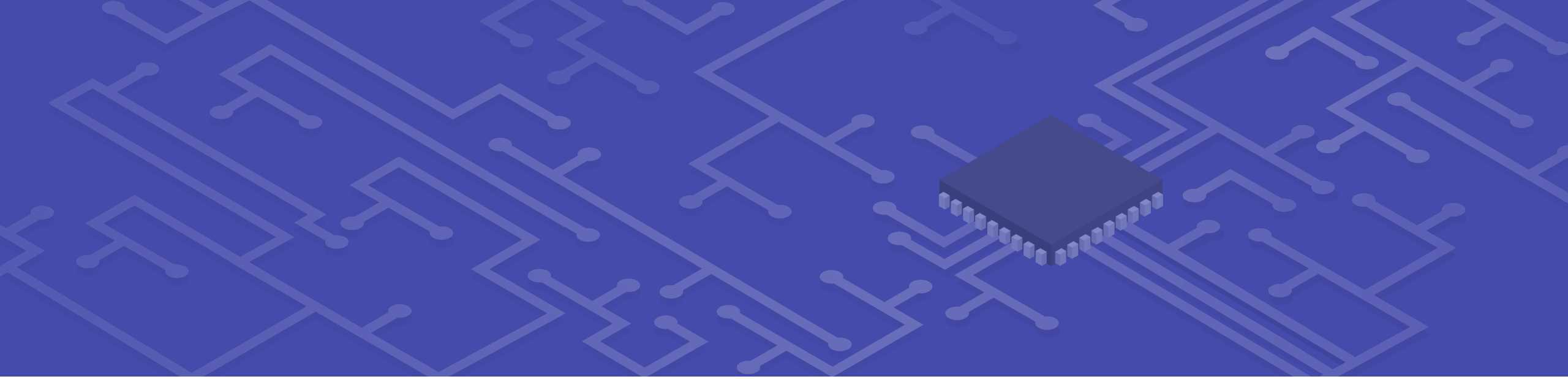
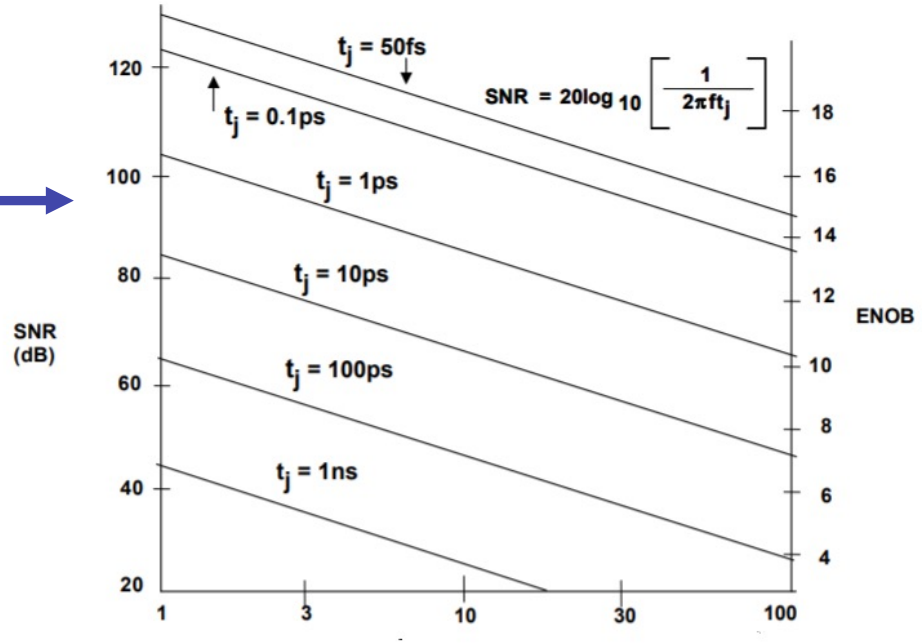


Часть восьмая

ПЛИС МОЖЕТ ВСЁ

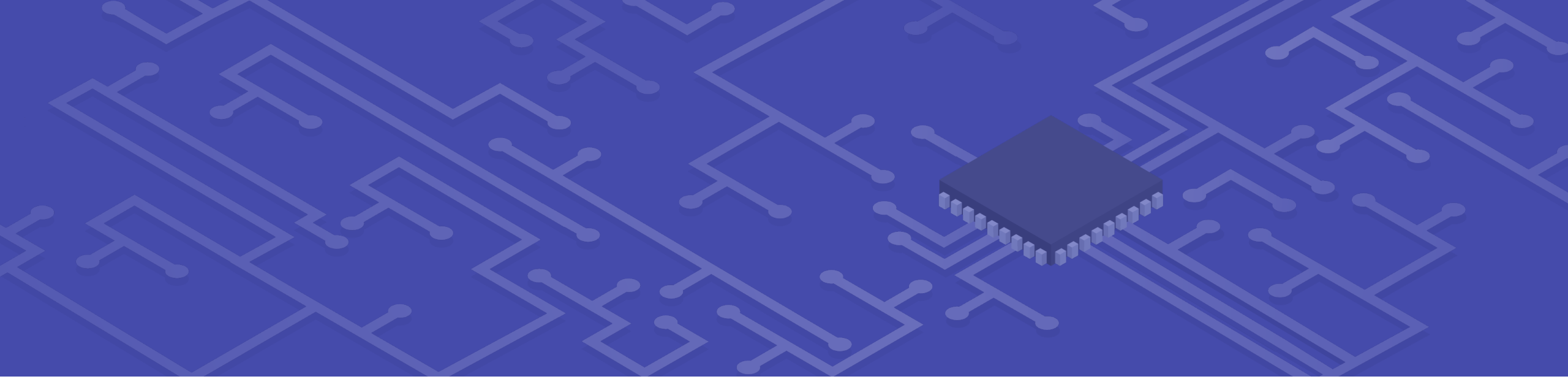
В каком качестве и когда нелогично использовать ПЛИС

- ПЛИС как источник тактового сигнала (особенно для быстрых многобитных АЦП)



- Если может справиться контроллер, (часто) лучше использовать его – редкие вычисления сложных функций, алгоритмы сортировки…

Kester W. Understand SINAD, ENOB, SNR, THD, THD+ N, and SFDR so you don’t get lost in the noise floor //MT-003 Tutorial, www.analog.com/static/importedfiles/tutorials/MT-003. pdf. – 2009.



Часть девятая

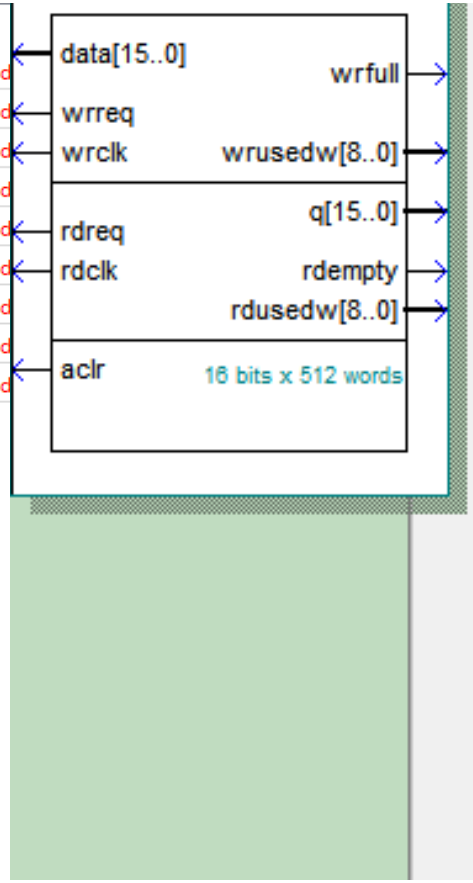
ИСТОРИЯ О СТАРОМ ПРОЕКТЕ, КОТОРЫЙ НУЖНО БЫЛО ВОССТАНОВИТЬ

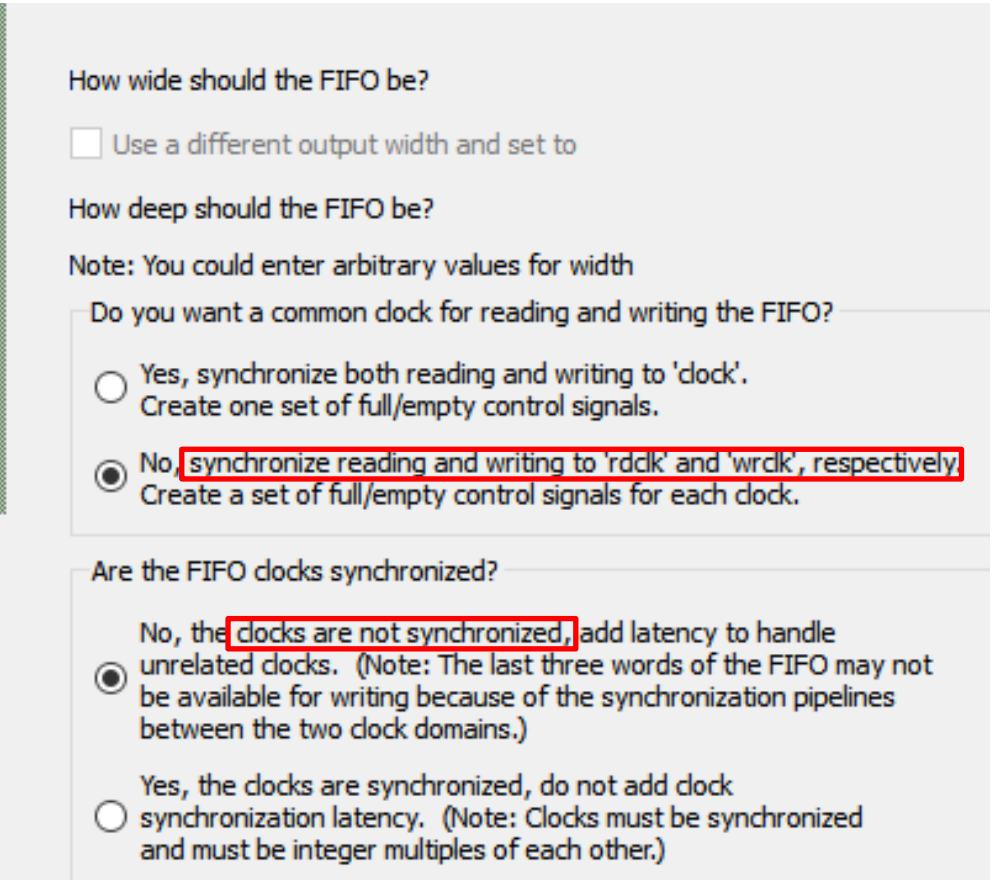
Завязка байки

- Есть старый проект, написанный неопытными разработчиками и переживший **много** редакций
- Видим провалы по setup, проваливаются пути **внутри IP Core Altera DCFIFO (!)**
- Провал в переносе между **асинхронными** клоковыми доменами 25 МГц и 60 МГц

Slack	From Node	
-3.691	Sdram_Control_4Port:u7|Sdram_FI..._a4o1:auto_generated|rdptr_g[0]	Sdram_Control_4Port:u7|Sd
-3.683	Sdram_Control_4Port:u7|Sdram_FI..._a4o1:auto_generated|rdptr_g[0]	Sdram_Control_4Port:u7|Sd
-3.683	Sdram_Control_4Port:u7|Sdram_FI..._a4o1:auto_generated|rdptr_g[4]	Sdram_Control_4Port:u7|Sd
-3.680	Sdram_Control_4Port:u7|Sdram_FI..._a4o1:auto_generated|rdptr_g[4]	Sdram_Control_4Port:u7|Sd
-3.676	Sdram_Control_4Port:u7|Sdram_FI..._a4o1:auto_generated|rdptr_g[9]	Sdram_Control_4Port:u7|Sd
-3.675	Sdram_Control_4Port:u7|Sdram_FI..._a4o1:auto_generated|rdptr_g[5]	Sdram_Control_4Port:u7|Sd

- DCFIFO, слово 16 бит, все настройки установлены корректно для работы с **асинхронными** клоками



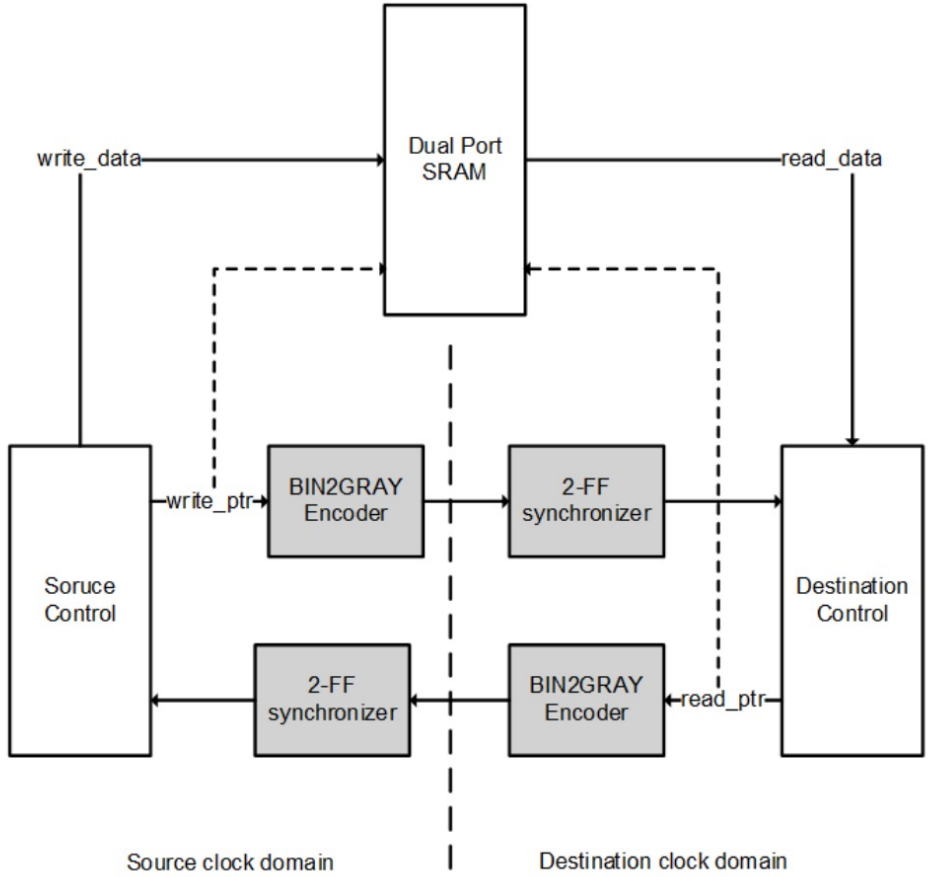


Немного о CDC – базовые вещи

МЕТОДОЛОГИЯ

1. Определяем свой случай (разная частота, полностью асинхронные домены и т.д.)
2. Реализуем нужную схему синхронизации
3. Прописываем констрейты
4. Проверяем правильность действий в разделе Report Clock Interaction / CDC Viewer
5. Проверяем корректность работы на реальной плате

Class	$\Delta\phi$	Δf	Synchronizer
Synchronous	0	0	None
Mesochronous	ϕ_c	**0**	**Phase compensation**
Multi-synchronous	drifts	0	Adaptive phase compensation
Plesiochronous	Varies	$f_d<\varepsilon$	Adaptive phase compensation
Periodic		$f_d>\varepsilon$	Predictive
Asynchronous			**Two-Flop**



Асинхронный CDC на основе FIFO

Виды CDC clock groups

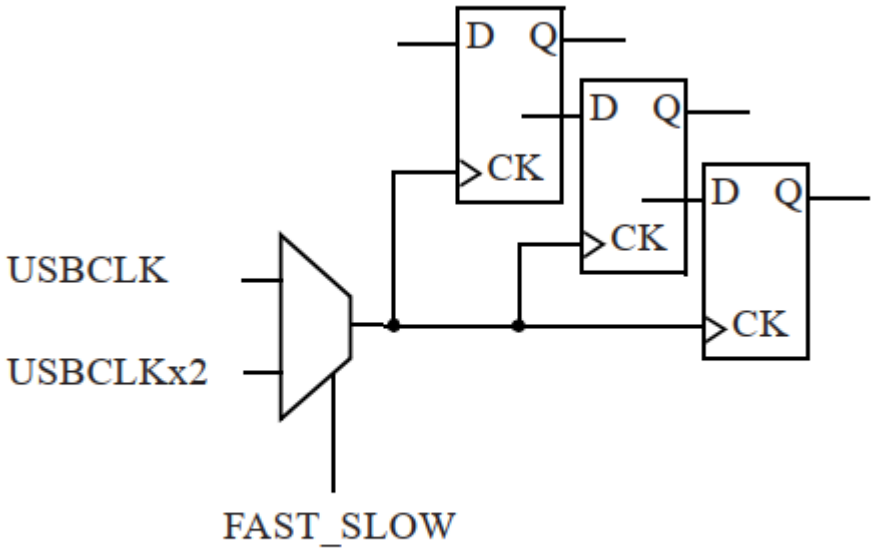
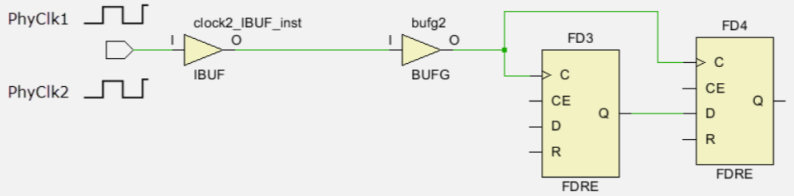


Figure 2-22 *Mutually-exclusive clocks.*

Асинхронные клоковые группы
(Asynchronous Clock Domain Crossings)

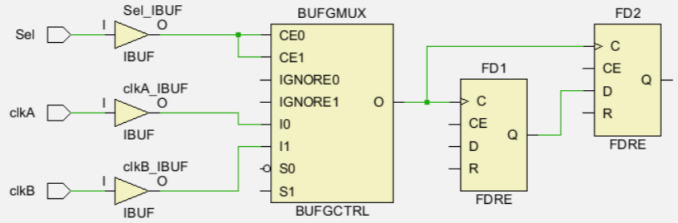
Physically Exclusive Clock Groups

Physically exclusive clocks are clocks that are defined on the same source point and propagate on the same clock tree. Figure 2-29 shows an example where two primary clocks are defined on the same input port.



Logically Exclusive Clock Groups with no Interaction

Logically exclusive clocks are clocks that are defined on different source points but share part of their clock tree due to a multiplexer or other combinational logic. The Timing Constraints wizard identifies such clocks and recommends a clock groups constraint directly on them when they do not have timing paths between each other except for the logic connected to their shared clock tree. Figure 2-31 shows an example of two clocks, `clkA` and `clkB`, which are defined on different input ports and start overlapping on the output of a `BUFGMUX`.



Особенности у Xilinx / Altera

- Вместо set_false_path : set_max_delay –datapath_only
  (В этом случае set_false_path убираем, т.к. он имеет больший приоритет и аннулирует set_max_delay!)
  - Другие варианты – set_clock_groups, set_max_skew, set_data_delay (Altera)
- Атрибут ASYNC_REG (XILINX) – assignment Synchronizer_Identification (Altera)
- Наличие инструментов – Report Clock Interaction и Report CDC в Xilinx
  - CDC Viewer у Altera только в Pro версии
- У Xilinx нет расчёта settling time, MTBF кроме Ultrascale

https://support.xilinx.com/s/article/794116?language=en_US ; https://www.youtube.com/watch?v=BNiXxmQlCCs

Расследование

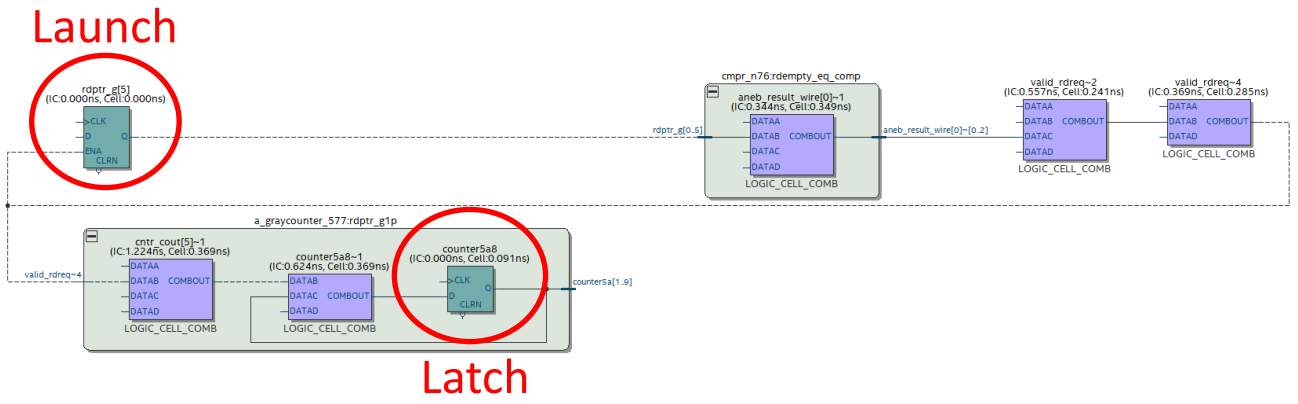
- Altera позаботилась* о том, чтобы Timing Analyzer не ругался на асинхронность DCFIFO: Автоматически устанавливаются **set_false_path** (их можно увидеть в выдаче Report SDC)

	SDC Command	Flags	From Flags	From	Through	To Flags	To	Location
1	set_false_path		-from	[get_keepers {*rdptr_g*}]		-to	[get_keepers {*ws_dgrp|dffpipe_se9:dffpipe8|dffe9a*}]	HDL-embedded SDC_STATEMENT
2	set_false_path		-from	[get_keepers {*delayed_wrptr_g*}]		-to	[get_keepers {*rs_dgwp|dffpipe_re9:dffpipe5|dffe6a*}]	HDL-embedded SDC_STATEMENT

- Пути не перегружены – clock и data path 8,236 нс, приемлемо для наших частот

	Total	Incr	RF	Type	Fanout	Location	
	500060.000	500060.000					launch edge time
>	500063.215	3.215					clock path
>	500068.236	5.021					data path

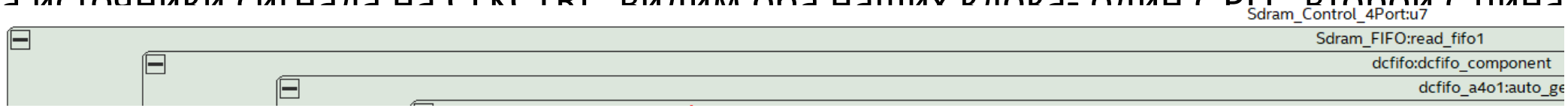
- В схеме также нет криминала



*DCFIFO Timing Constraints // FPGA Wiki, Intel community. https://community.intel.com/t5/FPGA-Wiki/DCFIFO-Timing-Constraints/ta-p/735793

Схема тактирования

- Смотрим на источники тактовых сигналов запускающего и защелкивающего триггеров
- Смотрим на источники сигнала на CLKCTRL - видим оба наших клока- один с PLL, второй с пина



Data Arrival Path

	Total	Incr	RF	Type	Fanout	Location	
1	500060.000	500060.000					launch edge time
2	500063.358	3.358					clock path
1	500060.000	0.000					source latency
2	500060.000	0.000			1	PIN_R8	CLOCK_50
3	500060.000	0.000	RR	IC	1	IOIBUF_X27_Y0_N22	CLOCK_50~input|i
4	500060.479	0.479	RR	CELL	3	IOIBUF_X27_Y0_N22	CLOCK_50~input|o
5	500062.357	1.878	RR	IC	1	PLL_4	u8|altpll_component|a
6	500056.821	-5.536	RR	COMP	3	PLL_4	u8|altpll_component|a
7	500056.821	0.000	RR	CELL	1	PLL_4	u8|altpll_component|a
8	500058.699	1.878	FF	IC	1	CLKCTRL_G19	u8|altpll_component|a
9	500058.699	0.000	FF	CELL	860	CLKCTRL_G19	u8|altpll_component|a
10	500060.096	1.397	FF	IC	1	LCCOMB_X27_Y15_N2	comb~1|dataa
11	500060.452	0.356	FR	CELL	1	LCCOMB_X27_Y15_N2	comb~1|combout
12	500061.829	1.377	RR	IC	1	CLKCTRL_G7	comb~1clkctrl|inclk[0]
13	500061.829	0.000	RR	CELL	45	CLKCTRL_G7	comb~1clkctrl|outclk
14	500062.839	1.010	RR	IC	1	FF_X37_Y18_N11	u7|read_fifo1|dcfifo_c
15	500063.358	0.519	RR	CELL	1	FF_X37_Y18_N11	Sdram_Control_4Port:

PROM_PLL:u8

Data Required Path

	Total	Incr	RF	Type	Fanout	Location	
1	500060.001	500060.001					latch edge time
2	500064.831	4.830					clock path
1	500060.001	0.000					source latency
2	500060.001	0.000			1	PIN_T10	GPIO_1[8]
3	500060.001	0.000	RR	IC	1	IOIBUF_X34_Y0_N15	GPIO_1[8]~input|i
4	500060.480	0.479	RR	CELL	3	IOIBUF_X34_Y0_N15	GPIO_1[8]~input|o
5	500061.934	1.454	RR	IC	1	LCCOMB_X27_Y15_N2	comb~1|datad
6	500062.059	0.125	RR	CELL	1	LCCOMB_X27_Y15_N2	comb~1|combout
7	500063.381	1.322	RR	IC	1	CLKCTRL_G7	comb~1clkctrl|inclk
8	500063.381	0.000	RR	CELL	45	CLKCTRL_G7	comb~1clkctrl|outc
9	500064.350	0.969	RR	IC	1	FF_X37_Y19_N7	u7|read_fifo1|dcfifo
10	500064.831	0.481	RR	CELL	1	FF_X37_Y19_N7	Sdram_Control_4P

Почему и как не стоит мультиплексировать клоки

- **Вообще,** не стоит (по возможности) мультиплексировать тактовые сигналы: это добавляет лишний уровень сложности для разработчиков и движка среды разработки (в контексте **CDC**)

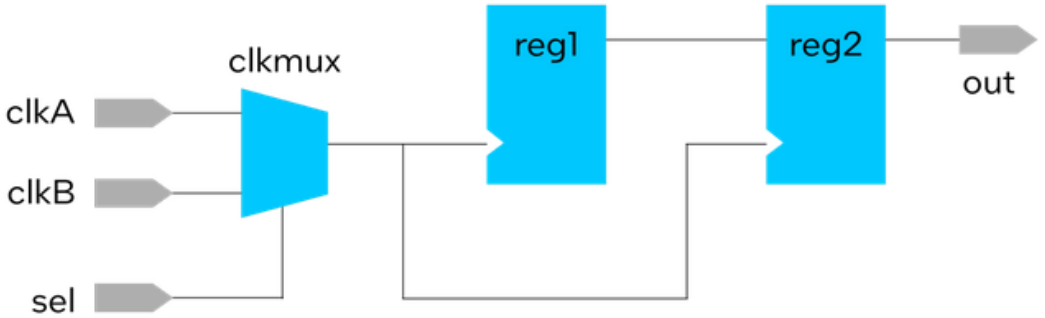
- Если всё же надо - не стоит мультиплексировать тактовые сигналы обычной, не специализированной **логикой**: это весьма тяжело корректно реализовать, а результирующая схема будет значительно сложнее

Мультиплексирование клоков

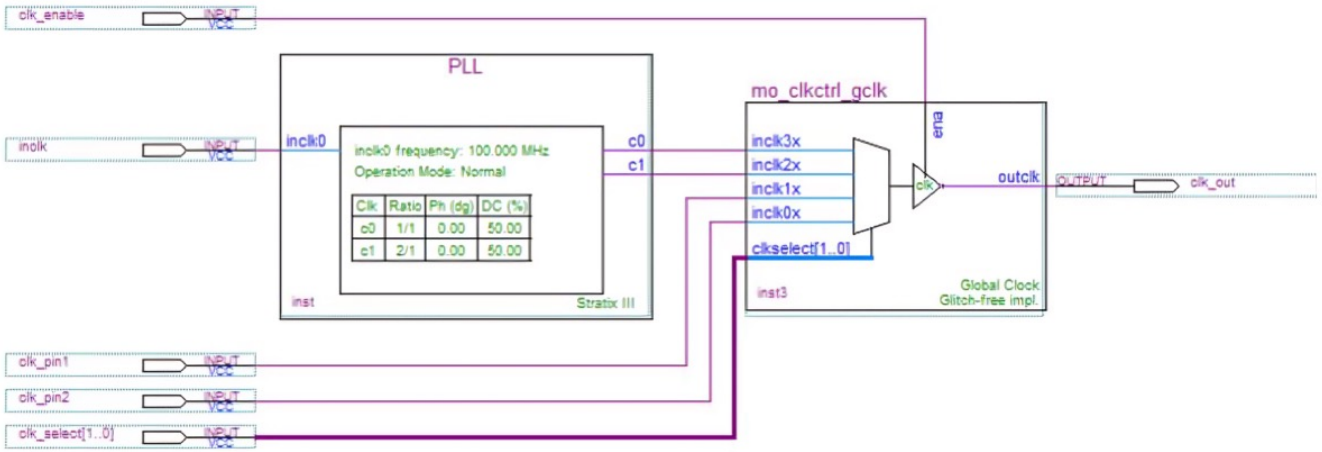
Как действовать, если всё-таки **нужно** переключать клоки?

- *Объявить, что клоки **взаимоисключающие**

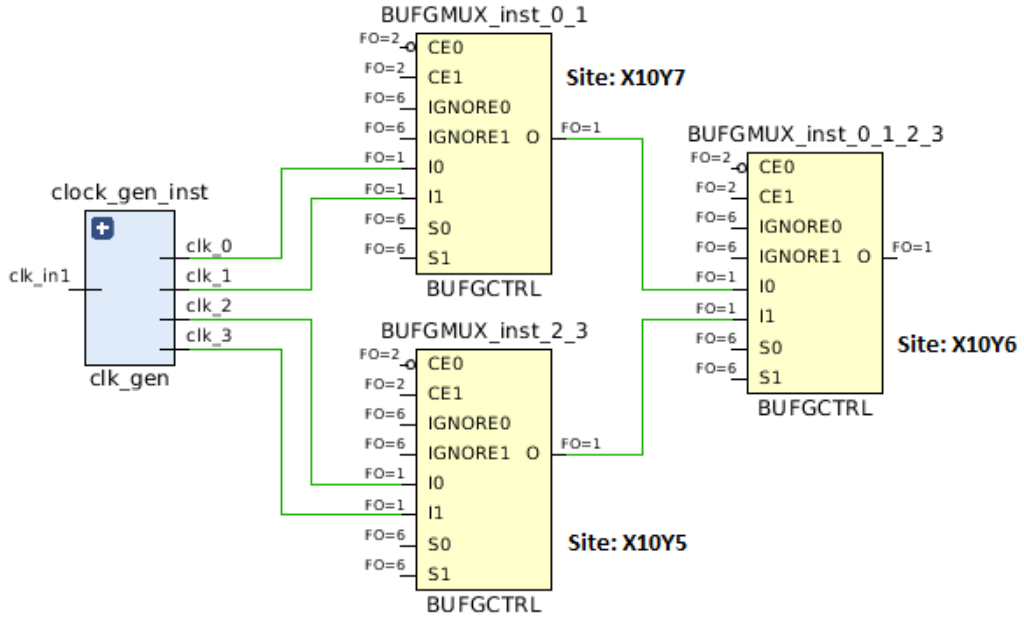
```
# Create the two clocks on the port
create_clock -name clk_100 -period 10 [get_ports clkA]
create_clock -name clk_125 -period 8 [get_ports clkB] -add
# Set the two clocks as exclusive clocks
set_clock_groups -exclusive -group {clk_100} -group {clk_125}
```



- Включать в схему элементы контроля клоков, вызывая их при помощи IP Core в явном виде

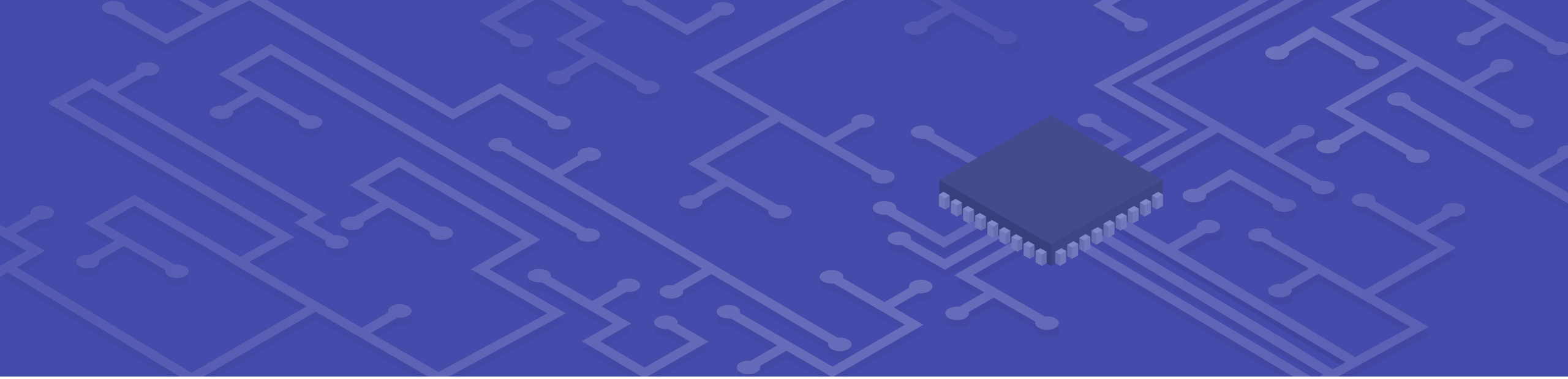


Пример ALTCLKCTRL Altera



Пример BUFGCTRL Xilinx

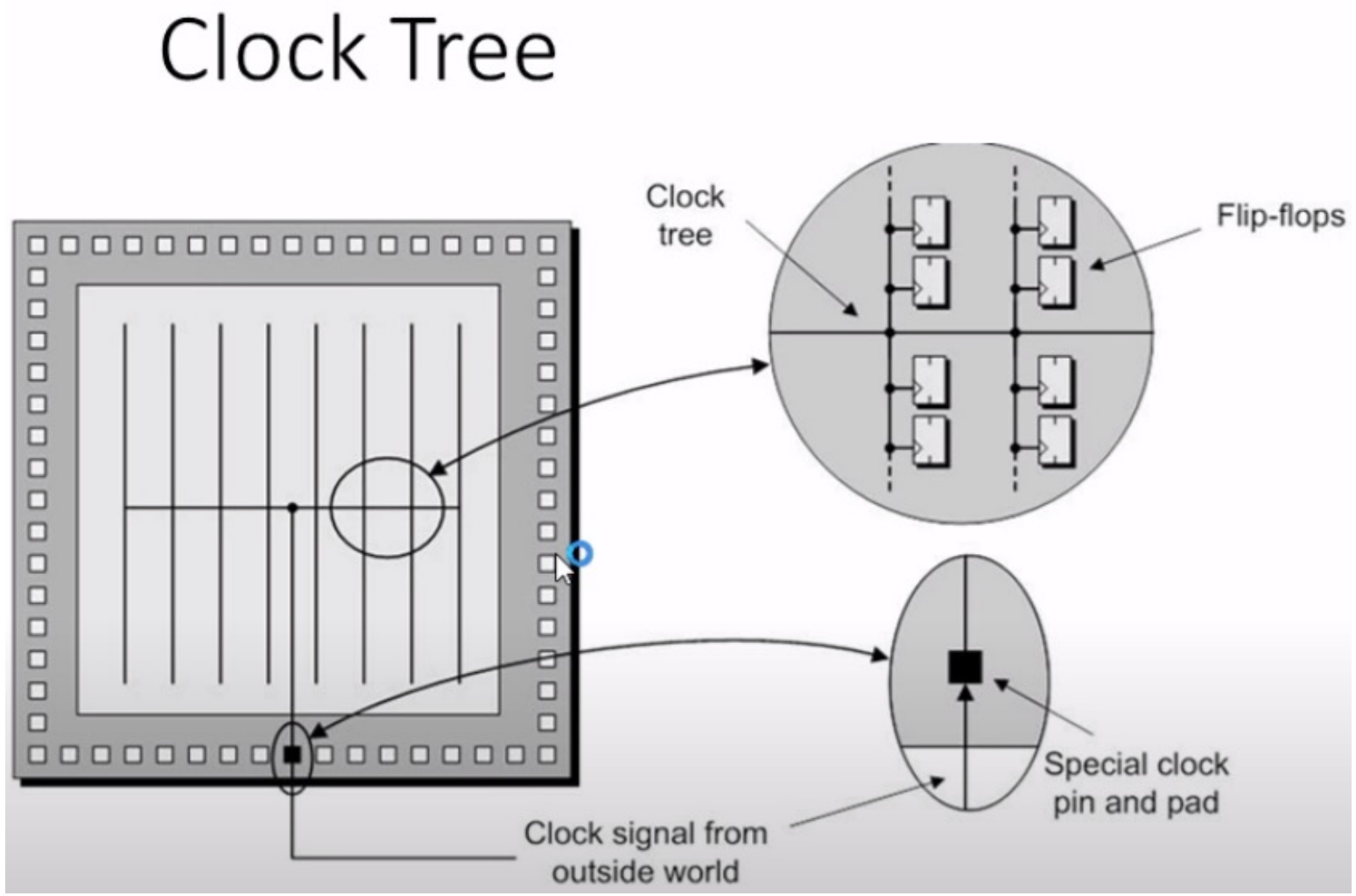
*Timing Analyzer Clock Multiplexer Examples https://www.intel.com/content/www/us/en/support/programmable/support-resources/design-examples/quartus/exm-tq-clock-mux.html



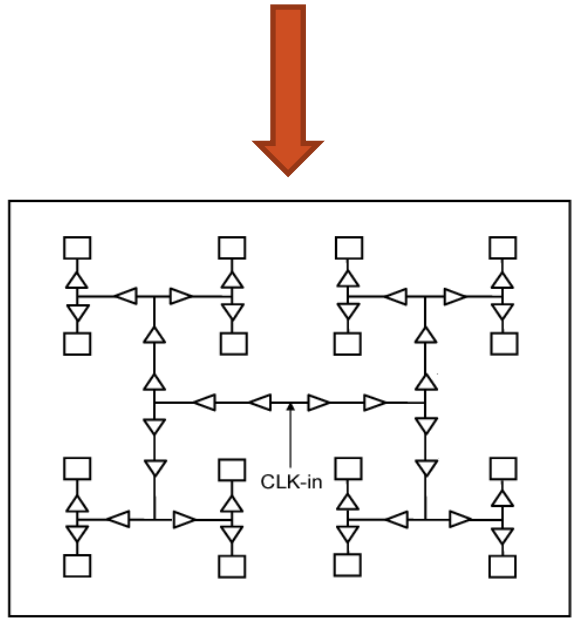
Часть десятая

УЧЁТ АРХИТЕКТУРЫ ТАКТОВЫХ СИГНАЛОВ

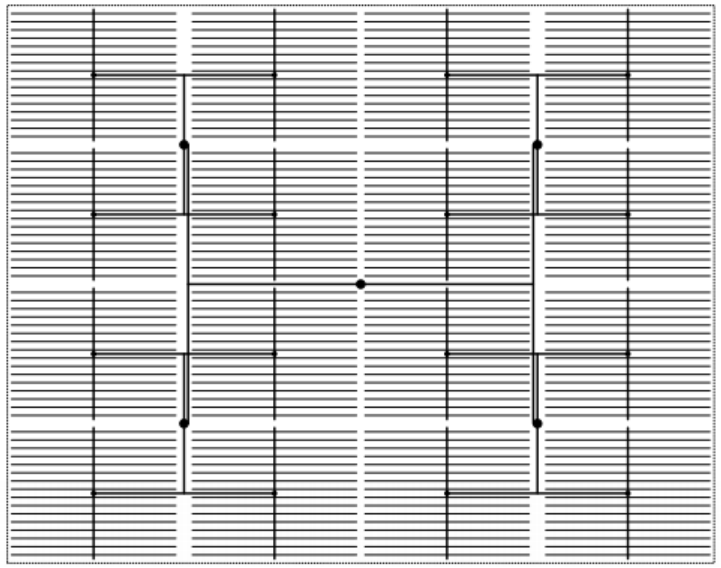
Архитектура тактовых сигналов – базовые сведения



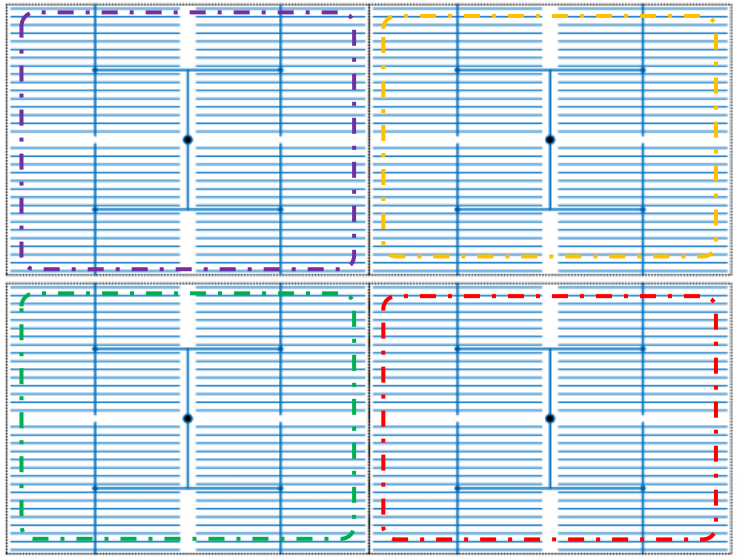
- Классический вид клокового древа в ПЛИС
- Проблема на первый взгляд – clock skew.
- Ideal final result – достижение клоком всех триггеров одновременно



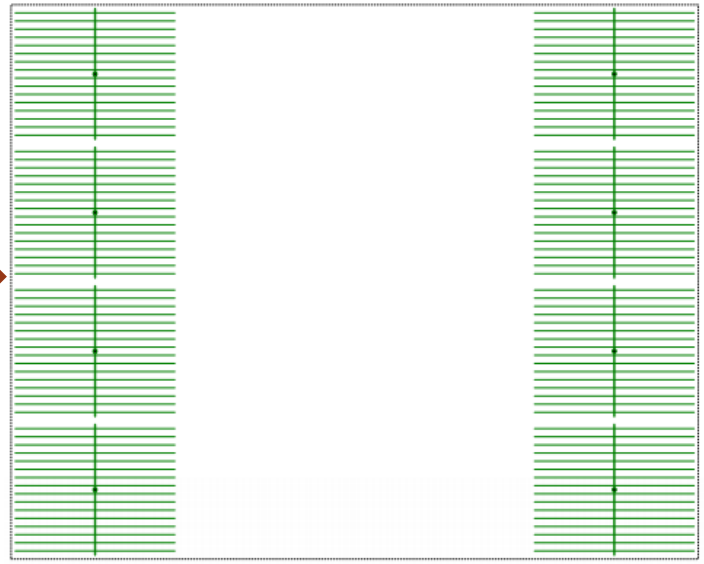
Архитектура тактовых сигналов – менее базовые сведения



Глобальные линии (Altera GCLK)

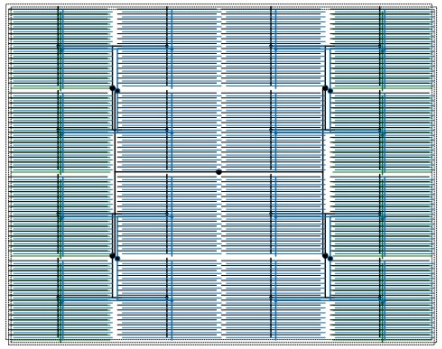


Региональные линии (Altera RCLK)

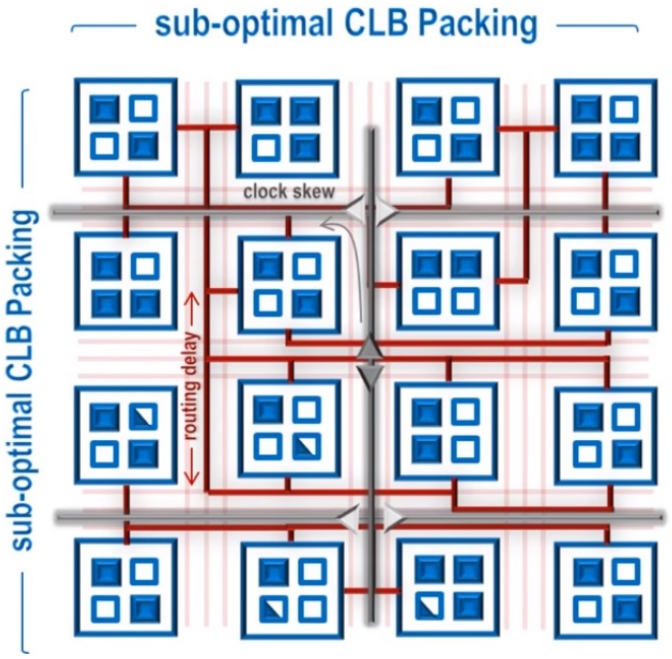


Периферийные линии (Altera PCLK)

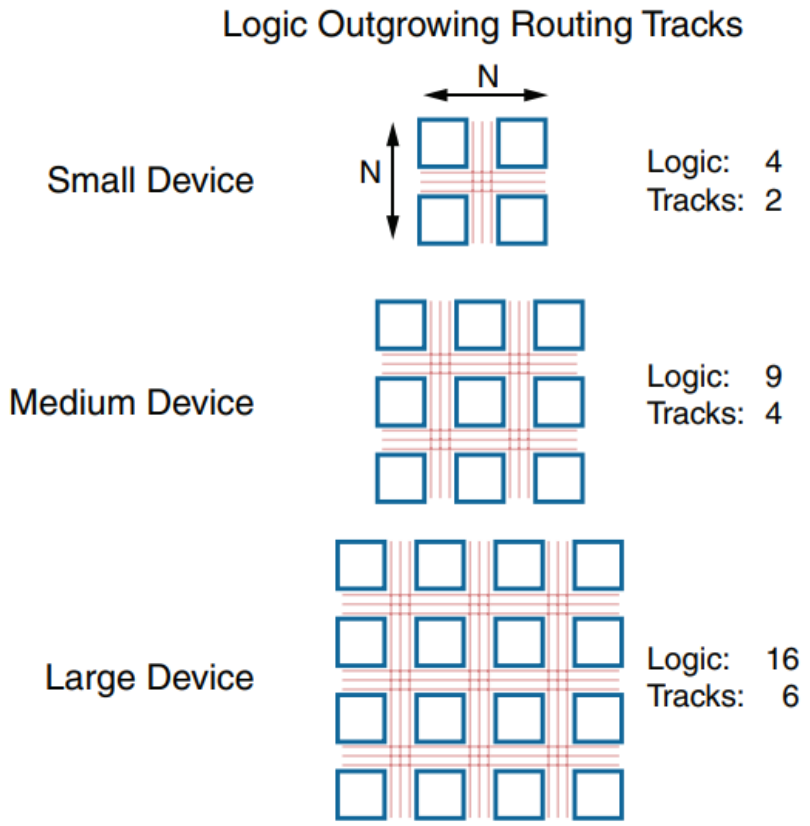
Все три представлены в итоговой архитектуре м/сх



Архитектура тактовых сигналов – проблематика



- Routing delay dominates overall delay
- Clock skew consumes more timing margin
- Sub-optimal CLB packing reduces performance and utilization



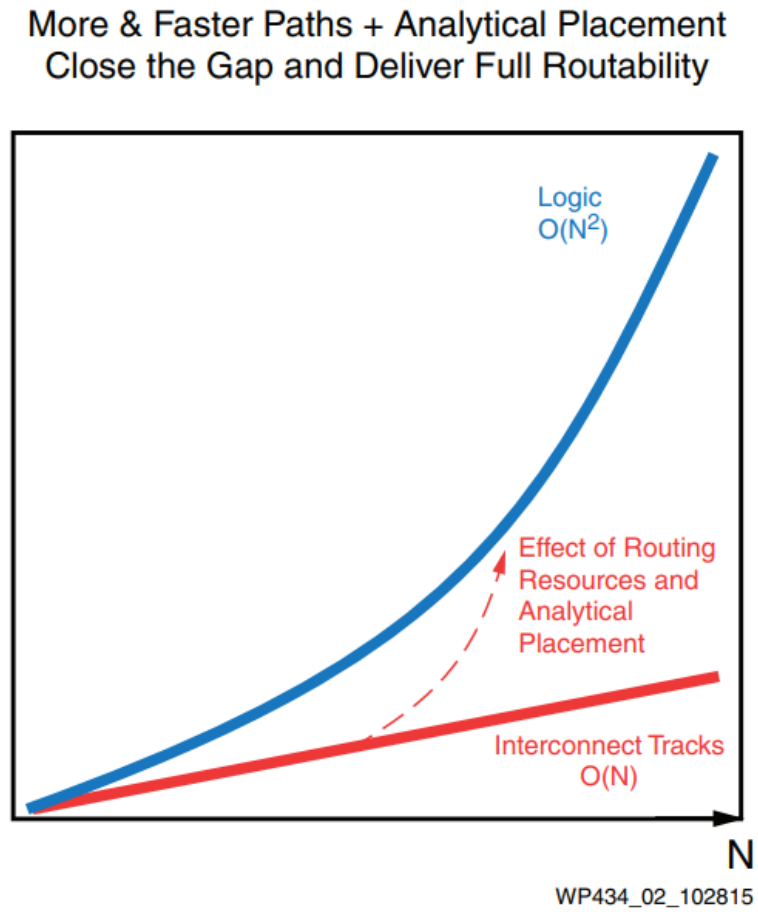
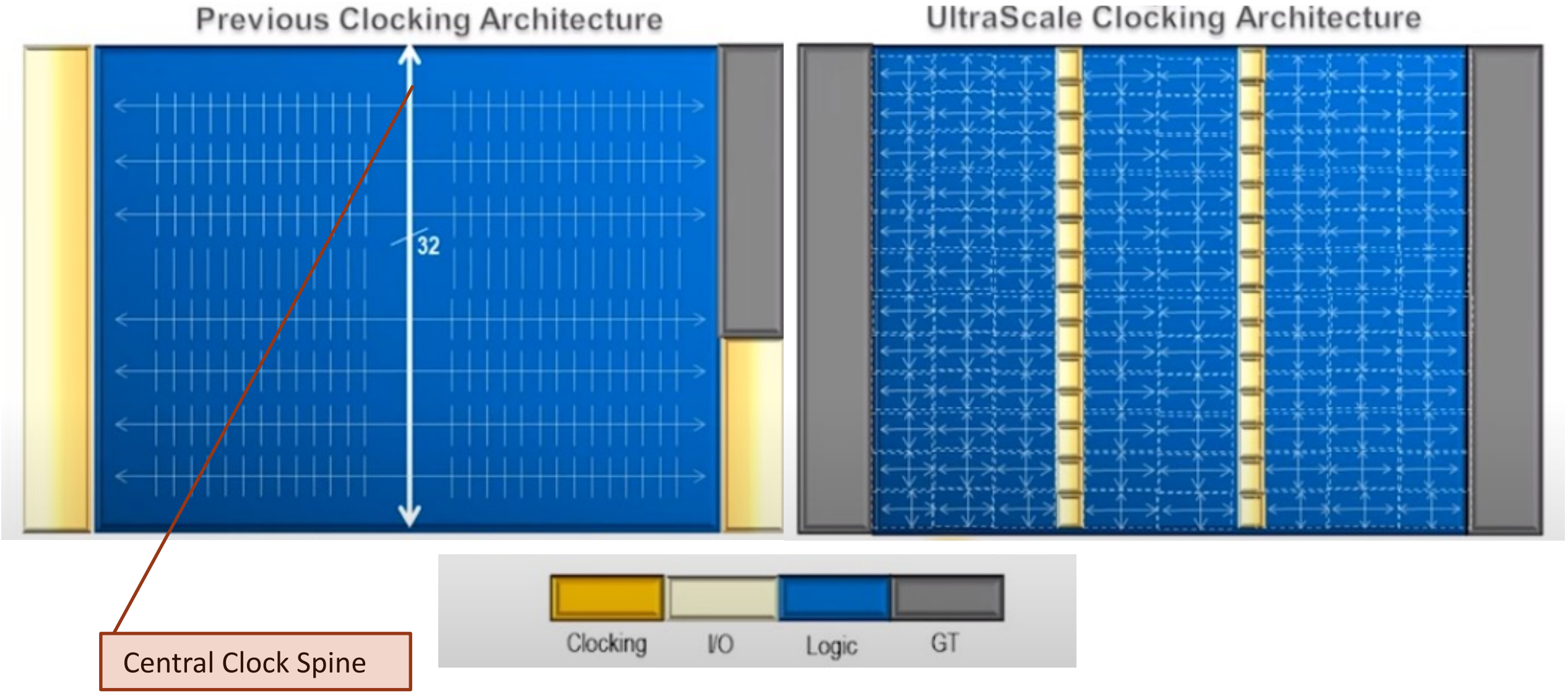


Figure 2: **Adding Routing in the UltraScale Architecture**

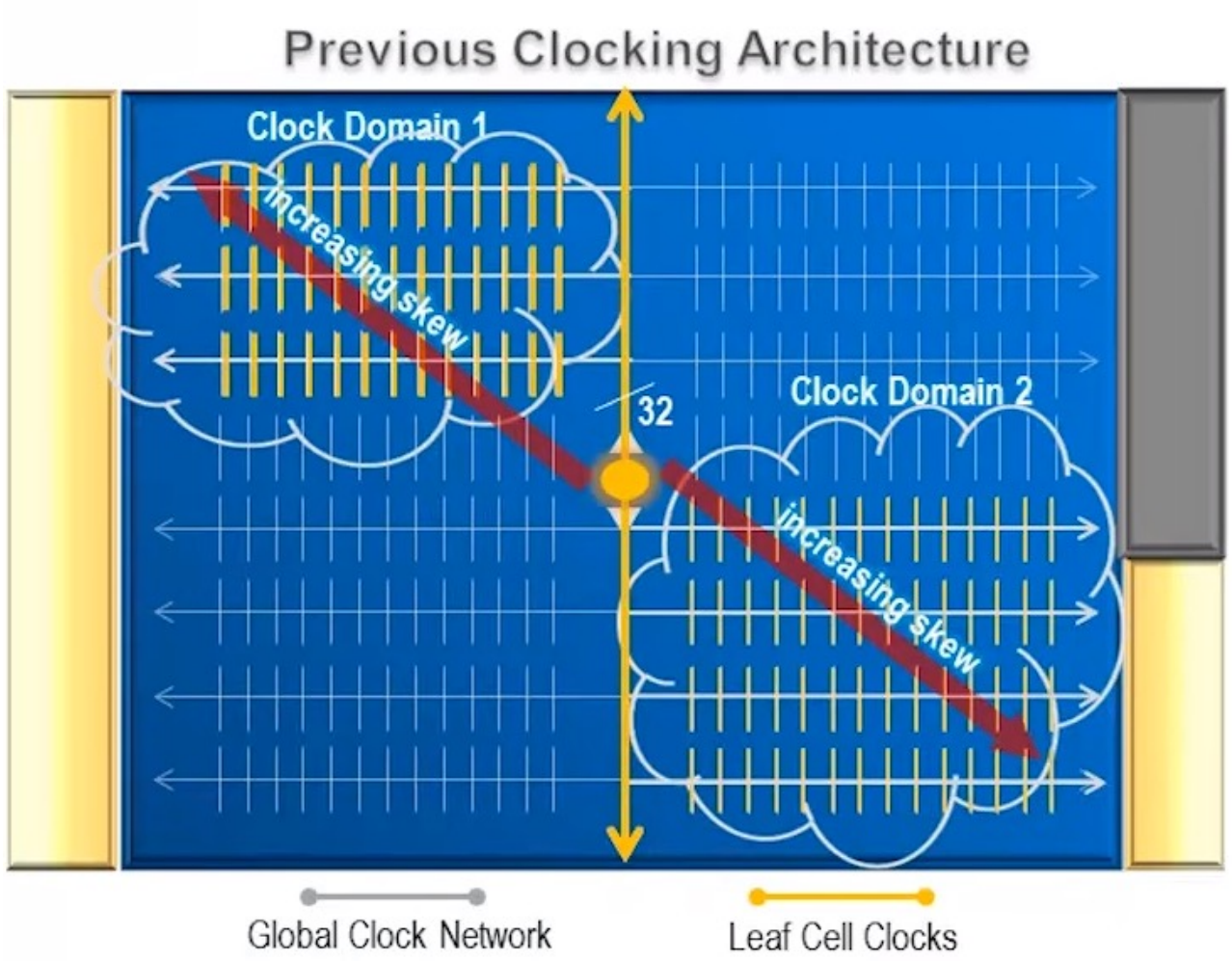
https://www.youtube.com/watch?v=lTp9cCEPwAI
https://docs.xilinx.com/v/u/en-US/wp434-ultrascale-smarter-systems

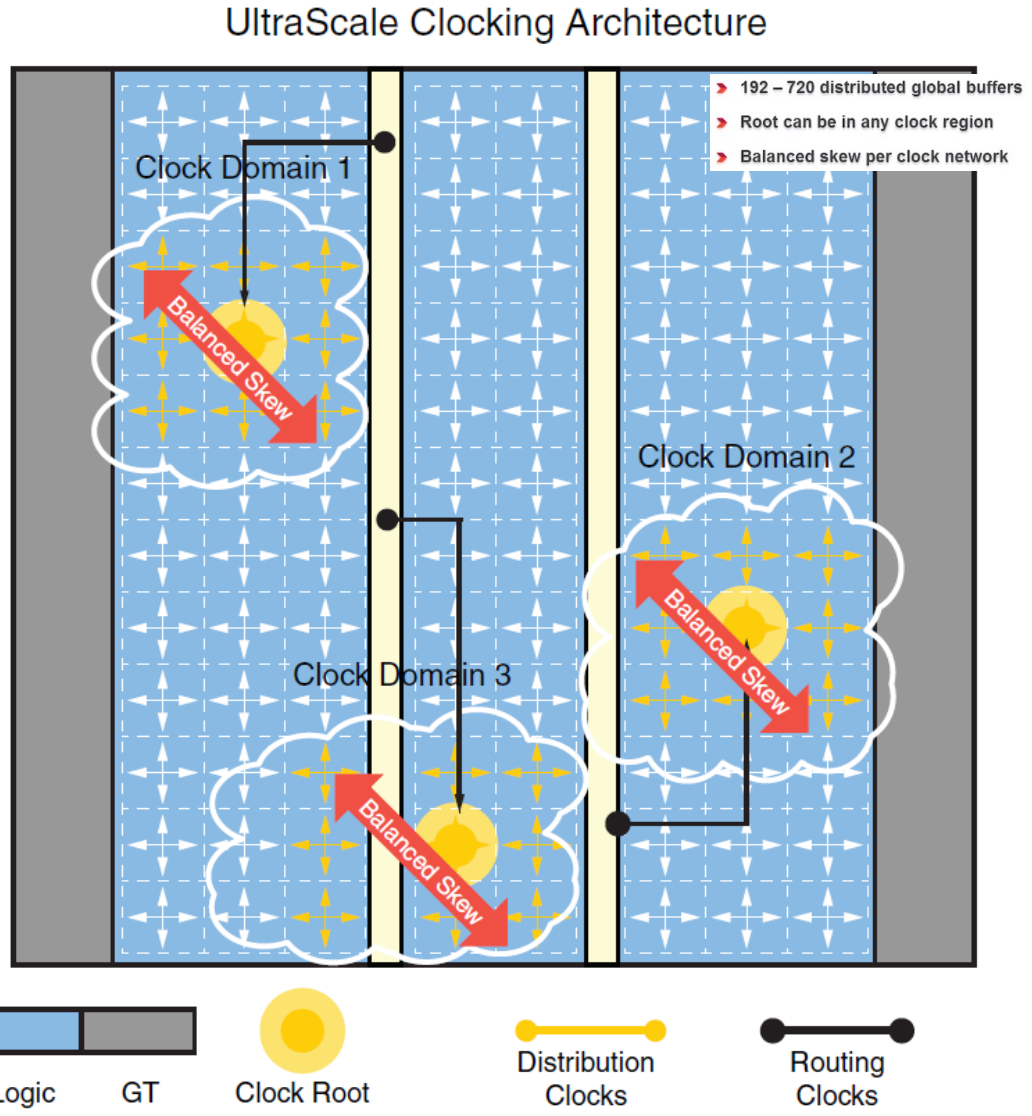
Архитектура тактовых сигналов в Ultrascale



https://www.youtube.com/watch?v=J-Fm7dBgGPU

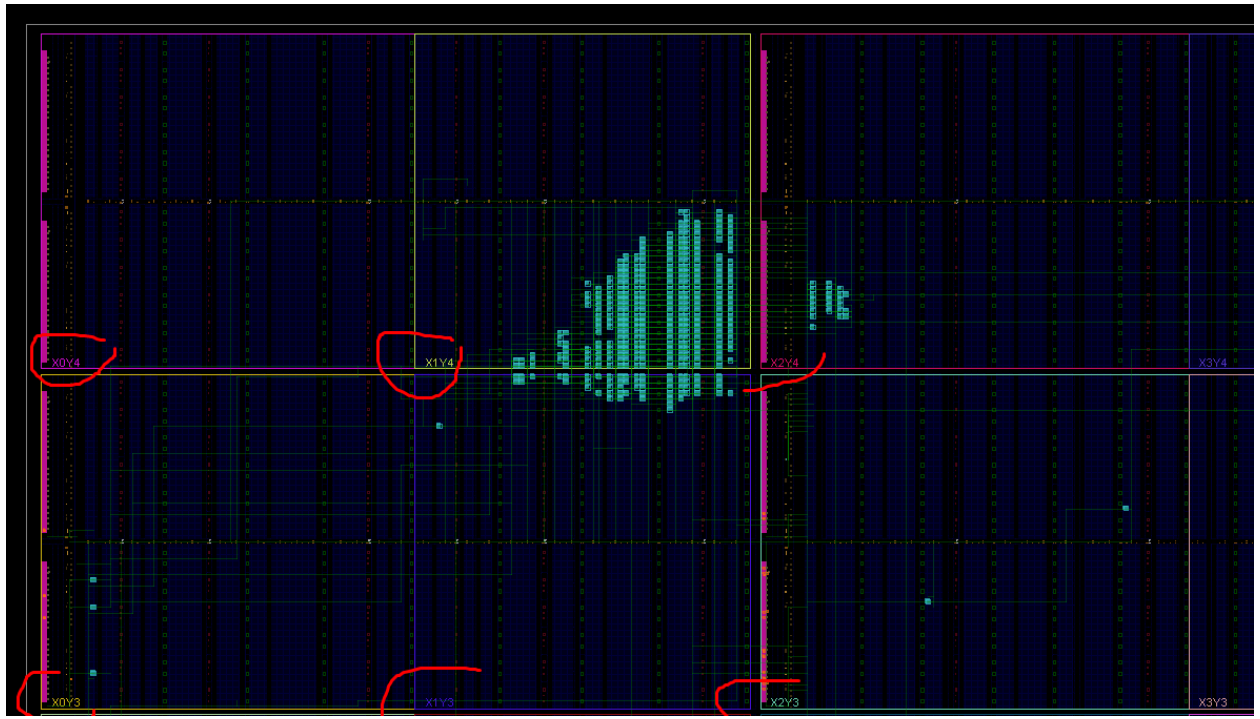
Архитектура тактовых сигналов в Ultrascale



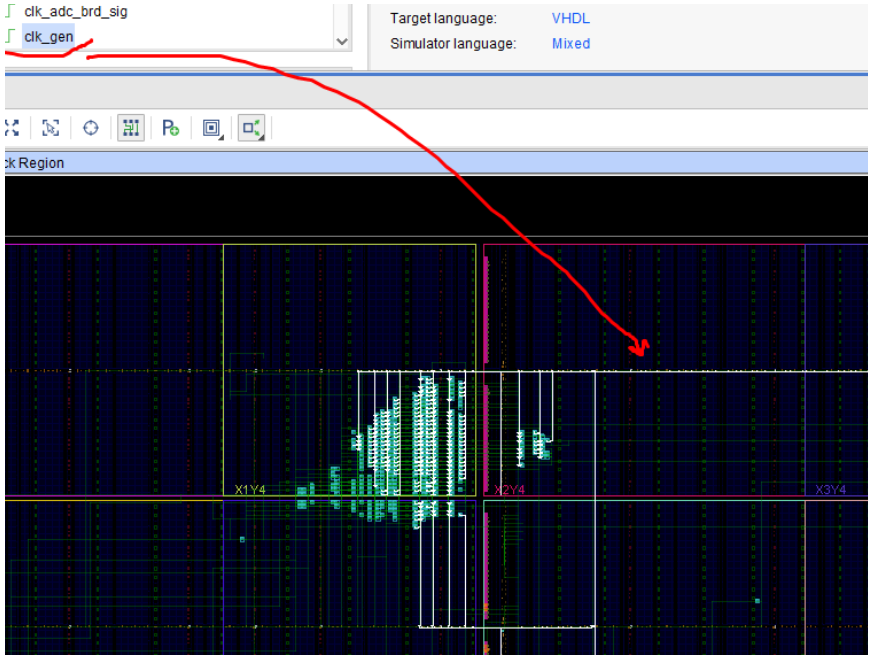


Архитектура тактовых сигналов в Ultrascale

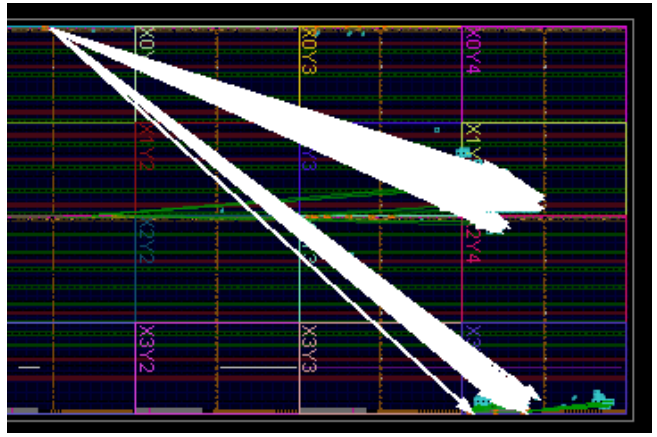
Все клоковые регионы одинакового размера – 60 CLBs в высоту и при этом такой же геометрической ширины
- The height of a CR is 60 CLBs, 24 DSP slices, and 12 block RAMs with a horizontal clock spine (HCS) at its center.



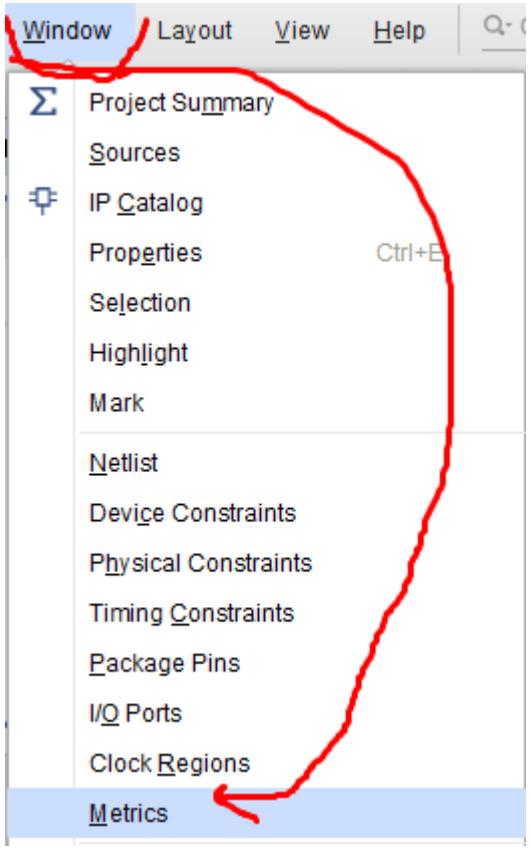
Клоковые регионы на Device view



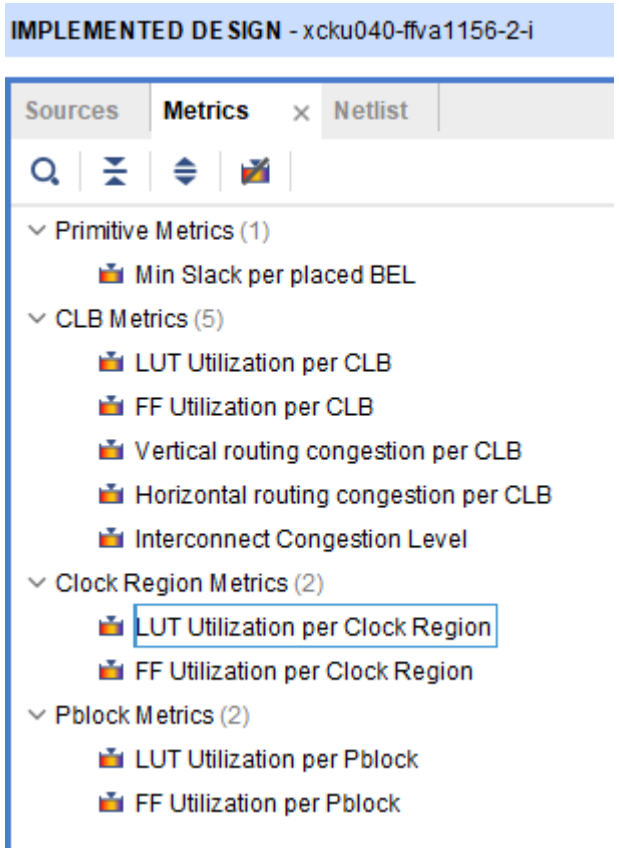
Распределение клока - пример



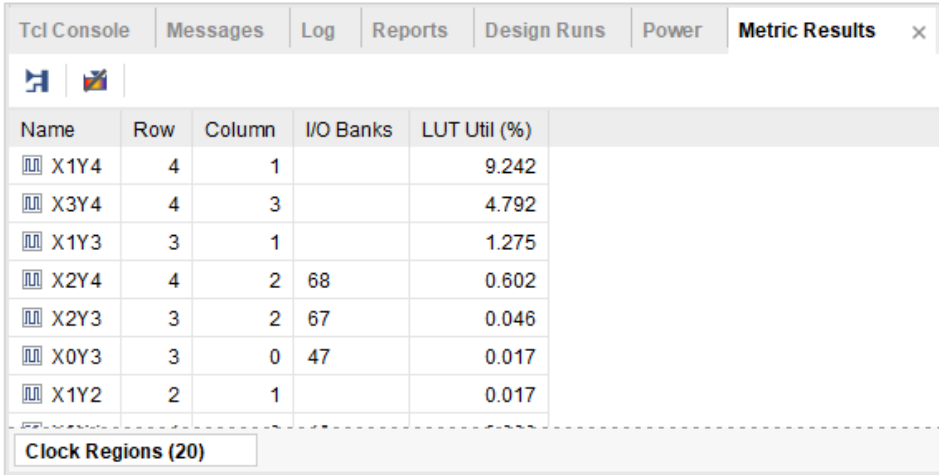
Использование метрики - metrics



Включение метрики в вивадо

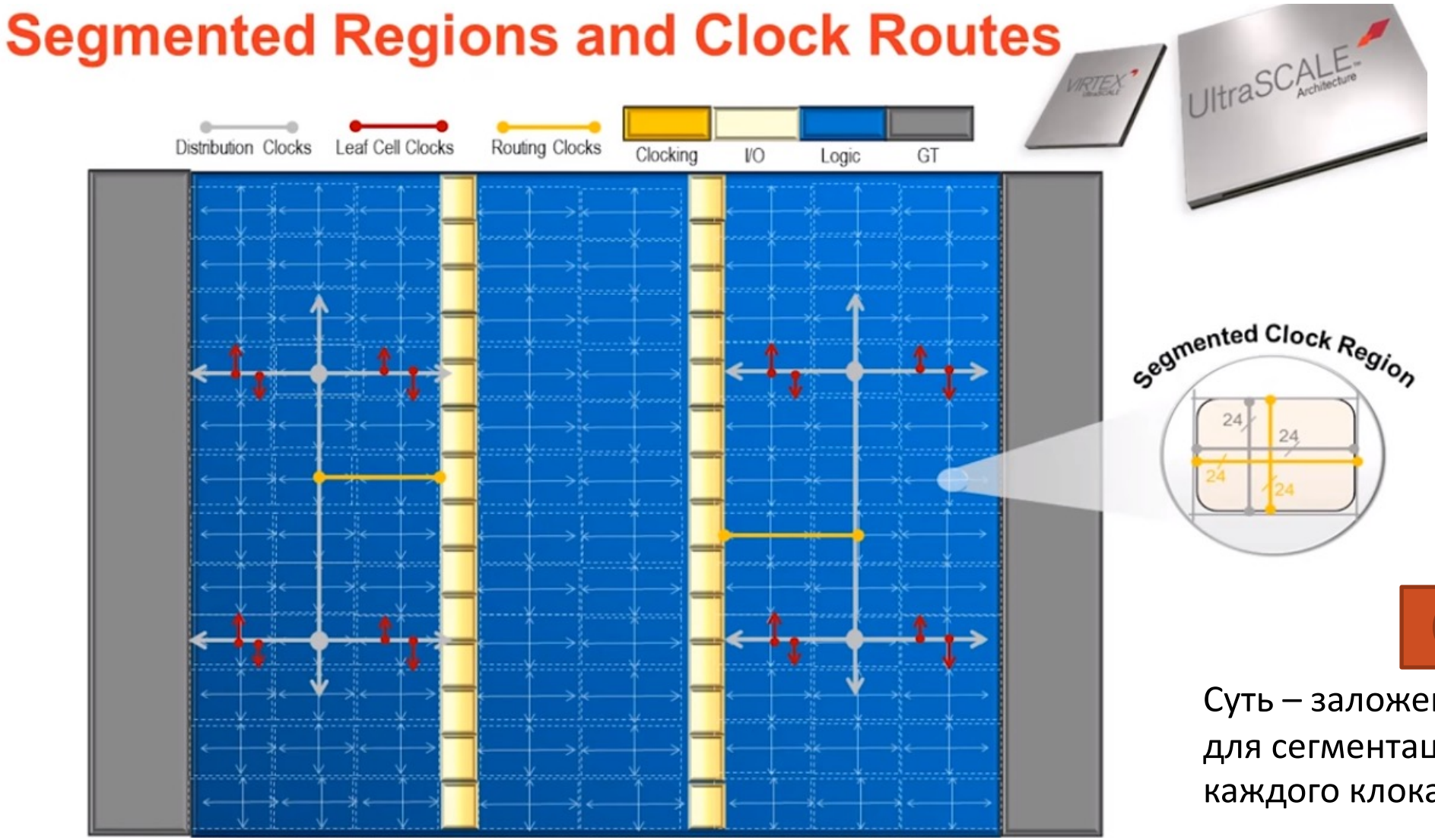


Окно метрики



Пример анализа загруженности ЛУТов по регионам

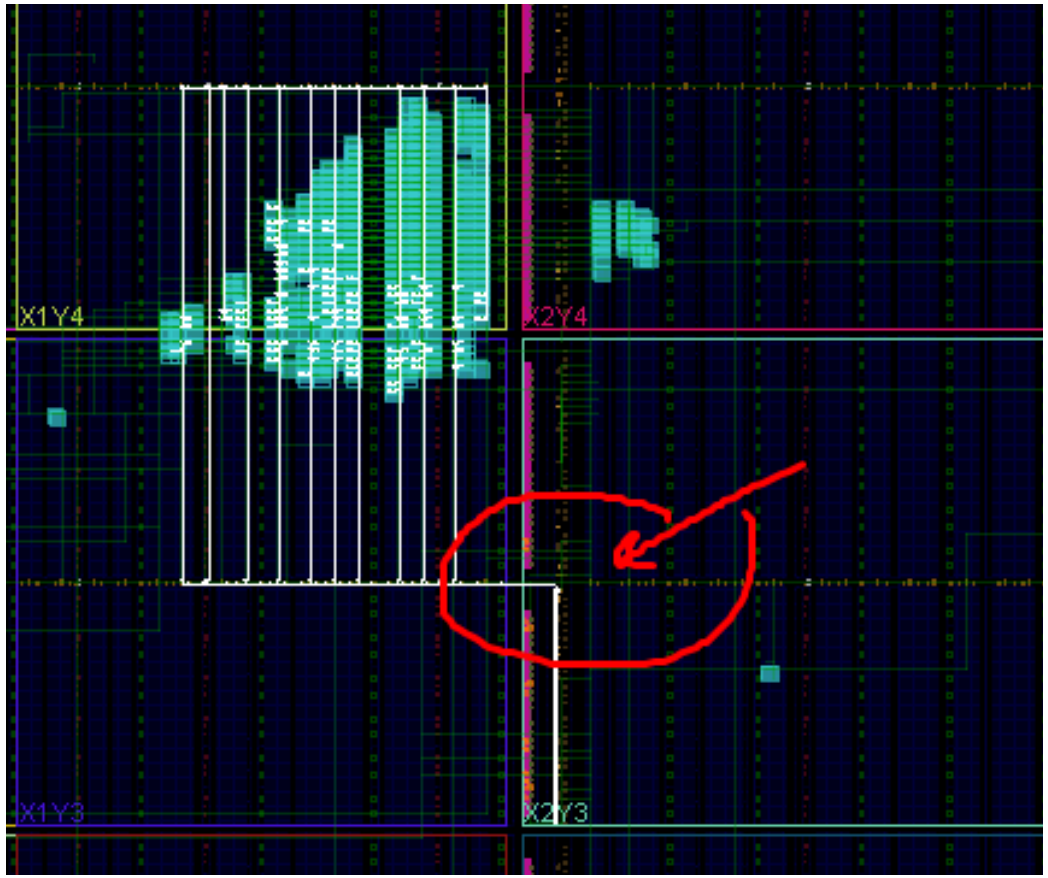
Архитектура тактовых сигналов в Ultrascale - Routing



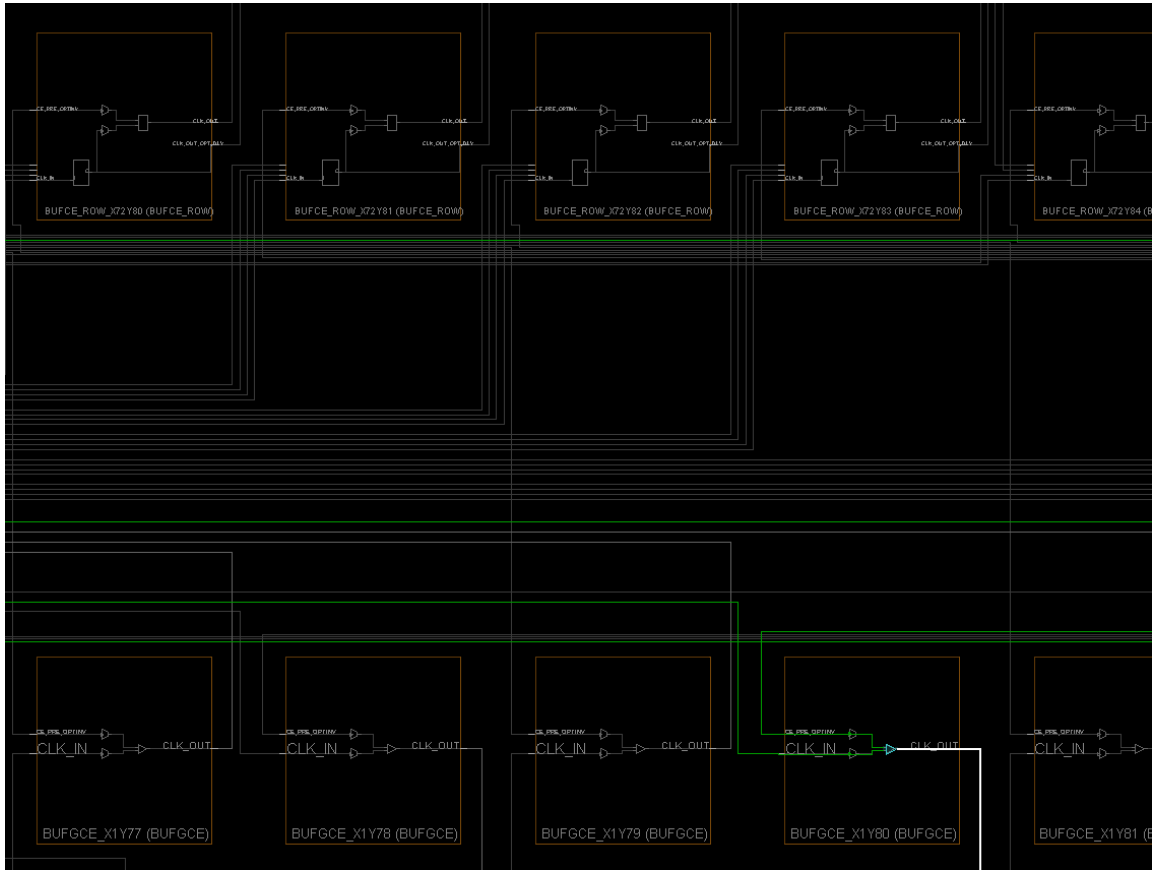
Суть – заложен потенциал
для сегментации
каждого клока, как в АСИКе!!

https://www.youtube.com/watch?v=J-Fm7dBgGPU

Архитектура тактовых сигналов в Ultrascale

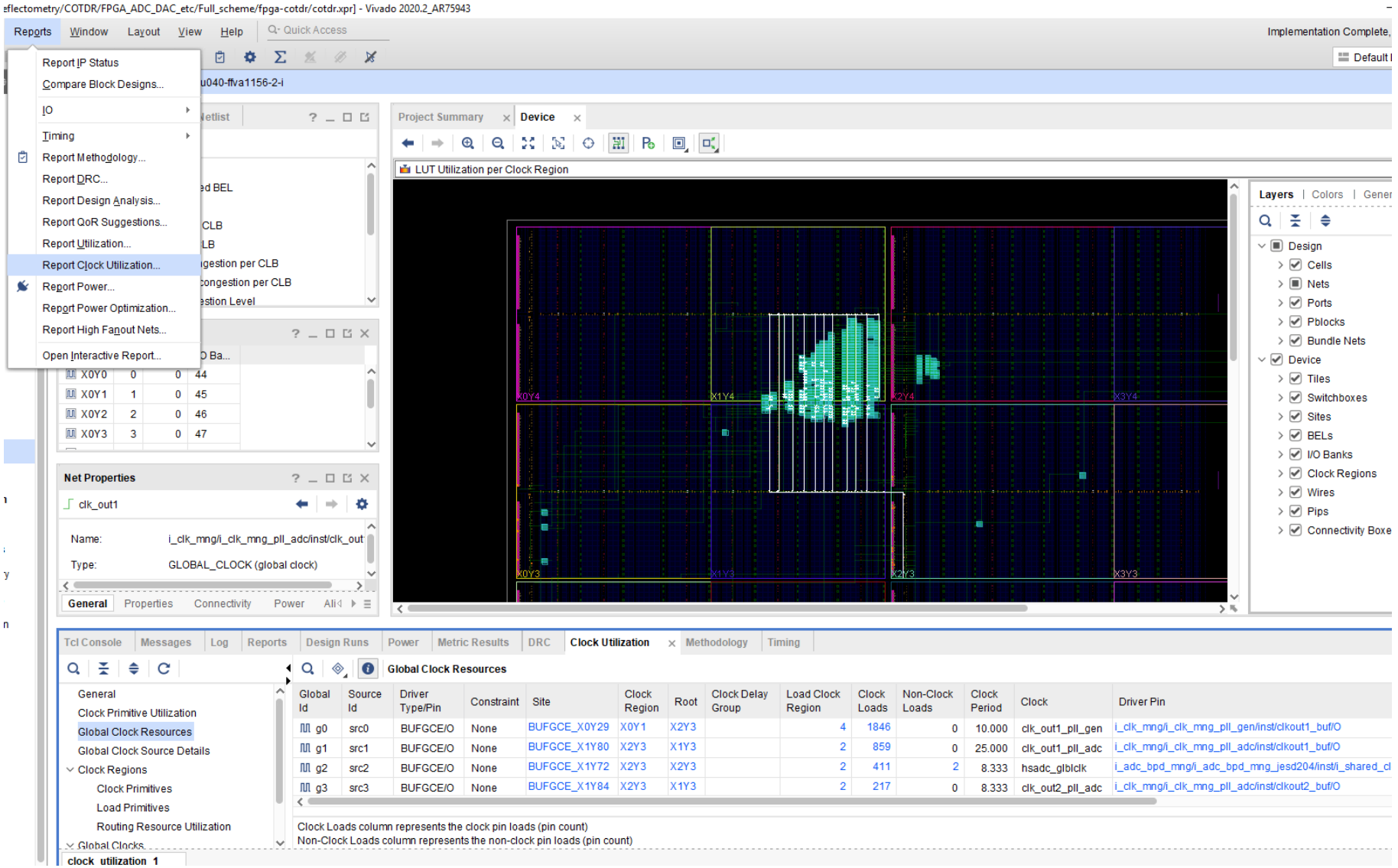


Где располагаются буферы на Device view



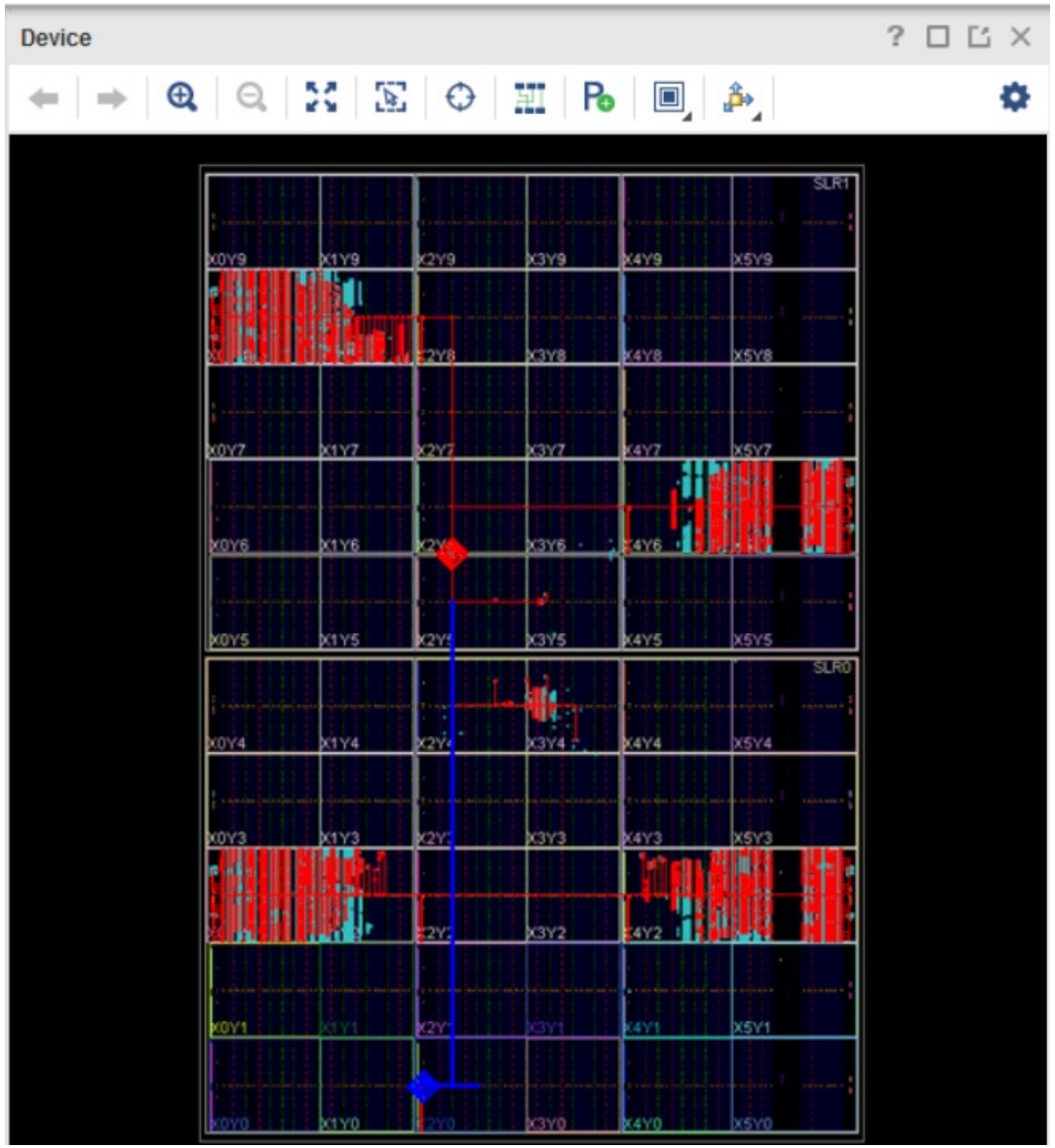
Зум – батарея буферов

Инструментарий анализа – Clock Utilization Report



https://docs.xilinx.com/r/en-US/ug949-vivado-design-methodology/UltraScale-Device-Clocking

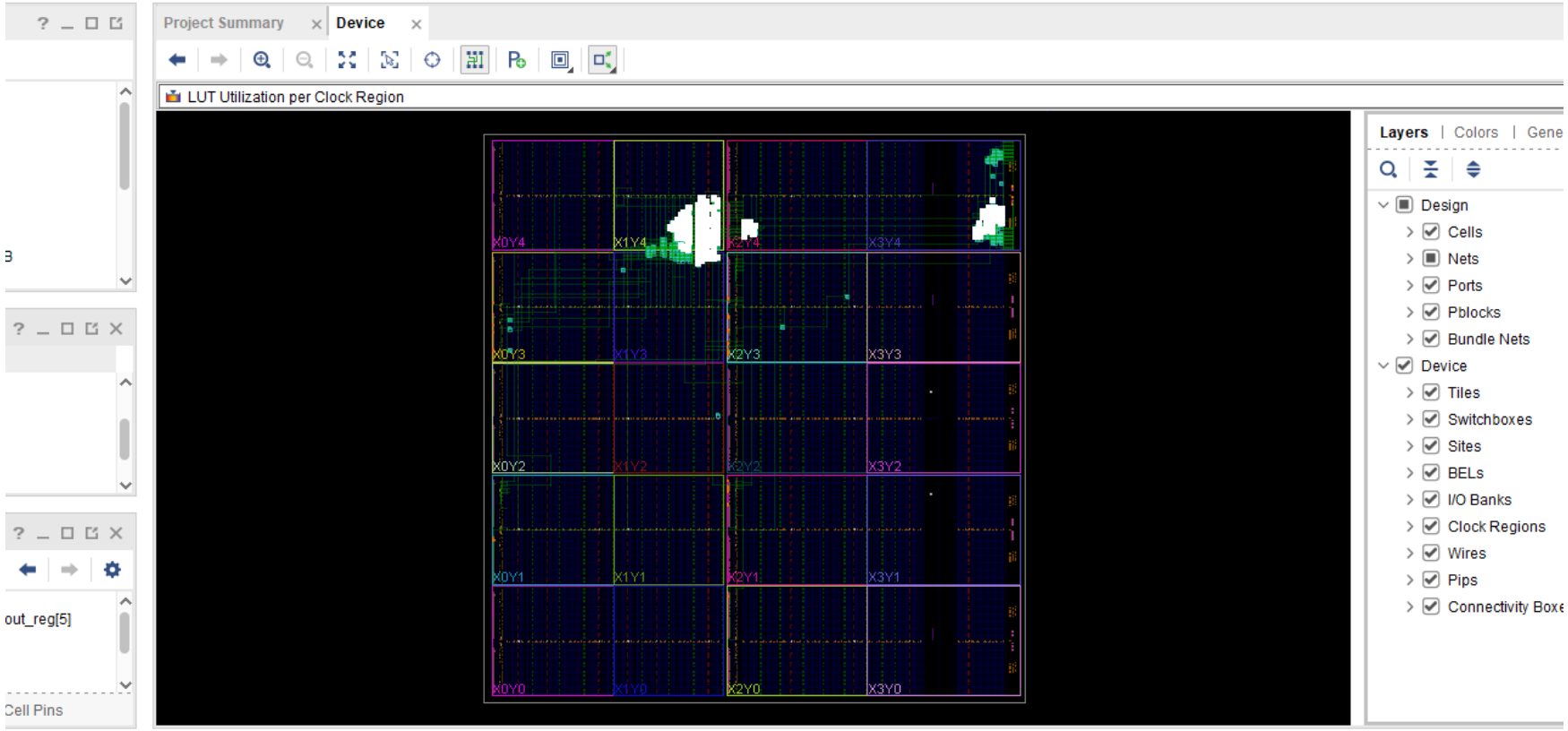
Пример централизации Loads (логики) вокруг Clock Root



– каноничный пример
Routed Device View of a Routed Clock Network

https://docs.xilinx.com/r/en-US/ug949-vivado-design-methodology/UltraScale-Device-Clocking

Пример централизации Loads (логики) вокруг Clock Root



Design Runs | Power | Metric Results | DRC | **Clock Utilization** | Methodology | Timing

Device Cell Placement

Global Id	Driver Type/Pin	Driver Region (D)	Clock	Period (ns)	Waveform (ns)	Root (R)	Slice Loads	IO Loads	Clocking Loads	GT Loads	Net
g0	BUFGCE/O	X0Y1	clk_out1_pll_gen	10.000	{0.000 5.000}	X2Y3	1844	0	0	2	i_clk_mng/i_clk_mng_pll_gen/inst/clk_out1
g1	BUFGCE/O	X2Y3	clk_out1_pll_adc	25.000	{0.000 12.500}	X1Y3	859	0	0	0	i_clk_mng/i_clk_mng_pll_adc/inst/clk_out1
g2	BUFGCE/O	X2Y3	hsadc_glblclk	8.333	{0.000 4.167}	X2Y3	402	0	1	2	i_adc_bpd_mng/i_adc_bpd_mng_jesd204/inst/i_shared_clocks/rx_core_clk_out

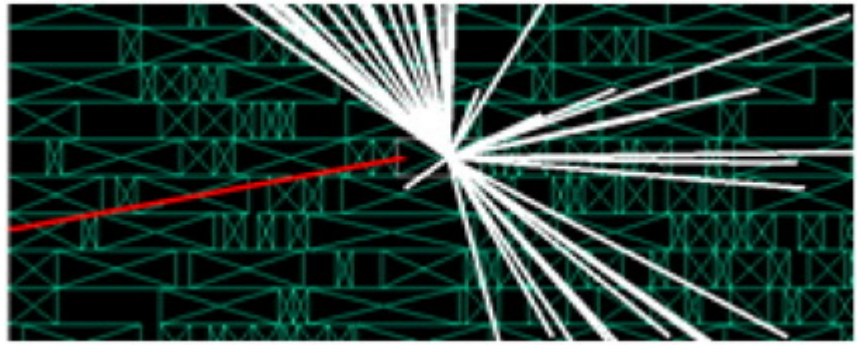
Slice Loads column represents load cell count of all cell types other than IO, GT and clock resources
IO Loads column represents load cell count of IO types
Clocking Loads column represents load cell count that are clock resources (global clock buffer, MMCM, PLL, etc)
GT Loads column represents load cell count of GT types

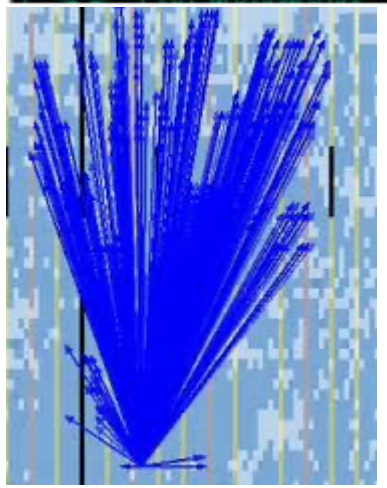
– пример из нашего проекта

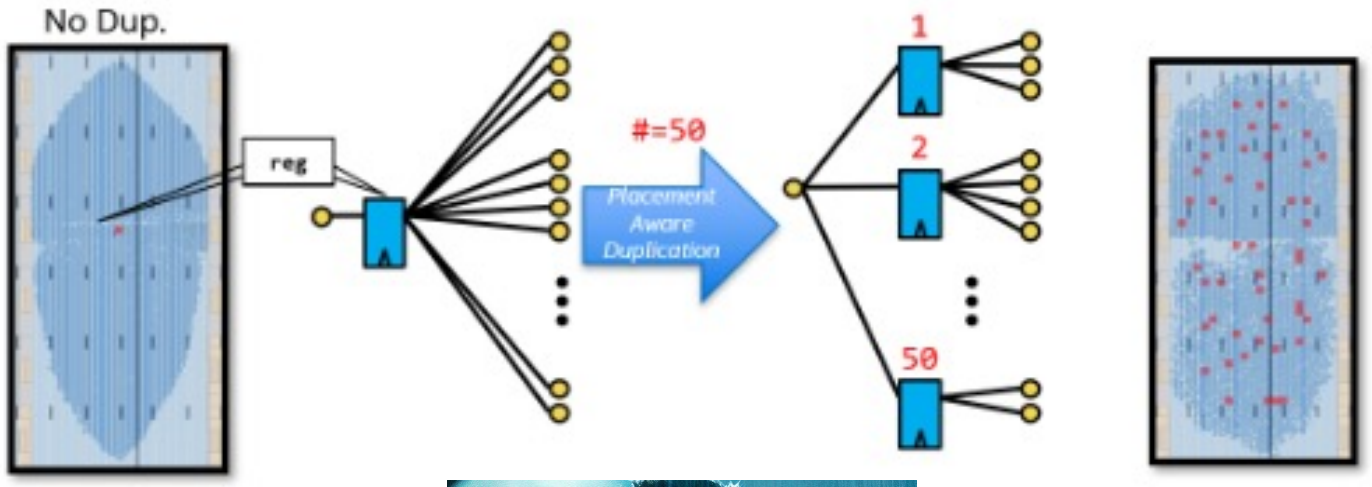
https://docs.xilinx.com/r/en-US/ug949-vivado-design-methodology/UltraScale-Device-Clocking

High fanout signals

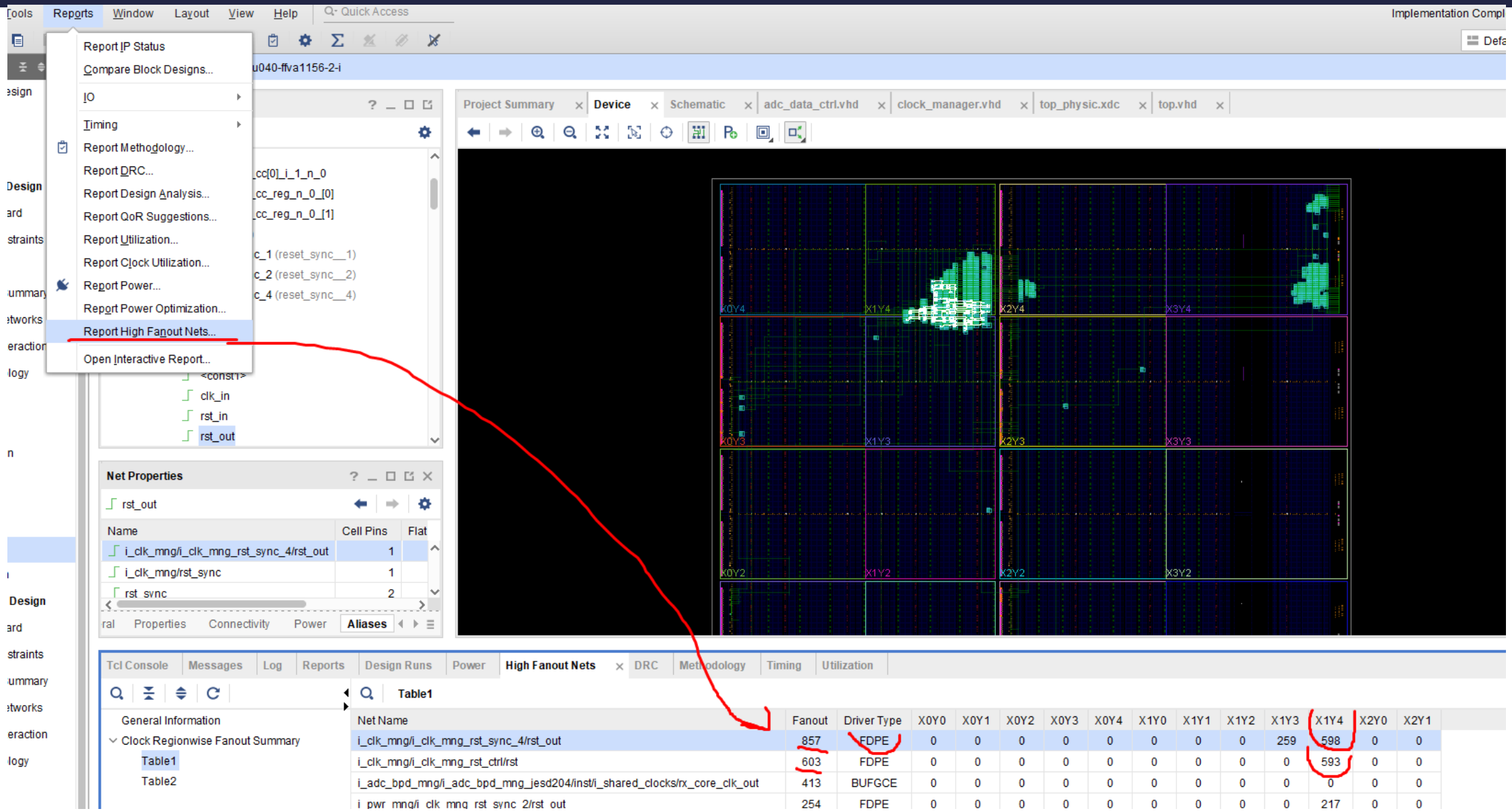
- В предыдущих архитектурах размещение high fanout signals на глобальных ресурсах иногда было предпочтительнее и давало лучшие результаты. Для UltraScale devices, лучше этого не делать.





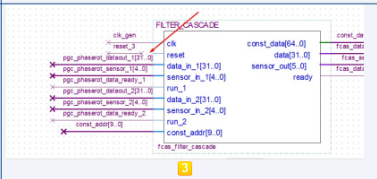


Report high fanout signals

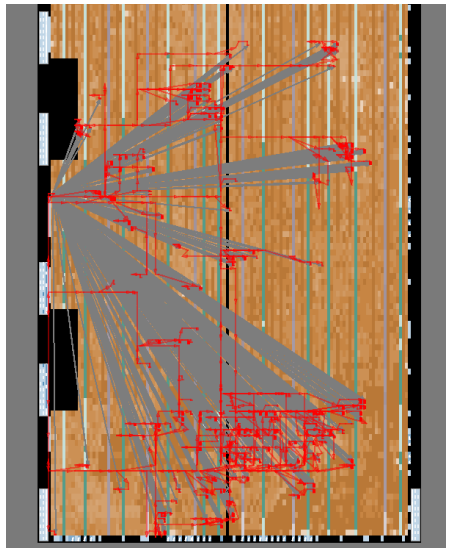


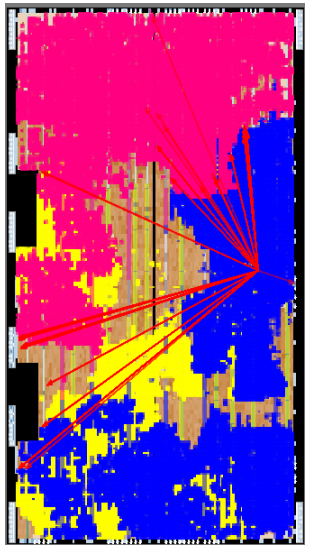
Итеративный подход к оптимизации проекта

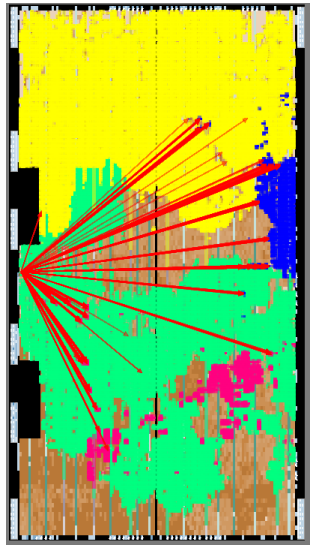
Тайминг фикс_проект 218_коса 160 МГц

	Проблематика	Список изменений	Результаты	Приложения
★ Итерация 1	Проект по умолчанию не лез в ПЛИС 5CEFA9F23I7 - 113 % DSP и 101% ALM. После настройки Optimization technique - AREA Проект влез, но люто не проходил по времянке - slack в районе 10 нс в блоке CIC Filter. Это изначальные данные.	1 Установил глоб. настройку Optimization technique - SPEED 2 Перевёл CIC настройку pipeline - 4; Decimator и Integrator - M10K 3 Оторвал reset_3 от FILTER_CASCADE	Family Cyclone V Device 5CEFA9F23I7 Timing Models Final Logic utilization (in ALMs) 94,737 / 113,560 (83 %) Total registers 217790 Total pins 205 / 224 (92 %) Total virtual pins 0 Total block memory bits 4,986,172 / 12,492,800 (40 %) Total DSP Blocks 340 / 342 (99 %) Результат по ресурсам Результат таймингов общий	

Xmind, Mindjet







Архитектура тактовых сигналов в Ultrascale – практические выводы

1. Системному архитектору для больших и быстрых проектов под Ultrascale всё больше важно мышление как у ASIC – инженера (понимание CDC, регионирования)
2. С учётом новых возможностей, ещё больше сокращается необходимость ручного влияния на трассировку путём выбора буферов, НО (!) возрастает возможность и важность регионирования для плотных проектов. Инструмент – Clock utilization Report
3. Для уменьшения джиттера (clock_uncertainty) можно использовать для деления клока BUFGCE_DIV вместо MMCM
4. *ADVANCED – не рекомендуется использовать Pblock для единственного (single) clock region.
5. *ADVANCED – при регионировании глобальных буферов вместо использования {LOCATION} property используем {CLOCK_REGION} property
6. *ADVANCED при анализе проблем с таймингами по setup используем другое эмпирическое правило

проверьте, нет ли высокой задержки на пути передачи данных (high datapath delay) из-за :

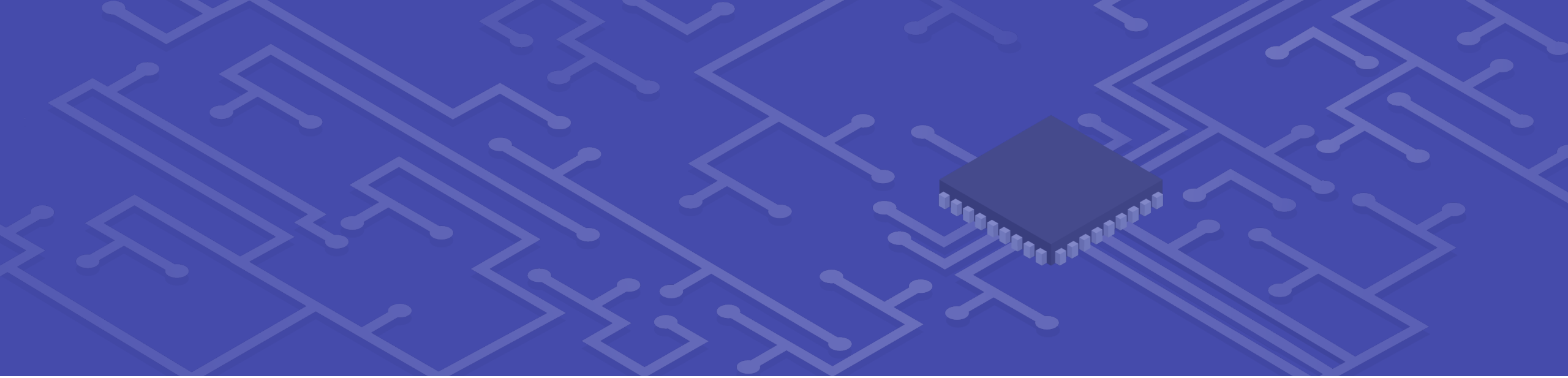
- Large cell delay (7 series > 25%, UltraScale devices > 50%)
- Large net delay (7 series > 75%, UltraScale devices > 50%)

История с некой фирмой

1. У некой фирмы не получалось поднять стабильно интерфейс работы с АЦП высокоскоростной АЦП

2. Тайминги не проходили на 7-ом Kintex, и был осуществлён переход на Kintex Ultrascale

3. Результат – стало хуже

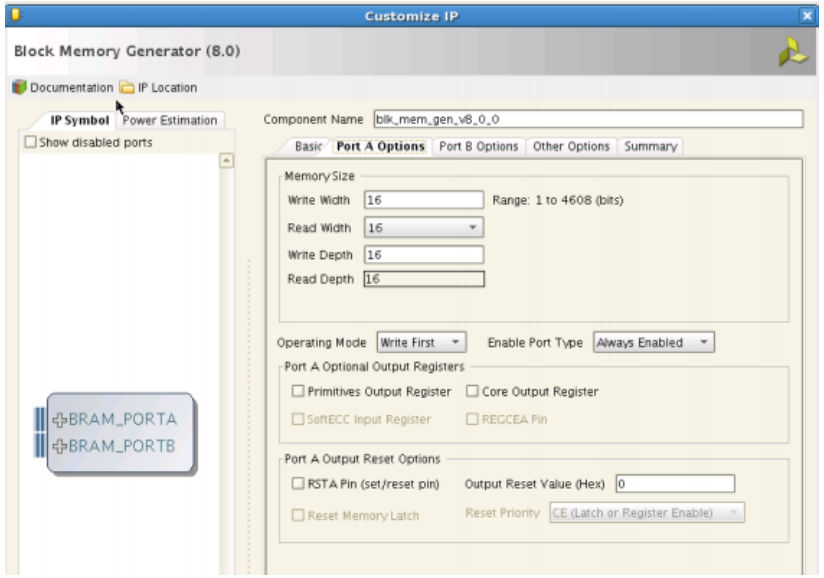


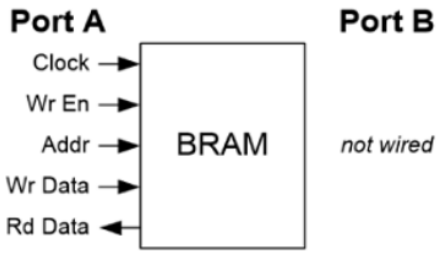
РАЗНОЕ

Способы использования аппаратных ресурсов ПЛИС в коде\проекте

Существует три основных варианта создания блоков кода:

GUI





Instantiate

```
SB_RAM256x16 ram256x16_inst (
.RDATA(RDATA_c[15:0]),
.RADDR(RADDR_c[7:0]),
.RCLK(RCLK_c),
.RCLKE(RCLKE_c),
.RE(RE_c),
.WADDR(WADDR_c[7:0]),
.WCLK(WCLK_c),
.WCLKE(WCLKE_c),
.WDATA(WDATA_c[15:0]),
.WE(WE_c),
.MASK(MASK_c[15:0])
);
```

Instantiate – создавать экземпляр , копию

Infer

```
module SRAM_Single_Port #(parameter WIDTH = 16,
                          parameter DEPTH = 256)
  (
   input                      i_Clk,
   input                      i_Wr_En,
   input [$clog2(DEPTH)-1:0] i_Addr,
   input [WIDTH-1:0]          i_Wr_Data,
   output reg [WIDTH-1:0]     o_Rd_Data,
   );

  reg [WIDTH-1:0] r_Mem [DEPTH-1:0];

  always @(posedge i_Clk)
  begin
    if (i_Wr_En)
      r_Mem[i_Addr] = i_Wr_Data;
  end

  assign o_Rd_Data = r_Mem[i_Addr];

endmodule // SRAM_Single_Port
```

Infer – подразумевать, делать вывод, выводить

https://nandland.com/lesson-17-inference-vs-instantiation-vs-gui-which-is-best/

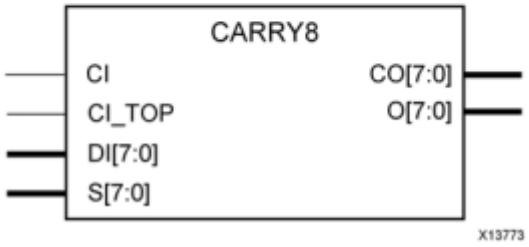
Способы использования аппаратных ресурсов ПЛИС в коде\проекте

Примеры – используем UG974 для Ultrascale

CARRY8

Primitive: Fast Carry Logic with Look Ahead

PRIMITIVE_GROUP: CLB
PRIMITIVE_SUBGROUP: CARRY
Families: UltraScale, UltraScale+



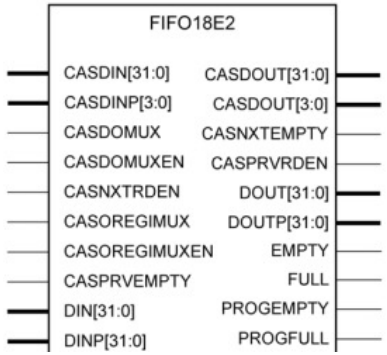
Design Entry Method

Instantiation	Yes
Inference	Recommended
IP and IP Integrator Catalog	Yes

FIFO18E2

Primitive: 18Kb FIFO (First-In-First-Out) Block RAM Memory

PRIMITIVE_GROUP: BLOCKRAM
PRIMITIVE_SUBGROUP: FIFO
Families: UltraScale, UltraScale+



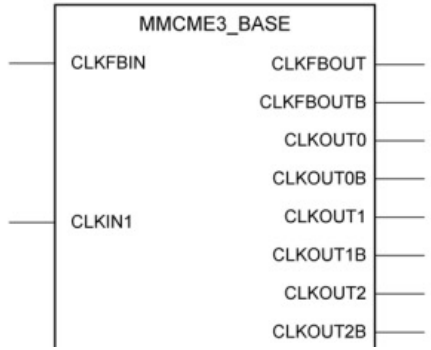
Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	Recommended

MMCME3_BASE

Primitive: Base Mixed Mode Clock Manager (MMCM)

PRIMITIVE_GROUP: CLOCK
PRIMITIVE_SUBGROUP: PLL
Families: UltraScale, UltraScale+



Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	Recommended

Архитектурные/системные решения на примере памяти

- **RAM / ROM** – рекомендуемый метод описания – **inference**. Используем шаблоны. Но есть возможность и использования IP-каталога и прямого Instantiation
- Отдельная важная **возможность** AMD (**Xilinx**) Parameterizable Macros (**XPMs**)

Некая **промежуточная** возможность **между inference** и **instantiation**. К преимуществам относят **быстрое моделирование** и **портативность** между семействами Xilinx. К **недостаткам** – **ограничение параметров** XPM. Фактически, XPMs – встроенные возможности создания блоков методом inference с использованием шаблонов, которые нельзя изменять.

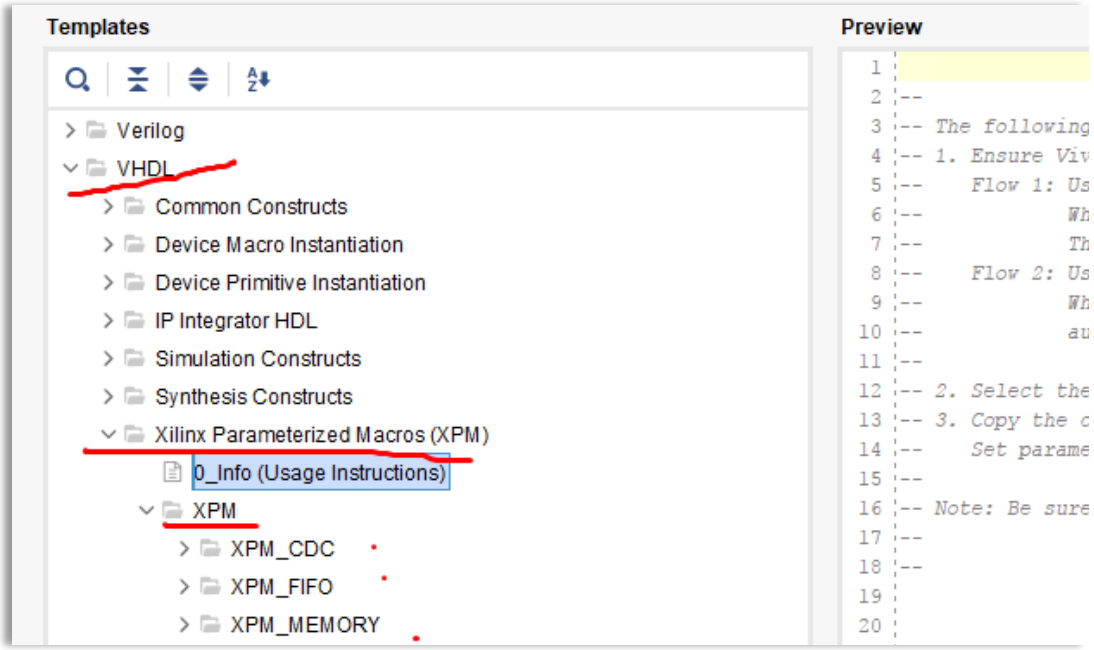
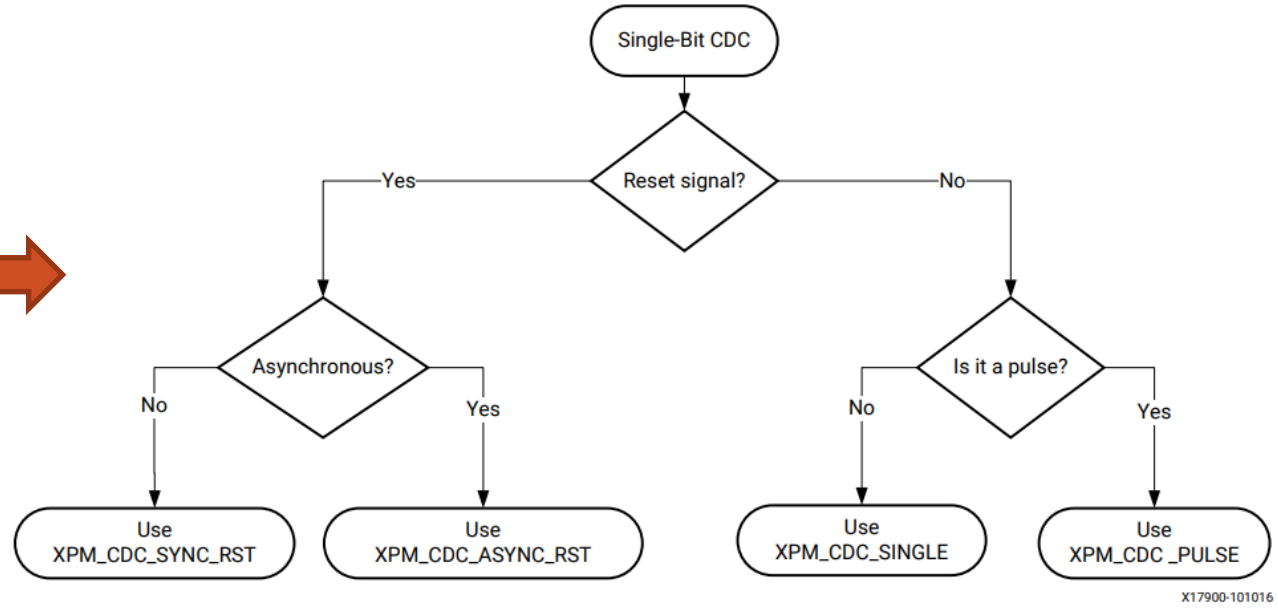
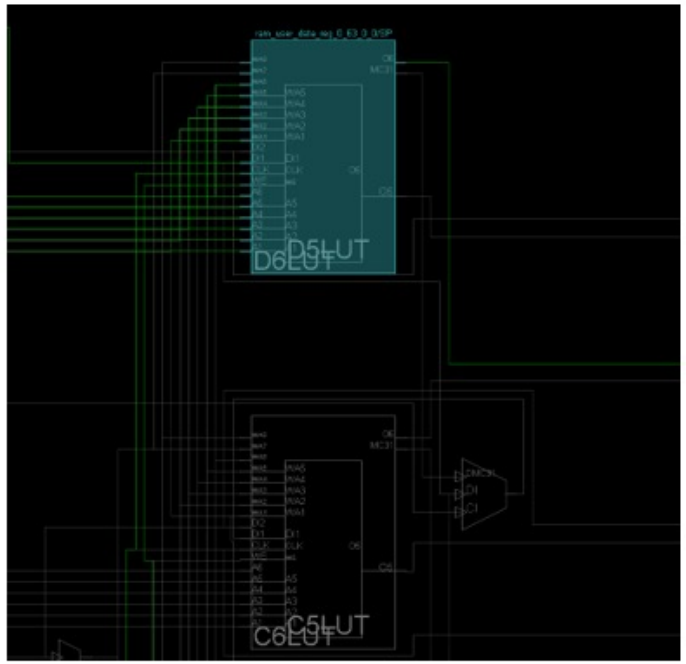


Figure 84: **Single-Bit CDC Decision Tree**

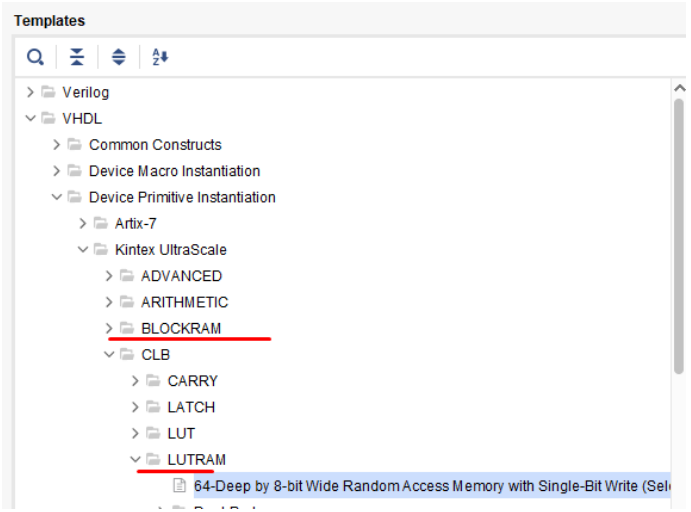


Архитектурные/системные решения на примере памяти

- **Элементы памяти** – RAM **могут** быть **имплементированы**, как и в 7 семействе, в блочной памяти или в **распределённой** (внутри логических элементов) – LUTRAM
- Примерное правило – память глубиной **менее 64 бит** реализуется в **LUTRAM**, **более 256** – **Block RAM**
- Массивы памяти **нельзя ресетить**. Только **выходной регистр** RAM. При этом ресет **только синхронный**! И сброс только в **0**!




```
architecture rtl of top_clb is
    ------------------------------------------------------------------------------
    -- DRAM
    ------------------------------------------------------------------------------
    attribute RAM_STYLE : string;
    constant RAM_BIT    : natural := 2 ** ram_user_addr'length;
    type RAM_TYPE is array (0 to RAM_BIT - 1) of std_logic;
    signal ram_user_data                : RAM_TYPE := (others => '0');
    attribute RAM_STYLE of ram_user_data : signal is "DISTRIBUTED";
begin
    process (clk)
    begin
        if (rising_edge(clk)) then
            if (ram_user_we = '1') then
                ram_user_data(to_integer(unsigned(ram_user_addr))) <= ram_user_in;
            end if;
        end if;
    end process;
    ram_user_out <= ram_user_data(to_integer(unsigned(ram_user_addr)));
    -->>>>>>> OR <<<<<<--
    RAM64X1S_1_inst : RAM64X1S_1
   generic map (
      INIT => X"0000000000000000")
   port map (
      O => O,        -- 1-bit data output
      A0 => A0,      -- Address[0] input bit
      A1 => A1,      -- Address[1] input bit
      A2 => A2,      -- Address[2] input bit
      A3 => A3,      -- Address[3] input bit
      A4 => A4,      -- Address[4] input bit
      A5 => A5,      -- Address[5] input bit
      D => D,        -- 1-bit data input
      WCLK => WCLK,  -- Write clock input
      WE => WE       -- Write enable input
   );
```




```
1  localparam WIDTH = 8;
2  localparam DEPTH = 64;
3  (* ram_style = "distributed" *) reg [WIDTH-1:0] mem [0:DEPTH-1];
4  always @(posedge clk) begin
5    if (wen)
6      mem[waddr] <= wdata;
7  end
8  assign rdata = mem[raddr];
```

В случае проблем с таймингами блоков RAM рекомендуется использовать Pipeline registers на выходах, и иногда и на входах

https://support.xilinx.com/s/question/0D52E00006hpQZhSAM/distributed-ram-synthesis-infers-more-slicem-resources-than-expected?language=en_US

Архитектурные/системные решения – DSP блоки - байка

- По DSP-блокам – регистры DSP48E2 в UltraScale содержат только ресеты.
- Не описывайте событие установки (Set) для регистров вокруг умножителей, сумматоров, счётчиков и другой логики, которая может быть имплементирована в виде DSP- блока.
- DSP-блоки используют signed арифметику
- Xilinx советует описывать HDL с использованием signed переменных!
- Утверждается, что использование unsigned из-за необходимости конверсии менее эффективно.

Результаты до и после использования атрибута (* use_dsp = "yes" *)

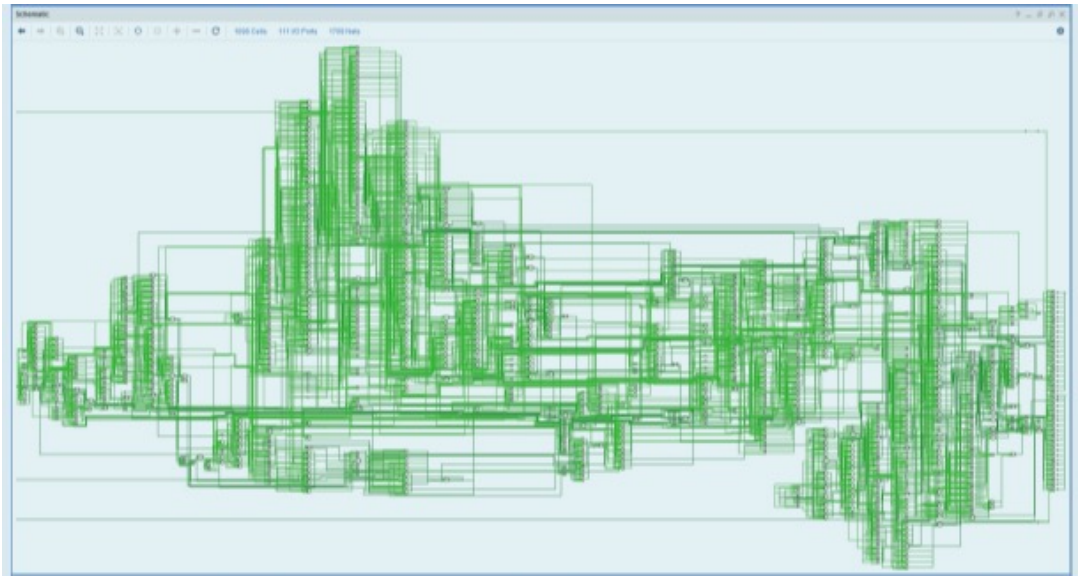


Схема без DSP-блоков

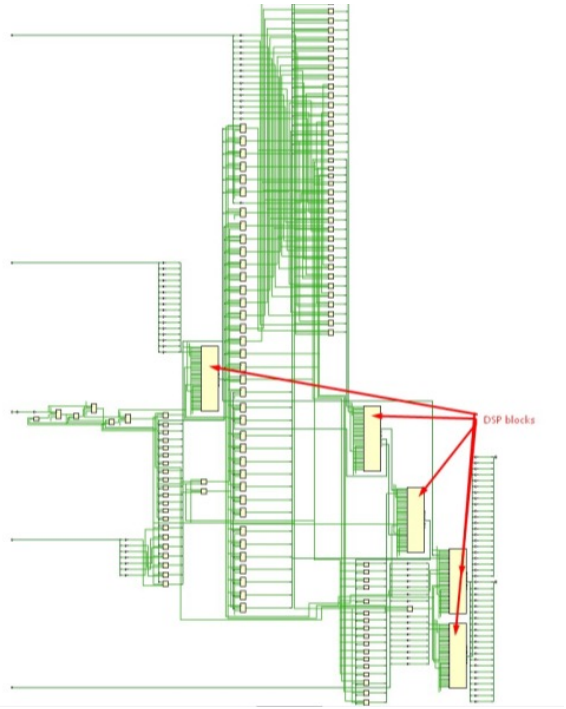


Схема с DSP-блоками

Атрибут использован для всего модуля, однако может быть применён для конкретного сигнала (VHDL) или регистра (Verilog),
пример использования тут :
https://www.xilinx.com/support/answers/54357.html

С….. Пасиба

УНИВЕРСИТЕТ ИТМО

ЦЕНТР ПРОГРАММИРУЕМОЙ ЭЛЕКТРОНИКИ

Проекты

Разработка оптоэлектронных приборов в НИЦ Световодной фотоники.

Презентация:

Магистратура

Инженерия цифровых систем в институте "Высшая инженерно-техническая школа"

Программа:

Курсы ДПО

Курсы по направлениям: Xilinx, IntelFPGA, Altium, STMicroelectronics

Сайт:

День открытых дверей
в НИЦ Световодной фотоники университета ИТМО

6 июня в 16.00

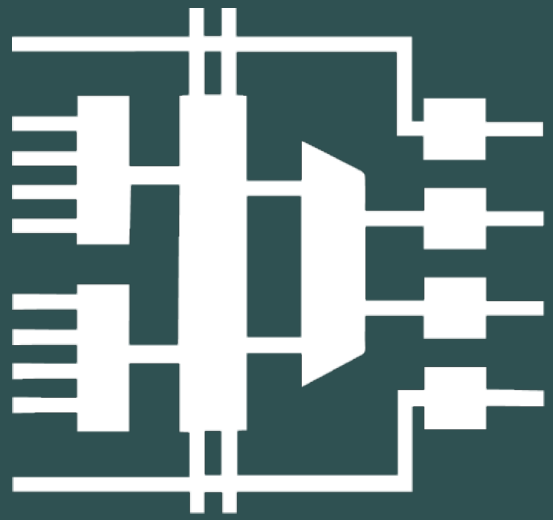
Генеральный партнёр конференции FPGA-Systems 2024.1

DeltaDesign

Первая современная отечественная САПР,
реализующая сквозной цикл проектирования печатных плат

www.eremex.ru

Где найти FPGA / RTL / Verification комьюнити?



FPGA-Systems.ru

Сайт комьюнити

FPGA-Systems Magazine (FSM)

Первый журнал о программируемой логике

@fpgasystems

Телеграм чат

admin@fpga-systems.ru

Электронная почта

Youtube.com/c/fpgasystems

Youtube канал