

The background features a light blue and green color palette. On the left side, there is a circular lensed light source, possibly a fiber optic or LED array, emitting a soft glow. The rest of the background is filled with a complex, semi-transparent pattern of circuit board traces and grid lines, creating a technical and digital atmosphere.

Szintézis attributumok és constraint-ek

Szintézis attributumok

- **BLACK BOX:** forrás szinten nem adott modul (pl. szintetizált huzalozási lista)

```
attribute black_box : string;  
attribute black_box of beh : architecture is yes;
```

- **CLOCK_BUFFER_TYPE**

```
entity test is port(  
  in1 : std_logic_vector (8 downto 0);  
  clk : std_logic;  
  out1 : std_logic_vector(8 downto 0));  
  attribute clock_buffer_type : string;  
  attribute clock_buffer_type of clk: signal is "BUFR";  
end test;
```

- Értékek: “BUFG”, “BUFH”, “BUFIO”, “BUFMR”, “BUFR” or “none”

Szintézis attributumok

- **EXTRACT_RESET: FF reset bemenet használat**

```
signal my_reg : std_logic;  
attribute extract_reset : string;  
attribute extract_reset of my_reg: signal is "no";
```

- **DIRECT_RESET: reset jel direct megadása**

```
attribute direct_reset : string;  
attribute direct_reset of rst: signal is "yes";
```

- **EXTRACT_ENABLE: FF CE bemenet használat**

```
signal my_reg : std_logic;  
attribute extract_enable : string;  
attribute extract_enable of my_reg: signal is "no";
```

- **DIRECT_ENABLE: CE jel direct megadása**

```
attribute direct_enable : string;  
attribute direct_enable of en_0: signal is "yes";
```

Szintézis attributumok

- **ASYNC_REG**

- Szinronizáció: az adat nem szinkron a mintavételező órajellel
- Az attributum hatására a szinkronizáló FF-kat a leehető legközelebb helyezi el egymáshoz

```
attribute ASYNC_REG : string;  
attribute ASYNC_REG of sync_regs : signal is "TRUE";
```


Szintézis attributumok

- **SHREG_EXTRACT**: shift regiszter felismerés

```
attribute shreg_extract : string;  
attribute shreg_extract of my_srl : signal is "no";
```

- **SRL_STYLE**: shift regiszter implementáció

```
attribute srl_style : string;  
attribute srl_style of my_srl : signal is "reg_srl_reg";
```

- „register”: CLB FF-kat használ
- „srl”: minden bit LUT SRL-ben
- „srl_reg”: utolsó bit CLB FF-ban, többi SRL-ben
- „reg_srl”: első bit CLB FF-ban, többi SRL-ben
- „reg_srl_reg”: első és utolsó bit CLB FF-ban, többi SRL-ben
- „block”: BRAM-ban

Szintézis attributumok

- **DONT_TOUCH:** jel megtartása a szintézis/implementáció során
 - Hatása: szintézis és place & route
 - Használható belső jeleken

```
signal sig1 : std_logic;  
attribute dont_touch : string;  
attribute dont_touch of sig1 : signal is "true";
```

- Vagy entity-n/module-on

```
entity example is port (  
  clk : in std_logic;  
  in1 : in std_logic;  
  out1 : out std_logic  
);  
attribute dont_touch : string;  
attribute dont_touch of example : entity is "yes";  
end example;
```

Szintézis attributumok

- **KEEP**

- Hatása csak a szintézisre van, belső jeleken használható

```
signal signal0 : std_logic;  
attribute keep : string;  
attribute keep of signal0 : signal is "true";  
...  
signal0 <= in1 and in2;  
out1 <= signal0 and in3;
```

Szintézis attributumok

- **KEEP_HIERARCHY: hierarchia (portok) megtartása**
 - Architecture-re/module-ra adható meg

```
entity test is port(  
  in1 : std_logic_vector (8 downto 0);  
  clk : std_logic;  
  out1 : std_logic_vector(8 downto 0)  
);  
end test;  
  
architecture rtl of test is  
  attribute keep_hierarchy : string;  
  attribute keep_hierarchy of rtl : entity is "yes";  
  .....  
end rtl;
```


Szintézis attributumok

- **KEEP_HIERARCHY**

- Példányosításnál:

```
entity test is port(  
  in1 : std_logic_vector (8 downto 0);  
  clk : std_logic;  
  out1 : std_logic_vector(8 downto 0)  
);  
end test;  
architecture rtl of test is  
  attribute keep_hierarchy : string;  
  attribute keep_hierarchy of inst0 : entity is "yes";  
begin  
  inst0: entity work.submodule(rtl)  
  port map(  
    clk => clk,  
    in0 => in1,  
    out1 => out1);  
end rtl;
```

Szintézis attributumok

- **MARK_DEBUG**

- Belső jelekre, portokra adható meg

```
signal debug_wire : std_logic;  
attribute MARK_DEBUG : string;  
attribute MARK_DEBUG of debug_wire : signal is "TRUE";
```

- A megadott jeleket automatikusan hozzáadja a logikai analizátorhoz a “Configure Debug” beállításánál (tehát nem kell megkeresni a szintetizált huzalozási listában)

Szintézis attributumok

- **FSM_ENCODING**

```
type count_state is (zero, one, two, three, four, five, six, seven);  
signal my_state : count_state;  
attribute fsm_encoding : string;  
attribute fsm_encoding of my_state : signal is "sequential";
```

- Értékek: “one_hot”, “sequential”, “johnson”, “gray”, “auto”, vagy “none”

- **FSM_SAFE_STATE**

```
type count_state is (zero, one, two, three, four, five, six, seven);  
signal my_state : count_state;  
attribute fsm_safe_state : string;  
attribute fsm_safe_state of my_state : signal is "power_on_state";
```

- NEM preferált, inkább kód szinten biztonságos állapotgépet kell írni!

Szintézis attributumok

- **IOB: I/O FF-k használata**

```
signal sig1:std_logic;  
attribute IOB: string;  
attribute IOB of sig1 : signal is "true";
```

- **IO_BUFFER_TYPE**

```
entity example is port(  
  in1 : std_logic_vector (8 downto 0);  
  clk : std_logic;  
  out1 : std_logic_vector(8 downto 0)  
);  
attribute io_buffer_type : string;  
attribute io_buffer_type of out1: signal is "none";  
end example;
```

- none: letiltja a bufferek automatikus példányosítását
- Ha almodulként akarunk beilleszteni egy szintetizált huzalozási listát, akkor kell

Szintézis attributumok

- **RAM_STYLE**

```
type marray is array (511 downto 0) of std_logic_vector(35 downto 0);  
signal memory : marray;  
attribute ram_style : string;  
attribute ram_style of memory : signal is "distributed";
```

- Értékek: „block”, „distributed”, „registers”, „ultra”

- **RAM_DECOMP**

- BRAM “vertikális” kaszkádosítása → nagyobb erőforrás igény, csökkenő fogyasztás

```
type marray is array (2047 downto 0) of std_logic_vector(35 downto 0);  
signal memory : marray;  
attribute ram_decomp : string;  
attribute ram_decomp of myram : signal is "power";
```

- Eredmény: 2 db 1Kx36 BRAMs

Szintézis attributumok

- **ROM_STYLE**

```
attribute rom_style : string;  
attribute rom_style of myrom : signal is distributed;
```

- Értékek: block, dsitributed

Szintézis attributumok

- **USE_DSP**

```
attribute use_dsp : string;  
attribute use_dsp of P_reg : signal is "no";
```

- Tipikusan a szorzást használó aritmetikai funkcióknál automatikusan DSP-t használ (kivéve kis bitszélesség, vagy konstanssal szorzás)
- Csak összeadás DSP-ben implementálásához általában kell az attributum

Szintézis attributumok

- Relatív elhelyezési constraint

```
attribute u_set : string;
attribute u_set of XORCY_L_DW: label is ("SET" & str(NUM, 10));
attribute u_set of REG_OUT_DW: label is ("SET" & str(NUM, 10));
attribute rloc: string;
attribute rloc of XORCY_L_DW : label is "X0Y0";
attribute rloc of REG_OUT_DW : label is "X0Y0";
muxcy_out(0) <= '0';
GEN_MUXCY:
for I in 0 to DW generate
mux_sel(I) <= (not op_a(I)) xor op_b(I); -- op_a != op_b
MUXCY_L_i : MUXCY_L port map(
    LO => muxcy_out(I+1),
    CI => muxcy_out(I),
    DI => op_a(I),
    S => mux_sel(I));
end generate;
XORCY_L_DW: XORCY_L port map(
    LO => sub_res(DW),
    CI => muxcy_out(DW+1),
    LI => mux_sel(DW) );
REG_OUT_DW: FDCE port map(
    Q => res(DW),
    C => clk,
    CE => en,
    CLR => '0',
    D => sub_res(DW));
```


XDC constraint-ek

- **PACKAGE_PIN**

```
set_property PACKAGE_PIN U16 [get_ports clk]
```

- **IOSTANDARD**

```
set_property IOSTANDARD LVCMOS25 [get_ports clk]
```

- **PULLUP, PULLDOWN, KEEPER**

```
set_property PULLUP true [get_ports i2c_scl]
```

- **SLEW: SLOW, FAST, (MEDIUM – US HP bank)**

```
set_property SLEW FAST [get_ports clk_out]
```

- **DRIVE_STRENGTH**

```
set_property DRIVE_STRENGTH 12 [get_ports clk_out]
```

XDC constraint-ek

- **Fizikai területek kijelölése (PBLOCKS)**
 - GUI-n is be lehet rajzolni

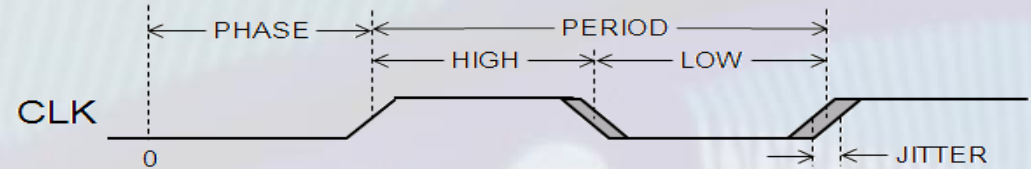
```
create_pblock Pblock_usbEngine
add_cells_to_pblock [get_pblocks Pblock_usbEngine] [get_cells -quiet [list usbEngine1]]
resize_pblock [get_pblocks Pblock_usbEngine] -add {SLICE_X8Y105:SLICE_X23Y149}
resize_pblock [get_pblocks Pblock_usbEngine] -add {DSP48_X0Y42:DSP48_X1Y59}
resize_pblock [get_pblocks Pblock_usbEngine] -add {RAMB18_X0Y42:RAMB18_X1Y59}
resize_pblock [get_pblocks Pblock_usbEngine] -add {RAMB36_X0Y21:RAMB36_X1Y29}
```

- A logika PBLOCK-okba rendezése nem egyértelmű, javíthat az elhelyezésen és időzítésen, de sokszor rosszabb, mint az automatikus place&route!

XDC constraint-ek

- **Elsődleges órajelek**

- 10 ns periódusidő
- Rise @ 0 ns, fall @ 5 ns



```
create_clock -name clk -period 10.0 -waveform {0.0 5.0} [get_ports clk_p]
```

- Differenciális órajel bemenetek esetén a P labra kell megadni.
- **MINDEN BEMENETI ÓRAJELRE KÖTELEZŐ!!!!**
- **Automatikusan generált órajel constraint-ek**
 - PLL, MMCM
 - BUFR

XDC constraint-ek

- **Generated**

- Ha az órajelet egyedi logika generálja (nem javasolt!)

```
process (clk)
begin
if (clk'event and clk='1') then
    reg_d2 <= not reg_d2;
end if;
end process;
```

- Constraint: A FF kimenete felezett frekvenciájú
 - Szintézis során a HDL kód jelneveihez egy “_reg” végződés kerül!

```
create_generated_clock -name clkdiv2 -source [get_ports clk] -divide_by 2 [get_pins reg_d2_reg/Q]
```


XDC constraint-ek

- **Órajel csoportok (clock groups)**
 - Szinkron
 - A különböző órajelek forrása ugyanaz (pl. ugyanaz az oszcillátor)
 - Aszinkron
 - Az egyes órajelek egymáshoz képesti fázisa nem meghatározható
 - Az időzítés analízis 1000 órajel alatt kalkulált legrosszabb értékkel számol



XDC constraint-ek

- **Azinkron órajel csoportok**

- Pl. clk0 és clk1 között

```
set_clock_groups -name async_clk0_clk1 -asynchronous -group {clk0} -group {clk1}
```

- Az időzítés analízis nem vizsgálja a két órajel tartomány közötti adatutakat

XDC constraint-ek

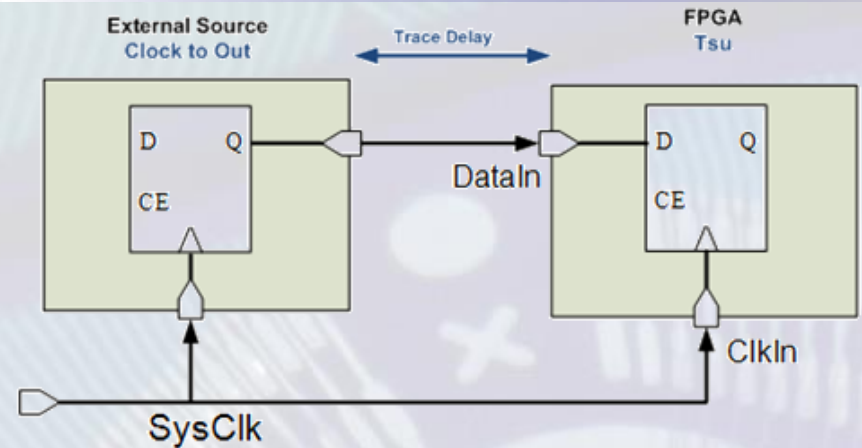
- **Bemeneti órajel jitter**
 - Belül generált órajelekre automatikusan számítódik

```
set_input_jitter [get_clocks -of_objects [get_ports clk]] 0.1
```

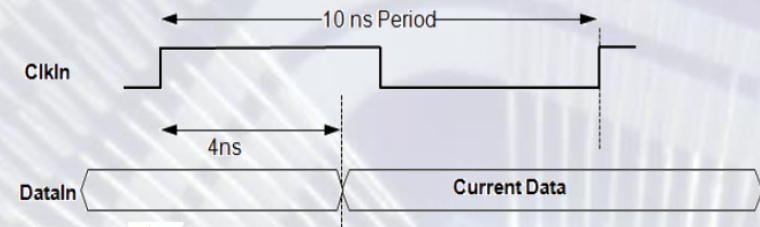
XDC constraint-ek

- **Bemeneti késleltetés**

- A bemeneti órajel és bemeneti adat fázisviszonya
- Lehet min és max értéke

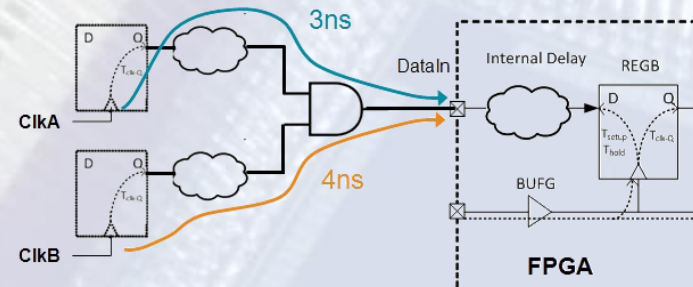


```
create_clock -name SysClk -period 10 [get_ports ClkIn]
set_input_delay -clock SysClk 4 [get_ports DataIn]
```



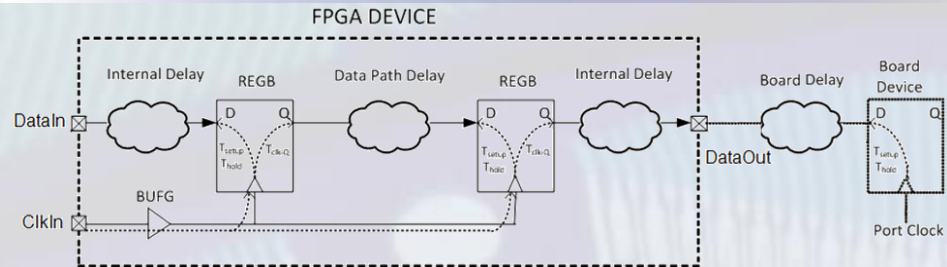
- **Több órajelre**

```
set_input_delay -clock ClkA 3 [get_ports DataIn]
set_input_delay -clock ClkB 4 [get_ports DataIn] -add_delay
```

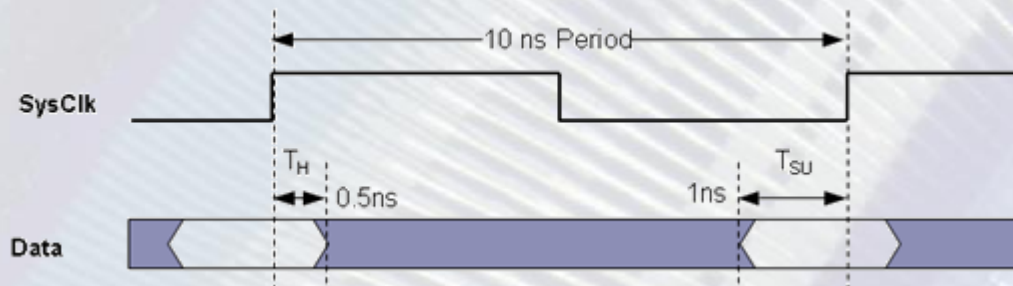


XDC constraint-ek

- **Kimeneti késleltetés**
 - Az FPGA által meghajtott külső eszköz időzítési paramétereit alapján adható meg



```
create_clock -name SysClk -period 10 [get_ports ClkIn]
set_output_delay -clock SysClk 1 [get_ports DataIn]
set_output_delay -clock SysClk -min -0.5 [get_ports DataIn]
```



XDC constraint-ek

- **False path: Időzítés analízisből kizárt adatutak**
 - Megadható órajelek alapján, pl. CLKA és CLKB között

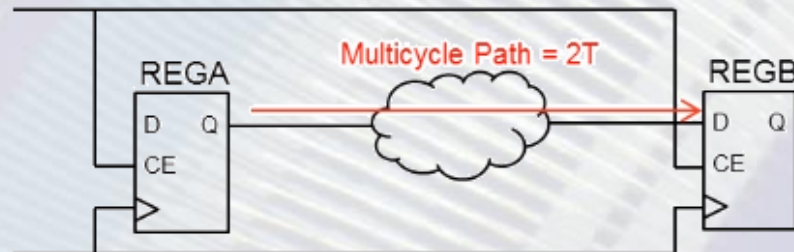
```
set_false_path -from [get_clocks CLKA] -to [get_clocks CLKB]  
set_false_path -from [get_clocks CLKB] -to [get_clocks CLKA]
```

- Vagy konkrét adatutakra

```
set_false_path -through [get_pins MUX1/a0] -through [get_pins MUX2/a1]
```

- **Multi-cycle path: ÓVATOSAN!**

- Olyan adatutak, amelyekben nem minden órajelben frissül a cél és forrás FF (pl. minden második órajelben van CE)



```
set_multicycle_path -from REGA/CLK to REGB/D 2  
set_multicycle_path -from REGA/CLK to REGB/D -hold 1
```