**Основы статического временного анализа.**

**Часть 4: Source Synchronous** **Output Delay Constraint.**

**Введение.**

В статье представлен временной анализ передачи сигналов из FPGA во внешнее устройство. Рассмотрены теоретические основы анализа для двух возможных случаев приема данных: по текущему и следующему фронту тактового сигнала. Разобраны практические примеры создания временных ограничений. Показан способ решения проблемы с временными ограничениями по *Setup* за счет инвертирования тактового сигнала и использования ODDR триггера.

1. **Передача данных для случая Source Synchronous.**

Данная статья частично опирается на материал, рассмотренный в предыдущих работах [1-2]. Предполагается, что читатель уже знаком с такими понятиями, как ограничение на максимальное (*Setup*) и минимальное (*Hold*) время распространения сигнала, запас (*Slack*) и т.д.

Ранее в [2] был представлен временной анализ передачи данных из FPGA во внешнее устройство в случае, когда тактовый сигнал формируется генератором, расположенным на плате (см. рисунке 1). Такой способ передачи называется System Synchronous.

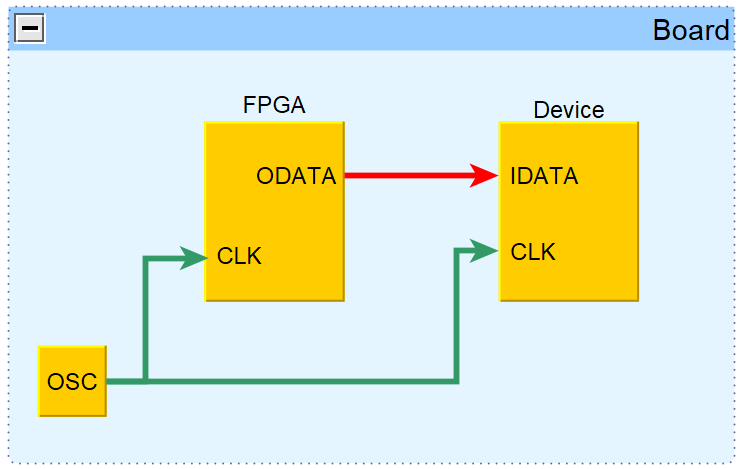


Рисунок 1. Соединение устройств на плате для случая System Synchronous.

В текущей статье будет рассмотрен другой способ, называемый Source Synchronous, при котором источник помимо данных также формирует тактовый сигнал (см. рисунке 2). В дальнейшем для краткости устройство, в которое из FPGA передаются данные и тактовый сигнал, будем иногда называть Device*.*

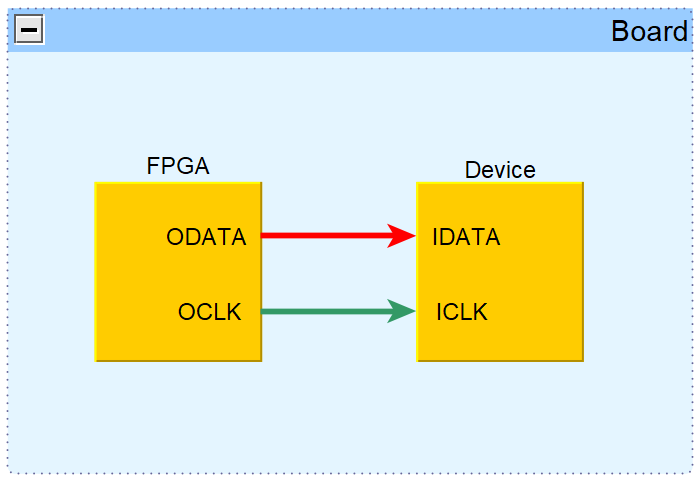


Рисунок 2. Соединение устройств на плате для случая Source Synchronous.

При передаче данных между FPGA и Device запускающий триггер располагается в FPGA, а защелкивающий – во внешнем устройстве. На рисунке 3 показан анализируемый путь, на который нанесены задержки сигналов. В случае Source Synchronous вместе с данными также передает тактовый сигнал, который тем или иным способом создается внутри FPGA.

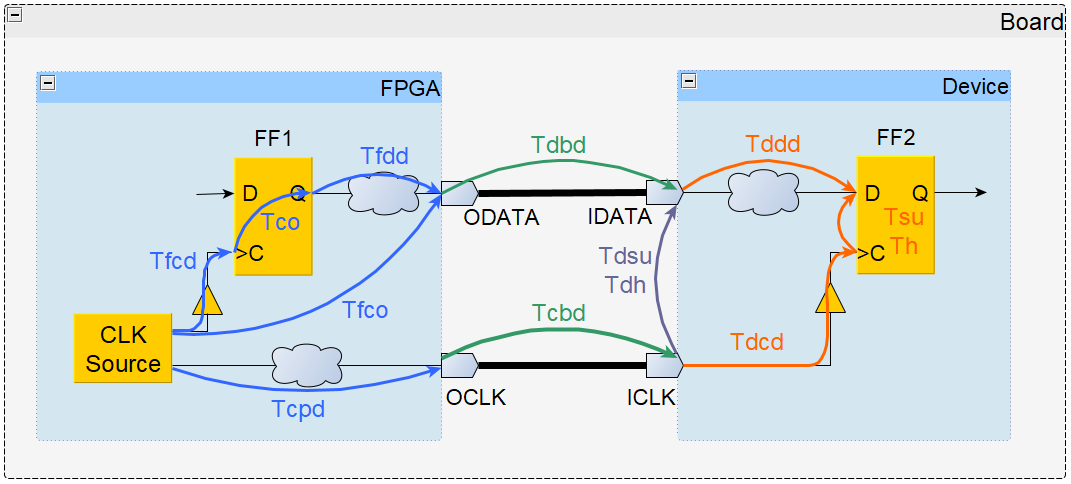


Рисунок 3. Путь с задержками для выходных данных и тактового сигнала.

Ниже даны определения задержек, представленных на рисунке 3.

* *Tdbd* (***D****ata* ***B****oard* ***D****elay*) – задержка распространения данных по дорожкам платы от FPGA до Device;
* *Tcbd* (***C****lock* ***B****oard* ***D****elay*) – задержка распространения тактового сигнала по дорожкам платы от FPGA до Device;
* *Tdcd* (***D****evice* ***C****lock* ***D****elay*) – задержка тактового сигнала от ножки ICLK Device до тактового входа защелкивающего триггера;
* *Tddd* (***D****evice* ***D****ata* ***D****elay*) – задержка распространения данных от ножки IDATA Device до защелкивающего триггера;
* *Tsu* (***S****et****U****p time*) – время установки защелкивающего триггера;
* *Th* (***H****old time*) – время удержания защелкивающего триггера;
* *Tcpd* (***C****lock to* ***P****in* ***D****elay*) – задержка тактового сигнала от места его формирования до ножки OCLK FPGA;
* *Tfcd* (***F****PGA* ***C****lock* ***D****elay*) – задержка тактового сигнала от места его формирования до тактового входа запускающего триггера;
* *Tco* (***C****lock to* ***O****utput*) – интервал времени между приходом фронта на тактовый вход триггера и появлением данных на его выходе Q;
* *Tfdd* (***F****PGA* ***D****ata* ***D****elay*) – задержка распространения данных от выхода Q запускающего триггера до ножки ODATA FPGA.

Период тактового сигнала будем обозначать *Tclk*. Оранжевым и зеленым цветом на рисунке 3 представлены задержки для участков пути, которые располагаются вне FPGA. Эти задержки необходимо указать временному анализатору Vivado.

1. **Максимальное время распространения.**

Для начала рассмотрим ограничения на максимальное время распространения (*Setup*). Напомним, что 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.

Глядя на рисунок 3, найдем фактическое время прибытия данных, как сумму задержек распространения запускающего фронта и данных:

где *Tfco\_max* (***F****PGA* ***C****lock to* ***O****utput time*) – временной интервал между формированием запускающего фронта и появлением данных на выходе FPGA.

В дальнейшем будут рассмотрены два варианта передачи данных между FPGA и Device. В первом случае данные будет запускаться из FPGA по одном фронту, а защелкиваться в Device – по следующему. Во втором случае данные будут запускаться и защелкиваться одним и тем же фронтом тактового сигнала. Чтобы описать оба этих варианта, введем дополнительную переменную *Tccd\_setup* (***C****apture* ***C****lock* ***D****elay*), которая задает задержку между появлением запускающего и защелкивающего фронта. До того, как к защелкивающему триггеру придет фронт, данные уже должны быть стабильны на его входе в течении времени *Tsu*. Поэтому требуемое время прибытия данных равно разнице между суммарной задержкой защелкивающего фронта и временем удержания триггера:

где *Tclk\_delay\_min* – задержка распространения защелкивающего фронта от его источника в FPGA до триггера в Device (см. рисунок 3).

Уравнение для *Slack* при анализе по *Setup* имеет вид [1]:

Подставим в него выражения (1) и (2) и получим:

где *Tdsu (****D****evice* ***S****e****T****up)* – время установки для данных на входе IDATA Device относительно тактового входа ICLK (см. рисунок 3).

Объединим все слагаемые, описывающие задержки вне FPGA, в одну переменную и получим выражения для *Slack* при анализе по *Setup*:

1. **Минимальное время распространения.**

Теперь изучим, как выполняется проверка ограничения на минимальное время распространения (*Hold*). При анализе по *Hold* рассматриваются соотношения между текущим запускающим и предыдущим защелкивающим фронтами. Также считается, что задержки для запускающего фронта и данных имеют минимальное значение, а для защелкивающего фронта – максимальное.

По аналогии с рассмотренным ранее анализом по *Setup* запишем уравнение для фактического времени прибытия данных:

Далее найдем требуемое время прибытия данных. Для этого нужно рассчитать максимальную задержку распространения защелкивающего фронта и добавить к ней время удержания триггера [1]:

где *Tccd\_hold* – временной интервал между текущим запускающим и предыдущим защелкивающим фронтами тактового сигнала.

Уравнение для *Slack* при анализе по *Hold* имеет вид [1]:

С учетом полученных ранее результатов имеем:

где *Tdh (****D****evice* ***H****old)* – время удержания для данных на входе IDATA Device относительно тактового входа ICLK (см. рисунок 3).

Объединим все задержки вне FPGA в одну переменную и получим окончательное выражение для *Slack* при анализе по *Hold*:

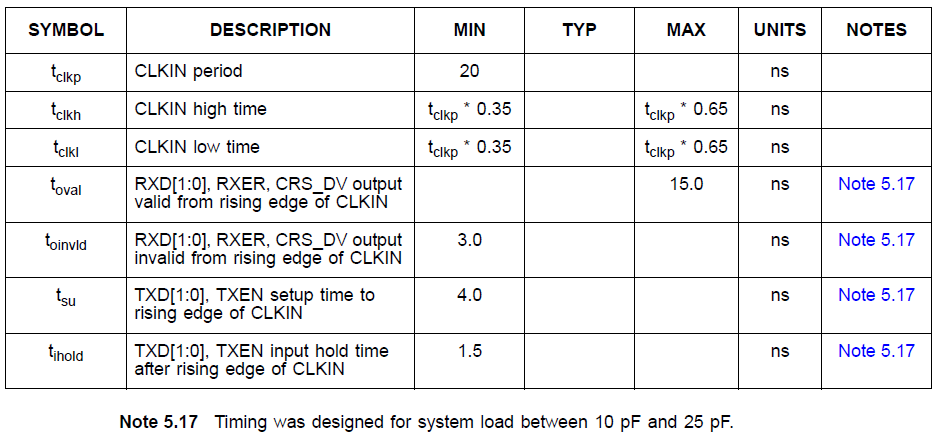
1. **Пример с защелкиванием данных по следующему фронту.**

Для начала разберем ситуацию, когда данные запускаются из FPGA одним фронтом тактового сигнала, а защелкиваются – следующим. В этом случае значение *Tccd\_setup* будет равно периоду тактового сигнала *Tclk*, а *Tccd\_hold* – нулю, так как текущий запускающий и предыдущий защелкивающий фронты появляются в одни и тот же момент времени.

С учетом этого для уравнений (3) и (4) получим:

Все задержки вне FPGA объединены в слагаемых *Tfpga\_ext\_setup* и *Tfpga\_ext\_hold*. Их необходимо указать Vivado в виде следующих временных ограничений:

В качестве практического примера рассмотрим передачу данных по RMII из FPGA в микросхему Ethernet PHY LAN8740A [3]. На рисунке 4 приведены таблица со значениями задержек и временная диаграмма сигналов из datasheet на LAN8740A. Для краткости ограничения будут продемонстрированы для одного выходного сигнала FPGA, который подключен к ножке TXD[0] микросхемы LAN8740A.



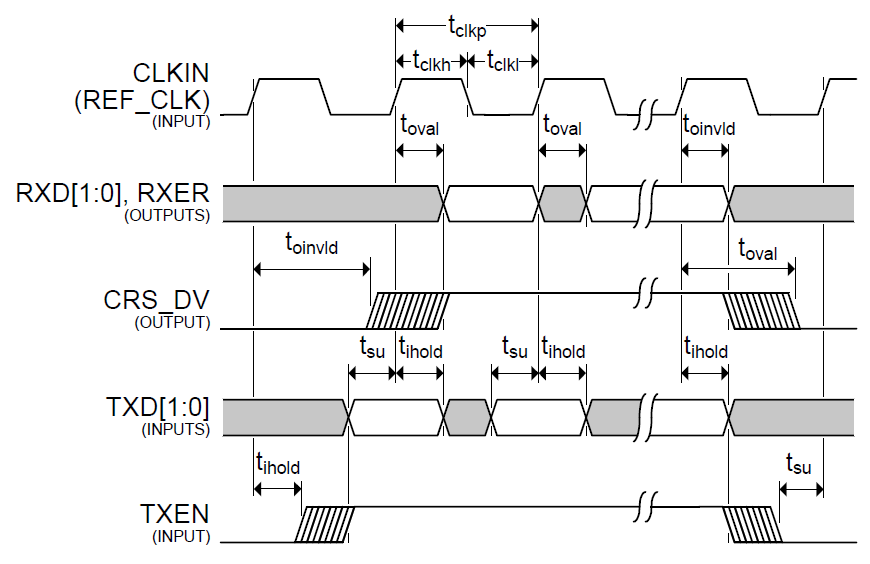


Рисунок 4. Задержки и временные диаграммы для LAN8740A.

В FPGA загрузим простой проект, состоящий из единственного триггера (см. рисунок 5). Этого вполне достаточно для демонстрации того, как в Vivado проводится временной анализ для выходных сигналов. Ниже показано описание проекта на System Verilog:

module top (

    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

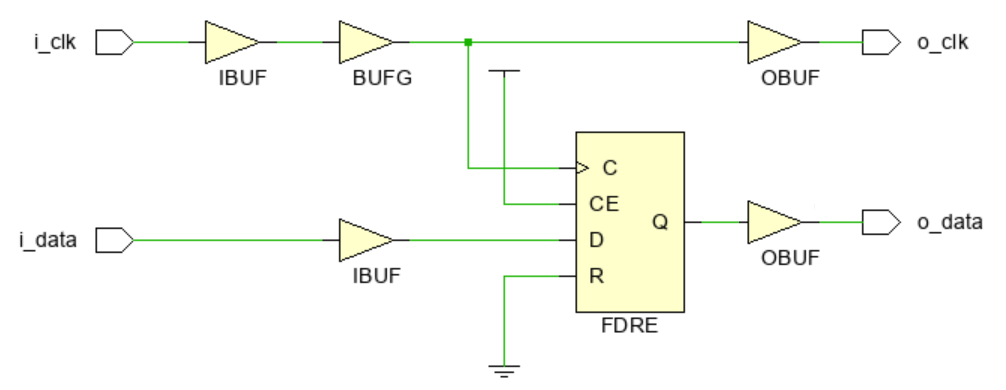


Рисунок 5. Схема FPGA проекта.

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. Эти значения заносятся в файл с временными ограничениями (xdc-файл):

# 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

Выходной тактовый сигнал на ножке o\_clk является копией сигнала, поступающего на ножку i\_clk, поэтому их периоды совпадают и равны *Tclk*. Из рисунка 4 находим период тактовых сигналов, а также время установки *Tdsu* и удержания *Tdh* для LAN8740A. Эти значения также внесем в xdc-файл и сразу создадим ограничение на период тактового сигнала i\_clk:

# период тактового сигнала

set Tclk 20

# время установки и удержания для микросхемы LAN8740A

set Tdsu 4.0

set Tdh 1.5

# ограничение на период входного тактового сигнала

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

Внутри FPGA новый тактовый сигнал всегда создается из уже существующего (опорного), например, с помощью PLL или деления частоты на основе счетчика. Несмотря на то, что мы фактически используем входной сигнал i\_clk для тактирования микросхемы LAN8740A, Vivado считает, что из i\_clk создается новый тактовый сигнал o\_clk. Его параметры нужно указать в xdc-файле, чтобы он мог учувствовать во временном анализе. Для создания временных ограничений на тактовые сигналы, формируемые внутри FPGA, используется команда *create\_generated\_clock*:

# ограничение на период выходного тактового сигнала

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

-divide\_by 1 [get\_ports o\_clk]

С помощью ключа -*source* задается источник опорного сигнала. а ключ   
-*divide\_by* указывает, на сколько делится его частота. В нашем случае опорный сигнал появляется на ножке FPGA i\_clk, поэтому для флага -*source* задается конструкция *[get\_ports i\_clk]*. Так как периоды опорного (i\_clk) и создаваемого (o\_clk) сигналов совпадают, значение флага *-divide\_by* равно единице. С помощью флага *-name* указывается имя формируемого тактового сигнала, а конструкция *[get\_ports o\_clk]* определяет место, которое считается его источником.

Теперь осталось создать ограничения на максимальное и минимальное время распространения выходного сигнала o\_data. Для этого воспользуемся формулами (6) и запишем в xdc-файл следующие команды:

# временные ограничения для выходного сигнала o\_data

set\_output\_delay -clock o\_clk -max [expr $Tdbd\_max - $Tcbd\_min + $Tdsu]

[get\_ports o\_data]

set\_output\_delay -clock o\_clk -min [expr $Tdbd\_min - $Tcbd\_max - $Tdh]

[get\_ports o\_data]

Обратите внимание, что флагу -*clock* присвоено имя тактового сигнала o\_clk, созданного ранее с помощью команды *create\_generated\_clock*. Более подробно о назначении других флагов можно прочитать в [2].

Полное содержимое xdc-файла представлено ниже:

# 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]

# временные ограничения для выходного сигнала o\_data

set\_output\_delay -clock o\_clk -max [expr $Tdbd\_max - $Tcbd\_min + $Tdsu]

[get\_ports o\_data]

set\_output\_delay -clock o\_clk -min [expr $Tdbd\_min - $Tcbd\_max - $Tdh]

[get\_ports o\_data]

Рассмотрим, как введенные ограничения будут отражены во временных отчетах, полученных после размещения и трассировки проекта. Для краткости в дальнейшем будет рассмотрено только ограничение по *Setup*.

На рисунке 6 представлен раздел *Summary* временного отчета. Из строки с именем *Source* можно увидеть, что путь внутри FPGA начинается на входе триггера o\_data\_reg, при этом запускающим является тактовый сигнал i\_clk. Строка *Destination* указывает, что путь заканчивается на выходной ножке o\_data, причем данные защелкиваются по сигналу o\_clk. Положительное значение *Slack*, равное 10.763 нс, означает, что временные ограничения выполнены.

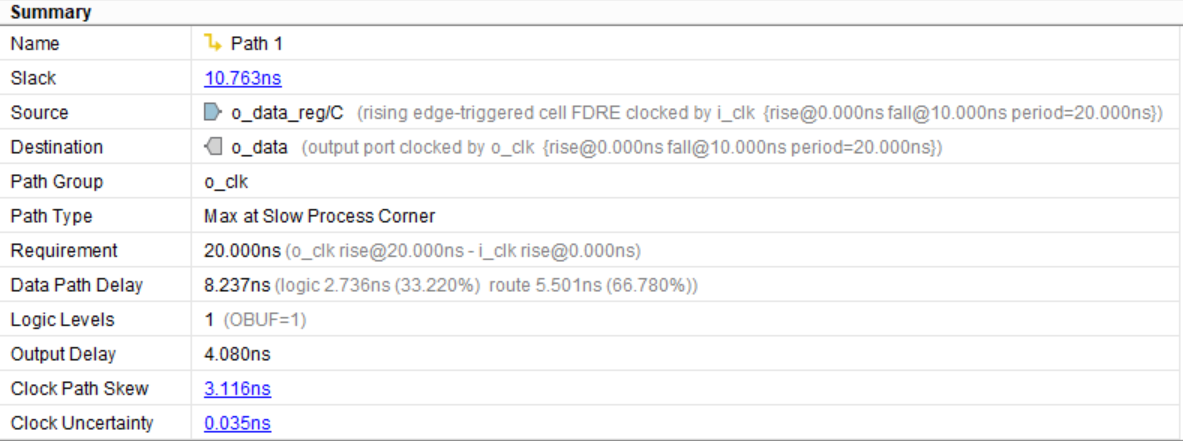


Рисунок 6. Раздел Summary для анализа по Setup.

На рисунке 7 показаны задержки распространения для запускающего фронта и данных. Можно увидеть, что фронт сигнала i\_clk появляется в нулевой момент времени и проходит через входной и тактовый буферы. Через 4.392 нс он достигает тактовой ножки триггера и запускает передачу данных. В свою очередь данные распространяются через выходной буфер и попадают на выход o\_data в момент времени 12.628 нс.

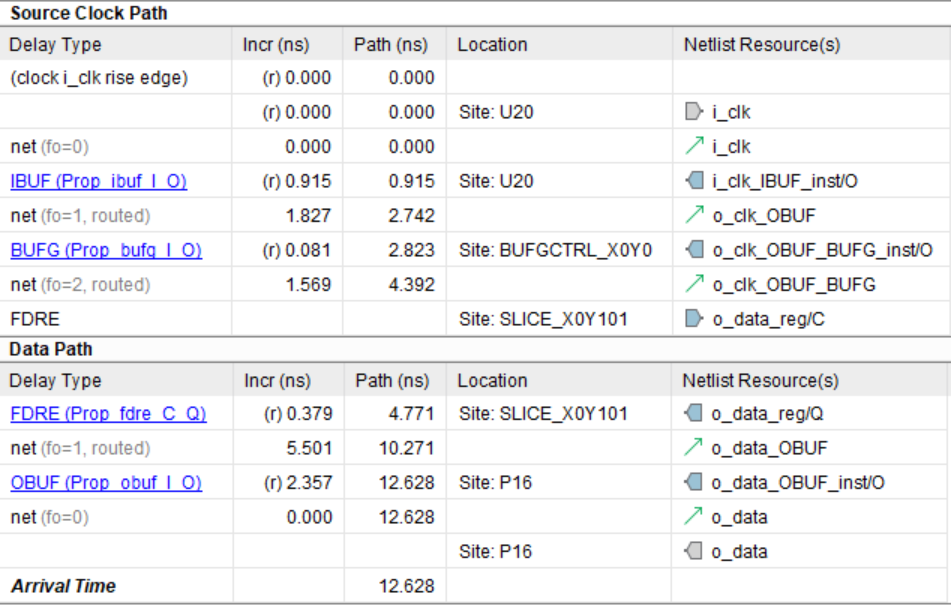


Рисунок 7. Задержки запускающего фронта и данных.

Как видно из рисунка 8, защелкивающий фронт o\_clk формируется из фронта опорного сигнала i\_clk, который появляется в момент времени 20 нс. Этот фронт распространяется через входной, тактовый и выходной буферы и появляется на ножке FPGA, когда время равно 27.214 нс.

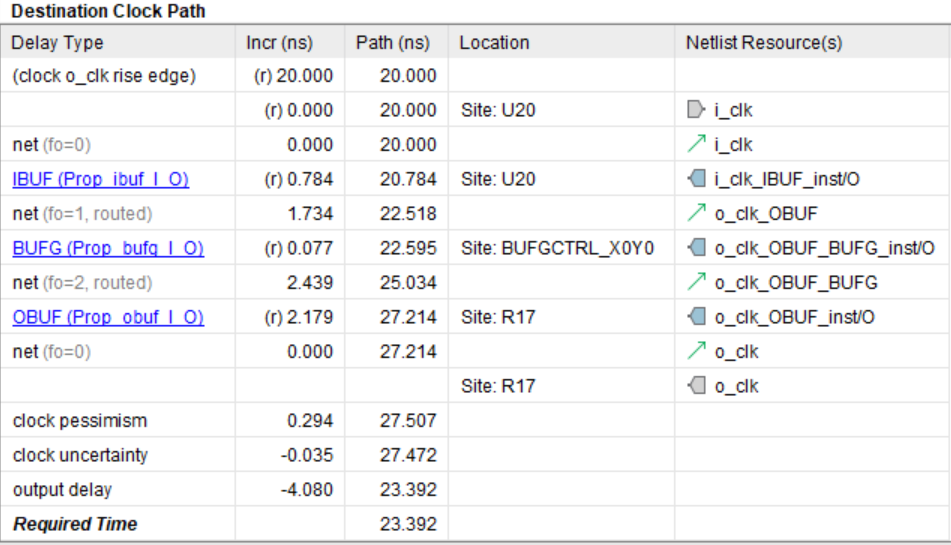


Рисунок 8. Задержки защелкивающего фронта.

Задержки распространения данных и тактового сигнала по дорожкам платы, а также время установки для LAN8740A учитываются с помощью слагаемого *output delay*, значение которого равно 4.0+0.15–0.07=4.08 нс. Добавляя неопределенность и пессимизм для тактового сигнала, получаем, что данные должны появиться на ножке o\_data FPGA не позже, чем через   
23.392 нс после запускающего фронта i\_clk. Вычитая из этого значения фактическое время прибытия данных на выход FPGA, получаем   
23.392–12.628=10.764 нс, что соответствует величине *Slack* из рисунка 6.

В предыдущем примере все задержки вне FPGA были объединены в одно слагаемое, что может усложнить создание ограничений. Часть задержек вносятся с положительными знаками, а часть – с отрицательными. За этим нужно внимательно следить, чтобы не перепутать и не совершить ошибку. Можно немного упростить себе жизнь, если задержки распространения тактового сигнала указывать отдельно с помощью команды *clock\_latency*. Для нашего примера в xdc-файл следует внести следующие команды:

# задержки распространения тактового сигнала от FPGA до LAN8740A

set\_clock\_latency -min $Tcbd\_min [get\_clocks o\_clk]

set\_clock\_latency -max $Tcbd\_max [get\_clocks o\_clk]

С помощью флагов *-max* и *-min* задаются минимальная (*Tcbd\_min*) и максимальная (*Tcbd\_max*) задержки распространения тактового сигнала от выхода FPGA до микросхемы LAN8740A. Так как эти задержки указываются отдельно, их нужно убрать из уравнений (6), которые теперь примут вид:

Полное содержимое обновленного xdc-файла представлено ниже:

# 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]

# задержки распространения тактового сигнала от FPGA до LAN8740A

set\_clock\_latency -min $Tcbd\_min [get\_clocks o\_clk]

set\_clock\_latency -max $Tcbd\_max [get\_clocks o\_clk]

# временные ограничения для выходного сигнала o\_data

set\_output\_delay -clock o\_clk -max [expr $Tdbd\_max + $Tdsu] [get\_ports o\_data]

set\_output\_delay -clock o\_clk -min [expr $Tdbd\_min - $Tdh]  [get\_ports o\_data]

Задержки для защелкивающего фронта представлены на рисунке 9. Можно увидеть, что в отчете появилась дополнительная строка *ideal clock network latency*, соответствующая времени распространения тактового сигнала от FPGA до LAN8740A. Все остальные задержки останутся теми же, что и на рисунках 6-8.

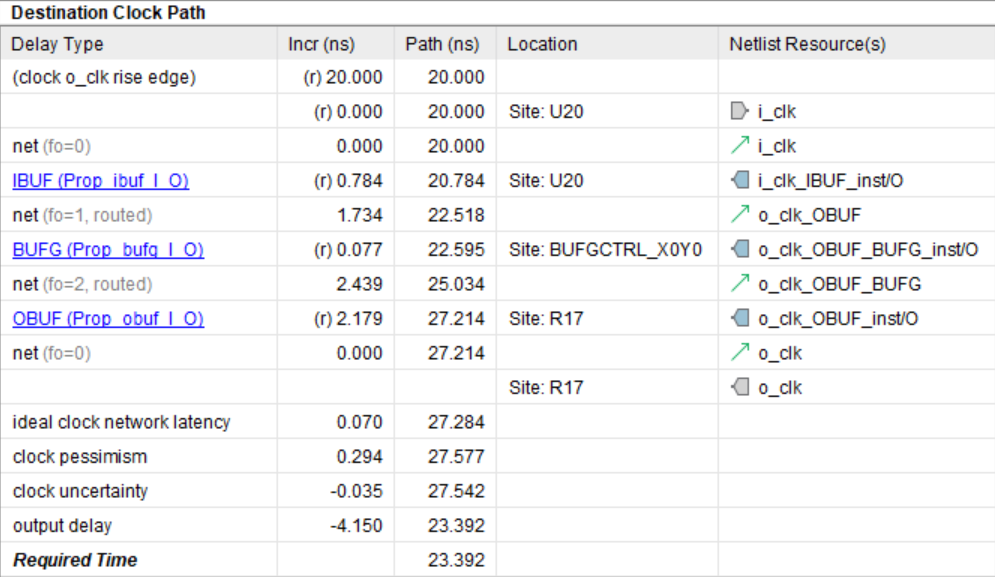


Рисунок 9. Задержки защелкивающего фронта.

Заметим также, что соотношения (7) можно обнаружить в Vivado Language Templates, если открыть вкладку XDC:

#  Rising Edge Source Synchronous Outputs

#

#  Setup and hold requirements for the destination device and board

# trace delays are known.

#

# forwarded         \_\_\_\_                      \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# clock                 |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|                   |\_\_\_\_\_\_\_\_\_\_\_\_

#                                            |

#                                     tsu    |    thd

#                                <---------->|<--------->

#                                \_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_

# data @ destination    XXXXXXXXX\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_XXXXX

set fwclk        <clock-name>;     # forwarded clock name

set tsu          0.000;            # destination device setup time requirement

set thd          0.000;            # destination device hold time requirement

set trce\_dly\_max 0.000;            # maximum board trace delay

set trce\_dly\_min 0.000;            # minimum board trace delay

set output\_ports <output\_ports>;   # list of output ports

# Output Delay Constraints

set\_output\_delay -clock $fwclk -max [expr $trce\_dly\_max + $tsu]

[get\_ports $output\_ports];

set\_output\_delay -clock $fwclk -min [expr $trce\_dly\_min - $thd]

[get\_ports $output\_ports];

Выходной тактовый сигнал FPGA, который в наших обозначениях имеет имя o\_clk, здесь называется fwclk (forwarded clock). Максимальные (*Tdbd\_max*) и минимальные (*Tdbd\_min*) задержки распространения данных по дорожкам платы указаны, как trce\_dly\_max и trce\_dly\_min. Задержки tsu и thd соответствуют времени установки и удержания для микросхемы LAN8740A.

1. **Пример с защелкиванием данных по текущему фронту.**

Рассмотрим второй вариант, при котором данные запускаются и защелкиваются одним и тем же тактовым фронтом. В этом случае защёлкивающий и запускающий фронты появляются одновременно, поэтому значение *Tccd\_setup* равно нулю. В свою очередь предыдущий защелкивающий фронт появляется на одни период раньше текущего запускающего фронта, а значит задержка между ними равна *Tccd\_hold* = *–Tclk*. С учетом всего вышесказанного уравнения (3) и (4) примут вид:

Важно отметить, что во время проведения временного анализа Vivado считает, что данные запускаются одним тактовым фронтом, а   
защелкиваются – следующим. По этой причине при расчете значения *Slack* Vivado будет использовать формулы (5) из предыдущего раздела.

Уравнение (8) для анализа по *Hold* отличается от (5) только наличием дополнительного слагаемого *Tclk*. В уравнении для *Setup* это слагаемое наоборот отсутствует. Если в него добавить и вычесть *Tclk*, то получим:

Введем новые переменные и представим уравнения (8) в следующем виде:

Полученные результаты с точностью до обозначений совпадают с уравнениями (5). Теперь с учетом предыдущего раздела можем записать временные ограничения для выходных данных:

В качестве примера опять рассмотрим передачу данных из FPGA в микросхему LAN8740A по RMII. Будем использовать те же самые значения задержек для данных и тактового сигнала. xdc-файл почти не изменится по сравнению с предыдущим случаем за исключением команды *set\_output\_delay*. Его содержимое представлено ниже:

# 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]

# задержки распространения тактового сигнала от FPGA до LAN8740A

set\_clock\_latency -min $Tcbd\_min [get\_clocks o\_clk]

set\_clock\_latency -max $Tcbd\_max [get\_clocks o\_clk]

# временные ограничения для выходного сигнала o\_data

set\_output\_delay -clock o\_clk -max [expr $Tclk + $Tdbd\_max + $Tdsu]

[get\_ports o\_data]

set\_output\_delay -clock o\_clk -min [expr $Tclk + $Tdbd\_min - $Tdh]

[get\_ports o\_data]

Проведем размещение и трассировку проекта и рассмотрим результаты для анализа по *Setup*. На рисунке 10 представлен раздел *Summary.* Отрицательное значение *Slack* указывает на нарушение временных ограничений. Чтобы выяснить источник проблемы, изучим задержки распространения данных и тактового сигнала, показанные на рисунках 11 и 12.

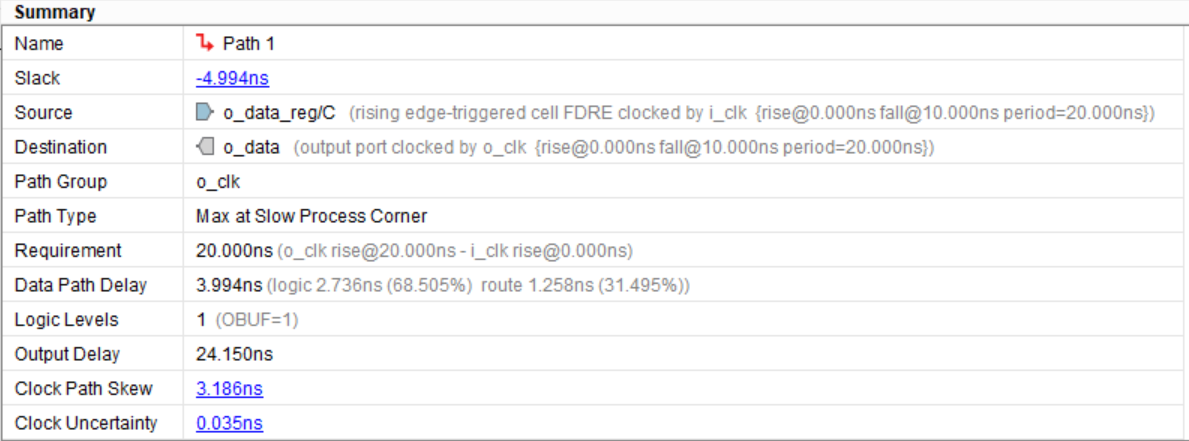


Рисунок 10. Раздел Summary для анализа по Setup.

Из рисунка 11 видно, что фронт сигнала i\_clk появляется в нулевой момент времени, распространяется до триггера o\_data\_reg и запускает передачу данных, которые спустя 8.363 нс достигнут выходной ножки FPGA.

Теперь взглянем на рисунок 12. Фронт сигнала i\_clk, из которого в дальнейшем будет сформирован защелкивающий фронт на выходе o\_clk, появляется через 20 нс, то есть спустя один период. Как уже упоминалось, это связано с тем, что Vivado при проведении временного анализа считает, что данные запускаются одним фронтом, а защелкиваются – следующим.

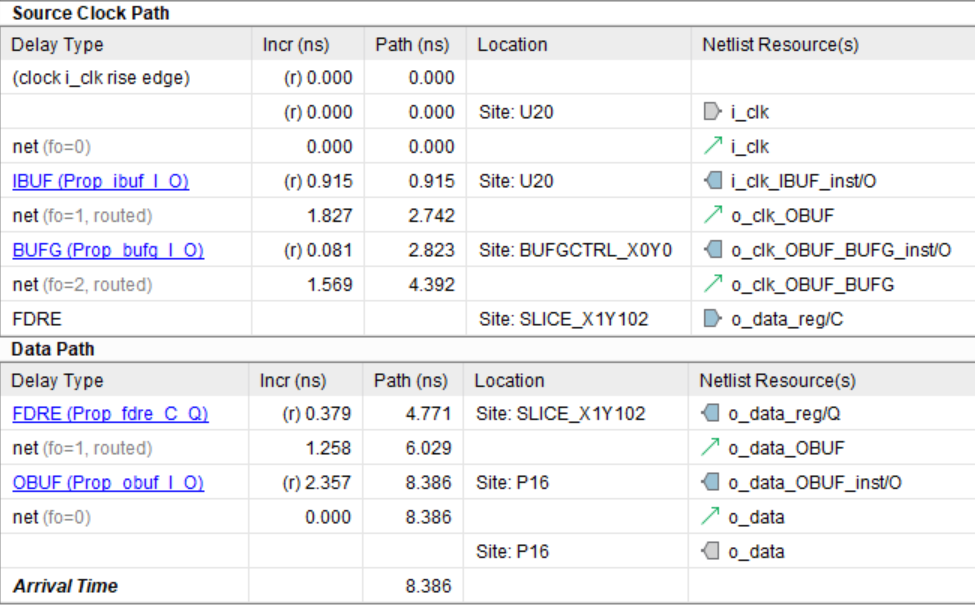


Рисунок 11. Задержки запускающего фронта и данных.

Если сравнить значение *output delay*, равное -24.15 нс, со значением на рисунке 9, то увидим, что они различаются как раз на 20 нс. За счет этой разницы компенсируется задержка появления фронта i\_clk в первой строке рисунка 12. С учетом неопределённости и пессимизма тактового сигнала получаем, что данные должны достигнуть ножки o\_data FPGA в момент времени 3.392 нс. Фактическое же время прибытия данных равно 8.386 нс. Это на 4.994 нс позже, чем требуется, а значит данные распространяются слишком медленно.

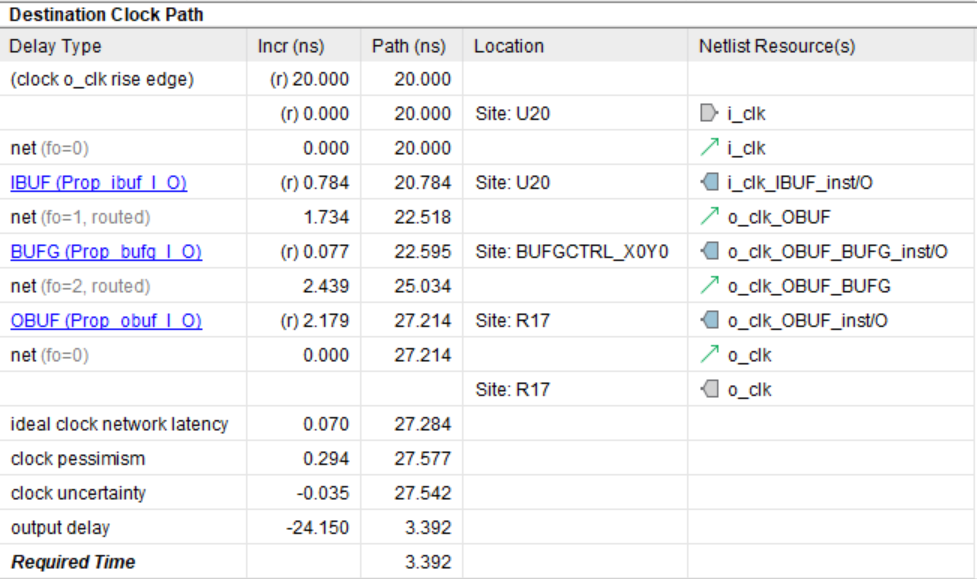


Рисунок 12. Задержки защелкивающего фронта.

Решение этой проблемы будет представлено в следующем разделе, а пока давайте рассмотрим еще один способ создания временных ограничений на выходные сигналы FPGA. Иногда бывает удобно вводить ограничения в виде задержек между данными и тактовым сигналом, которые измерены непосредственно на ножках FPGA. Такой подход называется FPGA Centric. При анализе по *Setup* считаем, что тактовый сигнал распространяются максимально быстро, а данные – максимально медленно, и поэтому тактовый фронт появляется раньше данных. Введем переменную *Tare\_skew* (***A****fter* ***R****ising* ***E****dge*) и из рисунка 3 получим:

Возвращаясь к уравнениям (8), запишем выражение для *Slack\_setup* в виде:

Значение *Tfpga\_ext\_setup* задается конфигурацией печатной платы и внешними микросхемами и с его помощью можно оценить предельно допустимую величину *Tare\_skew*, при которой временные ограничения все еще будут выполнены. Это соответствует ситуации, когда значение *Slack\_setup* равно нулю, откуда получаем:

Теперь рассмотрим задержки для анализа по *Hold*. Будем считать, что тактовый сигнал распространяется максимально медленно, а данные – максимально быстро, и введем переменную *Tbre\_skew* (***B****efore* ***R****ising* ***E****dge*):

Если провести аналогичные рассуждения и приравнять значение *Slack\_hold* в формуле (8) к нулю, то получим:

В итоге, если подставить полученные результаты в уравнения (9), то ограничения на выходной сигнал FPGA можно записать виде:

Эти соотношения также присутствуют в Vivado Language Templates:

#  Rising Edge Source Synchronous Outputs

#

#  Source synchronous output interfaces can be constrained either by the

# max data skew relative to the generated clock or by the destination

# device setup/hold requirements.

##

# forwarded                \_\_\_\_\_\_\_\_\_\_\_\_\_

# clock        \_\_\_\_\_\_\_\_\_\_\_|             |\_\_\_\_\_\_\_\_\_

#                         |

#                 bre\_skew|are\_skew

#                 <------>|<------>

#           \_\_\_\_\_\_        |        \_\_\_\_\_\_\_\_\_\_\_\_

# data      \_\_\_\_\_\_XXXXXXXXXXXXXXXXX\_\_\_\_\_\_\_\_\_\_\_\_XXXXX

#

set fwclk           <clock\_name>;   # forwarded clock name

set fwclk\_period    <period\_value>; # forwarded clock period

set bre\_skew        0.000;          # skew requirement before rising edge

set are\_skew        0.000;          # skew requirement after rising edge

set output\_ports    <output\_ports>; # list of output ports

# Output Delay Constraints

set\_output\_delay -clock $fwclk -max [expr $fwclk\_period - $are\_skew]

[get\_ports $output\_ports];

set\_output\_delay -clock $fwclk -min $bre\_skew [get\_ports $output\_ports];

1. **Инвертирование тактового сигнала.**

Рассмотрим один из возможных вариантов решения проблемы с нарушением временных ограничений из предыдущего раздела. Отрицательное значение *Slack\_setup* указывает, что данные распространяются слишком медленно. Чтобы они успели достичь защелкивающего триггера, можно ввести в дополнительную задержку для тактового сигнала. Для этого выполним инверсию сигнала i\_clk перед его выдачей на выход FPGA.

После инвертора фронты такового сигнала превратятся в спады, а   
спады – во фронты. По отношению к текущему фронту ближайший спад появляется спустя половину периода. Далее после инверсии он превращается во фронт и попадает на выходную ножку FPGA. Если считать, что период тактового сигнала равен 20 нс, то с помощью таких преобразований мы фактически дополнительно задерживаем защелкивающий фронт на 10 нс.

Код проекта для FPGA с внесенными изменениями представлен ниже. На рисунке 13 показана его схема. Можно увидеть, что в схеме появился дополнительный LUT, выполняющий инверсию сигнала i\_clk.

module top\_2 (

    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

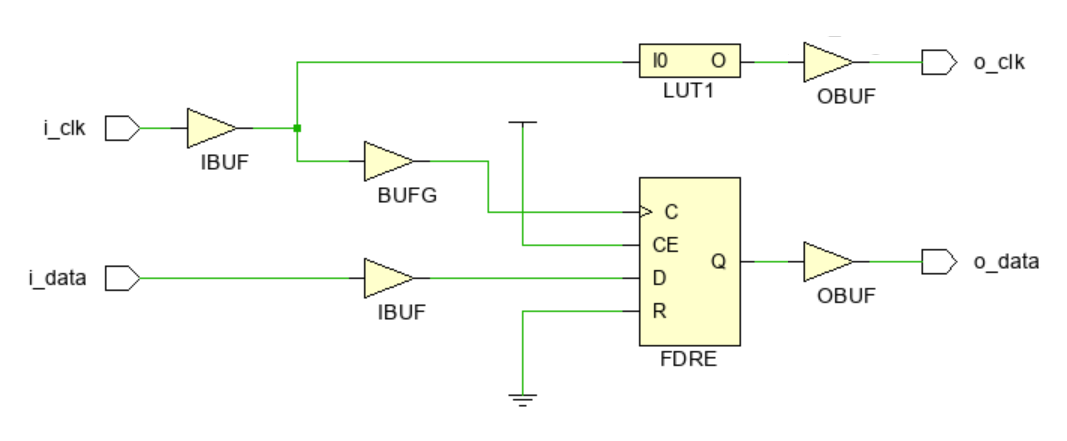


Рисунок 13. Обновленная схема FPGA проекта.

Обо всех действиях, которые выполняются при формировании выходного тактового сигнала o\_clk, необходимо сообщить Vivado. Для этого в команду *create\_generated\_clock* нужно добавить флаг -*inverted*. Полное содержимое xdc-файла представлено ниже:

# 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

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] -invert -divide\_by 1 [get\_ports o\_clk]

# задержки распространения тактового сигнала от FPGA до LAN8740A

set\_clock\_latency -min $Tcbd\_min [get\_clocks o\_clk]

set\_clock\_latency -max $Tcbd\_max [get\_clocks o\_clk]

# временные ограничения для выходного сигнала o\_data

set\_output\_delay -clock o\_clk -max [expr $Tdbd\_max + $Tdsu] [get\_ports o\_data]

set\_output\_delay -clock o\_clk -min [expr $Tdbd\_min - $Tdh]  [get\_ports o\_data]

На рисунке 14 представлены задержки распространения защелкивающего фронта при анализе по *Setup*. Из первой строки можно увидеть, что теперь дополнительная задержка перед появлением фронта составляет 10 нс, то есть половину периода тактового сигнал.

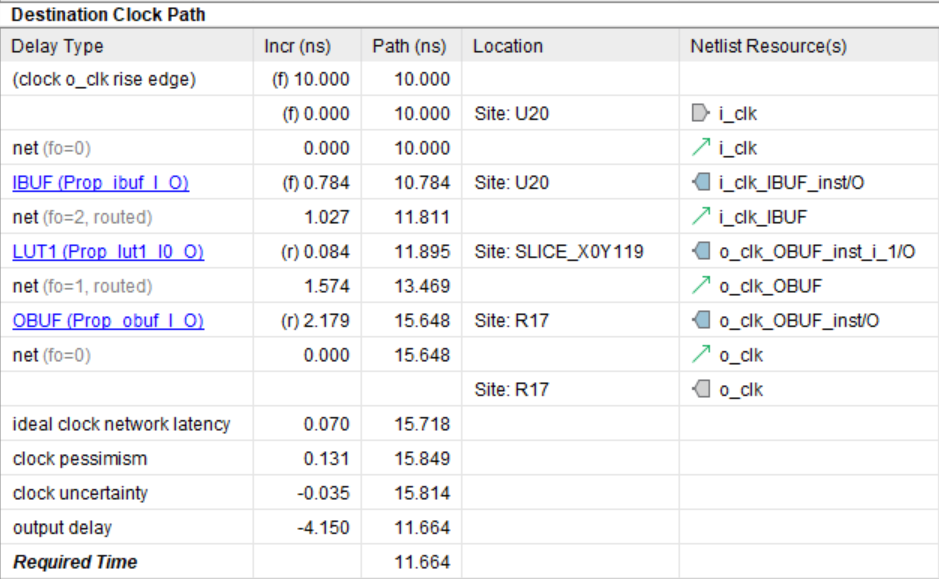


Рисунок 14. Задержки защелкивающего фронта.

Также обратите внимание, что несмотря на то, что мы защелкиваем данные текущим фронтом, в xdc-файле в командах *set\_output\_delay* отсутствуют слагаемые *Tclk* (см. уравнения 9). Напомним, что они вводились для компенсации задержки, которую Vivado автоматически добавлял для защелкивающего фронта. В данном случае эта задержка отсутствует.

Как видно из рисунка 15, теперь *Slack* имеет положительное значение, а значит временные ограничения выполнены, и проблема решена.

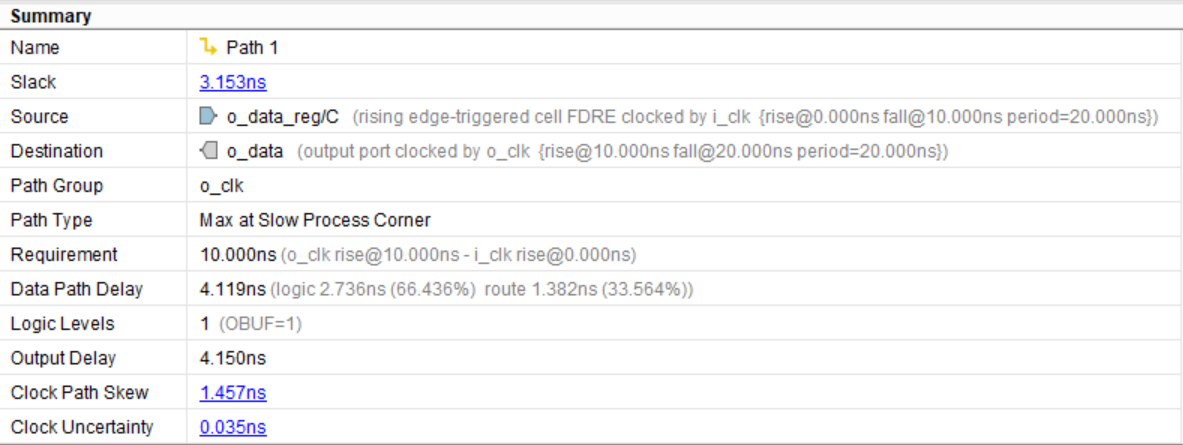


Рисунок 15. Раздел Summary для анализа по Setup.

1. **Использование ODDR триггера.**

Рассмотренный ранее способ инвертирования тактового сигнала с использованием LUT по ряду причин не является лучшим решением. Для формирования выходного тактового сигнала рекомендуется использовать DDR триггер. В кристаллах фирмы Xilinx такой триггер находится в IO Block, и его можно задействовать в своем проекте с помощью примитива ODDR [4]. Ниже представлен код FPGA проекта, в котором используется ODDR, а на рисунке 16 его схема.

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;

    ODDR ODDR\_reg (

        .C(i\_clk),

        .D1(1'b0),

        .D2(1'b1),

        .Q(o\_clk),

        .CE(1'b1),

        .R((1'b0),

        .S((1'b0)

    );

endmodule

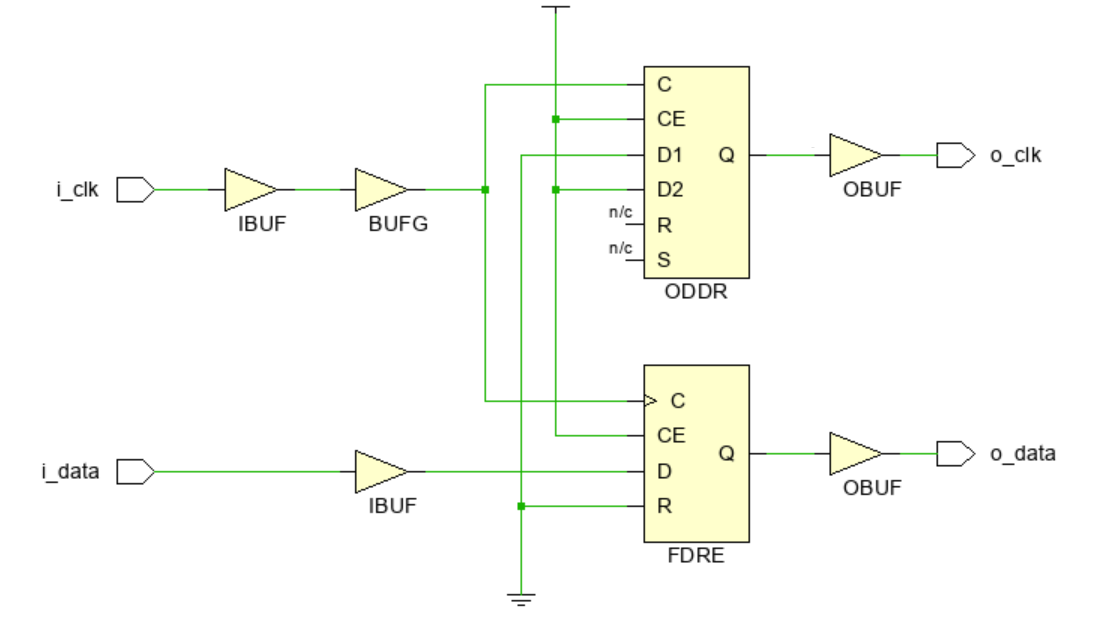


Рисунок 16. Схема FPGA проекта с ODDR.

Отметим некоторые моменты, на которые следует обратить внимание. На тактовый вход C ODDR поступает сигнал i\_clk. В качестве источника для o\_clk теперь выступает сигнал с выхода Q ODDR. По каждому фронту на входе   
C ODDR сигнал со его входа D1 передается на выход Q. В свою очередь значение на входе D2 попадает на выход Q по каждому спаду тактового сигнала.

По коду и на схеме можно увидеть, что на вход D1 ODDR постоянно подается нулевой сигнал, а на вход D2 – единичный. Таким образом, по фронту сигнала i\_clk значение выхода Q ODDR изменяется с единицы в ноль, и появляется спад. По каждому спаду i\_clk прошлое нулевое значение на   
выходе Q изменяется на единичное, и формируется фронт. В результате на выходе Q ODDR получается инвертированная копия сигнала i\_clk.

Для проведения временного анализа можно использовать xdc-файл из предыдущего раздела без каких-либо изменений. На рисунке 17 представлен раздел *Summary* временного отчета, который показывает, что ограничения выполнены, причем запас, равный 4.251 нс, оказался даже больше, чем при инвертировании тактового сигнала с помощью LUT.

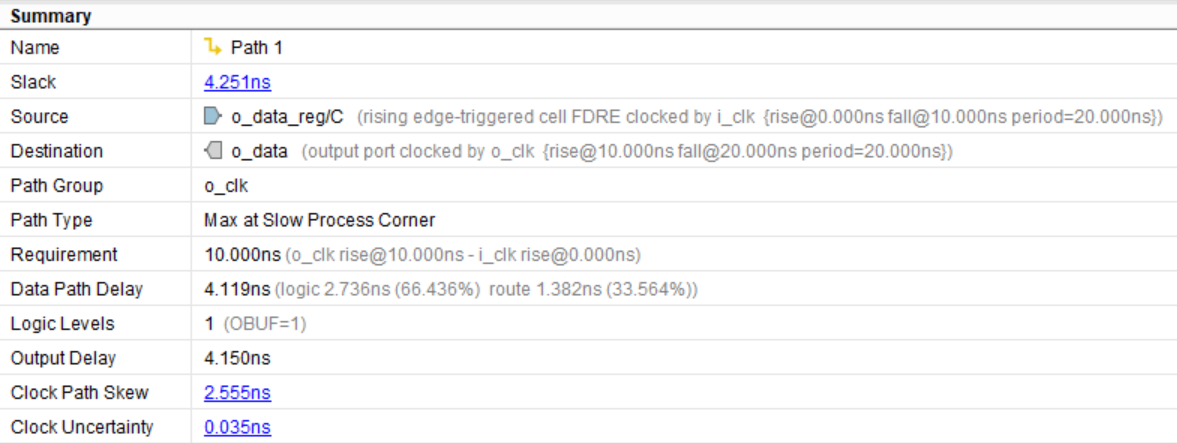


Рисунок 17. Раздел Summary для анализа по Setup.

Помимо возможности инвертировать сигнал, использование ODDR имеет еще несколько преимуществ [5]. Во-первых, как можно увидеть из рисунка 13, тактовый сигнал перед поступление на LUT не проходит через тактовый буфер (BUFG), то есть он передается по обычным трассировочным линиям FPGA. В случае с ODDR сигнал все время распространяется по специальным тактовым линиям. Во-вторых, наличие ODDR гарантирует сохранение duty cycle тактового сигнала, что может быть критично при выдачи данных из FPGA в DDR режиме.

Также в Vivado есть специальный помощник – Constraints Wizard, который анализирует проект и подсказывает, на какие его части требуется добавить ограничения. Он находит в netlist элементы, отвечающие за выдачу тактового сигнала с помощью ODDR, и предоставляет удобный способ задания флагов для команды *create\_generated\_clock*.

Использование DDR триггера никак не скажется на увеличении ресурсов проекта, так как он всегда присутствует в IO Block и в противном случае просто останется незадействованным. Даже наоборот, при инвертировании тактового сигнала можно сэкономить один LUT.

Исходя и всего выше сказанного, для выдачи тактового сигнала из FPGA крайне рекомендуется всегда использовать DDR триггер, хотя это и не строго обязательно.

**Заключение.**

В статье был рассмотрен временной анализ при Source Synchronous передаче сигналов из FPGA во внешнее устройство. Показан вывод уравнений статического временного анализа для двух способов приема данных: по текущему и по следующему фронту тактового сигнала. Приведены аргументы в пользу применения DDR триггера для выдачи тактового сигнала из FPGA.

**Ссылки.**

1. [Основы статического временного анализа. Часть 1: Period Constraint](https://fpga-systems.ru/static-timing-analysis-part-1-period-constraints)
2. [Основы статического временного анализа. Часть 2.2: System Synchronous Output Delay Constraint](https://fpga-systems.ru/sta-basic-system-synchronous-output-delay-constraint)
3. [Datasheet LAN8740A](https://fpga-systems.ru/go?http://www.datasheet.es/PDF/1021686/LAN8740A-pdf.html)
4. [Xilinx Libraries Guide for HDL Designs (UG 768)](https://www.xilinx.com/htmldocs/xilinx14_7/7series_hdl.pdf)
5. [Xilinx Forum](https://support.xilinx.com/s/question/0D52E00006hpcgDSAQ/why-oddr-for-forwarded-clock?language=en_US)