Szintézis attributumok és constraint-ek

Szintézis attributumok

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

```
(* black_box *) module mux_41 (input in0, in1, in2, in3, input [1:0] sel, output reg r);
```

- **CLOCK_BUFFER_TYPE**

```
module shr_module (
        (* clock_buffer_type = “none” *) input clk,
        input sh,
        input din,
        output dout
);
```

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

Szintézis attributumok

- **EXTRACT_RESET: FF reset bemenet használat**

```
(* extract_reset = "yes" *) reg [7:0] counter;
```

- **DIRECT_RESET: reset jel direct megadása**

```
module cntr (
        input clk,
        (* direct_reset = "yes" *) input rst,
        input ce,
        output [7:0] q);
```

- **EXTRACT_ENABLE: FF CE bemenet használat**

```
(* extract_enable = "yes" *) reg [7:0] counter;
```

- **DIRECT_ENABLE: CE jel direct megadása**

```
module cntr (
        input clk,
        input rst,
        (* direct_enable = "yes" *) input ce,
        output [7:0] q);
```

Szintézis attributumok

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

```
(* ASYNC_REG = “TRUE” *) reg [2:0] sync_regs;
```

Szintézis attributumok

- **SHREG_EXTRACT: shift regiszter felismerés**

```
(* shreg_extract = “no” *) reg [16:0] shift_reg;
```

- **SRL_STYLE: shift regiszter implementáció**

```
(* srl_style = “register” *) reg [16:0] shift_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

- **KEEP**
  - Hatása cask a szintézisre van, belső jeleken használható

```
(* keep = “true” *) wire sig1;
assign sig1 = in1 & in2;
assign out1 = sig1 & in2;
```

Szintézis attributumok

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

```
(* keep_hierarchy = “yes” *) module test_module (
                input in1,
                input in2,
                output out1);
```

  - Vagy példányosításnál

```
(* keep_hierarchy = “yes” *) test_module  inst_0 (
                .in1(in1),
                .in2(in2),
                .out1(temp1)
);
```

Szintézis attributumok

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

```
(* dont_touch = “yes” *) wire sig1;
assign sig1 = in1 & in2;
assign out1 = sig1 & in2;
```

  - Vagy entity-n/module-on

```
(* DONT_TOUCH = “yes” *) module test_module (
                input clk,
                input in1,
                input in2,
                output out1);
```

Szintézis attributumok

- **MARK_DEBUG**
  - Belső jelekre, portokra adható meg

```
(* MARK_DEBUG = “TRUE” *) wire debug_wire;
```

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

Szintézis attributumok

- **FSM_ENCODING**

```
(* fsm_encoding = “one_hot” *) reg [7:0] my_state;
```

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

```
(* fsm_safe_state = “reset_state” *) reg [7:0] my_state;
```

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

Szintézis attributumok

- **IOB: I/O FF-k használata**

```
(* IOB = “true” *) reg input_reg;
```

- **IO_BUFFER_TYPE**

```
module test_module (
        (* io_buffer_type = “none” *) input in1,
        (* io_buffer_type = “none” *) input in2,
        (* io_buffer_type = “none” *) output out1);
```

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

Szintézis attributumok

- **RAM_STYLE**

```
(* ram_style = “distributed” *) reg [data_size-1:0] memory [2**addr_size-1:0];
```

  - Értékek: „block”, „distributed”, „registers”, „ultra”
- **RAM_DECOMP**
  - BRAM “vertikális” kaszkádosítása → nagyobb erőforrás igény, csökkenő fogyasztás

```
(* ram_decomp = “power” *) reg [35:0] memory[2047:0];
```

  - Eredmény: 2 db 1Kx36 BRAMs

Szintézis attributumok

- **ROM_STYLE**

```
(* rom_style = “distributed” *) reg [data_size-1:0] rom_mem [2**addr_size-1:0];
```

  - Értékek: block, dsitributed

Szintézis attributumok

- **USE_DSP**

```
(* use_dsp = “yes” *) module adder(clk, in1, in2, out);
```

```
(* use_dsp = “yes” *) reg[47:0] add_reg;
```

  - Tipikusan a szorzást használó aritmetikai funkcióknál automatikusan DSP-t használ (kivéve kis bitszélesség, vagy konstanssal szotzás)
  - 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

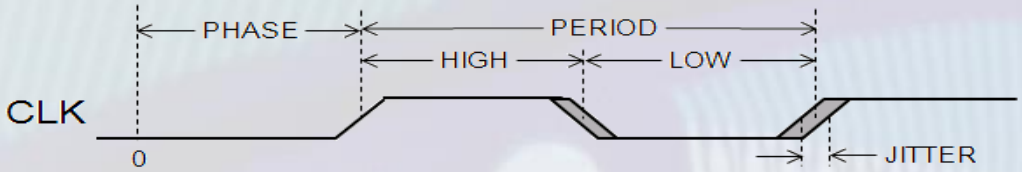
- **Fizikai területek kijelölése (PBLOCKS)**
  - GUI-n is be lehet rajzolni

```
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 egymáshjoz 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

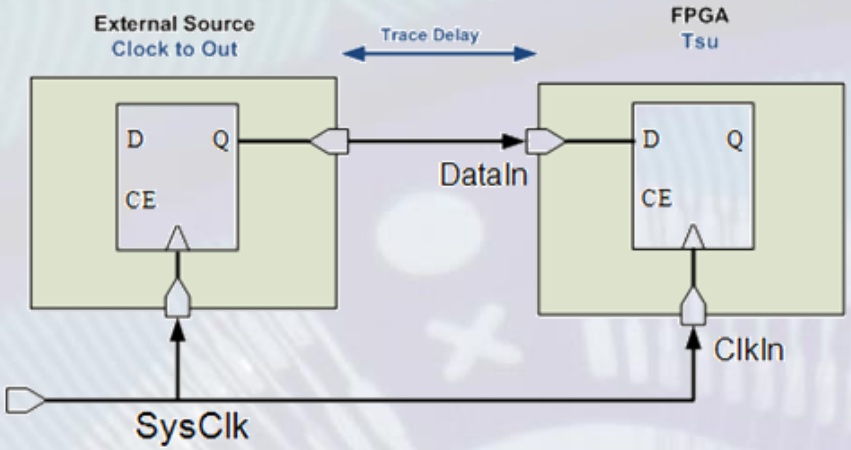
XDC constraint-ek

- **Bemeneti órajel jitter**
  - FPGA-ban generált órajelekre automatikusan számítódik

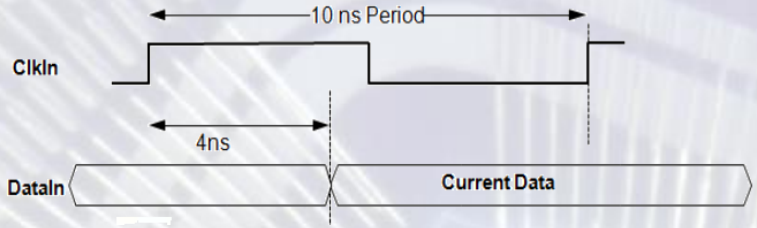
```
set_input_jitter [get_clocks -of_objects [get_ports clk]] 0.1
```

XDC constraint-ek

- **Bemeneti késleltetés**
  - A bemeneti órajel és bemeneti adat fázisviszonya
  - Lehet min és max értéke

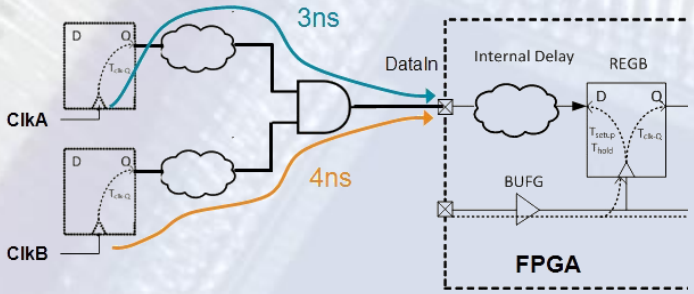



```
create_clock –name SysClk –period 10 [get_ports ClkIn]
set_input_delay –clock SysClk 4 [get_ports DataIn]
```



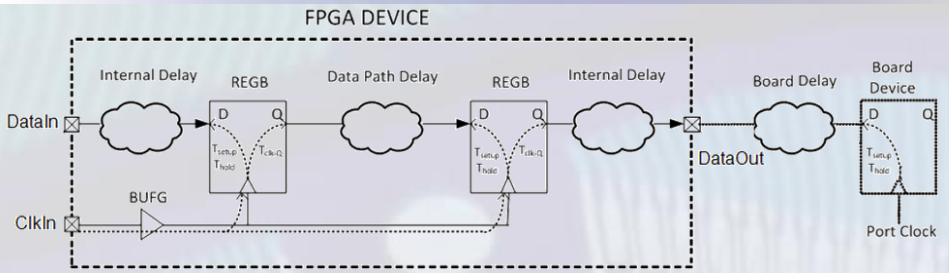
- Több órajelre

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

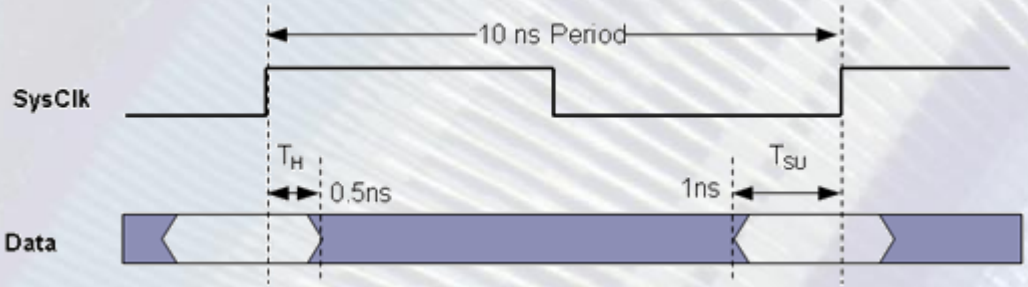


XDC constraint-ek

- **Kimeneti késleltetés**
  - Az FPGA által meghajtott külső eszköz időzítési paraméterei 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

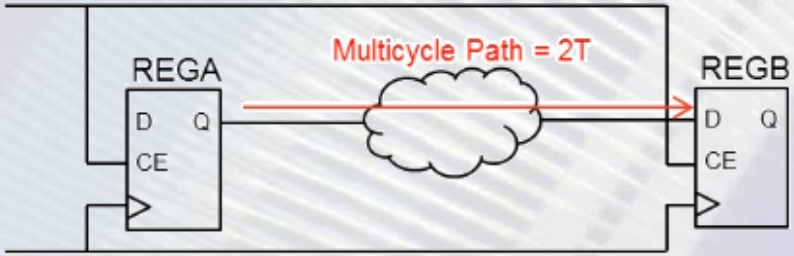
- **False path: Időzítés analízisből kizárt adatutak**
  - Megadható órajelek alapján, pl. CLKA és CLKB között

```
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
```