**Основы статического временного анализа.**

**Часть 1: Period Constraint.**

**Введение.**

Данная статья является первой из планируемой серии статей по временным ограничениям в FPGA. Основная цель – познакомить начинающих разработчиков с основами статического временного анализа. В этой статье будет рассмотрен анализ самого простого случая – передача данных между двумя последовательными элементами внутри FPGA с общим тактовым сигналом. Показан вывод уравнений временного анализа и продемонстрировано их применение анализатором Vivado.

1. **Цель статического временного анализа.**

Все производители FPGA в рекомендациях по разработке указывают на необходимость избегать наличия защелок (*latch*), поэтому грамотно сделанный проект представляет из себя синхронное последовательное цифровое устройство. Большая часть схемы любого синхронного устройства состоит из набора регистров, изменяющих свое состояние по фронту или спаду тактового сигнала, которые отделены друг от друга комбинационной логикой. Для определенности в дальнейшем будем считать, что данные передаются между регистрами по фронту тактового сигнала. Таким образом, типичный путь, который проходят данные внутри FPGA, имеет вид, представленный на рисунке 1.

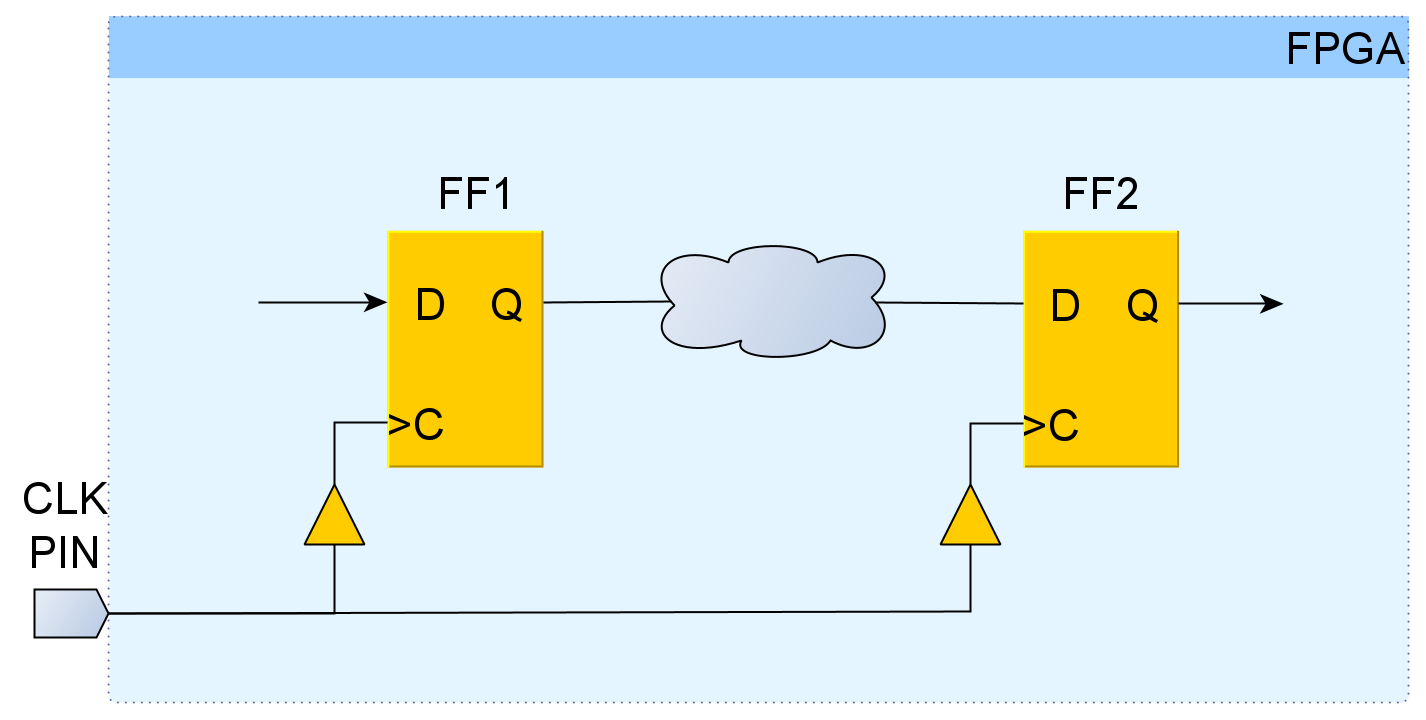


Рисунок 1. Типичный путь данных внутри FPGA.

При поступлении фронта данные с D входа триггера FF1 переходят на выход Q, распространяются через комбинационную логику и попадают на D вход триггера FF2. Данный фронт будем называть запускающий (*source edge* или *launch edge*). Спустя один период тактового сигнала появляется следующий фронт, по которому триггер FF2 защелкивает данные от FF1 на своем D входе и передает далее на свой выход. Одновременно с этим от FF1 начинают распространяться следующие данные. Такой фронт будем называть защелкивающий (*destination edge* или *latch edge*). Также будем называть FF1 запускающим триггером, а FF2 – защелкивающим.

Чтобы данные корректно распространялись от триггера к триггеру описанным выше образом, должны быть выполнены два ограничения:

* данные от FF1 должны распространяться достаточно быстро, чтобы успеть дойти до триггера FF2 раньше защелкивающего фронта (максимальное время распространения);
* следующие данные от FF1 должны распространяться достаточно медленно, чтобы защелкивающий фронт успел дойти до FF2 и захватить предыдущие данные от FF1 (минимальное время распространения).

Цель статического временного анализа заключается в том, чтобы для каждого пути (*path*) между двумя последовательными элементами рассчитать задержки распространения сигналов и установить, выполняются ли два приведенных выше ограничения. Считается, что путь данных начинается на тактовом входе запускающего элемента (триггер FF1) и заканчивается на информационном входе защелкивающего элемента (триггер FF2).

Рассмотрим, каким образом временной анализатор решает эту задачу. На рисунке 2 представлен путь, на который нанесены задержки для данных и тактового сигнала.

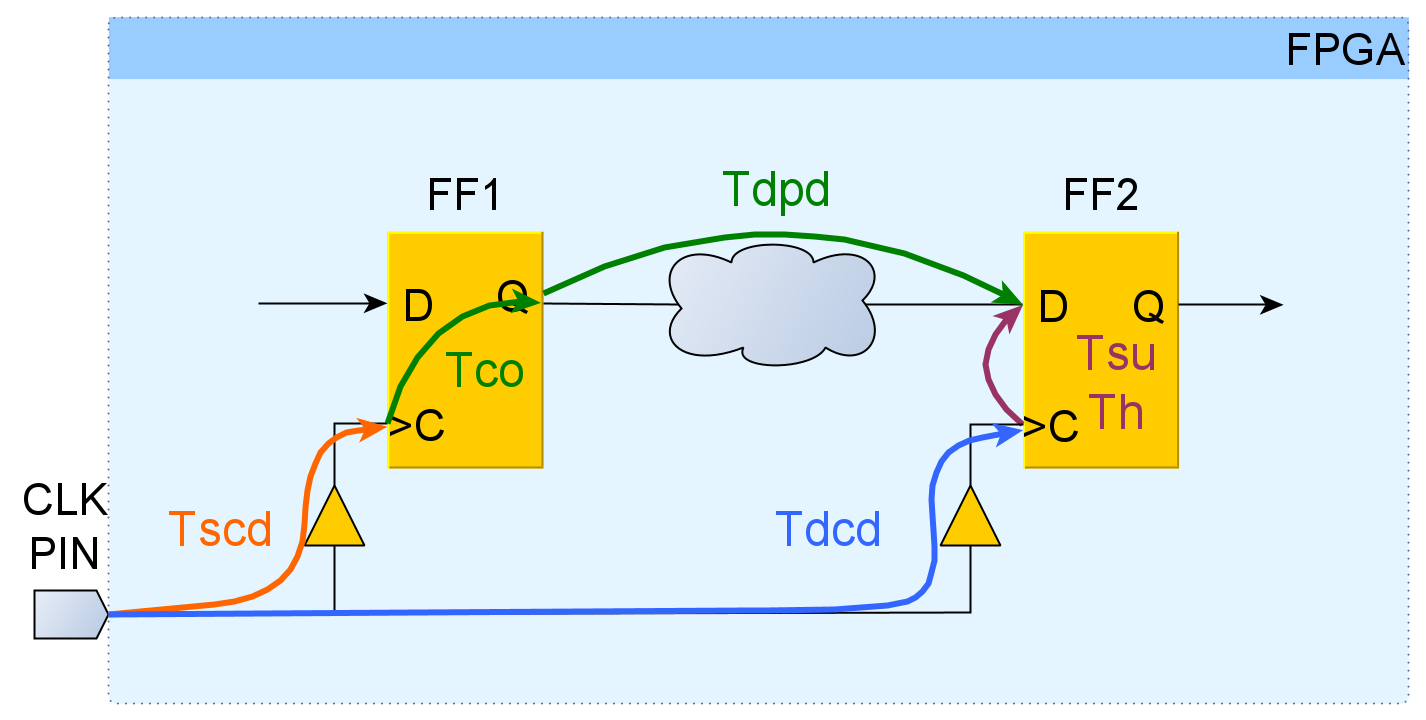


Рисунок 2. Путь с задержками для данных и тактового сигнала.

Ниже даны определения задержек, представленных на рисунке 2.

* (***S****ource* ***C****lock* ***D****elay*) – задержка тактового сигнала от источника, в нашем примере ножка FPGA clk\_pin, до тактового входа триггера FF1;
* (***D****estination* ***C****lock* ***D****elay*) – задержка тактового сигнала от источника до тактового входа триггера FF2;
* (***C****lock to* ***O****utput*) – интервал времени между приходом фронта на тактовый вход триггера и появлением данных на выходе Q;
* (***D****ata* ***P****ropagation* ***D****elay*) – задержка распространения данных по соединениям (*nets*) и через комбинационную логику;
* (***S****et****U****p time*) – время установки. До прихода защелкивающего фронта данные на D входе триггера уже должны быть стабильны в течении времени *Tsu*.
* (***H****old time*) – время удержания. После прихода защелкивающего фронта данные на D входе триггера не должны изменяться в течении времени *Th*.

Будем обозначать период тактового сигнала как *Tclk*. При проведении временного анализа все события отсчитываются от некоторого нулевого момента времени, в качестве которого обычно рассматривается появление запускающего фронта на ножке FPGA.

1. **Максимальное время распространения.**

Для начала рассмотрим, каким образом выполняется анализ для проверки ограничения на максимальное время распространения. Данный анализ также называют анализ по *Setup*.

Временной анализ по *Setup* всегда проводится для самого пессимистичного случая, которому соответствует максимально задержанный запускающий фронт, максимально медленное распространение данных и максимально быстро распространяющийся защелкивающий фронт.

На первом этапе рассчитывается время распространения данных до защёлкивающего триггера, считая, что запускающий фронт появляется на ножке FPGA в нулевой момент времени. Уравнения для расчета представлены ниже (см. рисунок 2):

* Время прибытия фронта к запускающему триггеру (***S****ource* ***С****lock* ***A****rrival time*):
* Задержка распространения данных (***D****ata* ***D****elay*):
* Время прибытия данных на вход защелкивающего триггера (***D****ata* ***A****rrival time*):

Далее вычисляется время прибытия защелкивающего фронта тактового сигнала и требуемое время прибытия данных. Защелкивающий фронт появляется через один такт после запускающего фронта, поэтому к задержке распространения добавлен один период тактового сигнала.

* Время прибытия фронта к защелкивающему триггеру (***D****estination* ***C****lock* ***A****rrival time*):
* Требуемое время прибытия данных (***D****ata* ***R****equired time*):

В предыдущем уравнении учитывается, что данные должны прийти на время *Tsu* раньше защелкивающего фронта. Таким образом, чтобы выполнялось требование на максимальное время распространения (*Setup*), данные должны дойти до конца пути не позднее времени *Tdr*.

Результат работы статического анализатора представляется в виде запаса для задержки данных (*Slack*), который вычисляется по формуле

Положительный *Slack* указывает на то, что данные доходят до места назначения раньше, чем это требуется. Отрицательное значение *Slack* означает нарушение ограничения по *Setup*.

Используя ранее полученные уравнения, можно получить ряд выражений для расчета *Slack*

Стоит заметить, что величина равная разности времени распространения тактового сигнал до запускающего и защелкивающего триггеров называется расфазировкой тактового сигнала (*Clock Skew*)

Можно увидеть, что положительное значение расфазировки при анализе по *Setup* увеличивает *Slack*.

1. **Минимальное время распространения.**

Аналогичным образом рассмотрим, как выполняется анализ для проверки ограничения на минимальное время распространения. Данный анализ также называется анализом по *Hold* и проводится для самого пессимистичного случая, при котором запускающий фронт и данные распространяются наиболее быстро, а защелкивающий фронт – максимально медленно.

Глядя на рисунок 2, можно получить следующие уравнения:

* Время прибытия фронта к запускающему триггеру (***S****ource* ***С****lock* ***A****rrival time*):
* Задержка распространения данных (***D****ata* ***D****elay*):
* Время прибытия данных на вход защелкивающего триггера (***D****ata* ***A****rrival time*):
* Время прибытия фронта к защелкивающему триггеру (***D****estination* ***C****lock* ***A****rrival time*):
* Требуемое время прибытия данных (***D****ata* ***R****equired time*):

Защелкивающий фронт для предыдущих данных появляется в тот же момент времени, что и запускающий фронт для следующих данных, поэтому к задержке распространения период тактового сигнала не добавляется.

Слагаемое в уравнении для учитывает, что данные не должны изменяться в течении времени удержания после защелкивающего фронта. Формулируя по-другому – следующие данные должны прийти на время *Th* позже защелкивающего фронта для предыдущих данных.

Чтобы было удовлетворено ограничение по *Hold*, следующие данные должны попасть на D вход защелкивающего триггера не раньше времени *Tdr*.

При анализе по *Hold* выражение для вычисления *Slack* имеет вид:

Если *Slack* положительный, то это значит, что следующие данные приходят позже, чем требуется. Отрицательное значение *Slack* указывает на нарушение ограничения по *Hold*.

Используя полученные выше уравнения, выражение для *Slack* можно записать в виде:

Интересно заметить, что в выражении для *Slack* расфазировка тактового сигнала входит с знаком противоположным тому, который был при анализе по *Setup*. То есть положительное значение расфазировки уменьшает *Slack*.

1. **Временной анализ в Vivado.**

Рассмотрим каким образом Vivado выполняет временной анализ пути между двумя триггерами внутри FPGA.

При проведении синтеза создается схема проекта, состоящая из логических примитивов, поэтому для каждого пути известно через какие комбинационные элементы (LUT, MUX, CARRY CHAIN) он проходит. Типовые задержки для этих элементов приводятся в datasheet для конкретного кристалла. Например, для Artix 7 (DS181) [1] в таблице 27 указано время распространения через LUT (*Tilo*), а также время установки (*Tas*) и удержания (*Tah*) для триггеров. В таблицах 32 – 35 можно найти задержки при распространении сигнала через различные виды буферов.

После размещения и трассировки проекта появляется информация о задержках сигналов при распространении через *nets* (соединения) с учетом их длины. Таким образом, на этом этапе анализатору Vivado известны значения все переменных, которые входят в уравнения 7 и 11, кроме периода тактового сигнала *Tclk*, Тактовый сигнал поступает от внешнего генератора, частота которого для Vivado напрямую неизвестна.

Рассмотрим самый простой пример проекта, который состоит из двух триггеров между которыми расположен LUT, реализующий логическое отрицание. Схема проекта показана на рисунке 3. Описание на System Verilog представлено ниже:

module top (

    input  logic clk\_pin,

    input  logic in1,

    output logic out1

);

    logic ff1, ff2;

    always\_ff @(posedge clk\_pin)

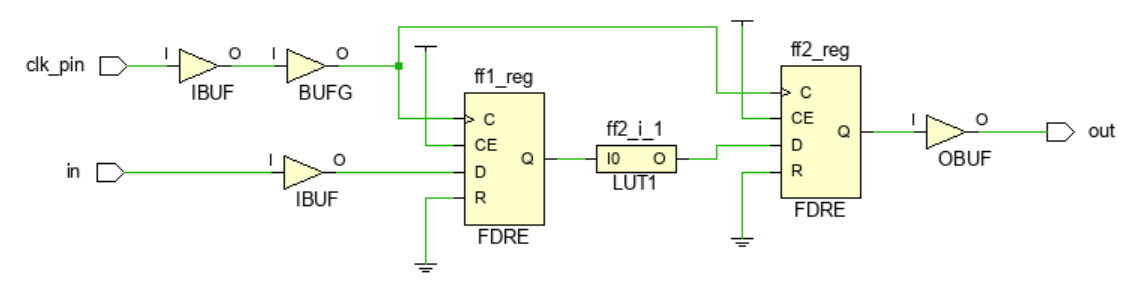
        ff1 <= in1;

    always\_ff @(posedge clk\_pin)

        ff2 <= ~ff1;

    assign out1 = ff2;

endmodule

Рисунок 3. Схема проекта.

Чтобы провести временной анализ и проверить проект на выполнение ограничений по *Setup* и *Hold*, в файле ограничений (xdc) требуется указать период тактового сигнала *Tclk*. Если считать, что значение *Tclk* равно 10 нс. (частота 100 МГц), то в xdc-файле необходимо записать следующую команду [2]:

create\_clock -period 10.000 -name clk\_100MHz [get\_ports clk\_pin]

Опция –period задает период в наносекундах. Конструкция [get\_ports clk\_pin] возвращает внешний порт проекта с именем clk\_pin, указывая источник сигнала. С помощью опции –name можно задать имя тактового сигнала. Если этого не сделать имя тактового сигнала будет совпадать с именем порта.

Увидеть пути, для которых проведен временной анализ, можно, если после размещения и трассировки открыть отчет Timing Summary на вкладке Intra-Clock Path / clk\_100MHz (рисунок 4).

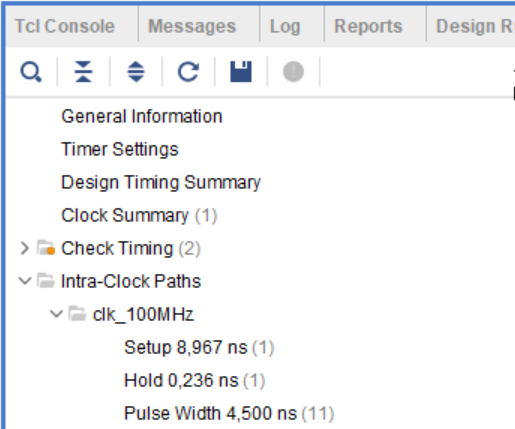


Рисунок 4. Разделы отчета Timing Summary.

1. **Анализ ограничения по Setup.**

Открыв вкладку Setup раздела Intra-Clock Path и дважды нажав на одни из показанных путей, можно получить расширенный отчет Path Report. Данный отчет состоит из четырех разделов. Рассмотрим их по порядку.

Первый раздел представлен на рисунке 5. В данном разделе представлены общие сведения, такие как имя пути (Name), рассчитанный для данного пути *Slack*, имя и период тактового сигнала (Path Group и Requirement). Началом пути является тактовый вход триггера FF1 (Source), а заканчивается путь на D входе триггера FF2 (Destination).

Также указывается задержка распространении данных (Data Path Delay), в наших обозначениях *Tdd* (уравнение 2), и количество уровней комбинационной логики. Для данного примера логика состоит из одного LUT, реализующего инвертор. Расфазировка тактового сигнала (*Tskew* из уравнения 8) в отчете обозначена как Clock Path Skew.

В конце раздела приводится значение неопределенности для тактового сигнала (Clock Uncertainty). Об этом параметре более подробно будет рассказано далее в статье в пункте 7.

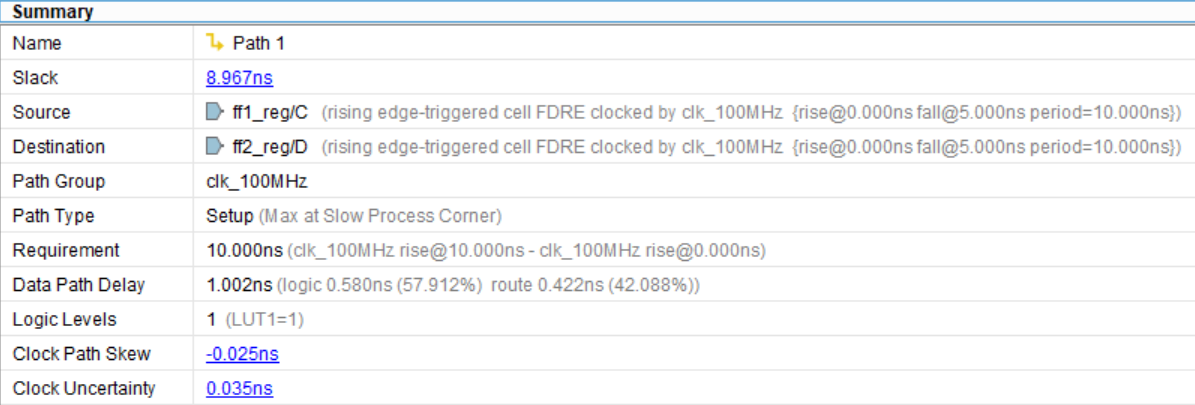


Рисунок 5. Общие сведения о пути.

В следующем разделе (рисунок 6) представлены задержки, оказывающие влияние на распространение запускающего фронта. В столбце *Incr* указаны значения отдельных задержек, а в столбце *Path* сумма текущей и всех предыдущих задержек. Можно увидеть, что запускающий фронт появляется в нулевой момент времени и распространяется через входной (IBUF) и тактовый (BUFG) буферы.

Общая задержка распространения запускающего фронта (*Tsca* из уравнения 1) составляет 4.708 нс. Символы (r) рядом со значениями задержек указывает, что анализируется фронт, а не спад тактового сигнала.

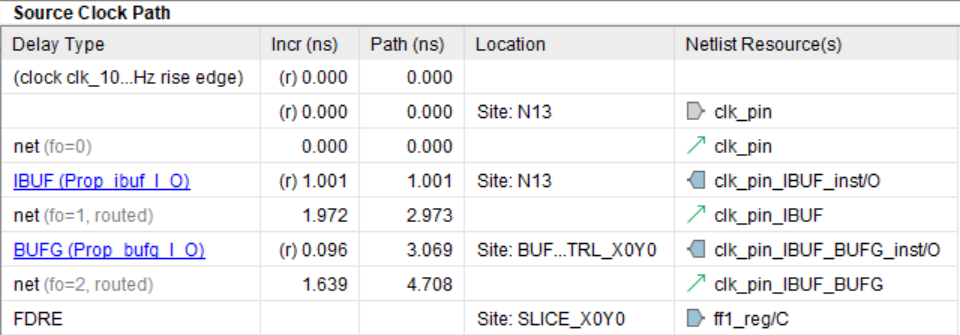


Рисунок 6. Задержка для запускающего фронта.

В разделе, представленном на рисунке 7, указаны задержки при распространении данных, которые состоят из задержки clock to output (*Tco*) для триггера FF1, задержки распространения через *nets* и LUT (*Tilo*). Просуммировав значения в столбце *Incr*, получим 1.002 нс, что совпадает со значением Data Path Delay на рисунке 5.

Обратите внимание, что значения в столбце *Path* не начинаются с нуля, так как учитывается задержка для запускающего фронта. Первое значение в столбце *Path* рассчитывается как сумма задержек *Tsca* (4.708 нс., рисунок 6) и *Tco* (0.456 нс.). Конечное значение в столбце *Path* (5.710 нс.) показывает время прибытия данных *Arrival Time* ко входу защелкивающего триггера (*Tda* из уравнения 3).

О том, что обозначают символы (r) и (f) рядом со значениями задержек, будет рассказано далее в статье в пункте 6.

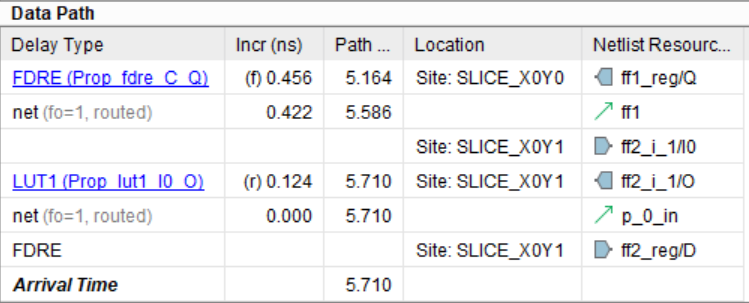


Рисунок 7. Задержки при распространении данных.

В последнем разделе (рисунок 8) приводится расчёт требуемого времени прибытия данных *Required Time*. Защелкивающий фронт появляется спустя период тактового сигнала, поэтому в первой строке отчета указано 10 нс.

Далее до строки с надписью FDRE указываются задержки при распространении защелкивающего фронта, который проходит через входной и тактовый буферы и попадает на С вход триггера FF2 в момент времени 14.347 нс. (*Tdca* из уравнения 4).

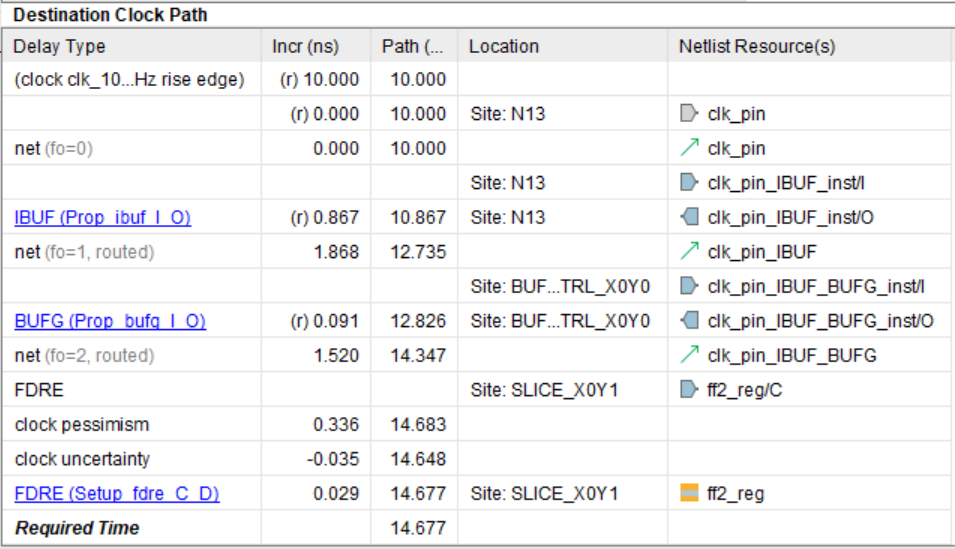


Рисунок 8. Требуемое время прибытия данных.

Далее после строки с надписью FDRE показаны еще три задержки, которые представляют из себя время установки *Tsu* для триггера FF2 (0.029 нс), а также пессимизм *Tcp* (***C****lock* ***P****essimism*, 0.336 нс.) и неопределенность *Tcu* (***C****lock* ***U****ncertainty*, -0.035 нс.) тактового сигнала. О последних двух задержках будет рассказано далее в статье в пунктах 7 и 8.

Требуемое время прибытия данных *Tdr* рассчитывается с помощью уравнения 5. Если учитывать дополнительные слагаемые *Tcp* и *Tcu*, то уравнение для *Tdr* будет иметь вид

*Slack* вычисляется по уравнению 6 и равен

,

Данный результат совпадает со значением, представленным в разделе Summary на рисунке 5.

1. **Анализ ограничения по Hold.**

Выбрав Hold в разделе Intra-Clock Path (рисунок 4), можно открыть расширенный отчет Path Report для анализа на минимальное время распространения. Отсчет также состоит из четырех разделов, которые представлены на рисунках 9-11. Данные в разделах Summary, Source Clock Path и Data Path имеют тот же самый смысл, что и при анализе по *Setup*.

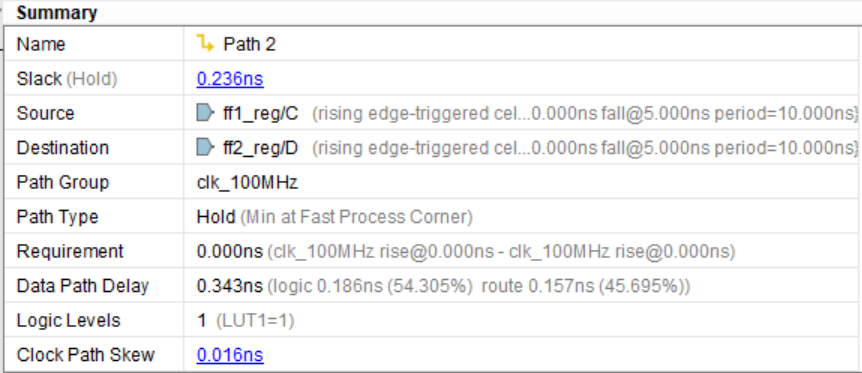


Рисунок 9. Общие сведения о пути

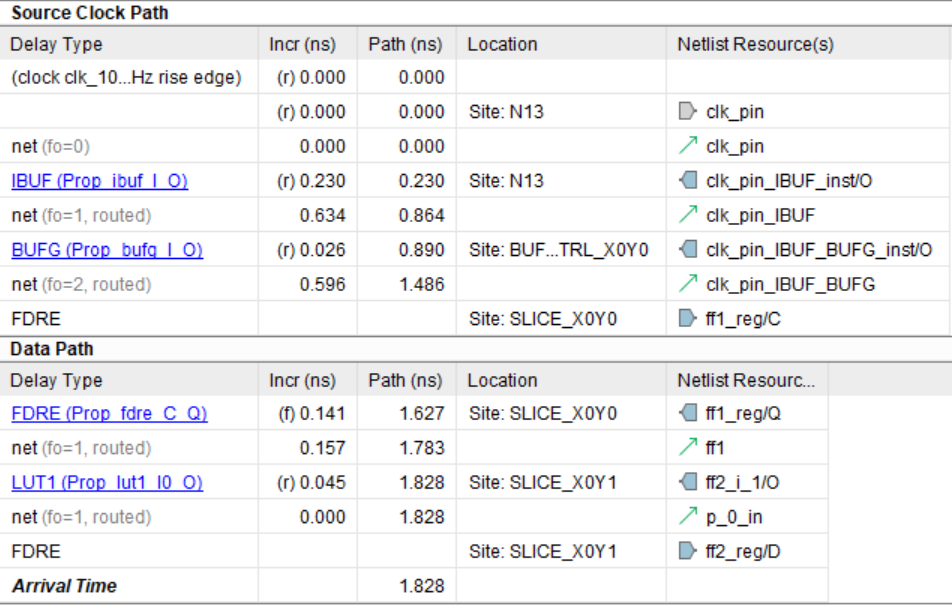


Рисунок 10. Расчет фактического времени прибытия данных

На рисунке 11 представлен расчёт требуемого времени прибытия данных *Tdr*. Обратите внимание, что начальное значение в столбцах *Incr* и *Path* равно 0 нс, так как при анализе по *Hold* защелкивающий фронт рассматривается относительно предыдущих данных и появляется в тот же момент времени, что и запускающий фронт для следующих данных.

Можно увидеть также, что в данном случае отсутствует слагаемое *Tcu* (***C****lock* ***U****ncertainty*). Причина будет раскрыта далее в статье в пункте 7.

*Slack* рассчитывается, использую уравнение 10, и равен

Важно заметить, что задержка через одни и те же элементы при анализе по *Setup* и *Hold* имеют разные значения. Например, задержка через LUT для *Setup* равна 0.124 нс., а для *Hold* – 0.045 нс. Значение, представленное в datasheet DS181 для speedgrade -1, равно 0.1 нс.

Как было сказано ранее, статический анализ производится для самого пессимистичного случая, поэтому при анализе по *Setup* рассматривается ситуация, когда данные будут распространятся максимально долго, что соответствует в уравнении 2. Для анализа по *Hold* самый худший случай – это максимально быстрое распространение данных, то есть в уравнении 9.

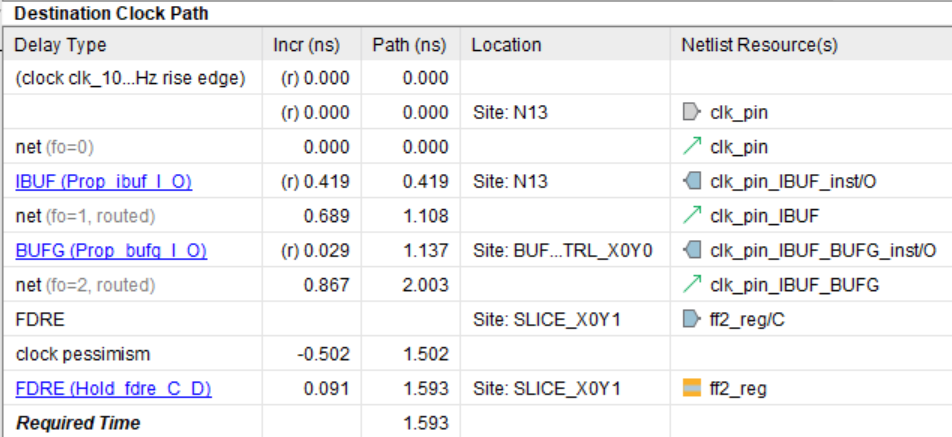


Рисунок 11. Расчет требуемого времени прибытия данных

В завершении данного пункта статьи рассмотрим, что же означают символы (r) и (f) рядом с задержками данных. Глядя на рисунок 3, можно подумать, что при временном анализе Vivado рассматривает всего один путь, однако это не так.

У каждого пути есть начало и конец. Точка, в которой заканчивается путь называется *Endpoint*. Несколько разных путей могут иметь один и тот же *Endpoint*. По умолчанию во временном отчете отображается всего один худший путь для каждого *Endpoint*. Можно настроить анализатор отображать больше путей, если при формировании отчета установить параметр *Maximum number of worst path per endpoint*. Задав значение этого параметра, например, равным 10, на рисунке 12 можно увидеть, что теперь в отчете отображается четыре пути.

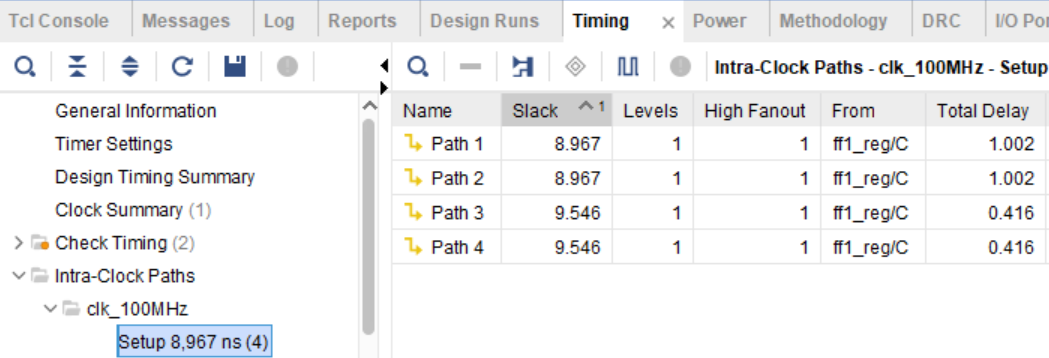


Рисунок 12. Отображение большего числа путей.

Задержки распространения через элемент цифровой схемы могут иметь разные значения, когда выход элемента изменяется с ‘0’ в ‘1’ или с ‘1’ в ‘0’. Символ (f) рядом с задержкой данных указывает, что в данном случае анализируется ситуация, при которой сигнал на выходе логического элемента изменяется с ‘1’ в ‘0’.

На рисунке 10 можно увидеть, что рассматривается случай, когда выход триггера FF1 изменяется с ‘1’ на ‘0’ (f), а выход LUT – c ‘0’ на ‘1’ (r). Всего есть четыре возможные комбинации (r) и (f) для LUT и FF1, поэтому на рисунке 12 переставлено четыре пути.

Очевидно, что, так как в рассматриваемом примере LUT реализует инвертор, то ситуация FF1 (f) и LUT (f), а также FF1 (r) и LUT (r) невозможны. Однако временной анализатор не рассматривает как именно работает схема, а просто рассчитывает все возможные задержки для самых пессимистичных случаев.

1. **Неопределенность тактового сигнала.**

Идеальный тактовый сигнал представляет из себя последовательность прямоугольных импульсов, повторяющихся через фиксированный период времени. У реального тактового сигнала фронты импульсов повторяются в среднем через период, однако каждый отдельный фронт может прийти чуть раньше или чуть позже.

Это явление называется дрожанием тактового сигнала или джиттером (*jitter*). В виде уравнения время между двумя фронтами тактового сигнала можно записать как

где *Tclk* – как и раньше период тактового сигнала, *Tj* – случайная величина с нулевым средним, обуславливающая неопределенность тактового сигнала.

Слагаемое *Tcu* (***C****lock* ***U****ncertainty*), которое встречалось при рассмотрении анализа по *Setup*, учитывает неопределенность тактового сигнала из-за джиттера. Нажав на значение *Clock Uncertainty* (рисунок 5), можно увидеть способ вычисления неопределенности (рисунке 13).

Уравнение для расчета неопределённости имеет вид [2]:

где

* *Ttsj* (***T****otal* ***S****ystem* ***J****itter*) – общий джиттер, присутствующий на тактовом входе каждого последовательного элемента схемы;
* *Ttij* (***T****otal* ***I****nput* ***J****itter*) – дополнительный джиттер для конкретного тактового сигнала;
* *Tdj* (***D****iscrete* ***J****itter*) – джиттер тактового сигнала, формируемого блоками PLL или MMCM;
* *Tpe* (***P****hase* ***E****rror*) – постоянная расфазировка тактового сигнала, формируемого блоками PLL или MMCM, относительного опорного сигнала.

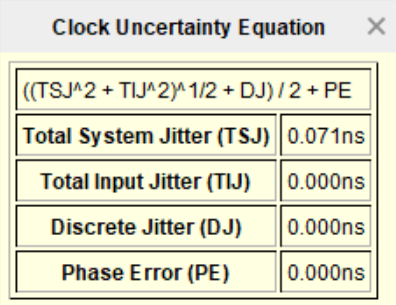


Рисунок 13. Выражение для расчета неопределенности.

Параметры *Tdj* и *Tpe* Vivado рассчитывает самостоятельно, и они отличны от нуля, только если данные запускаются или защёлкиваются тактовым сигналом, формируемым в PLL или MMCM.

По умолчанию значение *Ttsj* задается равным 0.071 нс. Это значение можно настраивать, указав следующую команду в xdc-файле [2]:

set\_system\_jitter Tsj

Значение *Tsj* (***S****ystem* ***J****itter*) задается в наносекундах, а величина общего джиттера *Ttsj* рассчитывается по формуле:

.

На рисунке 14, в качестве примера, представлен результат команды

set\_system\_jitter 0.15

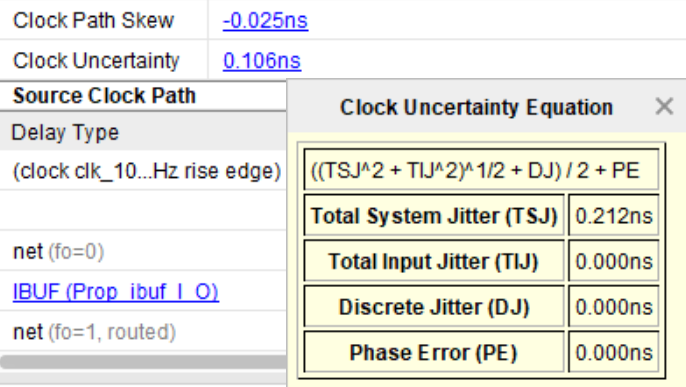


Рисунок 14. Настройка общего джиттера.

Как и ожидалось нc. и нс.

Если требуется, можно задать дополнительную неопределенность из-за джиттера для конкретного тактового сигнала с помощью команды [2]

set\_input\_jitter [get\_clocks clock\_name] Tij

Величина *Tij* (***I****nput* ***J****itter*) задается в наносекундах. Значение дополнительного джиттера для тактового сигнала равно , если запускающий и защёлкивающий фронты относятся к одному тактовому сигналу, и – если к разным.

На рисунке 15 представлен результат расчета неопределенности, в случае, когда содержимое xdc-файла имеет вид:

create\_clock -period 10.000 -name clk\_100MHz [get\_ports clk\_pin]

set\_system\_jitter 0.15

set\_input\_jitter [get\_clocks clk\_100MHz] 0.100

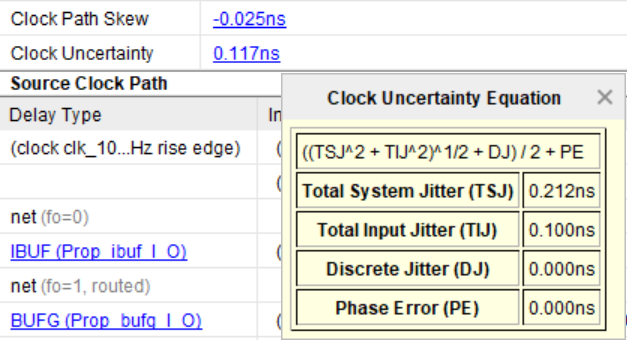


Рисунок 15. Настройка неопределенности тактового сигнала.

Подставив заданные значения в уравнение 12, получим:

Обобщая все выше сказанное, получаем, что из-за наличия джиттера интервал между двумя последовательными фронтами тактового сигнала отличается от *Tclk* на некоторую случайную величину, которую можно задавать с помощью команд set\_system\_jitter и set\_input\_jitter.

Также теперь можно объяснить, почему на рисунке 11 отсутствует слагаемое *Tcu*. При анализе по *Hold* рассматриваются запускающий фронт для следующих данных и защелкивающий фронт для предыдущих данных. Фактически это один и тот же фронт тактового сигнала, поэтому в данном случае отсутствует какая-либо неопределенность. Если запускающий и защелкивающий фронты относятся к разным тактовым сигналам, то слагаемое *Tcu* будет присутствовать в анализе по *Hold*.

1. **Устранение пессимизма для задержки тактового сигнала.**

Вернемся к анализу по *Hold*, чтобы выясним смысл последнего не рассмотренного слагаемого *Tcp* (***C****lock* ***P****essimism*) [3]. На рисунке 10 можно увидеть, что задержка распространения такового сигнала через входной буфер равна 0.23 нс. В то же время из рисунка 11 следует, что задержка через тот же буфер равна 0.419 нс. Но физически такого быть не может, так как при анализе по *Hold* запускающий и защелкивающий фронты совпадают, и задержки через буфер должны быть одинаковыми.

Данное противоречие связано с тем, что временной анализатор всегда рассматривает самый худший случай, и для анализа по *Hold* этому соответствуют минимальные задержки для запускающего и максимальные для защелкивающего фронтов.

На первом этапе анализатор рассчитывает задержки без учета топологии распространения тактового сигнала. Далее анализатор ищет общий участок пути для запускающего и защелкивающего фронтов. Задержки распространения по этому участку для обоих фронтов должны совпадать, поэтому анализатор вносит компенсацию в виде слагаемого *Tcp*. Например, задержки по общему пути от ножки FPGA до выхода тактового буфера (BUFG) равны 0.890 нс для запускающего фронта, и 1.137 нс. для защелкивающего (рисунки 10 и 11). Рассчитывая значение *Tcp*, получаем

.

В отчете это значение равно -0.502 нс. Это связано с тем, что Vivado также учитывает разность задержек распространения по общему последнему участку *net*, которым мы пренебрегли.

Рассмотрим временной анализ после синтеза, когда не учитываются задержки через *net*. Отдельные части отчета анализа по *Hold* представлены на рисунке 16.

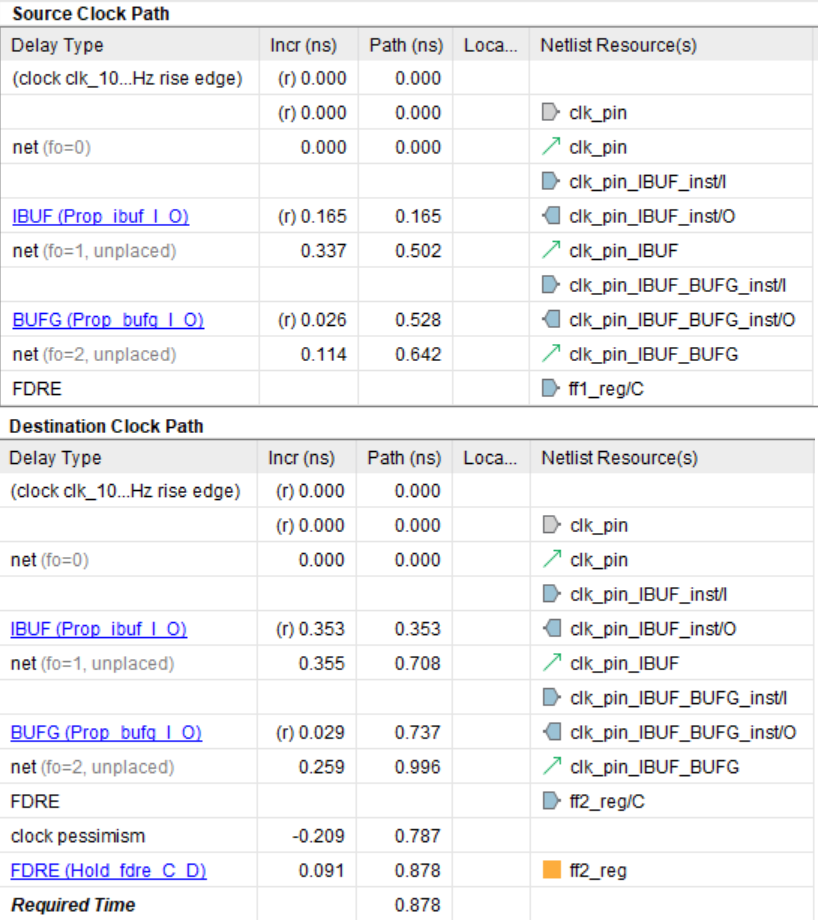


Рисунок 16. Анализ по Hold после синтеза.

Задержки по общему пути от ножки FPGA до выхода тактового буфера равны 0.528 нс. для запускающего и 0.737 нс. для защелкивающего фронтов соответственно. Тогда

Данный результат совпадает со значением *Clock Pessimism*, представленным на рисунке 16.

Также теперь можно понять, как рассчитывается расфазировка тактового сигнала, присутствующая в разделе Summary Path Report. Для этого необходимо воспользоваться уравнением 8 с учетом компенсации пессимизма распространения тактового сигнала. Глядя на рисунки 10 и 11, получаем

Полученное значение *Tskew* совпадает с *Clock Path Skew* в разделе Summary на рисунке 9.

**Заключение.**

В данной статье был рассмотрен временной анализ для простейшего случая – передачи данных между двумя триггерами внутри FPGA. Показан вывод уравнений статического временного анализа. Подробно рассмотрены разделы отчета Path Report. Рассказано о таких понятиях, как джиттер и пессимизм задержки тактового сигнала. В следующей части предполагается использовать тот же подход для рассмотрения System Synchronous Input Delay.

**Ссылки.**

1. <https://www.xilinx.com/support/documentation/data_sheets/ds181_Artix_7_Data_Sheet.pdf>
2. <https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_2/ug903-vivado-using-constraints.pdf>
3. <https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_2/ug906-vivado-design-analysis.pdf>