

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.

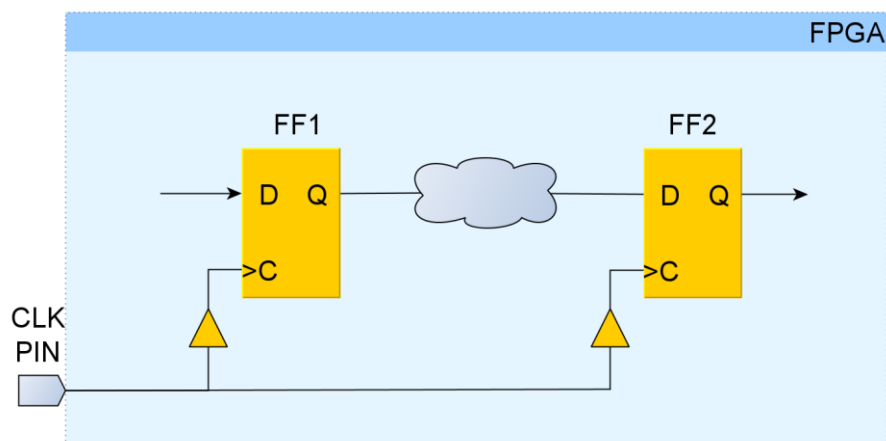


Рисунок 1. Типичный путь данных внутри FPGA.

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 запускаящий (*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 запускаящим 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 запускаящего 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.

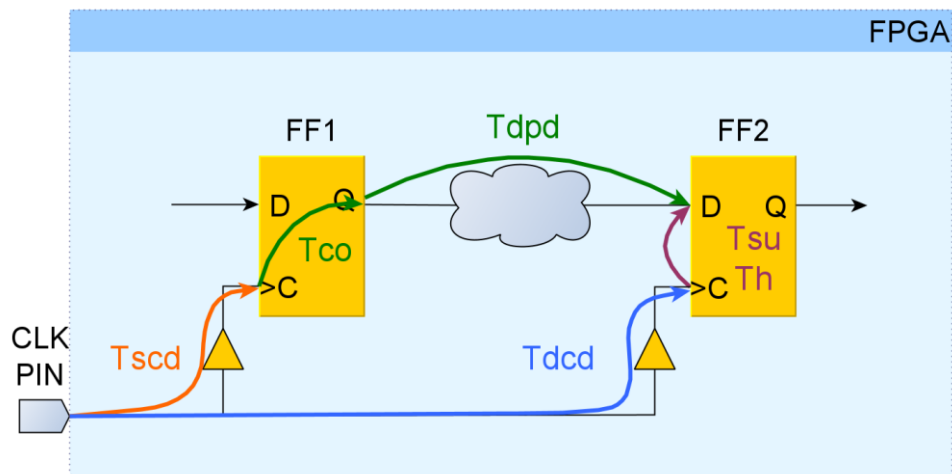


Рисунок 2. Путь с задержками для данных и тактового сигнала.

Ниже даны определения задержек, представленных на рисунке 2.

- ***Tscd*** (*Source Clock Delay*) – задержка тактового сигнала от источника, в нашем примере ножка FPGA clk_pin, до тактового входа триггера FF1;
- ***Tdcd*** (*Destination Clock Delay*) – задержка тактового сигнала от источника до тактового входа триггера FF2;
- ***Tco*** (*Clock to Output*) – интервал времени между приходом фронта на тактовый вход триггера и появлением данных на выходе Q;
- ***Tdpd*** (*Data Propagation Delay*) – задержка распространения данных по соединениям (*nets*) и через комбинационную логику;
- ***Tsu*** (*SetUp time*) – время установки. До прихода защелкивающего фронта данные на D входе триггера уже должны быть стабильны в течении времени ***Tsu***.
- ***Th*** (*Hold 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.

2. 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 (*Source Clock Arrival time*):

$$T_{sca_max} = T_{scd_max}. \quad (1)$$

- Задержка распространения данных (*Data Delay*):

$$T_{dd_max} = T_{co_max} + T_{dpd_max}. \quad (2)$$

- Время прибытия данных на вход защелкивающего триггера (*Data Arrival time*):

$$\begin{aligned} T_{da_max} &= T_{sca_max} + T_{dd_max} = \\ &= T_{scd_max} + T_{co_max} + T_{dpd_max}. \end{aligned} \quad (3)$$

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 (*Destination Clock Arrival time*):

$$T_{dca_min} = T_{dcd_min} + T_{clk}. \quad (4)$$

- Требуемое время прибытия данных (*Data Required time*):

$$T_{dr_min} = T_{dca_min} - T_{su} = T_{dcd_min} + T_{clk} - T_{su}. \quad (5)$$

В предыдущем уравнении учитывается, что данные должны прийти на время T_{su} раньше защелкивающего фронта. Таким образом, чтобы выполнялось требование на максимальное время распространения ($Setup$), данные должны дойти до конца пути не позднее времени T_{dr} .

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

$$Slack = T_{dr_min} - T_{da_max}. \quad (6)$$

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 $Slack$

$$Slack = T_{dca_min} - T_{sca_max} - T_{dd_max} - T_{su}.$$

$$Slack = T_{ck} + T_{dcd_min} - T_{scd_max} - T_{co_max} - T_{dpd_max} - T_{su}. \quad (7)$$

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

$$T_{skew} = T_{dcd} - T_{scd}. \quad (8)$$

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

3. 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 (*Source Clock Arrival time*):

$$T_{sca_min} = T_{scd_min}.$$

- Задержка распространения данных (*Data Delay*):

$$T_{dd_min} = T_{co_min} + T_{dpd_min}. \quad (9)$$

- Время прибытия данных на вход защелкивающего триггера (*Data Arrival time*):

$$T_{da_min} = T_{sca_min} + T_{dd_min} = T_{scd_min} + T_{co_min} + T_{dpd_min}.$$

- Время прибытия фронта к защелкивающему триггеру (*Destination Clock Arrival time*):

$$T_{dca_max} = T_{dcd_max}.$$

- Требуемое время прибытия данных (*Data Required time*):

$$T_{dr_max} = T_{dca_max} + T_h = T_{dcd_max} + T_h.$$

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

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

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

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

$$Slack = T_{da_min} - T_{dr_max}. \quad (10)$$

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:

$$Slack = Tsca_{min} + Tdd_{min} - Tdca_{max} - Th.$$

$$Slack = Tscd_{min} + Tco_{min} + Tdpd_{min} - Tdcd_{max} - Th. \quad (11)$$

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*.

4. 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

```

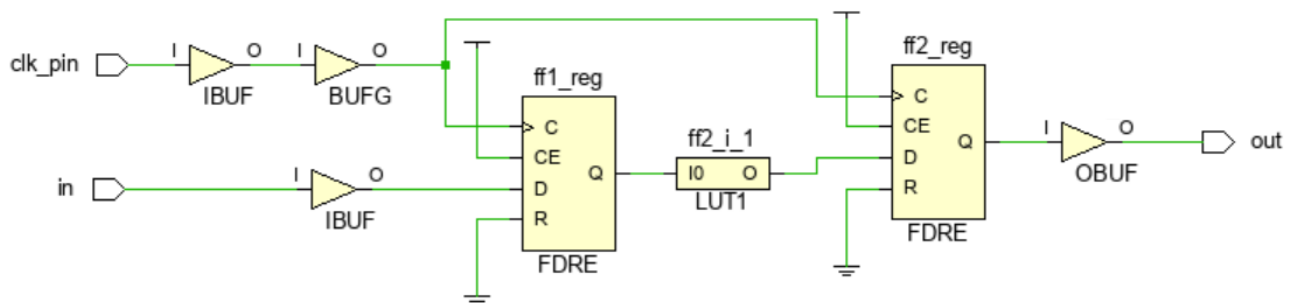


Рисунок 3. Схема проекта.

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).

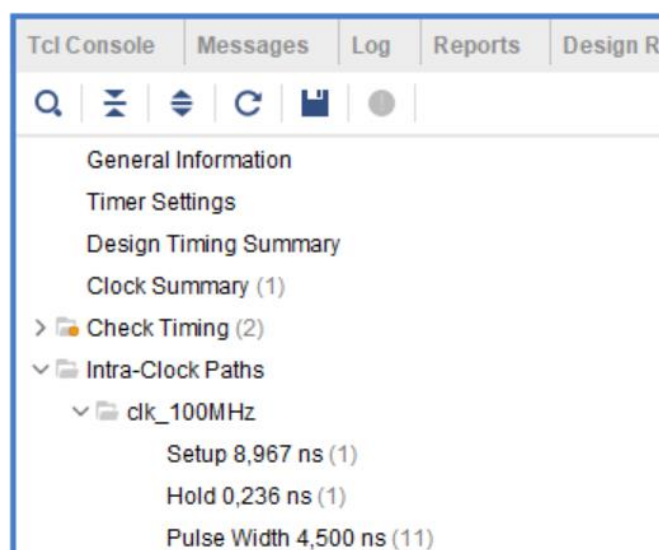


Рисунок 4. Разделы отчета Timing Summary.

5. Анализ ограничения по Setup.

Открыв вкладку Setup раздела Intra-Clock Path и дважды нажав на один 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 распространения 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.

Summary	
Name	Path 1
Slack	8.967ns
Source	ff1_reg/C (rising edge-triggered cell FDRE clocked by clk_100MHz)
Destination	ff2_reg/D (rising edge-triggered cell FDRE clocked by clk_100MHz)
Path Group	clk_100MHz
Path Type	Setup (Max at Slow Process Corner)
Requirement	10.000ns (clk_100MHz rise@10.000ns - clk_100MHz rise@0.000ns)
Data Path Delay	1.002ns (logic 0.580ns (57.912%) route 0.422ns (42.088%))
Logic Levels	1 (LUT1=1)
Clock Path Skew	-0.025ns
Clock Uncertainty	0.035ns

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 (T_{sca} из уравнения 1) составляет 4.708 нс. Символы (r) рядом со значениями задержек указывает, что анализируется фронт, а не спад тактового сигнала.

Source Clock Path				
Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock clk_10...Hz rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000	Site: N13	clk_pin
net (fo=0)	0.000	0.000		clk_pin
IBUF (Prop_ibuf I O)	(r) 1.001	1.001	Site: N13	clk_pin_IBUF_inst/O
net (fo=1, routed)	1.972	2.973		clk_pin_IBUF
BUFG (Prop_bufg I O)	(r) 0.096	3.069	Site: BUF...TRL_X0Y0	clk_pin_IBUF_BUFG_inst/O
net (fo=2, routed)	1.639	4.708		clk_pin_IBUF_BUFG
FDRE			Site: SLICE_X0Y0	ff1_reg/C

Рисунок 6. Задержка для запускающего фронта.

В разделе, представленном на рисунке 7, указаны задержки при распространении данных, которые состоят из задержки clock to output (T_{co}) для триггера FF1, задержки распространения через *nets* и LUT (T_{ilo}). 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 T_{sca} (4.708 нс., рисунок 6) и T_{co} (0.456 нс.). Конечное значение в столбце *Path* (5.710 нс.) показывает время прибытия данных *Arrival Time* ко входу защелкивающего триггера (T_{da} из уравнения 3).

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







Data Path				
Delay Type	Incr (ns)	Path ...	Location	Netlist Resourc...
FDRE (Prop fdre_C_Q)	(f) 0.456	5.164	Site: SLICE_X0Y0	 ff1_reg/Q
net (fo=1, routed)	0.422	5.586		 ff1
			Site: SLICE_X0Y1	 ff2_i_1/I0
LUT1 (Prop lut1_I0_O)	(r) 0.124	5.710	Site: SLICE_X0Y1	 ff2_i_1/O
net (fo=1, routed)	0.000	5.710		 p_0_in
FDRE			Site: SLICE_X0Y1	 ff2_reg/D
Arrival Time		5.710		

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. (T_{dca} из уравнения 4).

Destination Clock Path				
Delay Type	Incr (ns)	Path (...)	Location	Netlist Resource(s)
(clock clk_10...Hz rise edge)	(r) 10.000	10.000		
	(r) 0.000	10.000	Site: N13	clk_pin
net (fo=0)	0.000	10.000		clk_pin
			Site: N13	clk_pin_IBUF_inst/I
IBUF (Prop_ibuf I_O)	(r) 0.867	10.867	Site: N13	clk_pin_IBUF_inst/O
net (fo=1, routed)	1.868	12.735		clk_pin_IBUF
			Site: BUF...TRL_X0Y0	clk_pin_IBUF_BUFG_inst/I
BUFG (Prop_bufg I_O)	(r) 0.091	12.826	Site: BUF...TRL_X0Y0	clk_pin_IBUF_BUFG_inst/O
net (fo=2, routed)	1.520	14.347		clk_pin_IBUF_BUFG
FDRE			Site: SLICE_X0Y1	ff2_reg/C
clock pessimism	0.336	14.683		
clock uncertainty	-0.035	14.648		
FDRE (Setup_fdre C_D)	0.029	14.677	Site: SLICE_X0Y1	ff2_reg
Required Time		14.677		

Рисунок 8. Требуемое время прибытия данных.

Далее после строки с надписью FDRE показаны еще три задержки, которые представляют из себя время установки T_{su} для триггера FF2 (0.029 нс), а также пессимизм T_{cp} (*Clock Pessimism*, 0.336 нс.) и неопределенность T_{cu} (*Clock Uncertainty*, -0.035 нс.) тактового сигнала. О последних двух задержках будет рассказано далее в статье в пунктах 7 и 8.

Требуемое время прибытия данных T_{dr} рассчитывается с помощью уравнения 5. Если учитывать дополнительные слагаемые T_{cp} и T_{cu} , то уравнение для T_{dr} будет иметь вид

$$\begin{aligned}
 T_{dr} &= T_{dca} - T_{su} + T_{cu} + T_{cp} = \\
 &= 14.347 - 0.029 - 0.035 + 0.336 = 14.677.
 \end{aligned}$$

$Slack$ вычисляется по уравнению 6 и равен

$$Slack = 14.677 - 4.708 = 8.967,$$

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

6. 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.

Summary	
Name	Path 2
Slack (Hold)	0.236ns
Source	ff1_reg/C (rising edge-triggered cel...0.000ns fall@5.000ns period=10.000ns)
Destination	ff2_reg/D (rising edge-triggered cel...0.000ns fall@5.000ns period=10.000ns)
Path Group	clk_100MHz
Path Type	Hold (Min at Fast Process Corner)
Requirement	0.000ns (clk_100MHz rise@0.000ns - clk_100MHz rise@0.000ns)
Data Path Delay	0.343ns (logic 0.186ns (54.305%) route 0.157ns (45.695%))
Logic Levels	1 (LUT1=1)
Clock Path Skew	0.016ns

Рисунок 9. Общие сведения о пути

Source Clock Path				
Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock clk_10...Hz rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000	Site: N13	clk_pin
net (fo=0)	0.000	0.000		clk_pin
IBUF (Prop_ibuf_I_O)	(r) 0.230	0.230	Site: N13	clk_pin_IBUF_inst/O
net (fo=1, routed)	0.634	0.864		clk_pin_IBUF
BUFG (Prop_bufg_I_O)	(r) 0.026	0.890	Site: BUF...TRL_X0Y0	clk_pin_IBUF_BUFG_inst/O
net (fo=2, routed)	0.596	1.486		clk_pin_IBUF_BUFG
FDRE			Site: SLICE_X0Y0	ff1_reg/C
Data Path				
Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resourc...
FDRE (Prop_fdre_C_Q)	(f) 0.141	1.627	Site: SLICE_X0Y0	ff1_reg/Q
net (fo=1, routed)	0.157	1.783		ff1
LUT1 (Prop_lut1_I_O)	(r) 0.045	1.828	Site: SLICE_X0Y1	ff2_i_1/O
net (fo=1, routed)	0.000	1.828		p_0_in
FDRE			Site: SLICE_X0Y1	ff2_reg/D
Arrival Time		1.828		

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 T_{cu} (*Clock Uncertainty*). Причина будет раскрыта далее в статье в пункте 7.

$Slack$ рассчитывается, используя уравнение 10, и равен

$$Slack = T_{da} - T_{dr} = 1.828 - 1.593 = 0.235.$$

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 распространяться максимально долго, что соответствует T_{dpd_max} в уравнении 2. Для анализа по $Hold$ самый худший случай – это максимально быстрое распространение данных, то есть T_{dpd_min} в уравнении 9.









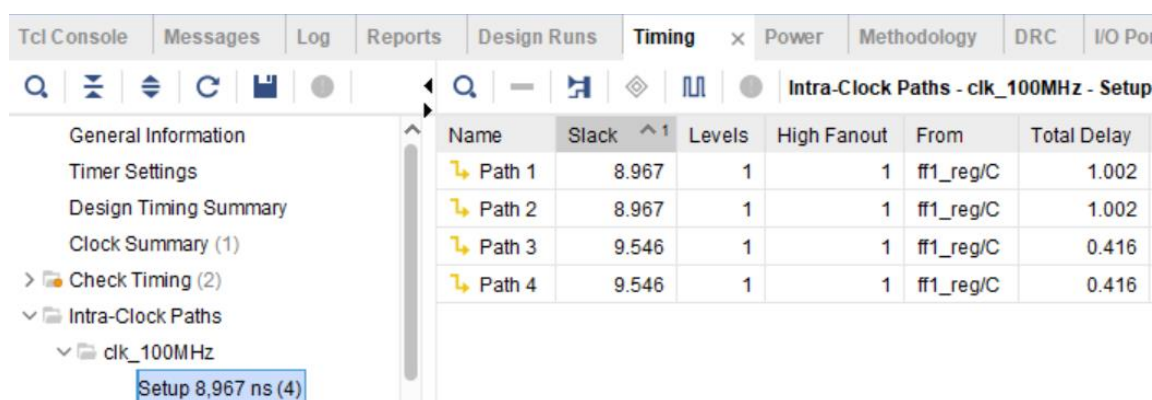
Destination Clock Path				
Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock clk_10...Hz rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000	Site: N13	 clk_pin
net (fo=0)	0.000	0.000		 clk_pin
IBUF (Prop_ibuf I O)	(r) 0.419	0.419	Site: N13	 clk_pin_IBUF_inst/O
net (fo=1, routed)	0.689	1.108		 clk_pin_IBUF
BUFG (Prop_bufg I O)	(r) 0.029	1.137	Site: BUF...TRL_X0Y0	 clk_pin_IBUF_BUFG_inst/O
net (fo=2, routed)	0.867	2.003		 clk_pin_IBUF_BUFG
FDRE			Site: SLICE_X0Y1	 ff2_reg/C
clock pessimism	-0.502	1.502		
FDRE (Hold fdre C D)	0.091	1.593	Site: SLICE_X0Y1	 ff2_reg
Required Time		1.593		

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.



Name	Slack	Levels	High Fanout	From	Total Delay
Path 1	8.967	1	1	ff1_reg/C	1.002
Path 2	8.967	1	1	ff1_reg/C	1.002
Path 3	9.546	1	1	ff1_reg/C	0.416
Path 4	9.546	1	1	ff1_reg/C	0.416

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 – с '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.

7. 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

$$\Delta T = T_{clk} \pm T_j,$$

где T_{clk} – как и раньше период тактового сигнала, T_j – случайная величина с нулевым средним, обуславливающая неопределенность тактового сигнала.

Слагаемое T_{cu} (*Clock Uncertainty*), 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 (рисунок 13).

Уравнение для расчета неопределённости имеет вид [2]:

$$T_{cu} = \frac{\sqrt{T_{tsj}^2 + T_{tij}^2} + T_{dj}}{2} + T_{pe}, \quad (12)$$

где

- T_{tsj} (*Total System Jitter*) – общий джиттер, присутствующий на тактовом входе каждого последовательного элемента схемы;
- T_{tij} (*Total Input Jitter*) – дополнительный джиттер для конкретного тактового сигнала;
- T_{dj} (*Discrete Jitter*) – джиттер тактового сигнала, формируемого блоками PLL или MMCM;
- T_{pe} (*Phase Error*) – постоянная расфазировка тактового сигнала, формируемого блоками PLL или MMCM, относительно опорного сигнала.

Clock Uncertainty Equation	
((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE	
Total System Jitter (TSJ)	0.071ns
Total Input Jitter (TIJ)	0.000ns
Discrete Jitter (DJ)	0.000ns
Phase Error (PE)	0.000ns

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 (*System Jitter*) задается в наносекундах, а величина общего джиттера $Ttsj$ рассчитывается по формуле:

$$Ttsj = \sqrt{Tsj^2 + Tsj^2} = \sqrt{2} Tsj.$$

На рисунке 14, в качестве примера, представлен результат команды

```
set_system_jitter 0.15
```

Clock Path Skew	-0.025ns
Clock Uncertainty	0.106ns
Source Clock Path	<div> <div>Clock Uncertainty Equation</div> <div> <div>((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE</div> <div>Total System Jitter (TSJ) 0.212ns</div> <div>Total Input Jitter (TIJ) 0.000ns</div> <div>Discrete Jitter (DJ) 0.000ns</div> <div>Phase Error (PE) 0.000ns</div> </div> </div>
Delay Type	
(clock clk_10...Hz rise edge)	
net (fo=0)	
IBUF (Prop ibuf I O)	
net (fo=1, routed)	

Рисунок 14. Настройка общего джиттера.

Как и ожидалось $Ttsj = \sqrt{2} \cdot 0.15 = 0.212$ нс. и $Tcu = Ttsj/2 = 0.106$ 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 T_{ij} (*Input Jitter*) задается в наносекундах. Значение дополнительного джиттера для тактового сигнала равно $T_{tij} = T_{ij}$, если запускающий и защёлкивающий фронты относятся к одному тактовому сигналу, и $T_{tij} = \sqrt{T_{ij1}^2 + T_{ij2}^2}$ – 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
```

Clock Path Skew	-0.025ns
Clock Uncertainty	0.117ns
Source Clock Path	
Delay Type	In
(clock clk_10...Hz rise edge)	(
net (fo=0)	(
IBUF (Prop_ibuf I O)	(
net (fo=1, routed)	(
BUFG (Prop_bufg I O)	(

Clock Uncertainty Equation		
((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE		
Total System Jitter (TSJ)	0.212ns	
Total Input Jitter (TIJ)	0.100ns	
Discrete Jitter (DJ)	0.000ns	
Phase Error (PE)	0.000ns	

Рисунок 15. Настройка неопределенности тактового сигнала.

Подставив заданные значения в уравнение 12, получим:

$$T_{cu} = \frac{\sqrt{0.212^2 + 0.1^2}}{2} = \frac{\sqrt{0.054944}}{2} = 0.117.$$

Обобщая все выше сказанное, получаем, что из-за наличия джиттера интервал между двумя последовательными фронтами тактового сигнала отличается от T_{clk} 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 T_{cu} . 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 T_{cu} будет присутствовать в анализе по *Hold*.

8. Устранение пессимизма для задержки тактового сигнала.

Вернемся к анализу по *Hold*, чтобы выясним смысл последнего не рассмотренного слагаемого T_{cp} (*Clock Pessimism*) [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 T_{cp} . 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 T_{cp} , получаем

$$T_{cp} = 0.89 - 1.137 = -0.247.$$

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.

Source Clock Path				
Delay Type	Incr (ns)	Path (ns)	Loca...	Netlist Resource(s)
(clock clk_10...Hz rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000		clk_pin
net (fo=0)	0.000	0.000		clk_pin
				clk_pin_IBUF_inst/I
IBUF (Prop_ibuf I O)	(r) 0.165	0.165		clk_pin_IBUF_inst/O
net (fo=1, unplaced)	0.337	0.502		clk_pin_IBUF
				clk_pin_IBUF_BUFG_inst/I
BUFG (Prop_bufg I O)	(r) 0.026	0.528		clk_pin_IBUF_BUFG_inst/O
net (fo=2, unplaced)	0.114	0.642		clk_pin_IBUF_BUFG
FDRE				ff1_reg/C
Destination Clock Path				
Delay Type	Incr (ns)	Path (ns)	Loca...	Netlist Resource(s)
(clock clk_10...Hz rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000		clk_pin
net (fo=0)	0.000	0.000		clk_pin
				clk_pin_IBUF_inst/I
IBUF (Prop_ibuf I O)	(r) 0.353	0.353		clk_pin_IBUF_inst/O
net (fo=1, unplaced)	0.355	0.708		clk_pin_IBUF
				clk_pin_IBUF_BUFG_inst/I
BUFG (Prop_bufg I O)	(r) 0.029	0.737		clk_pin_IBUF_BUFG_inst/O
net (fo=2, unplaced)	0.259	0.996		clk_pin_IBUF_BUFG
FDRE				ff2_reg/C
clock pessimism	-0.209	0.787		
FDRE (Hold fdre C D)	0.091	0.878		ff2_reg
Required Time		0.878		

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

$$T_{cp} = 0.528 - 0.737 = -0.209.$$

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

$$\begin{aligned} Tskew &= T_{dcd} - T_{scd} + T_{cp} = \\ &= 2.003 - 1.486 - 0.502 = 0.015. \end{aligned}$$

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