**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 вариантов защелкивания данных. Разобраны практические примеры создания временных ограничений. Показан способ решения проблем с 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.

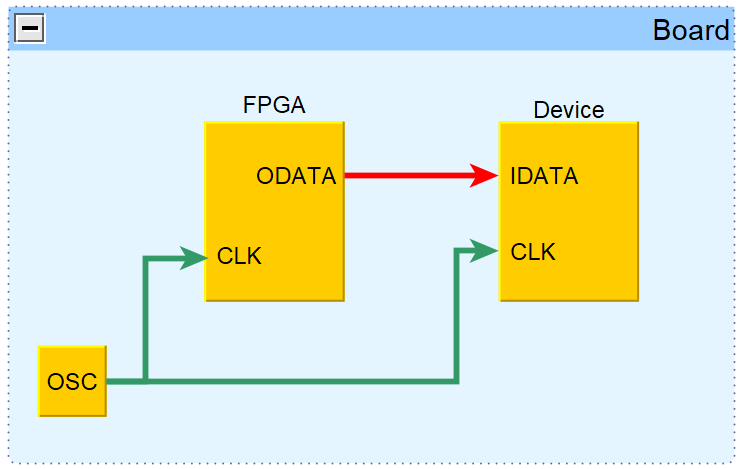


Рисунок 1. Соединение устройств на плате для случая System Synchronous.

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*.*

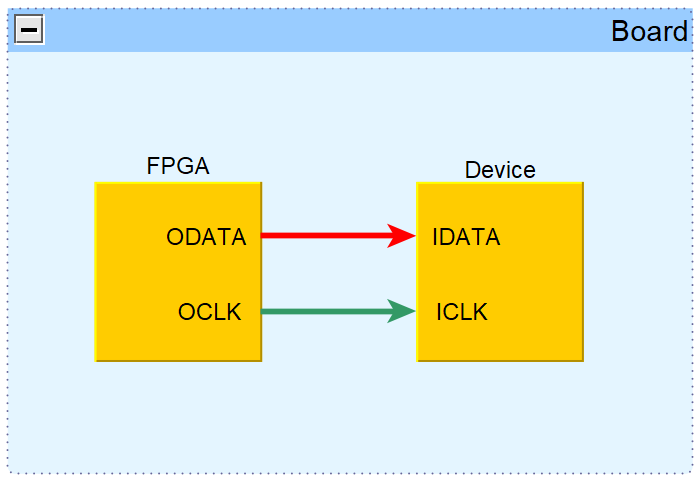


Рисунок 2. Соединение устройств на плате для случая Source Synchronous.

При передаче данных между FPGA и Device запускающий триггер располагается в FPGA, а защелкивающий – во внешнем устройстве. На рисунке 4 показан анализируемый путь, на который нанесены задержки сигналов. В случае Source Synchronous вместе с данными передает тактовый сигнал, которые может внутри FPGA может формироваться с помощью PLL или браться непосредственно с входной тактовой ножки.

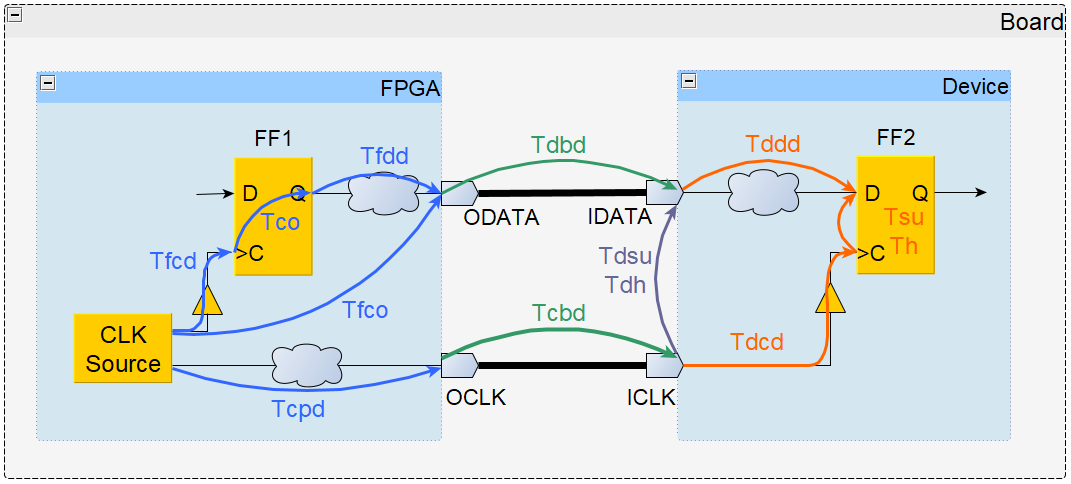


Рисунок 3. Путь с задержками для выходных данных и тактового сигнала.

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 выходе Q запускающего триггера ножки ODATA FPGA.

Период тактового сигнала будем обозначать *Tclk*. Оранжевым и зеленым цветом на рисунке 4 представлены задержки для участков пути, которые располагаются вне FPGA. Эти задержки необходимо указать временному анализатору Vivado.

1. **Максимальное время распространения.**

Для начала рассмотрим, каким образом выполняется анализ для проверки 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, можно определить фактическое время прибытия данных, как сумму задержек для запускающего фронта и задержек распространения данных

где введено обозначение (*Tfco* – ***F****PGA* ***C****lock to* ***O****utput time*):

В дальнейшем будет рассмотрено два варианта передачи данных между FPGA и Device. В первом случае данные будет запускаться из FPGA по одному фронту, а защелкиваться в Device - по следующему. При втором варианте данные будут запускаться и защелкиваться по одному и тому же тактовому фронту. Чтобы описать оба эти варианта введем дополнительную переменную *Tccd\_setup* (***C****apture* ***C****lock* ***D****elay*), которая описывает интервал времени между запускающим и защелкивающим фронтами.

Тогда из рисунка 3 можно получить требуемое время прибытия данных, как разницу между временем прибытия защелкивающего фронта и временем удержания триггера.

где *Tclk\_delay\_min* – задержка распространения тактового сигнала от его источника в FPGA до входа защелкивающего триггера в Device

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

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

где *Tdsu (****D****evice* ***S****e****T****up)*– время установке 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 окончательное выражения для *Slack* при анализе по *Setup*:

1. **Минимальное время распространения.**

Теперь рассмотрим, как выполняется анализ для проверки ограничения на минимальное время распространения (*Hold*). При анализе по *Hold* считается, что задержки для запускающего фронта и данных имеют минимальное значение, а для защелкивающего фронта – максимальное.

По аналогии с анализом по *Setup* уравнения для фактического времени прибытия данных имеют вид:

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

Переменная *Tccd\_hold* описывает интервал времени между текущим запускающим и предыдущим защелкивающим фронтами тактового сигнала.

Уравнение для *Slack* при анализе по *Hold* имеет вид:

С учетом полученных ранее результатов получим:

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 выражения для анализа по *Hold*:

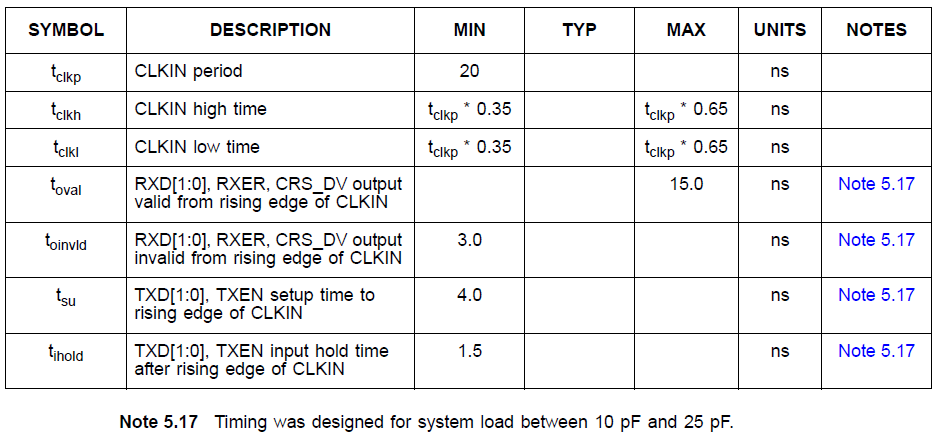
1. **Пример с защелкивание данных по следующему фронту.**

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

Учитывая эти результаты, уравнения (3) и (4) примут вид:

В данном случае все задержки вне FPGA объединены в слагаемых *Tfpga\_ext\_setup* и *Tfpga\_ext\_hold*. Эти задержки необходимо указать Vivado в виде следующих значений:

В качестве примера рассмотрим передачу данных по RMII из FPGA в микросхему Ethernet PHY LAN8740A [2]. На рисунке 4 приведены таблица со значениями задержек и временная диаграмма сигналов из datasheet на LAN8740A. Для краткости ограничения будут продемонстрированы для одного выходного порта FPGA, который подключен к ножке TXD[0] микросхемы LAN8740A.



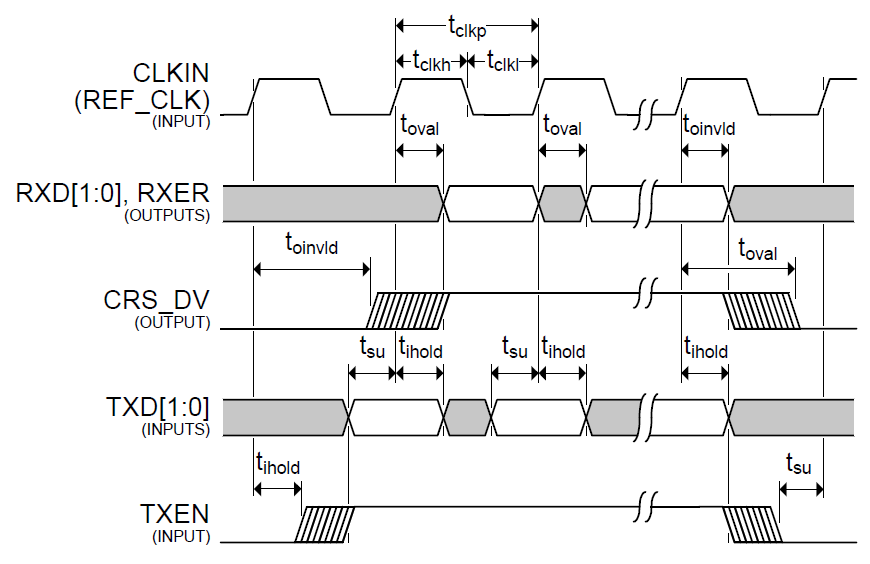


Рисунок 4. Задержки и временные диаграммы для LAN8740A.

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

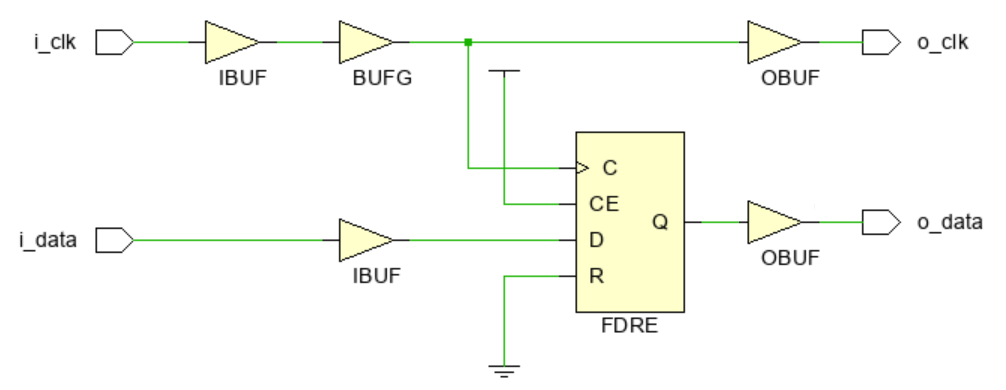


Рисунок 5. Схема FPGA проекта.

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

Из рисунка 4 находим период тактового сигнала *Tclk*, а также время установки *Tdsu* и удержания *Tdh* для LAN8740A, которые на рисунке 4 обозначены *Tsu* и *Tihold* соответственно. Эти значения также внесем в   
xdc-файл и сразу создадим ограничение на период тактового сигнала, поступающего в FPGA через порт 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]

Тактовый сигнал поступает на выходную ножку FPGA o\_clk и затем используется для тактирования микросхемы LAN8740A. При этом считается, что на ножке o\_clk формируется новый тактовый сигнал. Чтобы этот сигнал мог учувствовать во временном анализе, его параметры нужно указать Vivado. 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]

Новый тактовый сигнал всегда создается из уже существующего (опорного), например, с помощью PLL или деления на основе счетчика. С помощью ключа –source указывается источник опорного сигнала, а с помощью ключа –divide\_by – соотношение частот опорного и создаваемого сигналов, то есть на сколько делится частота опорного сигнала. В нашем случае опорный сигнал появляется на ножку FPGA i\_clk, поэтому используется флаг –source [get\_ports i\_clk]. Так как такой сигнал с i\_clk просто передается на выход o\_clk и его частота не изменяется значение флага –divide\_by равно единице. С помощью флага –name указывается имя формируемого тактового сигнала, а с помощью конструкции [get\_ports o\_clk] указывается место формирования нового сигнала.

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 ограничение 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 для защелкивание данных внутри LAN8740A используется выходной тактовый сигнал из FPGA, поэтому флагу –clock присвоено имя тактового сигнала o\_clk, созданного 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 ограничение 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 будут рассматриваться только ограничения по *Setup*. На рисунке 6 показаны расчеты фактического и требуемого времени прибытия данных для анализа по *Setup*.

На рисунке 6 представлен раздел Summary временного отчета, из строки Source которого можно увидеть, что путь внутри FPGA начинается на тактовом входе триггера o\_data\_reg и тактовый сигнал i\_clk является запускающим. Из строки Destination видно, что путь заканчивается на выходной ножке o\_data, а данные защелкиваются по сигналу o\_clk. Положительное значение Slack, равное 10.763 нс, указывает на то, что временные ограничения выполнены.

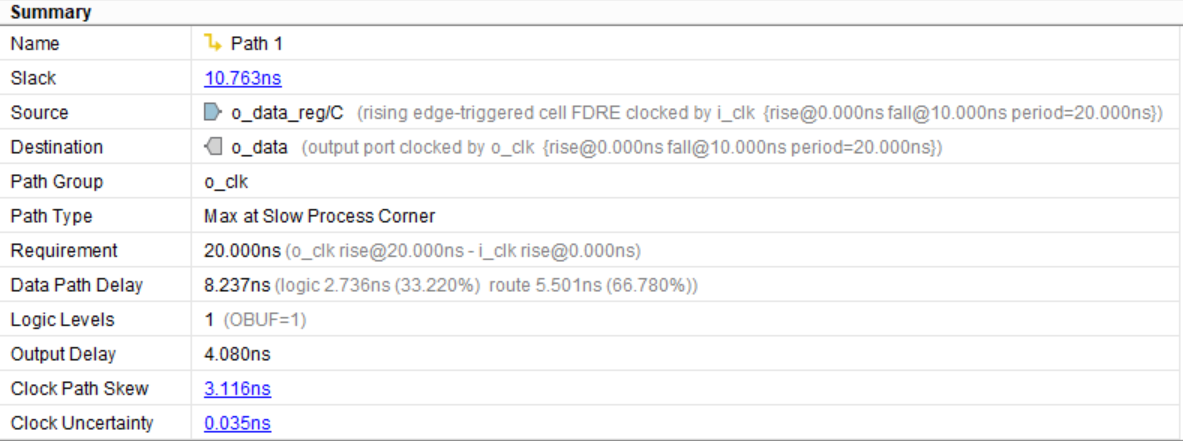


Рисунок 6. Раздел Summary для анализа по Setup.

На рисунке 7 представлены задержки распространения запускающего фронта и данных. Можно увидеть, что фронт появляется в нулевой момент времени и распространяется через входной и тактовый буферы. Через 4.392 нс он достигает тактовой ножки триггера и запускает передачу данные. В свою очередь данные распространяются через выходной буфер и появляются на ножке FPGA o\_data в момент времени 12.628 нс.

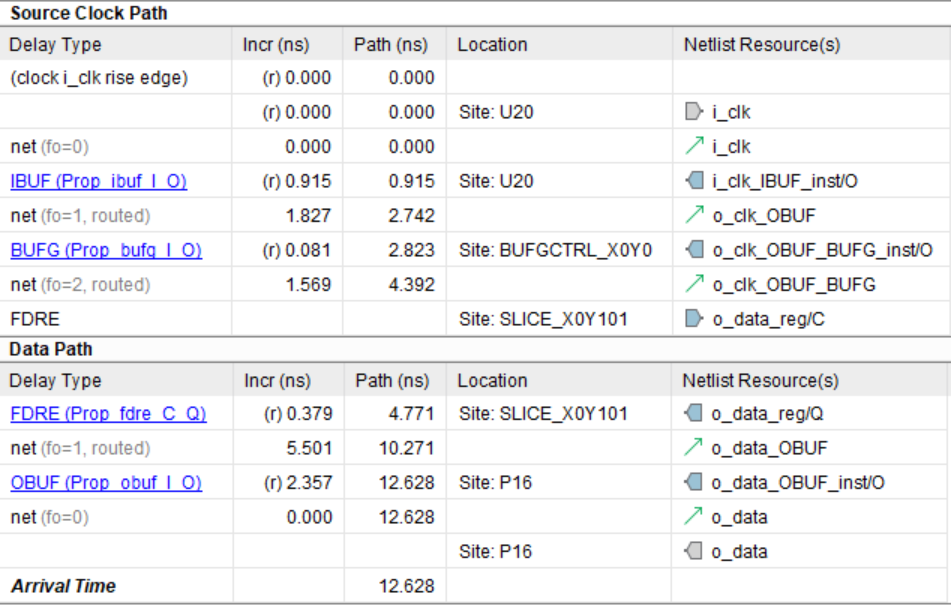


Рисунок 7. Задержки запускающего фронта и данных.

Задержки для защелкивающего фронта представлены на рисунке 8. Выходной тактовый сигнал формируется из сигнала i\_clk, следующий фронт которого формируется на ножке FPGA через 20 нс после защелкивающего фронта. Защелкивающий фронт распространяется через входной, тактовый и выходной буферы и появляется на ножке o\_clk FPGA в момент времени 27.214 нс.

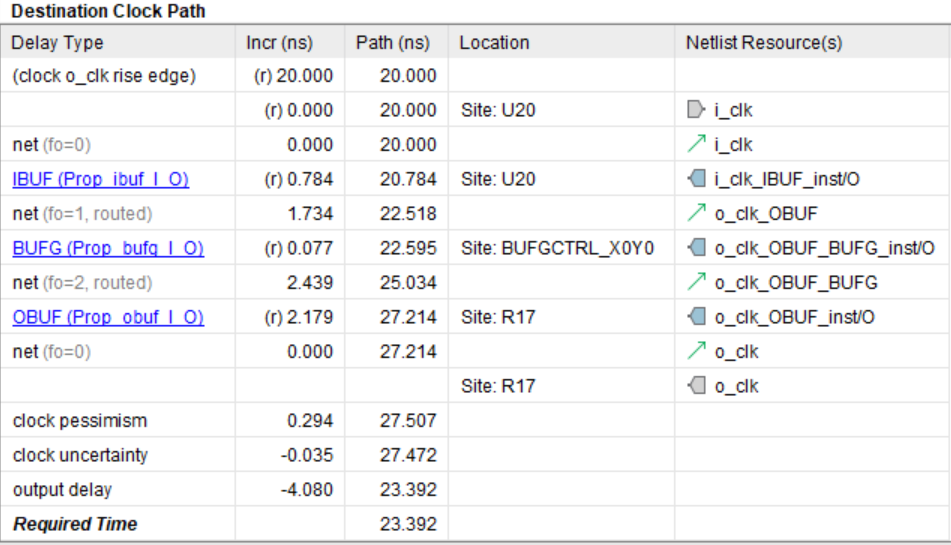


Рисунок 8. Задержки защелкивающего фронта.

Задержки на распространение данных и тактового сигнала по дорожкам печатной платы, а также время установки для LAN8740A учитываются с помощью слагаемого output delay, равного 4.0 + 0.15 – 0.07 = 4.08 нс. Учитывая 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. Вычитая из этого значения фактическое время прибытия данных, 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, например, Tcbd\_max и Tdbd\_max. За этим нужно внимательно следить, чтобы не совершить ошибку.

Чтобы немного упростить, вспомним, что для тактового сигнала можно отдельно указывать задержки распространения с помощью команды clock\_latency. Для нашего примера следует внести в xdc-файл 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 указывается минимальная (Tcbd\_min) и максимальная (Tcbd\_max) задержки распространения тактового сигнала с выход o\_clk FPGA до микросхемы LAN8740A. Так как эти задержки уже указаны, их нужно брать из уравнений (6), которые теперь примут вид:

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]

# 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 ограничение 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, которая отдельно указывает задержку тактового сигнала вне FPGA. В остальном все задержки останутся теми же, что и на рисунках 6-8, и значение Slack также будет равно 10.763 нс.

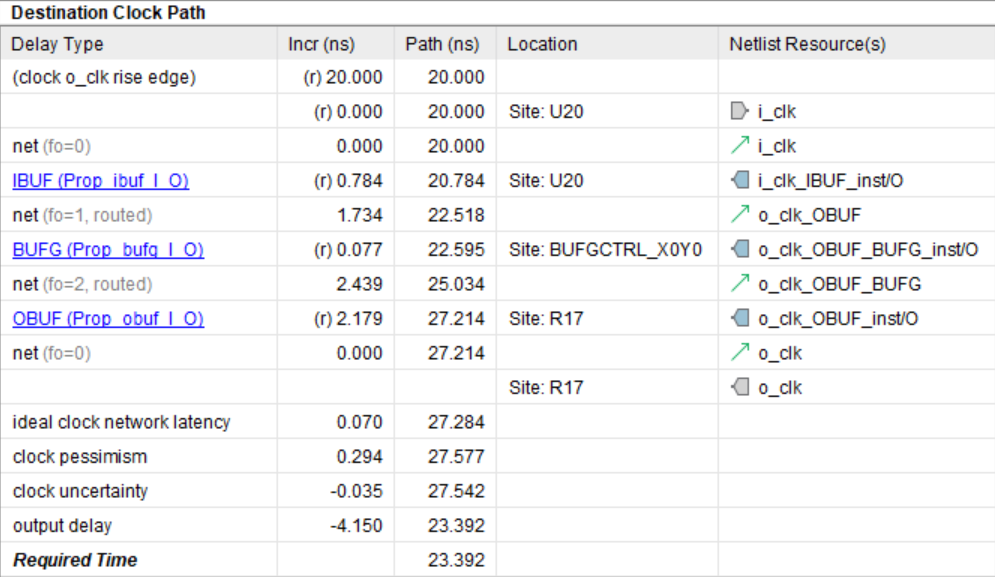


Рисунок 9. Задержки защелкивающего фронта.

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, называется fwclk (forwarded clock). Максимальные (*Tdbd\_max*) и минимальные (*Tdbd\_min*) задержки распространения данных по дорожкам платы обозначены, как trce\_dly\_max и trce\_dly\_min. Задержки tsu и thd соответствуют времени установки и удержания для микросхемы LAN8740A.

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

Теперь рассмотрим второй вариант, при котором данные запускаются и защелкиваются одним и тем же фронтом тактового сигнала. Так как защёлкивавший и запускающий фронты появляются одновременно значение Tccd\_setup в этом случае равно нулю. Предыдущий защелкивающий фронт появляется на одни период раньше текущего защелкивающего фронта, который формируется одновременно с текущим запускающий фронтом. Поэтому задержка между предыдущим защелкивающим и текущим запускающим фронтами равна Tccd\_hold = -Tclk. С учетом всего вышесказанного уравнения (3) и (4) примут вид:

Важно запомнить, что Vivado в время проведения временного анализа всегда считает, что данные запускаются одним тактовым фронтом, а   
защелкиваются - следующим. По этой причине для расчета значения Slack будут использоваться уравнения (5) из предыдущего раздела.

Уравнение (8) для анализа по Hold отличается от (5) только наличием дополнительного слагаемого Tclk. В уравнении для Setup наоборот отсутствует слагаемое Tclk. Добавим и вычтем это слагаемое и получим следующее уравнение для анализа по *Setup*:

Введем новое обозначения:

Уравнения (8) теперь можно записать в таком виде:

Полученные результаты с точностью до обозначений совпадают с уравнениями (5). Теперь с учетом предыдущего раздела можем записать выражения для ограничений на распространение выходных данных:

В качестве примера опять рассмотрим передачу данных из FPGA в микросхему LAN8740A по RMII. Будем использовать те же, что и в предыдущем разделе, задержки распространения данных и тактового сигнала по дорожкам платы. xdc-файл почти не изменится по сравнению с предыдущим случаем за исключением команды set\_output\_delay. Его полное 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 ограничение 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. Как и раньше для далее для краткости будем рассматривать результаты для анализа по Setup. На рисунке 10 представлен раздел Summary временного отчета. Отрицательное значение Slack указывает на нарушение временных ограничений. Чтобы выяснить причину нарушений изучим задержки распространения данных и тактового сигнала, представленные на рисунках 11 и 12.

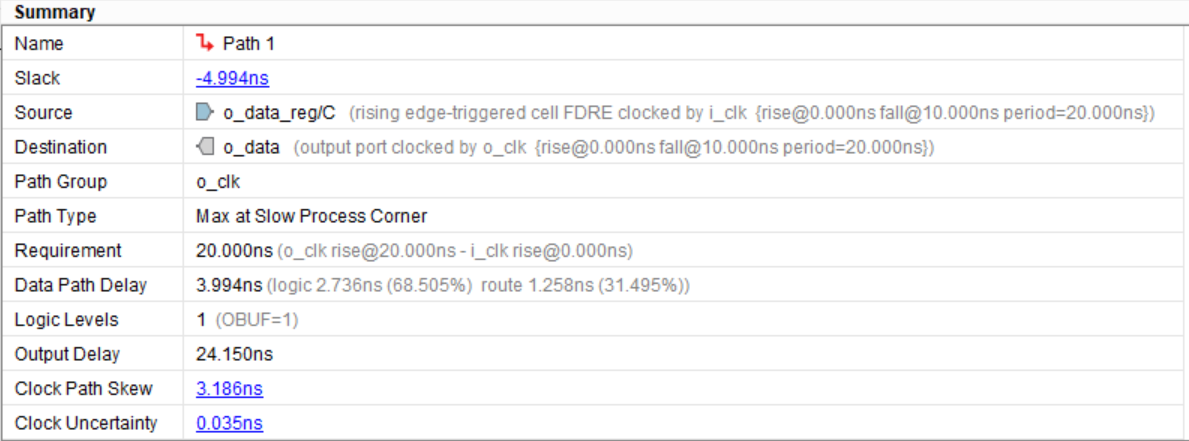


Рисунок 10. Раздел Summary для анализа по Setup.

Из рисунка 11 видно, что запускающий фронт формируется в нулевой момент времени, распространяется до триггера и запускает передачу данных. Спустя 8.363 нс после поступления тактового фронта данные появляются на выходной ножке o\_data FPGA.

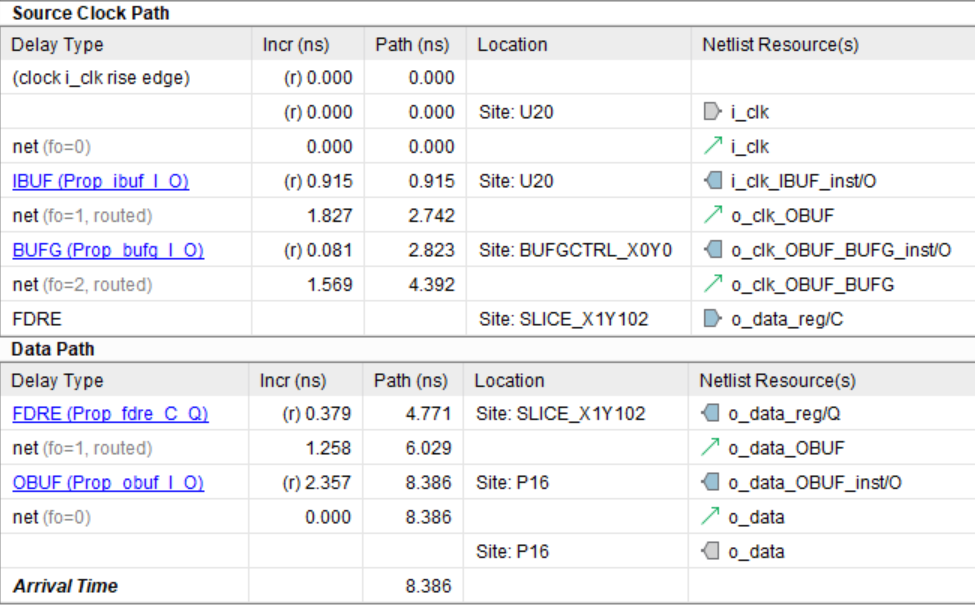


Рисунок 11. Задержки запускающего фронта и данных.

Далее из рисунка 12 находим, что защелкивающий фронт появляется через 20 нс после запускающего. Как уже упоминалось, это связано с тем, что Vivado при проведении временного анализа всегда считает, что данные запускаются одним фронтом, а защелкиваются – следующим. В момент времени равный 27.214 нс защелкивающий фронт достигнет ножки o\_clk FPGA. Обратите внимание на значение output delay, которое равно -24.15 нс. Если сравнить его со значением на рисунке 9, то получим, что они различаются на 20 нс, то есть на один период тактового сигнала. За счет этой разницы компенсируется задержка появления защелкивающего фронта в первой строке рисунка 12.

С учетом неопределённости и пессимизма тактового сигнала получаем, что данные должны уже появиться на выходе FPGA в момент времени 3.392 нс. Фактическое время появления данных равно 8.364 нс. Это на 4.994 нс позже, чем требуемое время прибытия, то есть данные распространяются слишком медленно.

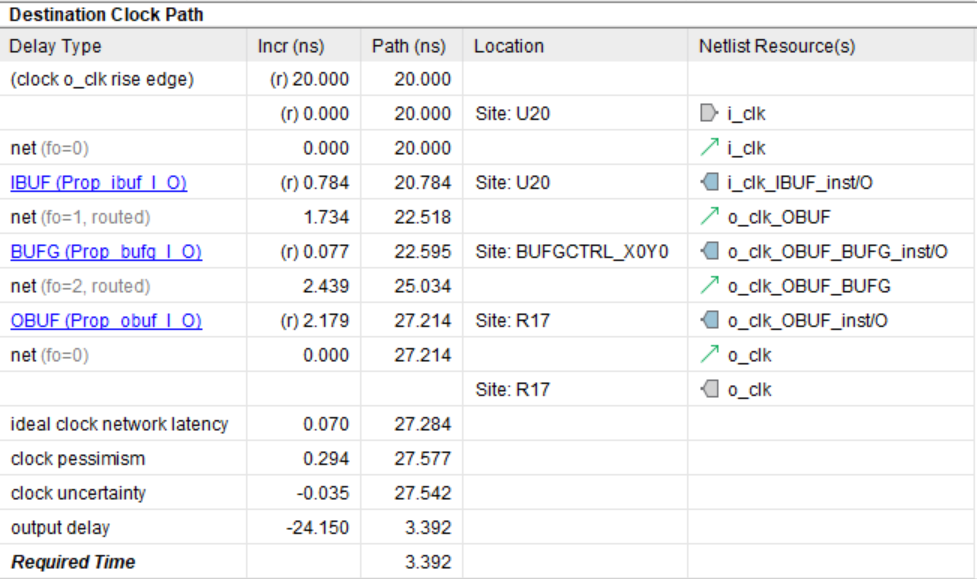


Рисунок 12. Задержки защелкивающего фронта.

Решение проблемы с отрицательным значением Slack будет представлено в следующем разделе, а пока давайте рассмотрим еще один способ создания временных ограничений на выходной сигнал. Иногда бывает удобно вводить ограничения в виде задержек между данными и тактовым сигналом на ножках FPGA. Такой подход называется FPGA Centric. Для анализа по Setup рассчитаем промежуток времени между появлением защелкивающего тактового фронта и данных на выходах FPGA. Будем считать, что тактовый сигнал распространяются максимально быстро, а данные – максимально медленно, поэтому 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 к FPGA микросхемами и поэтому фиксировано. Отсюда можно оценить предельно допустимое значение *Tare\_skew*, при котором временные ограничения все еще будут выполняться. Это соответствует ситуации, когда значение *Slack\_setup* равно нулю, поэтому можем записать:

Теперь рассмотрим противоположный случай для анализа по Hold. Будем считать, что тактовый распространяются 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 уравнении (8) к нулю, то получим:

В итоге подставив полученные результаты в уравнения (9), можем записать ограничения на выходной сигнал FPGA виде:

Эти соотношения также можно обнаружить 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 данные успели достичь защелкивающего триггера раньше фронта, можно ввести в дополнительную задержку для тактового сигнала. Это можно сделать, если инвертировать тактовый сигнала перед его выдачей на выходную ножку FPGA.

После инвертора фронта такового сигнала превратятся в спады, а спады - во фронты. Ближайший спад после текущего фронта появляется спустя половину периода. После инверсии он превратится во фронт и попадет на выходную ножку FPGA. Если, как и ранее, считает, что период равен 20 нс, то с помощью таких преобразований мы фактически задерживаем тактовый сигнал дополнительно на 10 нс.

Код проекта FPGA с внесенными изменениями представлен ниже, а на рисунке 13 показана его схема. Можно увидеть, что в схеме появился дополнительный LUT, выполняющий инверсию тактового сигнала перед его выдачей на ножку o\_clk FPGA.

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

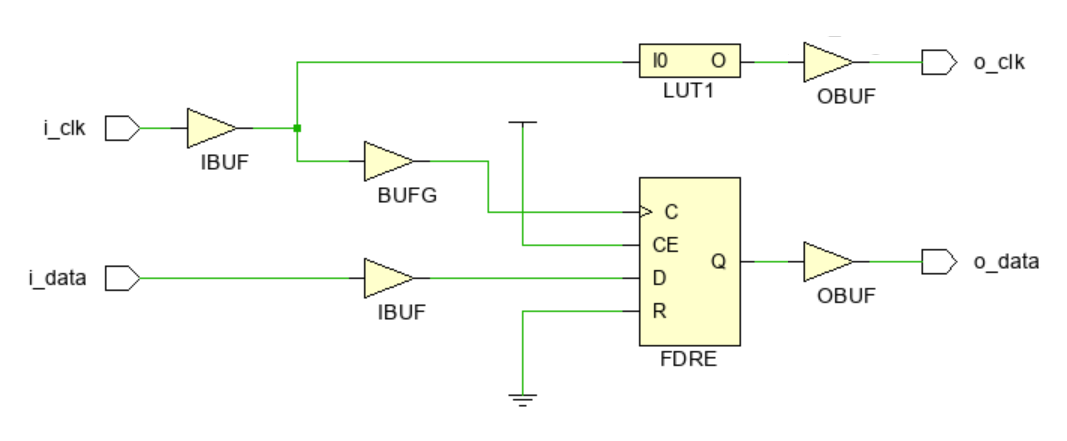


Рисунок 13. Обновленная схема FPGA проекта.

О дополнительных преобразованиях, которые теперь выполняются при формировании выходного тактового сигнала, необходимо сообщить Vivado. Для этого в команду create\_generated\_clock нужно добавить флаг –inverted, указывающий, на инверсию сигнала. 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 ограничение 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. Можно увидеть, что теперь защелкивающий фронт появляется в момент времени 10 нс, то есть спустя половину периода тактового сигнал.

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). Эти слагаемые вводились для компенсации задержки в *Tclk* нс, которую Vivado автоматически добавлял для защелкивающего фронта. В данном случае эта задержка отсутствует. Также, как видно из рисунка 15, теперь Slack имеет положительное значение, а значит временные ограничения выполнены.

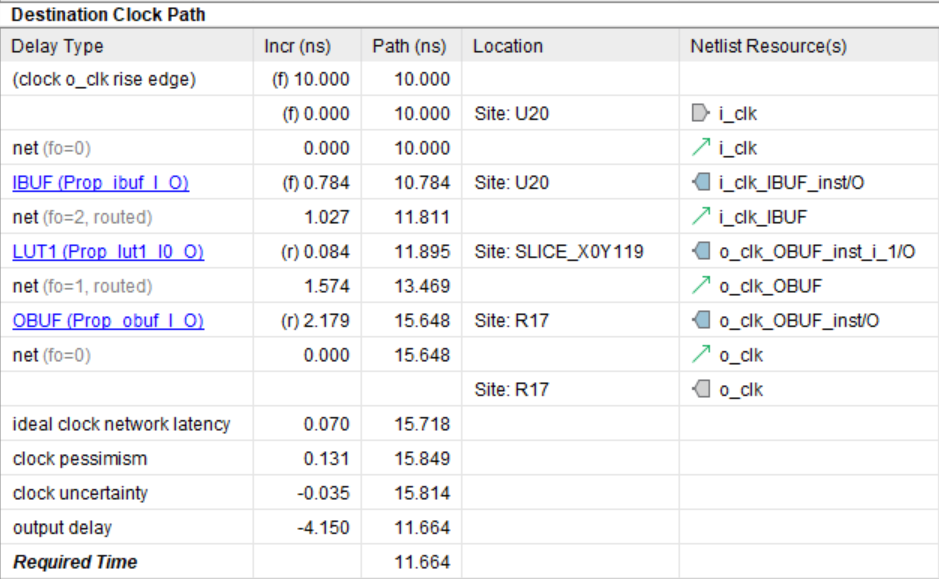


Рисунок 14. Задержки защелкивающего фронта.

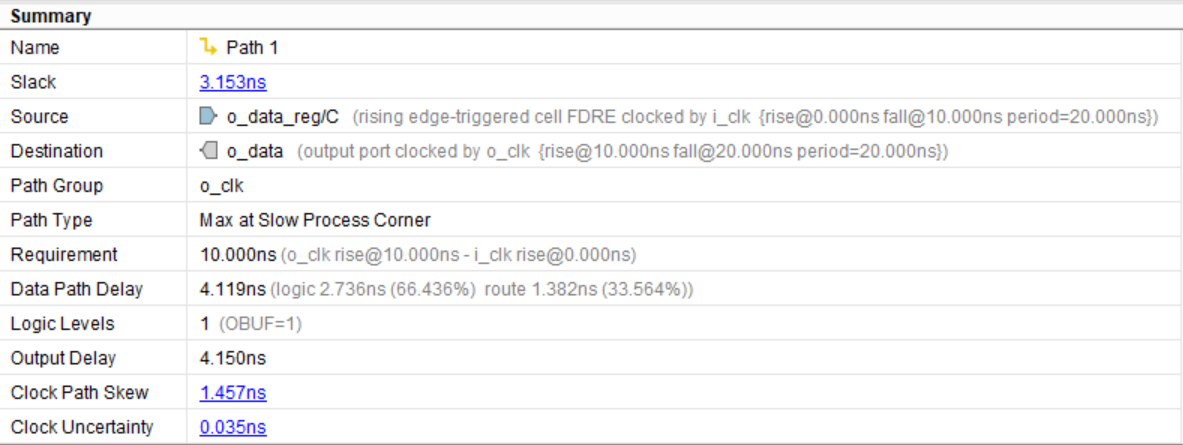


Рисунок 15. Раздел Summary для анализа по Setup.

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 [1]. 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

Отметим моменты, на которые следует обратить внимание. На тактовый вход C ODDR поступает тактовый сигнал с ножки i\_clk FPGA. В качестве выходного тактового сигнала FPGA выступает сигнал с выхода Q ODDR. По каждому фронту на входе C сигнал со входа D1 передается на выход Q. Сигнал со входа D2 передается на выход Q по каждому спаду тактового сигнала.

Можно увидеть, что на вход D1 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, то есть формируется спад. В свою очередь, по спаду i\_clk прошлое нулевое значение на выходе Q изменяется на единичное, то есть формируется фронт. В результате на выходе Q ODDR получается инвертированная копия сигнала i\_clk.

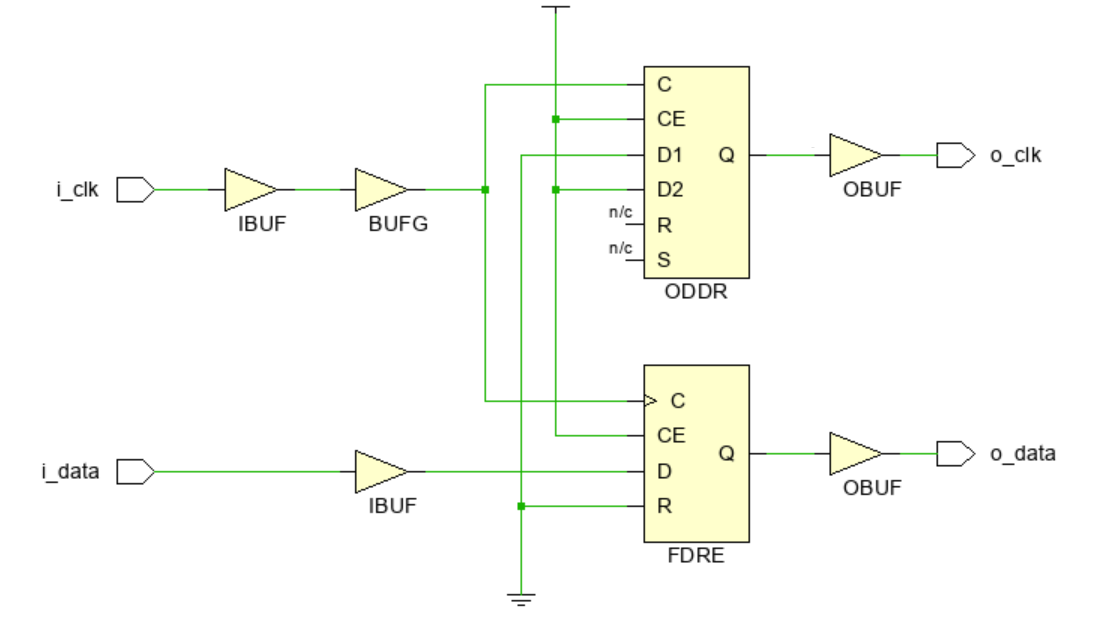


Рисунок 16. Схема FPGA проекта с ODDR.

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

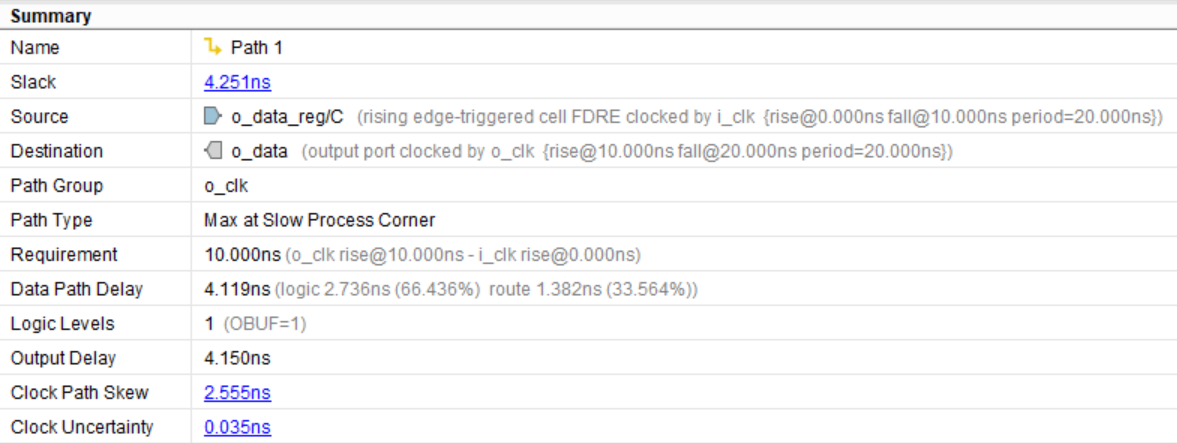


Рисунок 17. Раздел Summary для анализа по Setup.

Помимо возможности инвертировать сигнал использование ODDR имеет еще несколько преимуществ [2]. 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. В случае с ODDR тактовый сигнал все время распространяется по специальным тактовым линия. Во-вторых, наличие ODDR гарантирует сохранение duty cycle тактового сигнала, что может быть критично при передачи данных в DDR режиме.

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

Применение DDR триггера никак не скажется на увеличении ресурсов проекта, так как он всегда присутствует в IO Block и в противном случае просто останется неиспользованным. При инвертировании тактового сигнала с его помощью наоборот 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 вариантов защелкивания данных: по текущему и по следующему тактовому фронту. 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
3. Xilinx Libraries Guide for HDL Designs (UG 768)
4. Datasheet LAN8740A
5. Xilinx Forum