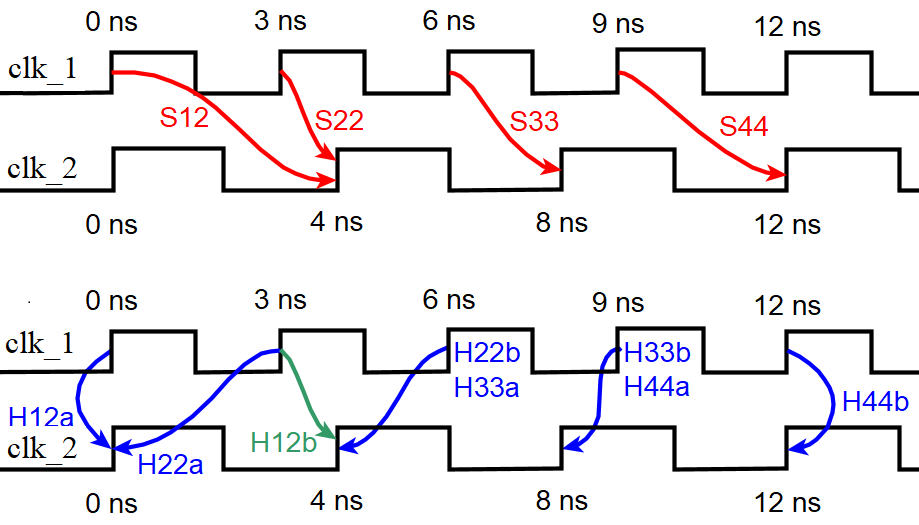


Рисунок 2. Диаграммы сигналов для временного анализа.

Напомним, каким образом анализатор получает значения *ΔTsu* и *ΔTh* [2]. Для начала каждый фронт сигнала clk\_1 рассматривается в качестве запускающего и ему сопоставляется соответствующий защелкивающий фронт сигнала clk\_2. По умолчанию выбирается ближайший фронт clk\_2, который следует после запускающего фронта clk\_1. На рисунке 2 полученные пары фронтов обозначены как S12, S22, S33 и S44. Для каждой пары по формуле (1) рассчитывается величина *ΔTsu*. В уравнение для *Slack* слагаемое *ΔTsu* входит с положительным знаком, поэтому в дальнейшем для временного анализа выбирается пара фронтов с минимальным значение *ΔTsu*, что соответствует самому пессимистичному случаю. В нашем примере это пара S22, для которой *ΔTsu* = 1 нс.

Далее временной анализатор вычисляет значение *ΔTh*. Для этого для каждой пары фронтов, полученной при расчете *ΔTsu*, проверяются два   
условия [2]:

1. данные, которые передаются по запускающему фронту clk\_1 не должны быть приняты фронтом clk\_2, который является предыдущим по отношению к текущим защелкивающему фронту clk\_2;
2. данные, которые передаются фронтом clk\_1, следующим после запускающего, не должны быть приняты текущим защелкивающим фронтом clk\_2.

Для нашего примера эти условия изображены на рисунке 2 в виде синих стрелок. Например, для пары S22 первому условию соответствует   
стрелка H22a. Запускающий фронт clk\_1 начинает передачу данных в момент времени 3 нс. Эти данные не должны быть защелкнуты предыдущим фронтом clk\_2, который появляется в нулевой момент времени. Второму условию для пары S22 соответствует стрелка H22b. Следующий фронт clk\_1 после запускающего появляется в момент времени 6 нс и запускает новую передачу данных. Эти данные не должны быть приняты текущим защелкивающим фронтом из пары S22.

Для каждого условия по формуле (1) вычисляется значение *ΔTh*. Например, рассматривая пару фронтов S22, для условия H22а получаем, что *ΔTh =* –3 нс*,* а для условия H22b – *ΔTh =* –2 нс. Временной анализатор рассчитывает величины *ΔTh* для всех условий, показанных синими стрелками на рисунке 2. Самому пессимистичному случаю при анализе по *Hold* соответствует максимальное значение *ΔTh*, так как в уравнение для *Slack* это слагаемое входит с отрицательным знаком. Для нашего примера максимальное значение *ΔTh* равно нулю, что соответствует условию H12a.

Обратите внимание, что на рисунке 2 условие H12b обозначено зеленым цветом. Данные запущенные фронтом clk\_1 в момент времени 3 нс не должны быть приняты фронтом clk\_2, который появляется в момент времени 4 нс. Для этого условия получаем, что *ΔTh* = 4 – 3 = 1 нс, что является максимальными значение и соответствует самому пессимистичному случаю. Однако, условие H12b противоречит условию S22, которое говорит, что в момент времени 4 нс данные как раз должны быть приняты фронтом clk\_2. Из-за этого противоречия временной анализатор в дальнейшем не рассматривает условие H12b и использует значение *ΔTh*, равное нулю.

Проверим проведенные выше рассуждения на практике и рассмотрим простой проект, в котором присутствует два синхронных тактовых домена. Для простоты пусть каждый домен состоит всего из одного триггера. На рисунке 3 показаны настройки PLL, а на рисунке 4 – схема проекта.

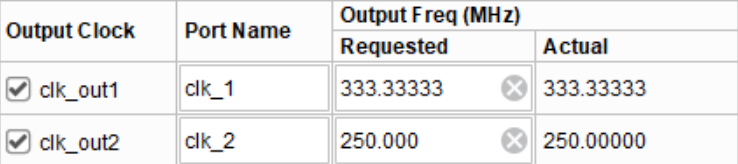


Рисунок 3. Настройки PLL.

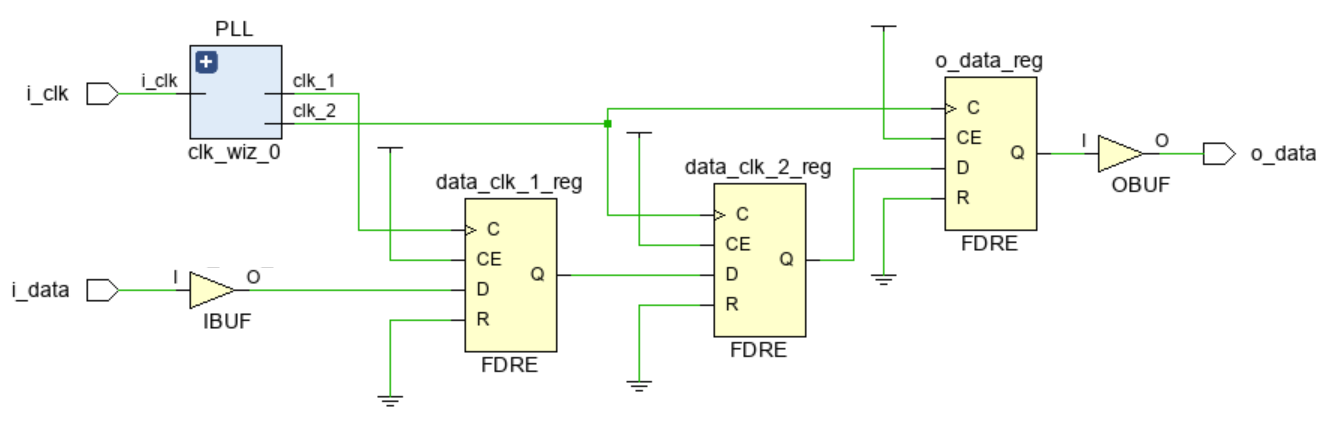


Рисунок 4. Схема проекта.

Описание на SystemVerilog представлено ниже:

module top (

    input  logic i\_clk,

    input  logic i\_data,

    output logic o\_data

);

    logic clk\_1, clk\_2;

    logic data\_clk\_1, data\_clk\_2;

    // PLL для формирования синхронных сигналов

    clk\_wiz\_0 PLL (

        .i\_clk(i\_clk),

        .clk\_1(clk\_1),

        .clk\_2(clk\_2)

    );

    // передающий домен

    always\_ff @(posedge clk\_1) begin

        data\_clk\_1 <= i\_data;

    end

    // приемный домен

    always\_ff @(posedge clk\_2) begin

        data\_clk\_2 <= data\_clk\_1;

        o\_data <= data\_clk\_2;

    end

endmodule

На ножку FPGA приходит тактовый сигнал i\_clk с периодом 10 нс, из которого с помощью PLL формируются два сигнала clk\_1 и clk\_2. Добавим в xdc-файл следующие команды:

# период входного тактового сигнала

create\_clock -period 10.0 -name i\_clk [get\_ports i\_clk]

# объявление переменных, которые указывают места формирования тактовых сигналов

set pll\_input    [get\_pins PLL/inst/mmcm\_adv\_inst/CLKIN1]

set pll\_output\_1 [get\_pins PLL/inst/mmcm\_adv\_inst/CLKOUT0]

set pll\_output\_2 [get\_pins PLL/inst/mmcm\_adv\_inst/CLKOUT1]

# обновление имен сгенерированных с помощью PLL тактовых сигналов

create\_generated\_clock -name clk\_1 -source $pll\_input

-master\_clock [get\_clocks i\_clk] $pll\_output\_1

create\_generated\_clock -name clk\_2 -source $pll\_input

-master\_clock [get\_clocks i\_clk] $pll\_output\_2

Временному анализатору с помощью команды *create\_clock* необходимо указать период сигнала i\_clk. Зная его, а также учитывая настройки PLL, анализатор самостоятельно определит периоды сигналов clk\_1 и clk\_2. Как и в [1], используя команду *create\_generated\_clock* с ключом *-name*, заменим автоматически сгенерированные имена сигналов на выходе PLL на более простые.

На рисунке 5 показан раздел *Summary* временного отчета для пути между тактовыми доменами при анализе по *Setup*. Из строки *Requirement* можно увидеть, что запускающий фронт появляется в момент времени 3 нс, а защелкивающий – в 4 нс. Значение *ΔTsu* равно 1 нс, что соответствует паре фронтов S22 на рисунке 2.

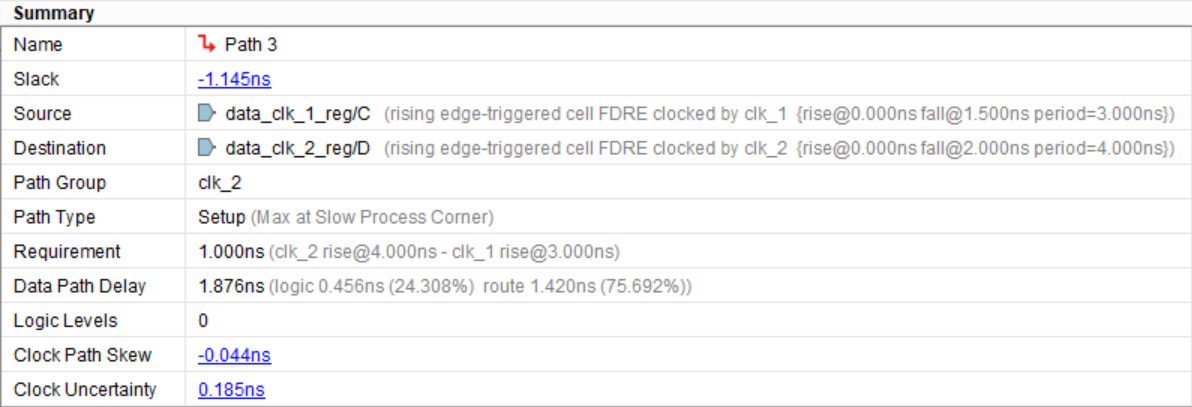


Рисунок 5. Раздел *Summary* временного отчета для анализа по *Setup*.

На рисунке 6 представлен раздел *Summary* временного отчета для анализа по *Hold*. И запускающий, и защелкивающий фронты формируются в нулевой момент времени, что на рисунке 2 соответствует условию H12a.

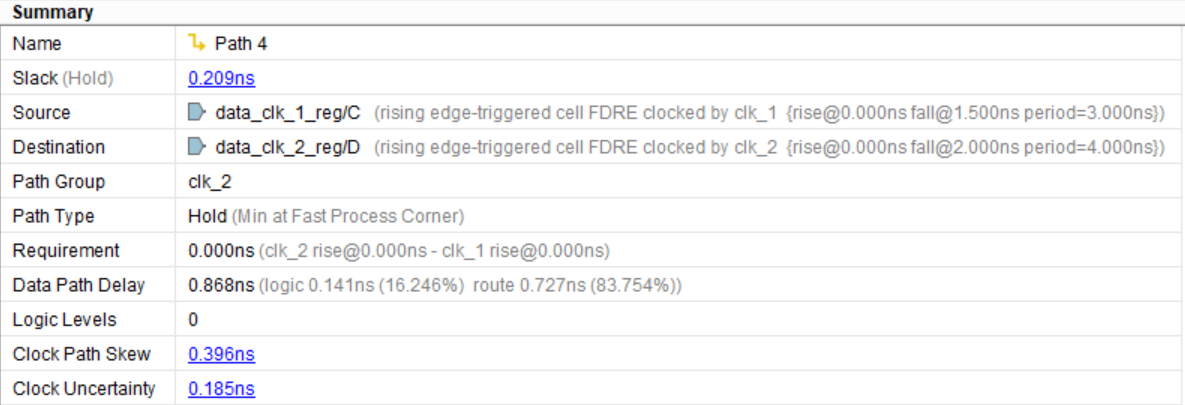


Рисунок 6. Раздел *Summary* временного отчета для анализа по *Hold*.

Из-за того, что интервал времени между появлением фронтов из пары S22 равен всего 1 нс, мы получили трудновыполнимое ограничение при анализе по *Setup*. Как можно увидеть из рисунка 5 это ограничение нарушается, и данные опаздывают на 1.145 нс.

1. **Изменение положения защелкивающего фронта.**

Как отмечалось ранее, при формировании пар фронтов тактовых сигналов для анализа по *Setup* каждому запускающему фронту сигнала clk\_1 сопоставляется ближайший по времени защелкивающий фронт сигнала clk\_2. Для временного анализатора это правило обозначается, как *Setup* = 1. Его можно модифицировать для конкретного пути с помощью команды *set\_multicycle\_path* и, например, сдвинуть положение защелкивающего фронта на один фронт сигнала clk\_2.

Добавим в xdc-файл следующую команду:

# сдвиг защелкивающего фронта вправо для анализа по Setup

set\_multicycle\_path 2 -setup -end -from [get\_pins data\_clk\_1\_reg/C]

-to [get\_pins data\_clk\_2\_reg/D]

С помощью ключей *-from* и *-to* задается путь, для которого изменяются правила временного анализа. Ключ -*setup* говорит, что модификации вносятся в анализ по *Setup*, а ключ *-end* – что изменяется положение защелкивающего фронта. Число 2 обозначает, что новое правило для анализа выглядит, как *Setup* = 2.

Начало и конец пути можно также задавать через триггеры с помощью команды *get\_cells* или через тактовые сигналы, используя команду *get\_clocks*:

# другие способы задания начала и конца пути

set\_multicycle\_path 2 -setup -end -from [get\_cells data\_clk\_1\_reg]

-to [get\_cells data\_clk\_2\_reg]

set\_multicycle\_path 2 -setup -end -from [get\_clocks clk\_1]

-to [get\_clocks clk\_2]

Временная диаграмма с обновленным положением фронтов представлена на рисунке 7. Положение всех защелкивающих фронтов сдвинулись вправо на один фронт по сравнению со случаем без команды *set\_multicycle\_path*   
(см. рисунок 2). Самым пессимистичная ситуация по-прежнему соответствует паре фронтов S22, но теперь *ΔTsu* равно 8 – 3 = 5 нс.

Так правила для анализа по *Hold* привязаны к парам фронтов, полученных на этапе вычисления *ΔTsu*, их защелкивающие фронты также смещаются. Например, правило H12a обозначает, что данные запущенные в нулевой момент времени не должны быть приняты фронтом clk\_2, который предшествует защелкивающему. Это соответствует моменту времени 4 нс, так как защелкивающий фронт для пары S12 теперь появляется в 8 нс.

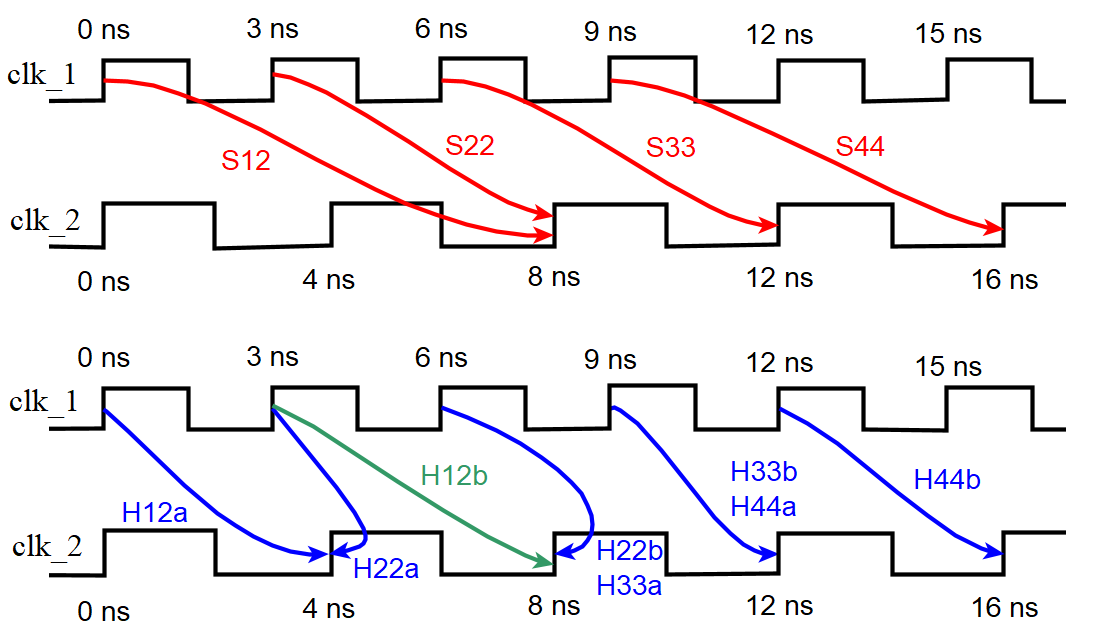


Рисунок 7. Диаграммы сигналов для временного анализа.

Для анализа по *Hold* будет использоваться правило H12a, для которого *ΔTh* = 4 – 0 = 4 нс. Правило H12b, обозначенное зеленым цветом, игнорируется из-за конфликта с требованиями для пары S22 при анализе по *Setup*.

Ниже показаны обновленные разделы *Summary* временного отчета. На рисунке 8 можно увидеть, что при анализе по *Setup* (*Requirement*) защелкивающий фронт сигнала clk\_2 появляется в момент времени 8 нс, а запускающий фронт сигнала clk\_1 – в 3 нс. Это полностью согласуется с фронтами S22 на рисунке 7.

Также обратите внимание на строку *Timing Exceptions*, которая указывает, что правила временного анализа были изменены с помощью команды *set\_multicycle\_path*.

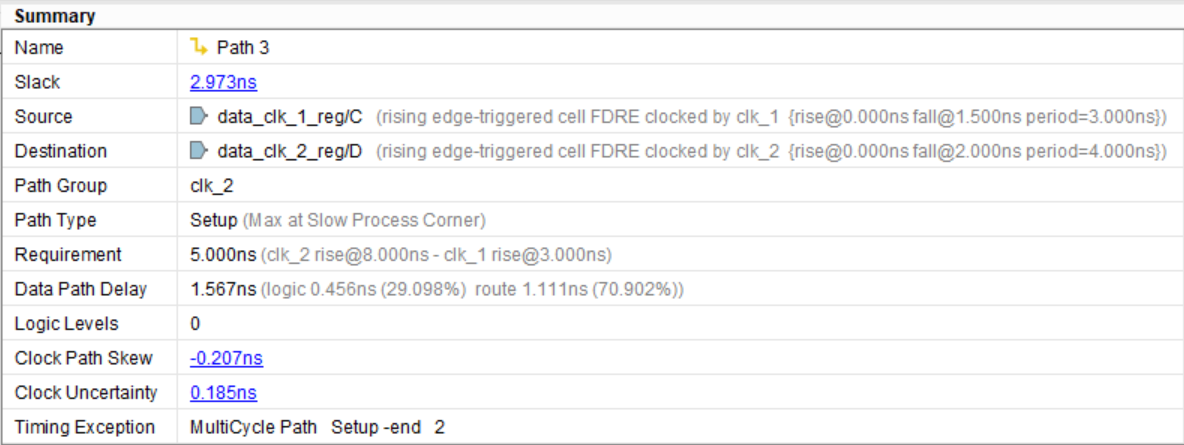


Рисунок 8. Раздел *Summary* временного отчета для анализа по *Setup*.

При анализе по *Hold* рассматривается условие H12a, для которого  
 *ΔTh* = 4 нс (см. *Requirement* на рисунке 9). Это ограничение оказалось слишком сложным и привело к нарушению временных ограничений. Данные поступают на вход защелкивающего триггера на 3.926 нс раньше, чем требуется.

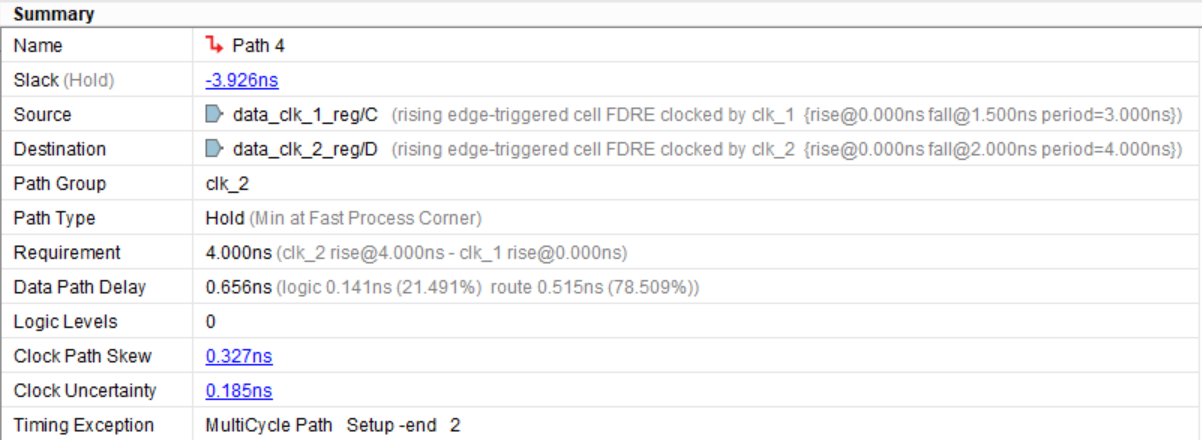


Рисунок 9. Раздел *Summary* временного отчета для анализа по *Hold*.

Чтобы исправить новые нарушения временных ограничений, можно сдвинуть положение защелкивающих фронтов для анализа по *Hold*. Для этого добавим в xdc-файл следующую команду:

# сдвиг защелкивающего фронта влево для анализа по Hold

set\_multicycle\_path 1 -hold -end -from [get\_pins data\_clk\_1\_reg/C]

-to [get\_pins data\_clk\_2\_reg/D]

Ключ -*hold* говорит, что модификации вносятся в анализ по *Hold*. Напомним, что при данном анализе проверяются ограничения для текущего запускающего фронта относительно прошлого защелкивающего [3]. Для временного анализатора это правило обозначается, как *Hold* = 0. Число 1 в команде *set\_multicycle\_path* изменяет это правило на *Hold* = 1. Это приводит к смещению защелкивающий фронтов на один фронт влево. Обновленная временная диаграмма для тактовых сигналов представлена на рисунке 10.

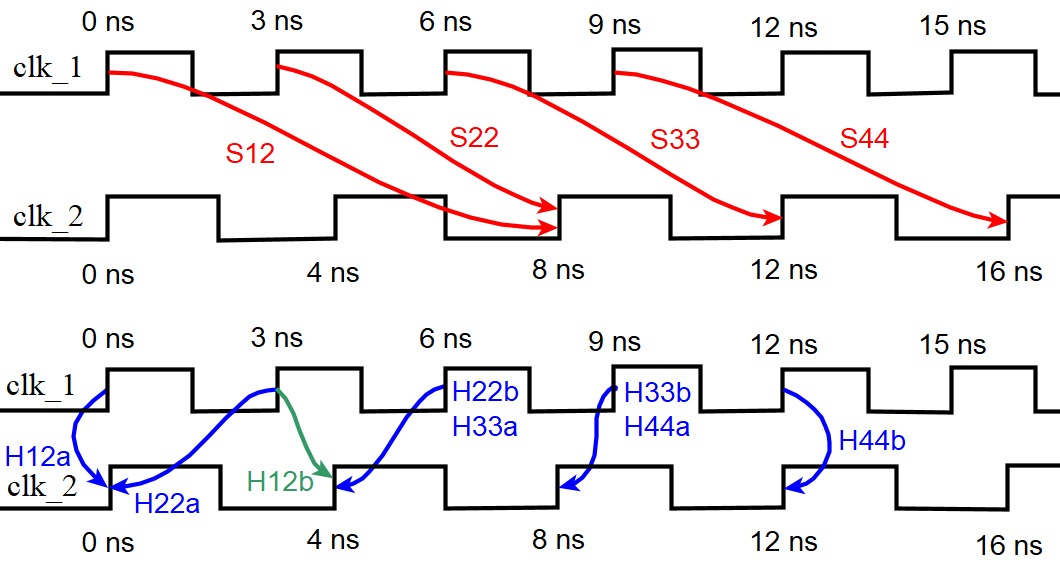


Рисунок 10. Диаграммы сигналов для временного анализа.

Изучим, что изменилось во временных отчетах. Для анализа по *Setup*   
по-прежнему самым критичным случаем является пара фронтов S22   
(см. рисунок 11). При анализе по *Hold* рассматривается условие H12a, для которого после изменения положения защелкивающего фронта получаем  
 *ΔTh* = 0 нс (см. *Requirement* на рисунке 12). Это условие не такое сложное, и теперь временные ограничения выполнены.

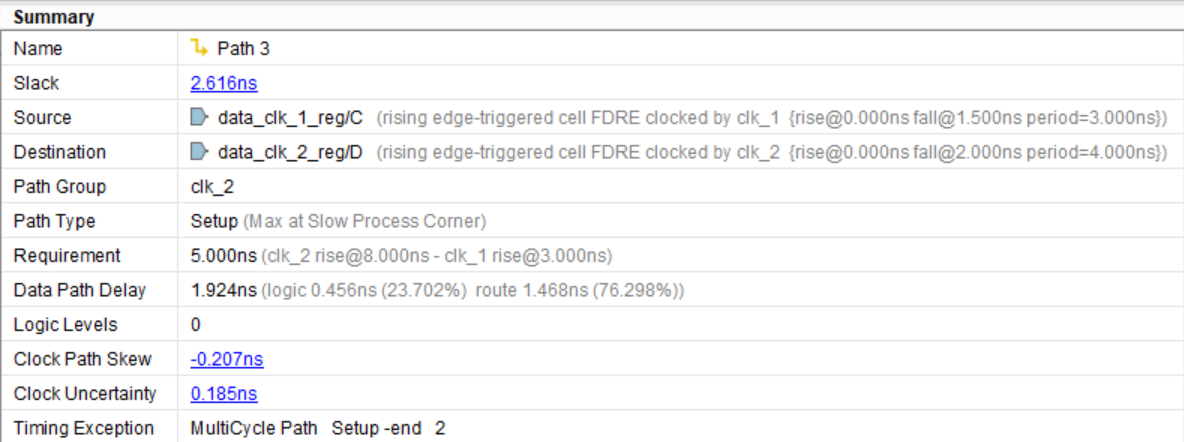


Рисунок 11. Раздел *Summary* временного отчета для анализа по *Setup*.

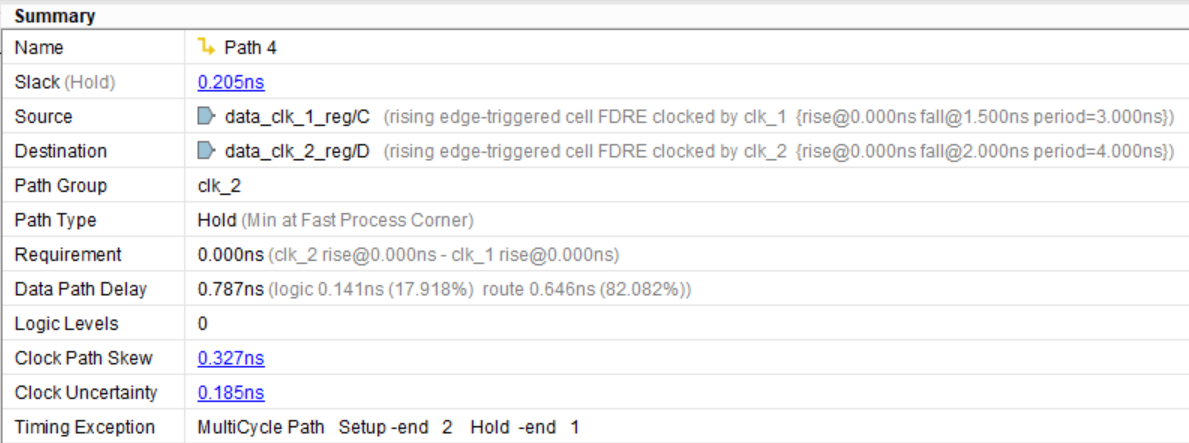


Рисунок 12. Раздел *Summary* временного отчета для анализа по *Hold*.

К сожалению, радоваться пока рано. Из рисунка 10 видно, что, несмотря на тот, что условие H12b больше не с чем не конфликтует, оно по-прежнему игнорируется при временном анализе. Это приводит к следующей проблеме.

Данные запущенные в момент времени 3 нс, точно не будут защелкнуты фронтом clk\_2, который появляется в нулевой момент времени (H22a). Также эти данные точно будут приняты фронтом clk\_2 в момент времени 8 нс (S22). Вспоминая, что анализ проводится для *ΔTh* = 0 нс, получаем, что данные на входе защелкивающего триггера изменятся в интервале между 3 и 8 нс. Может случится так, что это изменение произойдет в момент времени 4 нс, как раз, когда появляется фронт clk\_2. Это в свою очередь может привести к тому, что защёлкивающий триггер перейдет в метастабильное состояние.

Таким образом, изменения, внесенные нами в правила временного анализа, убрали нарушения, однако, это не гарантирует полной защиты от метастабильности. Всегда важно понимать, как именно проводится временной анализ и какие выводы можно сделать на его основе.

1. **Изменение положения запускающего фронта.**

С помощью команды *set\_multicycle\_path* можно также изменять положение запускающего фронта. Для этого нужно добавить ключ -*start*. Уберем из xdc-файл все ранее записанные команды *set\_multicycle\_path* и внесем следующее:

# сдвиг запускающего фронта влево для анализа по Setup

set\_multicycle\_path 2 -setup -start

-from [get\_pins data\_clk\_1\_reg/C] -to [get\_pins data\_clk\_2\_reg/D]

Для анализа по *Setup* запускающий фронт перемещается влево на одно позицию, что можно увидеть на рисунке 13. Так для пары S22 запускающий фронт теперь появляется в нулевой момент времени, а значение *ΔTsu*равно 4 нс. Для анализа по *Hold* фронты также изменяют свое положение. Самым сложным является условие H12a, для которого *ΔTh* = 0 – (–3) = 3 нс. Условие H12b, как и раньше, игнорируется из-за конфликта с S22.

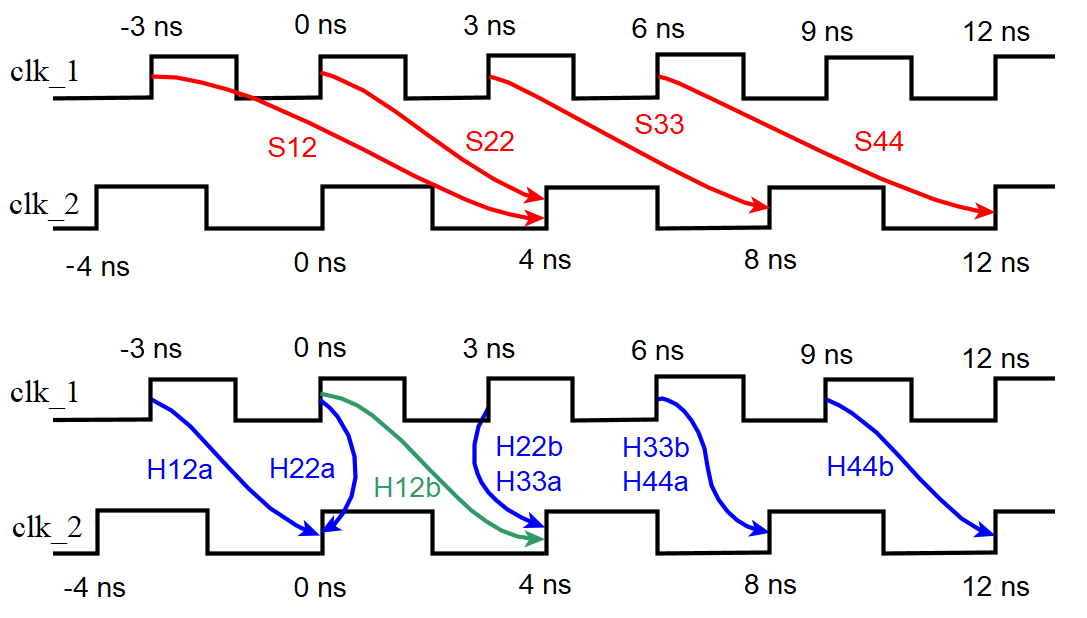


Рисунок 13. Диаграммы сигналов для временного анализа.

Рассмотрим, что изменилась во временных отчетах. Для анализа по *Setup*, представленного на рисунке 14, получаем, что ситуация только еще сильнее ухудшилась. Значение *Slack* равно –4.028 нс, хотя до использования команды *set\_multicycle\_path* его значение было равно –1.145 нс. Причиной   
является очень большое время распространения данных, равное 7.568 нс   
(см. строку отчета *Data Path Delay*). При этом 94% задержки связано с распространением данных через трассировочные линии.

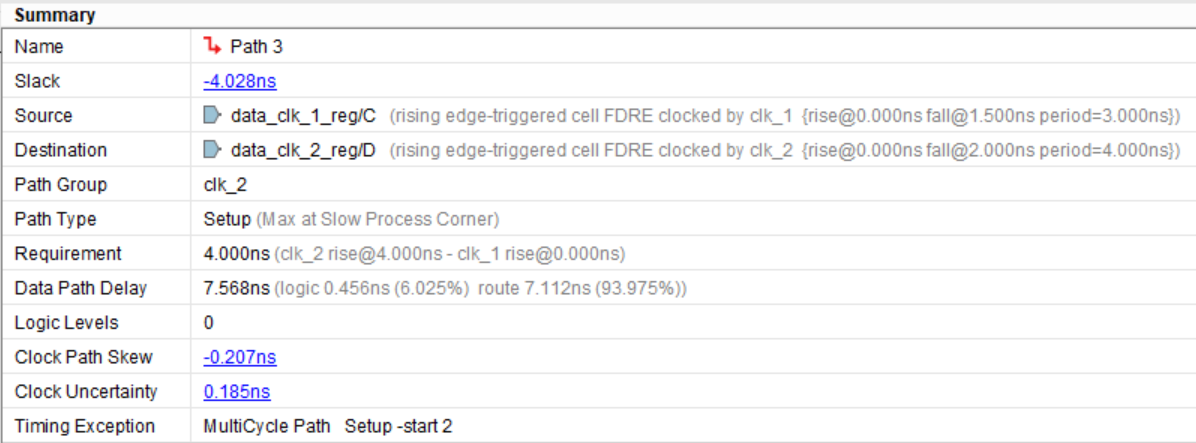


Рисунок 14. Раздел *Summary* временного отчета для анализа по *Setup*.

Чтобы понять, откуда взялась такая большая задержка, нужно рассмотреть результаты анализ по *Hold*, представленные на рисунке 15. Из строки *Requirement* можно увидеть, что проверяется условие H12a, для которого *ΔTh* = 3 нс. Чтобы выполнить временные ограничение, необходимо увеличить время распространения данных между триггерами. Это делается за счет увеличения длины трассировочных линий.

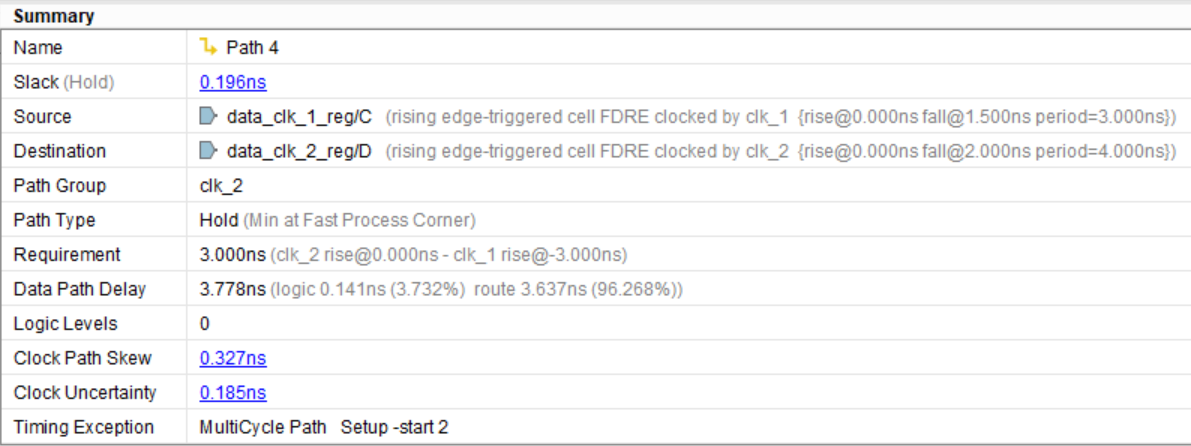


Рисунок 15. Раздел *Summary* временного отчета для анализа по *Hold*.

В этом также можно убедиться, если рассмотреть результаты размещения и трассировки проекта, показанные на рисунке 16. Путь между триггерами выделен белым цветом.

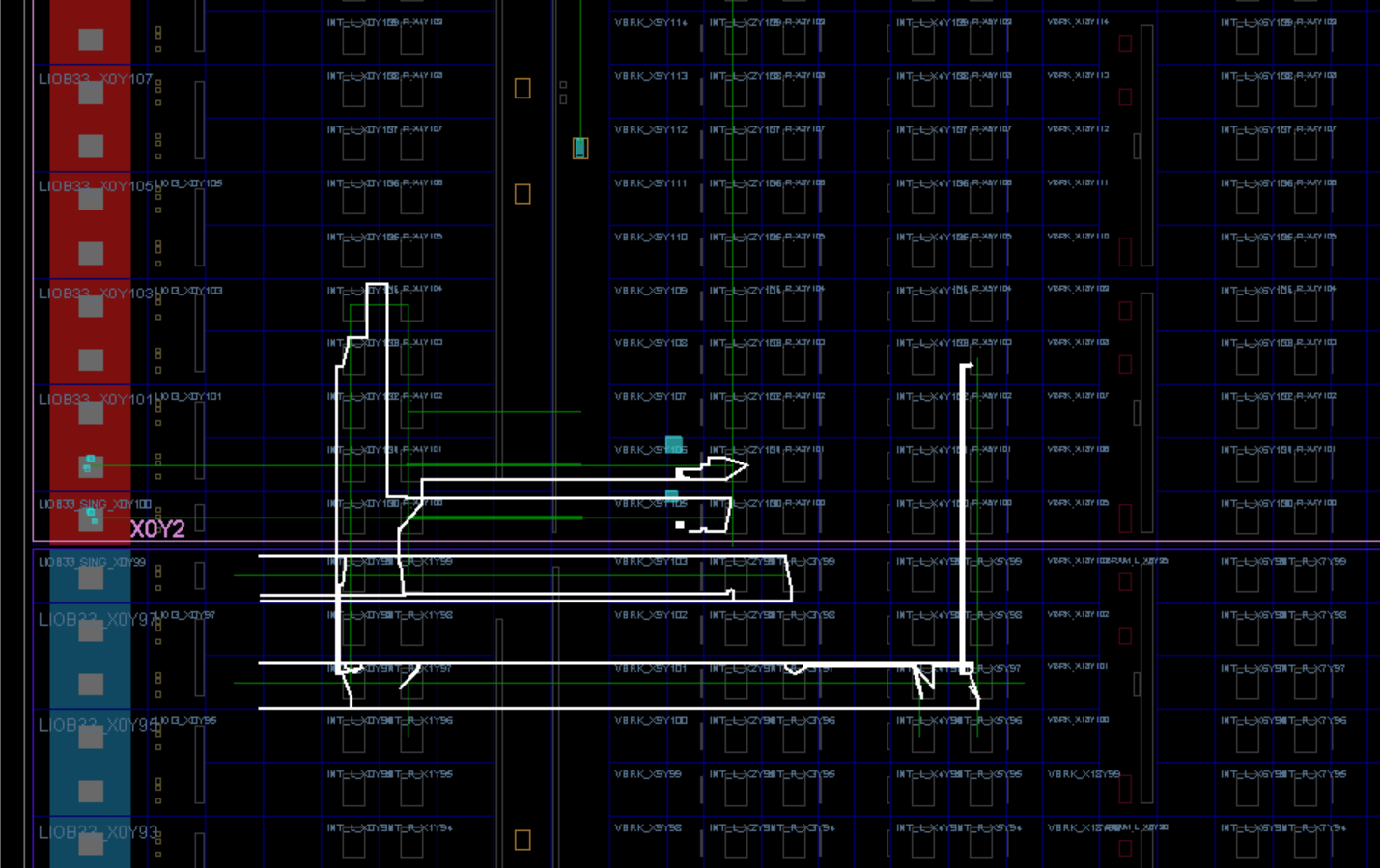


Рисунок 16. Результат размещения и трассировки проекта.

Чтобы ослабить такие сложные ограничения, можно изменить положение запускающих фронтов при анализе по *Hold*. Для этого добавим в xdc-файл команду *set\_multicycle\_path* с ключами -*hold* и -*start*:

# сдвиг запускающего фронта вправо для анализа по Hold

set\_multicycle\_path 1 -hold -start

-from [get\_pins data\_clk\_1\_reg/C] -to [get\_pins data\_clk\_2\_reg/D]

Запускающие фронты при этом будут сдвинуты на одну позицию вправо, что можно увидеть на рисунке 17.

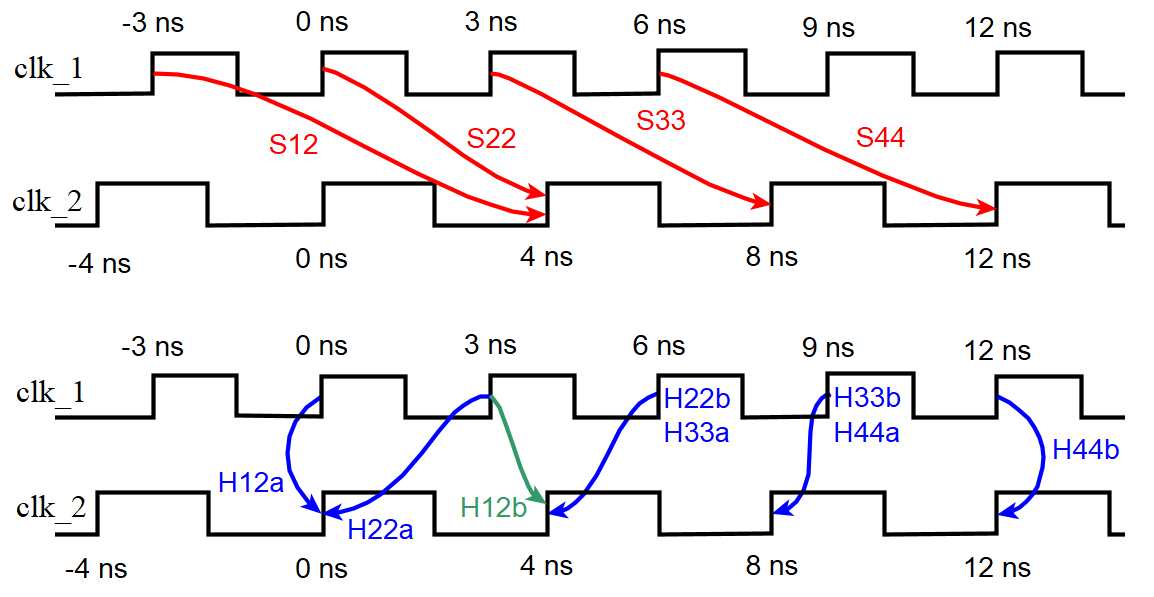


Рисунок 17. Диаграммы сигналов для временного анализа.

Требования по *Setup* при этом не изменяются, и самое сложное ограничение по-прежнему соответствует паре фронтов S22 (см. рисунок 18).

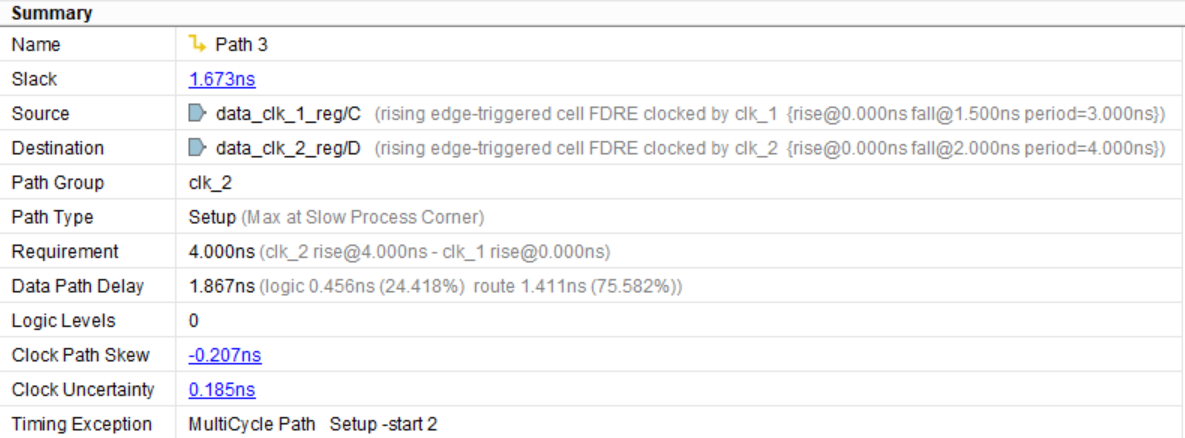


Рисунок 18. Раздел *Summary* временного отчета для анализа по *Setup*.

Анализ по Hold проводится при условии H12a, для которого значение *ΔTh* теперь равно 0 нс, что можно увидеть из строки *Requirement* на рисунке 19. Как и раньше, несмотря на тот, что условие H12b больше не с чем не конфликтует, оно игнорируется при временном анализе.

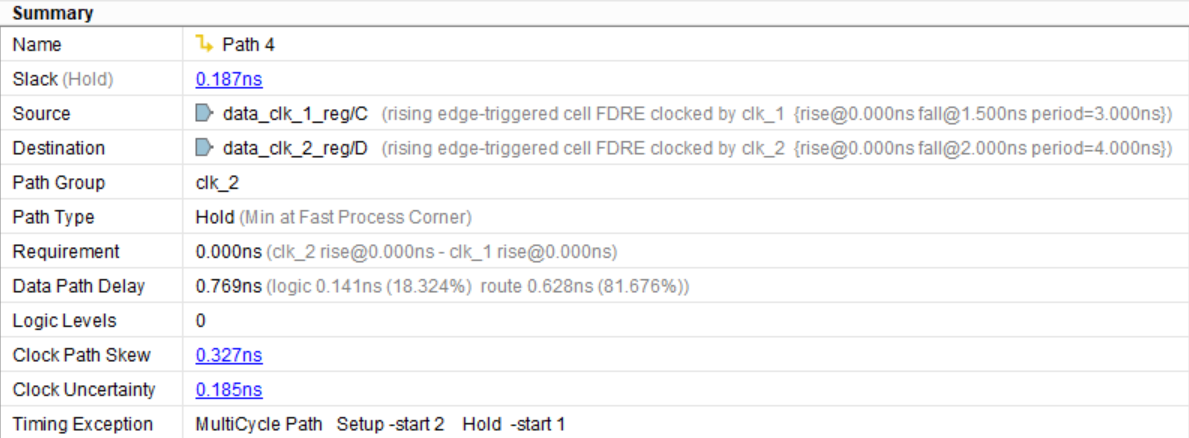


Рисунок 19. Раздел *Summary* временного отчета для анализа по *Hold*.

Требования по *Hold* теперь менее сложны, что и задержку распространения данных можно сделать меньше. Из рисунка 20 можно увидеть, что длина трассировочных линий теперь намного короче. Исходя из результатов отчетов, получаем, что все временные ограничения выполнены.

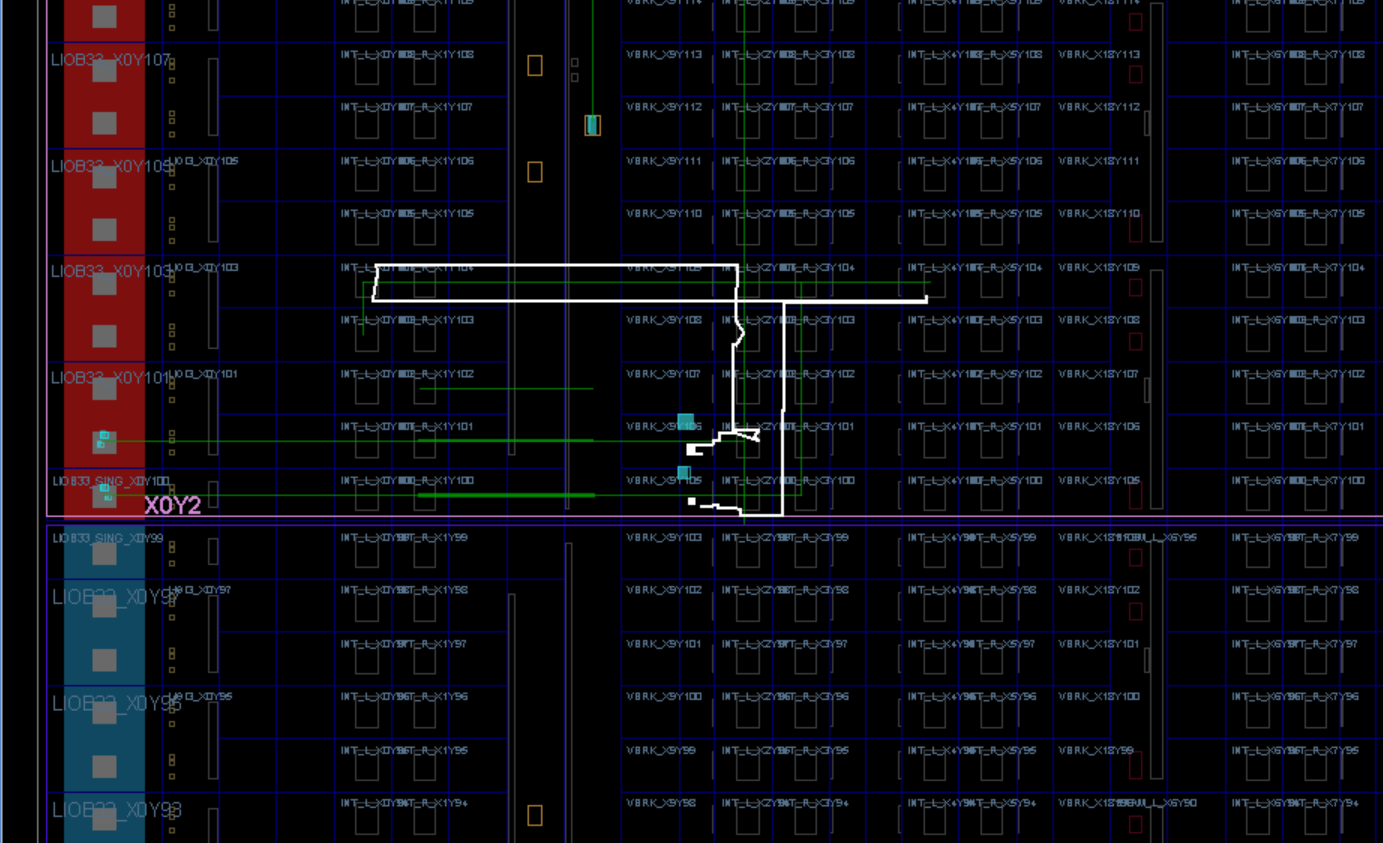


Рисунок 20. Результат размещения и трассировки проекта.

Однако, как и в предыдущем разделе, из-за игнорирования условия H12b мы не можем гарантировать отсутствие проблем с метастабильностью. Например, данные запущенные в нулевой момент времени точно будут приняты фронтом clk\_2 через 4 нс (пара S22 на рисунке 17). В свою очередь данные передаваемые следующим фронтом будут приняты в момент времени 8 нс (пара фронтов S33). Из-за того, что условие не учитывается, эти же данные могут быть защелкнуты фронтом из пары S22 или даже начать изменяться в момент прихода фронта. Такая ситуация может приводить к тому, что защелкивающий триггер будет попадать в метастабильное состояние.

Рассмотренный выше пример передачи данных между синхронными тактовыми доменами иллюстрирует возможности и способы применения команды *set\_multicycle\_path*. Однако, конкретно в этом случае этого недостаточно, чтобы гарантировать корректность передачи данных. Единственным способ решения проблемы – это добавить между доменами синхронизатор и исключить путь из временного анализа. Несколько ситуация, где применение команды *set\_multicycle\_path* является крайне полезным, рассмотрены в следующих разделах.

1. **Сдвиг фазы тактового сигнала.**

В качестве первого практического примера разберем ситуацию, когда требуется передать данные между двумя доменами, которые тактируются сигналами, сдвинутыми по фазе. Рассмотрим тот же проект, что и в предыдущих разделах. Его схема представлена на рисунке 4.

На ножку FPGA приходит тактовый сигнал i\_clk с периодом 10 нс, из которого с помощью PLL формируются два сигнала clk\_1 и clk\_2 с   
периодом 4 нс. Пусть также сигнал clk\_2 задержан относительно clk\_1   
на 90 градусов (1 нс). Настройки PLL представление на рисунке 21.

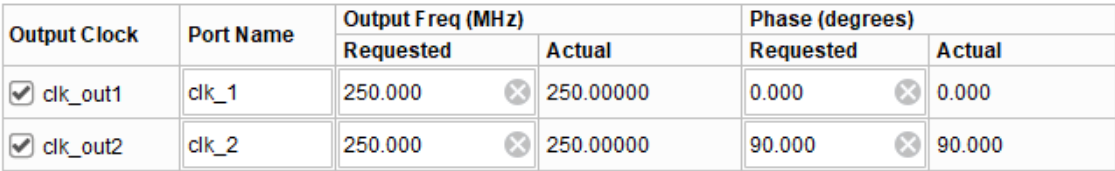


Рисунок 21. Настройки PLL.

Добавим в xdc-файл следующие команды, задающие период входного тактового сигнала и изменяющие имена выходных сигналов PLL:

# период входного тактового сигнала

create\_clock -period 10.0 -name i\_clk [get\_ports i\_clk]

# объявление переменных, которые указывают места формирования тактовых сигналов

set pll\_input    [get\_pins PLL/inst/mmcm\_adv\_inst/CLKIN1]

set pll\_output\_1 [get\_pins PLL/inst/mmcm\_adv\_inst/CLKOUT0]

set pll\_output\_2 [get\_pins PLL/inst/mmcm\_adv\_inst/CLKOUT1]

# обновление имен сгенерированных с помощью PLL тактовых сигналов

create\_generated\_clock -name clk\_1 -source $pll\_input

-master\_clock [get\_clocks i\_clk] $pll\_output\_1

create\_generated\_clock -name clk\_2 -source $pll\_input

-master\_clock [get\_clocks i\_clk] $pll\_output\_2

Временная диаграмма сигналов показана на рисунке 22. Для каждой пары запускающих и защелкивающих фронтов получаем, что *ΔTsu* = 1 нс. Для   
всех условия для анализа по *Hold* также получаем одинаковые значения  
 *ΔTh* = -3 нс.

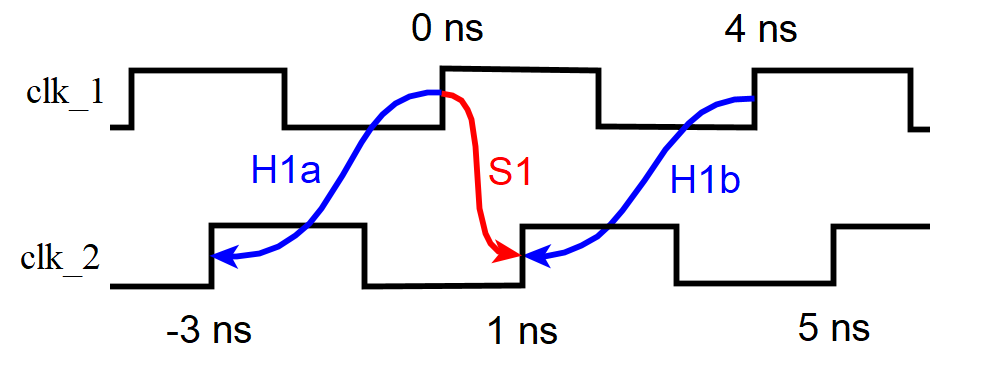


Рисунок 22. Диаграммы сигналов для временного анализа.

Данные рассуждения подтверждаются временными отчетами, представленными на рисунках 23 и 24. Можно увидеть, что анализ по *Setup* проводится для пары фронтов S1, а анализ по *Hold* – для условия H1b. Требование *ΔTsu* = 1 нс слишком сложное, что приводит к нарушению временных ограничений. Данные опаздывают на 0.215 нс.

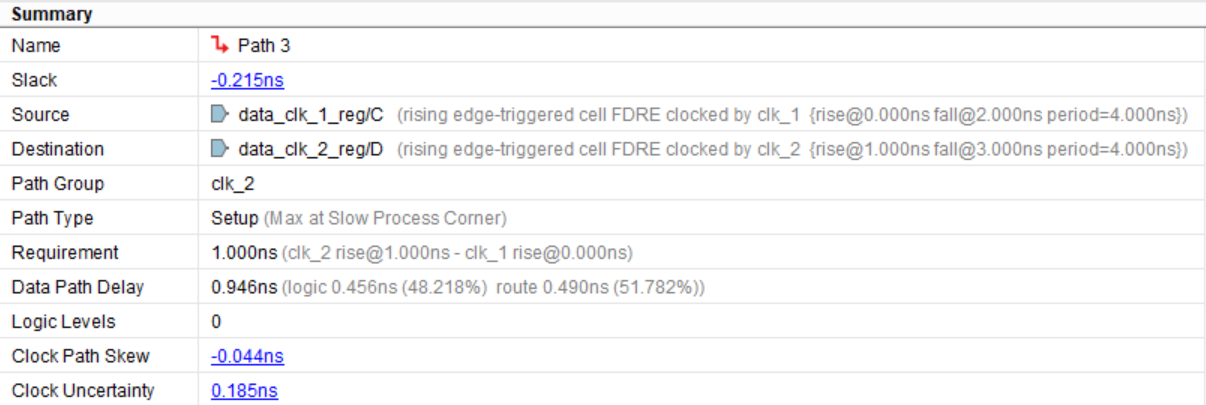


Рисунок 23. Раздел *Summary* временного отчета для анализа по *Setup*.

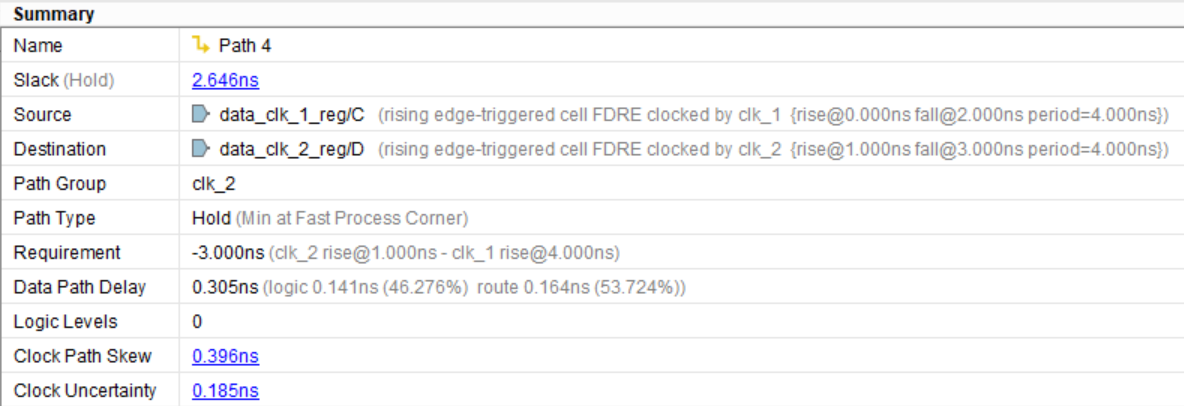


Рисунок 24. Раздел *Summary* временного отчета для анализа по *Hold*.

Ограничения по *Setup* можно упростить, если защелкивать данные не ближайшим фронтом clk\_2, а следующим через один. То есть сдвинуть защелкивающий фронт на одну позицию вправо. Для этого добавим в   
xdc-файл команду *set\_multicycle\_path*:

# сдвиг защелкивающего фронта вправо для анализа по Setup

set\_multicycle\_path 2 -setup -end -from [get\_pins data\_clk\_1\_reg/C]

-to [get\_pins data\_clk\_2\_reg/D]

Обновленная временная диаграмма для тактовых сигналов показана на рисунке 25. Для анализа по *Setup* получаем, что *ΔTsu* = 5 нс. Защелкивающие фронты для анализа по *Hold* также смещаются, и теперь значение *ΔTh* равно единице.

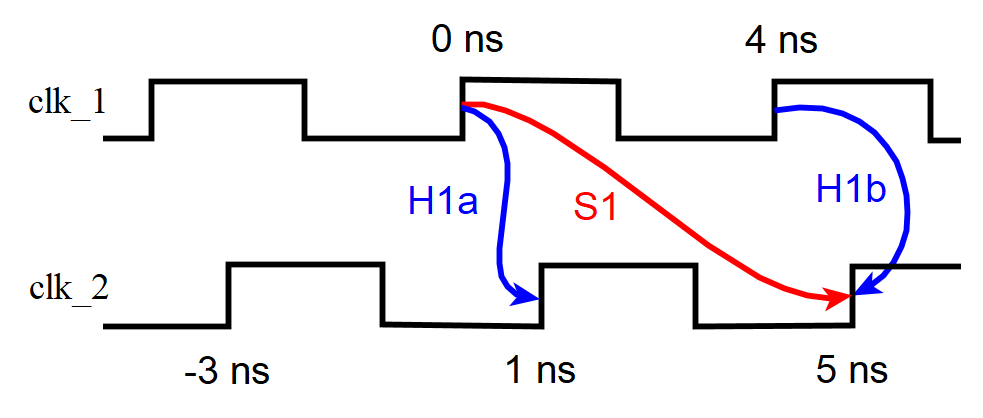


Рисунок 25. Диаграммы сигналов для временного анализа.

Временные отчеты после внесения изменений представлены на рисунках 26 и 27. Можно увидеть, что все временные ограничение выполнены.

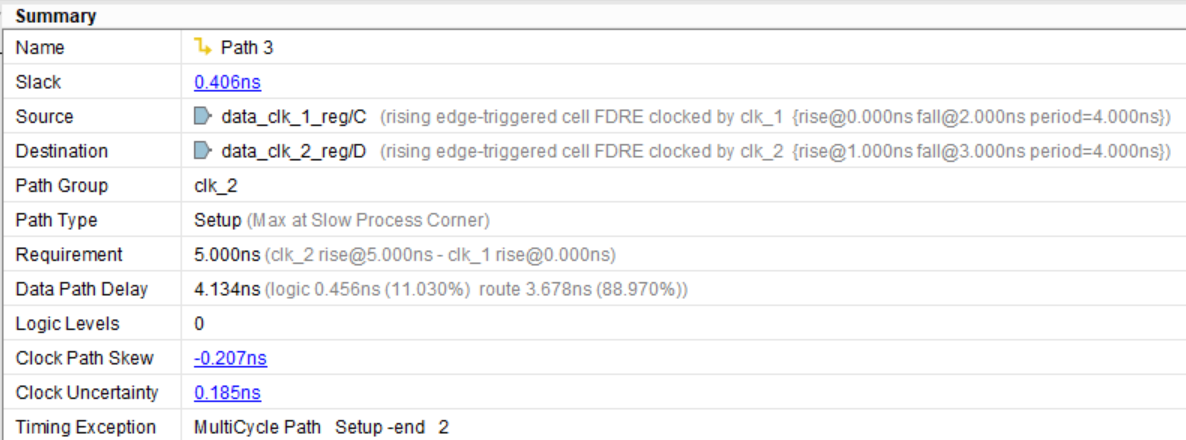


Рисунок 26. Раздел *Summary* временного отчета для анализа по *Setup*.

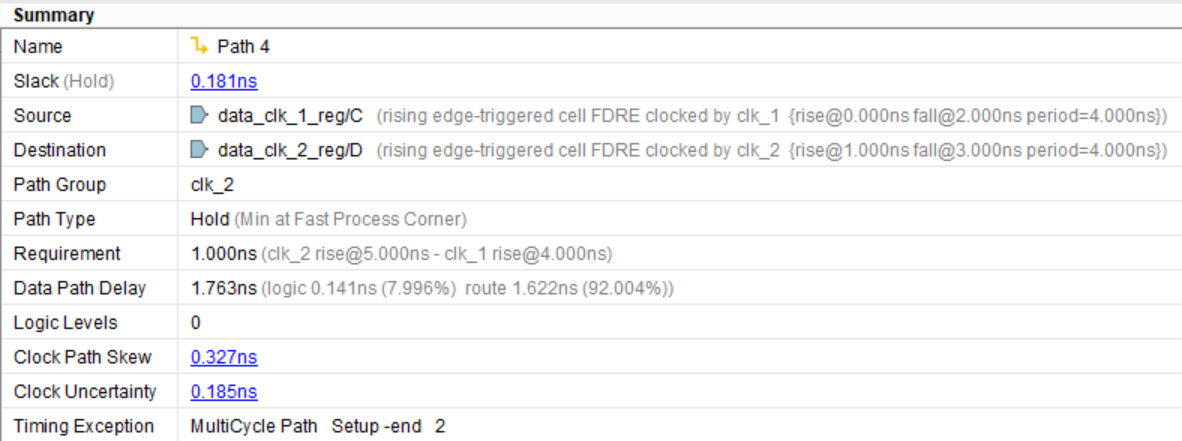


Рисунок 27. Раздел *Summary* временного отчета для анализа по *Hold*.

1. **Работа модуля по строб-сигналу**

Команда *set\_multicycle\_path* может оказаться очень полезной для изменения правил временного анализа для модулей, имеющих управляющий строб-сигнал. Рассмотрим проект, описание которого представлено ниже:

module top\_2 (

    input  logic        clk,

    input  logic        reset,

    output logic [31:0] q

);

    // строб-сигнал для счетчика

    logic counter\_ce;

    // формирование строб-сигнала

    always\_ff @(posedge clk)

        if (reset)

            counter\_ce <= 1'b0;

        else

            counter\_ce <= ~counter\_ce;

    // счетчик

    always\_ff @(posedge clk)

        if (reset)

            q <= '0;

        else if (counter\_ce)

            q <= q + 1'b1;

endmodule

Проект состоит из 32-битного счетчика и формирователя строб-сигнала. Строб-сигнал *counter\_ce* инвертирует свое значение по каждому фронту тактового сигнала *clk*. Этот сигнал поступает на входы clock enable (CE) триггеров, которые входят в состав счетчика. Счетчик изменяет свое состояние по фронту *clk* только при условии, что значение строб-сигнала *counter\_ce* равно единице, то есть каждый второй такт.

Пусть требуется, чтобы счетчик тактировался сигналом с периодом   
2.5 нс. В xdc-файл с ограничениями добавим следующую команду:

# период входного тактового сигнала

create\_clock -period 2.5 -name clk [get\_ports clk]

Диаграмма сигналов, используемая при временном анализе показана на рисунке 28. По умолчанию данные должны быть приняты ближайщим фронтом сигнала, который тактирует защелкивающий триггер. Таким образом, данные запущенные в нулевой момент времени должны быть приняты спустя 2.5 нс (пара фронтов S1, *ΔTsu* = 2.5 нс). При анализе по *Hold* для всех условий получаем, что *ΔTh* = 0 нс.

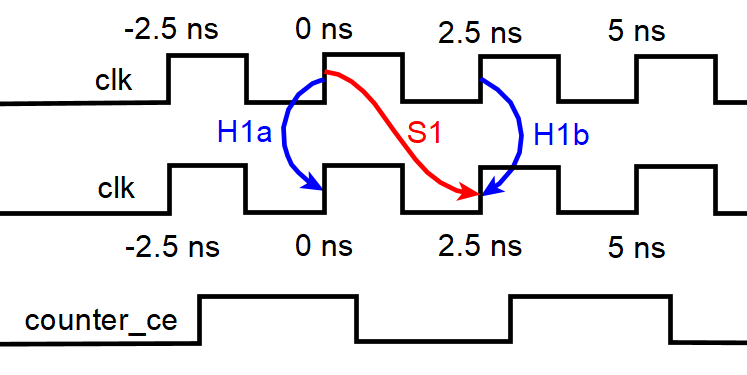


Рисунок 28. Диаграммы сигналов для временного анализа.

Результаты временного анализа показаны ниже (см. рисунок 29). Временные ограничение при анализе по *Setup* не выполняются для шести путей внутри счетчика. Однако, так как для защелкивающего фронта из пары S1 строб-сигнал *counter\_ce* имеет неактивный уровень. То есть, триггеры счетчика не будут реагировать на этот защелкивающий фронт.

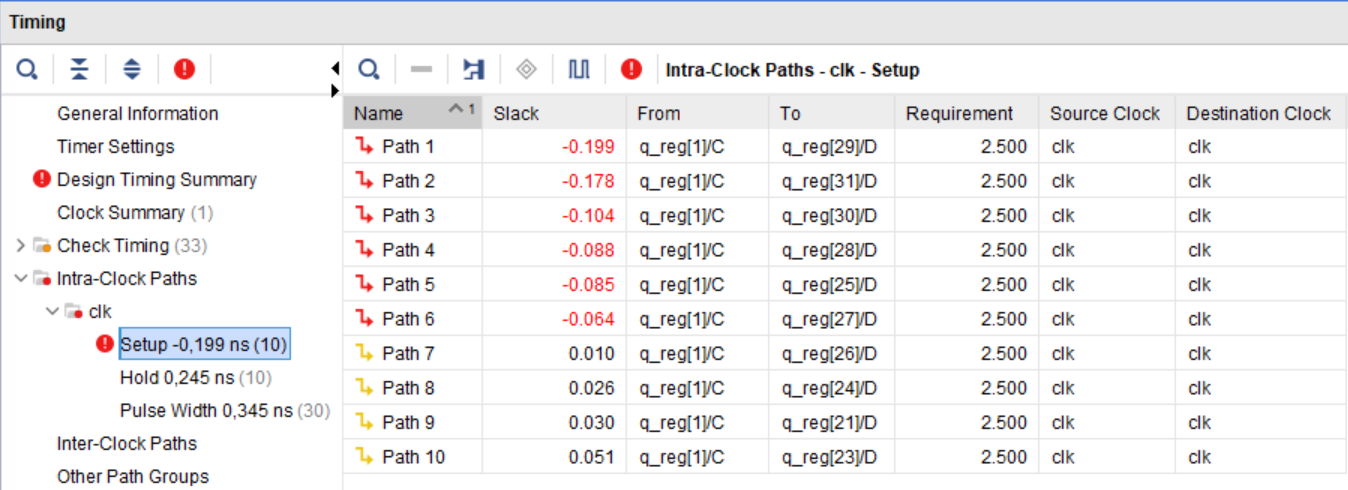


Рисунок 29. Результаты временного анализа.

Данные запущенные в нулевой момент времени должны быть приняты ближайшим фронтом, для которого сигнал *counter\_ce* имеет единичное значение. Этот фронт формируется в момент времени 5 нс. Глядя на рисунок 29, можно увидеть, что наименьшее значение *Slack* равно –0.199 нс, то есть фронт, появляющийся спустя 2.5 нс после защелкивающего фронта из   
пары S1, корректно примет данные. Таким образом, несмотря на наличие нарушений временных ограничений, триггеры счетчика не будут попадать в метастабильное состояние.

Чтобы каждый раз не проводить вручную подобные рассуждения, можно модифицировать правила выполнения временного. Добавим в xdc-файл следующие команды:

# получение триггеров счетчика по имени ячеек netlist

set counter\_flops [get\_cells q\_reg[\*]]

# добавление multicycle path

set\_multicycle\_path 2 -setup -end -from $counter\_flops

-to $counter\_flops

set\_multicycle\_path 1 -hold  -end -from $counter\_flops

-to $counter\_flops

С помощью команды *get\_cells* находим все примитивы с именем *q\_reg*[\*], что соответствует поиску всех триггеров счетчика. Рузельтат записывается в переменную *counter\_flops*, которая в дальнейшем используется для указания начала и конца путей с измененными правилами временного анализа. Далее, первая команда *set\_multicycle\_path* сдвигает защелкивающий фронт при анализе по *Setup* на одну позицию вправо. Это приводит к соответствующему сдвигу защелкивающий фронтов для анализа по *Hold*. Вторая команда *set\_multicycle\_path* возвращает эти фронты на прежнее место. Обновленная диаграмма сигналов, используемая для временного анализа, показана на рисунке 30.

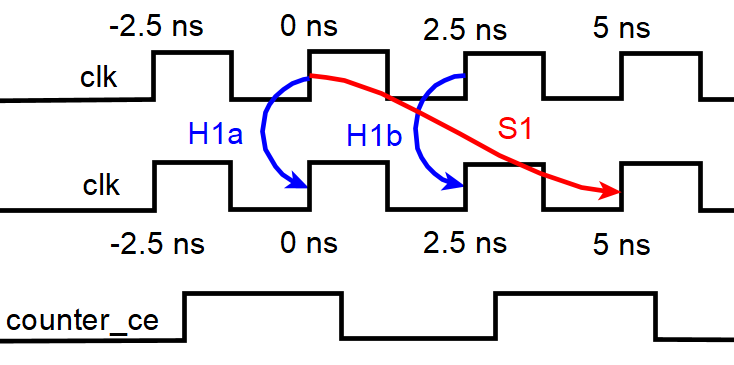


Рисунок 30. Диаграммы сигналов для временного анализа.

Отчет с результатами анализа представлен на рисунке 31. Можно увидеть, что временные ограничения выполняются. Теперь для путей между триггерами счетчика значение *ΔTsu* равно 5 нс. Правила временного анализа для путей между формирователем строб-сигнала *counter\_ce* и счетчиком не модифицировались, и для них по-прежнему *ΔTsu* = 2.5 нс.

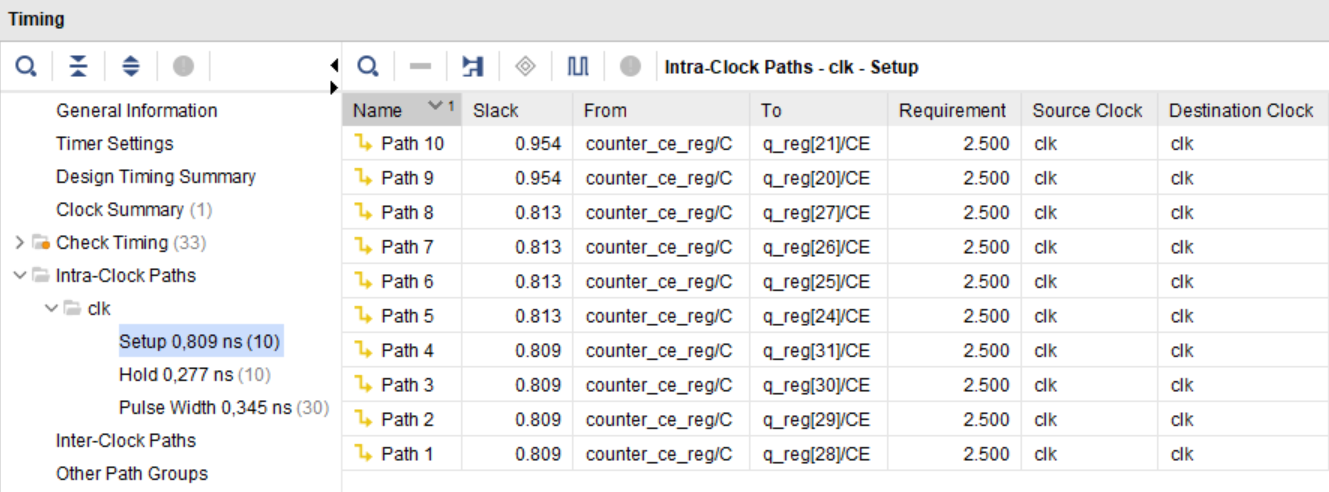


Рисунок 31. Результаты временного анализа.

Дополнительно рассмотрим еще пару способов указания начала и конца путей, на которое должна воздействовать команда *set\_multicycle\_path*. После проведения оптимизаций к именам триггеров могут добавляться дополнительные суффикса, поэтому иногда удобно выполнять поиск следующим образом:

# получение триггеров счетчика по имени переменной и типу

set counter\_flops [get\_cells q\*

-filter {PRIMITIVE\_TYPE =~ FLOP\_LATCH.flop.\*}]

С помощью команды *get\_cells* находим все примитивы проекта, имя которых начинается с символа *q*. Это могут быть и триггеры, и LUT счетчика. Далее с помощью ключа -*filter* удаляем все примитивы, которое не являются триггерами.

Также можно постараться найти все триггеры, на вход CE которых поступает строб-сигнал *counter\_ce*. Для этого воспользуемся следующими командами:

# получение триггеров счетчика по сигналу counter\_ce

set ce\_net [get\_nets counter\_ce]

set flops\_ce\_pins [get\_pins -of\_objects $ce\_net

-filter {NAME =~ "\*CE"}]

set counter\_flops [get\_cells -of\_objects $flops\_ce\_pins]

С помощью команды *get\_nets* получаем наш строб-сигнал *counter\_ce*. Далее используем команду *get\_pins* для поиска пинов, на которые приходит строб-сигнал и имя которых заканчивается на CE. Наконец, командой *get\_cells* получаем все примитивы, которым принадлежат найденные ранее пины.

1. **Ограничения для входных и выходных сигналов.**

В завершении рассмотрим, как можно использовать команду *set\_multicycle\_path* для изменения правил временного анализа входных и выходных сигналов.

В [4] был рассмотрен временной анализ при передаче данных из FPGA во внешнее устройство для случая Source Synchronous. На рисунке 32 показан анализируемый путь, на который нанесены задержки сигналов. Более подробно о введенных обозначениях можно также прочитать в [4].

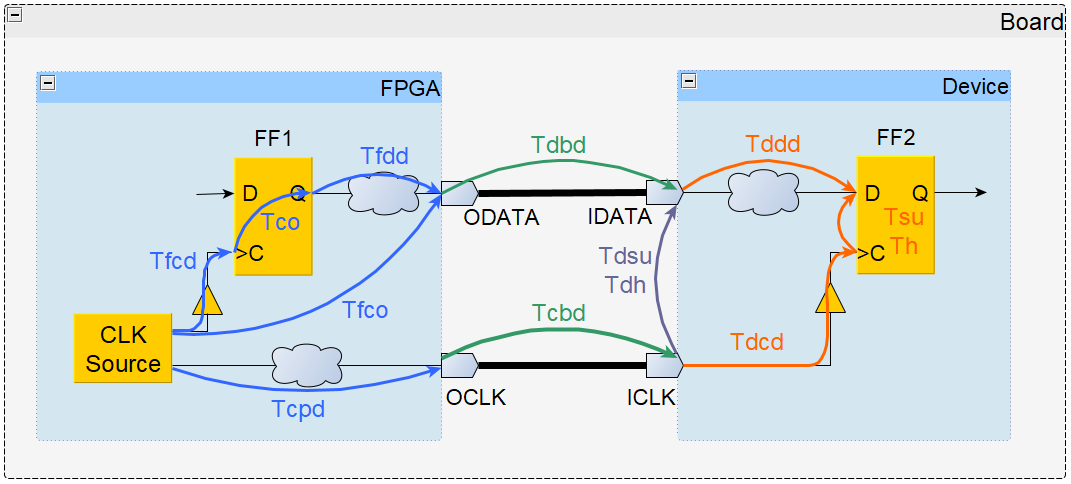


Рисунок 32. Путь с задержками для данных и тактовых сигналов.

В качестве примера, пусть в FPGA загружен простой проект, состоящий всего из одного триггера. Его описание на SystemVerilog представлено ниже:

module top\_3 (

    input  logic i\_clk,

    input  logic i\_data,

    output logic o\_clk,

    output logic o\_data

);

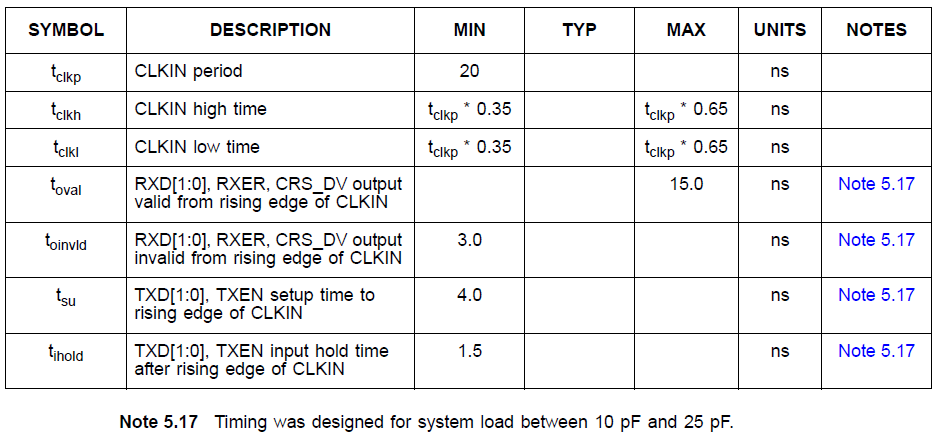
    always\_ff @(posedge i\_clk)

        o\_data <= i\_data;

    assign o\_clk = i\_clk;

endmodule

Как и в [4] в качестве внешнего устройства рассмотрим в микросхему Ethernet PHY LAN8740A [5]. На рисунке 33 приведены таблица со значениями задержек и временная диаграмма сигналов из datasheet на LAN8740A. Для краткости временные ограничения будут продемонстрированы для одного выходного сигнала FPGA, который подключен к ножке TXD[0] микросхемы LAN8740A.



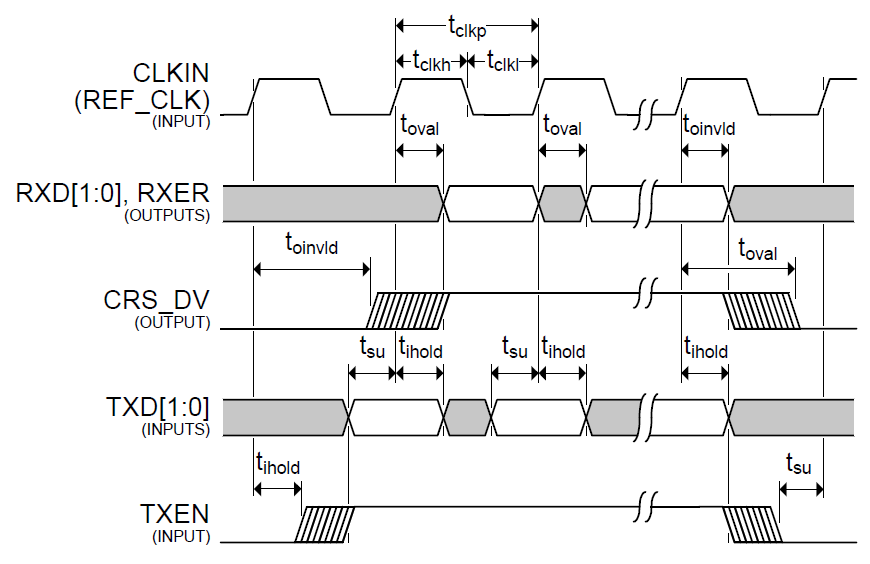


Рисунок 33. Задержки и временные диаграммы для LAN8740A.

Из рисунка 33 получим время установки *Tdsu* и удержания *Tdh* для LAN8740A, а также минимальные период такового сигнала. Также будем считать, что минимальные и максимальные задержки распространения данных и тактового сигнала по дорожкам печатной платы известны. В качестве примера примем следующие значения в наносекундах: *Tdbd\_max* = 0.15, *Tdbd\_min* = 0.1, *Tcbd\_max* = 0.12 и *Tcbd\_min* = 0.07.

Занесем все задержки в xdc-файл и сразу добавим ограничения на период входного (i\_clk) и выходного (o\_clk) тактовых сигналов:

# период тактового сигнала

set Tclk 20

# время установки и удержания для микросхемы LAN8740A

set Tdsu 4.0

set Tdh 1.5

# минимальное и максимальное время распространения данных по дорожкам платы

set Tdbd\_max 0.15

set Tdbd\_min 0.1

# минимальное и максимальное время распространения тактового сигнала по дорожкам платы

set Tcbd\_max 0.12

set Tcbd\_min 0.07

# ограничение на период входного тактового сигнала

create\_clock -period $Tclk -name i\_clk [get\_ports i\_clk]

# ограничение на период выходного тактового сигнала

create\_generated\_clock -name o\_clk -source [get\_ports i\_clk] -divide\_by 1 [get\_ports o\_clk]

Пусть требуется запускать данные из FPGA и принимать их микросхемой LAN8740A с помощью одного и того же фронта тактового сигнала. В [4] для этого случая были получены уравнения для задания временных ограничений:

Так как по умолчанию при временном анализе по *Setup* защелкивающим фронтом считается следующий фронт тактового сигнала (правило *Setup* = 1), а нам требуется, чтобы запускающий и защелкивающий фронты совпадали, в уравнение для *output\_delay\_max* пришлось добавить дополнительное слагаемое *Tclk*. По той же причине слагаемое *Tclk* появляется в уравнении для анализа по *Hold*.

Основываясь на уравнениях (2), добавим ограничения для выходного сигнала *o\_data* в xdc-файл:

# временные ограничения для выходного сигнала o\_data

set\_output\_delay -clock o\_clk

-max [expr $Tclk + $Tdbd\_max + $Tdsu - $Tcbd\_min] [get\_ports o\_data]

set\_output\_delay -clock o\_clk

-min [expr $Tclk + $Tdbd\_min - $Tdh  - $Tcbd\_max] [get\_ports o\_data]

Ниже показаны разделы временного отчета, описывающие задержки для защёлкивающего фронта такового сигнала. При анализе по *Setup*   
(см. рисунок 34) считается, что защелкивающий фронт формируется в момент времени 20 нс, что на один период позже появления запускающего фронта. Дополнительная задержка в 20 нс компенсируется за счет дополнительно слагаемого *Tclk* в уравнениях (2), поэтому в отчете величина *output delay* равна –24.080 нс.

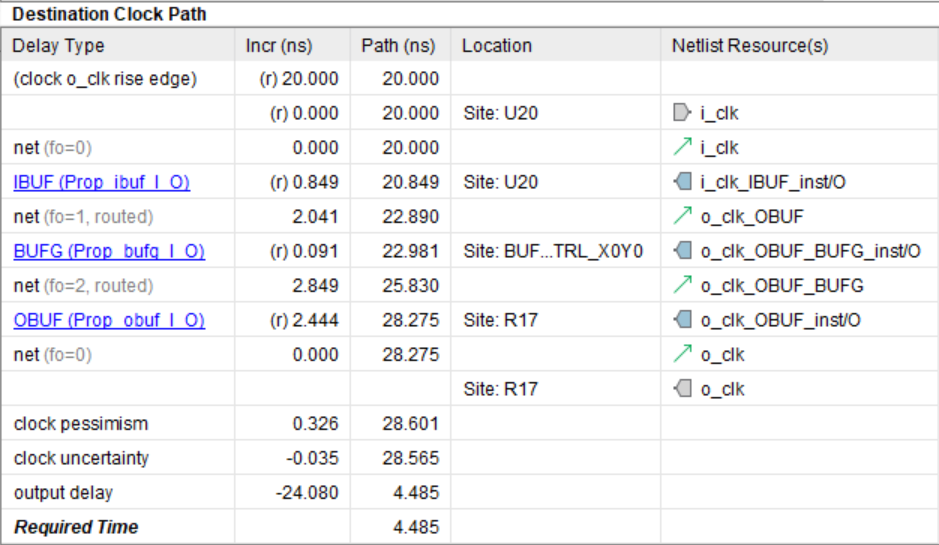


Рисунок 34. Задержки для защелкивающего фронта при анализе по *Setup*.

Та же картина наблюдается при анализе по *Hold* (см. рисунок 35). Предыдущий защелкивающий фронт должен формироваться на один период раньше запускающего, то есть в –20 нс. Однако, он появляется в нулевой момент времени. Это противоречие исправляется за счет того, что в отчет значение *output delay* равно –18.480 нс.

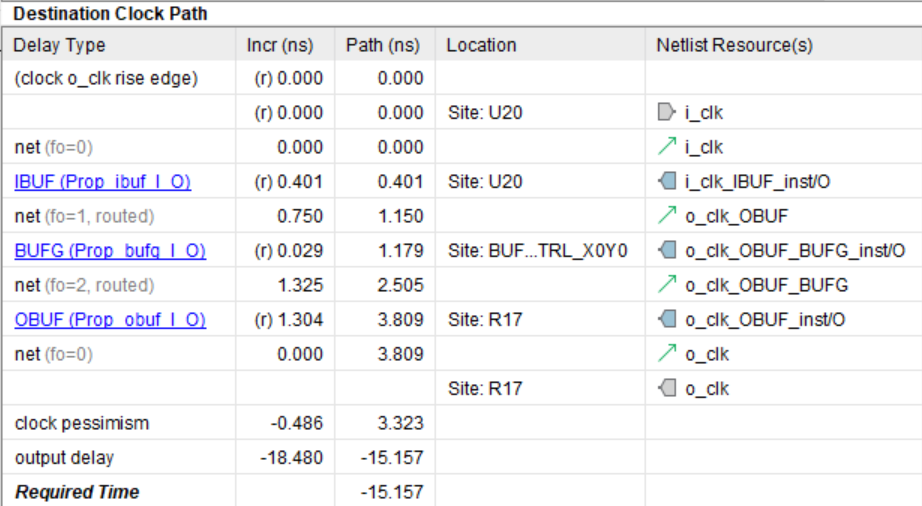


Рисунок 35. Задержки для защелкивающего фронта при анализе по *Hold*.

Все описанные выше несоответствия усложняют интерпретацию временных отчетов. Эту проблему можно решить, если модифицировать правила временного анализа с помощью команды *set\_multicycle\_path*. Для начала удалим из уравнений (2) слагаемые *Tclk*. В этом случае они примет вид:

Соответствующим образом обновим временные ограничения для выходного сигнала *o\_data* в xdc-файле:

# временные ограничения для выходного сигнала o\_data

set\_output\_delay -clock o\_clk

-max [expr $Tdbd\_max + $Tdsu - $Tcbd\_min] [get\_ports o\_data]

set\_output\_delay -clock o\_clk

-min [expr $Tdbd\_min - $Tdh  - $Tcbd\_max] [get\_ports o\_data]

Для анализа по *Setup* по умолчанию временной анализатор считает, что защелкивающий фронт формируется спустя один период после запускающего   
(правило *Setup* = 1). Нам же требуется, чтобы они появлялись в один и тот же момент времени. Для этого подвинем защелкивающий фронт на одну позицию влево. Это обозначается в виде правила *Setup* = 0.

Добавим в xdc-файл следующую команду:

# добавление multicycle path

set\_multicycle\_path 0 -setup -end -to [get\_ports o\_data]

Как и ранее ключ -*end* указывает, что изменяется положение защелкивающего фронта. Обратите внимание, что выше отсутствует ключ   
-*from*, то есть не указано начало пути. Так можно делать, и в этом случае модифицируется анализ для всех путей, которое заканчиваются на выходной ножке FPGA o\_data.

Рассмотрим, какие изменения произошли во временных отчетах, результаты которых представлены. При анализе по *Setup* теперь, как и ожидается, защелкивающий фронт появляется в нулевой момент времени. Также обратите внимание, что на рисунке 36 отсутствует величина *clock uncertainty*, задающая неопределенность для тактового сигнала из-за джиттера. Запускающий и защелкивающий фронты теперь это один и тот же фронт, поэтому джиттер не оказывает никакого влияния на задержки. Таким образом, временной отчет стал не только более понятным, но и более точным.

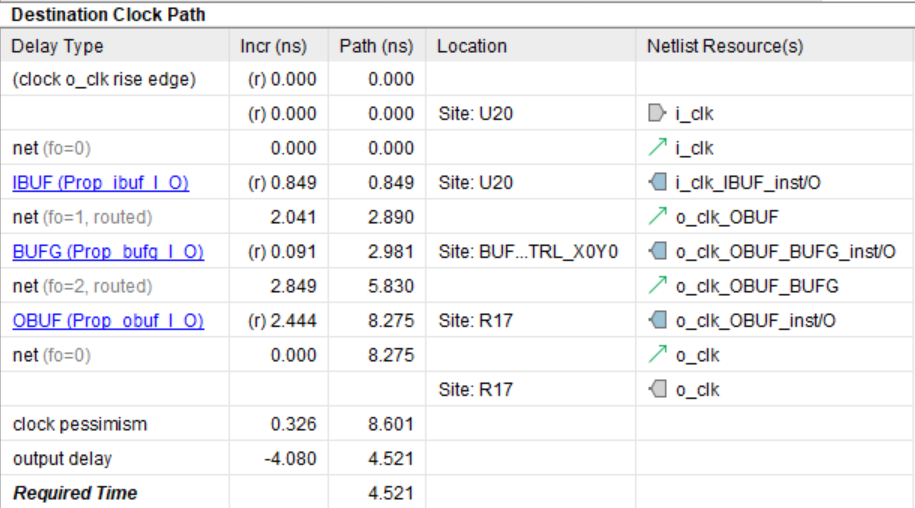


Рисунок 36. Задержки для защелкивающего фронта при анализе по *Setup*.

При анализе по *Hold* защелкивающий фронт теперь формируется в момент времени –20 нс (см. рисунок 37). Строка *clock uncertainty* теперь наоборот появляется, так как теперь запускающий и предыдущий защелкивающий фронты – это разные фронт.

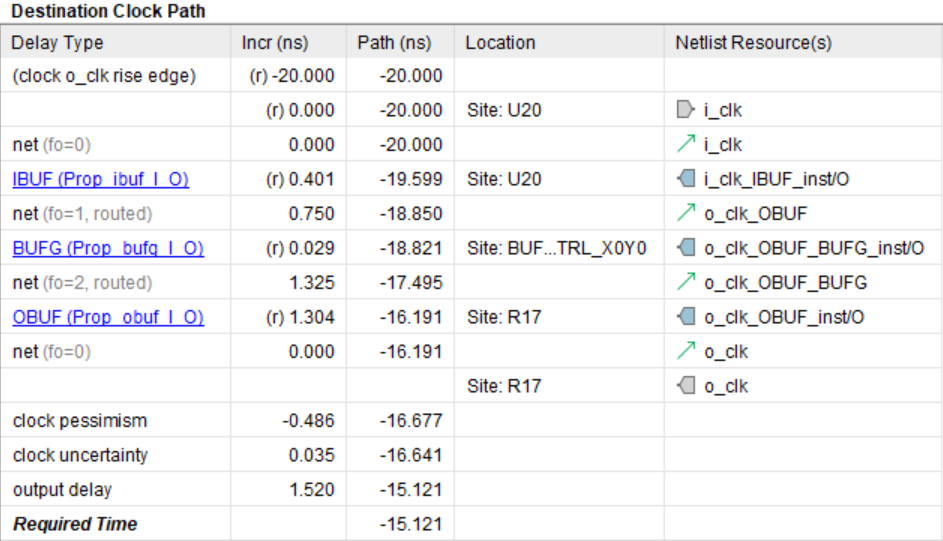


Рисунок 37. Задержки для защелкивающего фронта при анализе по *Hold*.

**Заключение.**

В статье представлен способ модификации правил проведения временного анализа. Показаны возможность использования команды *set\_multicycle\_path*, для исправления нарушений временных ограничений. Продемонстрировано несколько практических примеров.

**Ссылки.**

1. Основы статического временного анализа. Часть 5: False Path
2. [Vivado Design Analysis and Closure Techniques (UG 906)](https://docs.xilinx.com/v/u/2017.3-English/ug906-vivado-design-analysis)
3. [Основы статического временного анализа. Часть 1: Period Constraint](https://fpga-systems.ru/static-timing-analysis-part-1-period-constraints) [Vivado Using Constraints (UG 903)](https://docs.xilinx.com/v/u/2013.1-English/ug903-vivado-using-constraints)
4. [Datasheet LAN8740A](https://fpga-systems.ru/go?http://www.datasheet.es/PDF/1021686/LAN8740A-pdf.html)
5. Основы статического временного анализа. Часть 4: Source Synchronous Output Delay Constraint