Programmable Logic Design

Grzegorz Budzyń

Lecture 9: Attributes and Constraints

Choose yourself and new technologies

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!

Wrocław University of Technology

EUROPEAN SOCIAL FUND

Project co-financed from the EU European Social Fund

Plan

- Introduction
  - Attributes vs Constraints
- UCF
- Constraints in FPGAs

Introduction

Project co-financed from the EU European Social Fund

Attributes vs Constraints

- There are two classes of attributes:
  - predefined as a part of the 1076 standard,
  - introduced outside of the standard:
    - by the designer
    - by the design tool supplier (like Xilinx or Altera)

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

Custom Xilinx attributes

- Attributes vs Constraints:
  - Names can be used interchangably for custom attributes
  - We can define subcategories:
    - Attributes - property associated with a device architecture primitive component that generally affects an instantiated component's functionality or implementation
    - Synthesis constraints - direct the synthesis tool optimization technique for a particular design or piece of HDL code
    - Implementation constraints - instructions given to the FPGA implementation tools to direct the mapping, placement, timing, etc. for the implementation tools

FPGA constraints

- FPGA constraints can ge split into:
  - **Grouping Constraints**
  - **Logical Constraints**
  - **Physical Constraints**
  - **Mapping Directives**
  - **Placement Constraints**
  - **Routing Directives**
  - **Synthesis Constraints**
  - **Timing Constraints**
  - **Configuration Constraints**

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

Constraints usage in VHDL

- Before a constraint can be used, it must be declared with the following syntax:
  - **attribute** *attribute_name* : **string;**
- Example:
  - **attribute RLOC : string;**

- Once the attribute is declared, a VHDL attribute can be specified as:
  - **attribute bufg of my_clock: signal is "clk";**

Constraints usage in VHDL

- An attribute can be declared in an entity or architecture.
  - If the attribute is declared in the entity, it is visible both in the entity and the architecture body.
  - If the attribute is declared in the architecture, it cannot be used in the entity declaration.

Where to define constraints

- Depending on a attribute they can be defined in different places:
  – Constraints Editor – *Timing Constraints*
  – Floorplanner – *Non-timing placement constraints*
  – PACE – *IO placement and area constraints*
  – Floorplan Editor - *IO placement and area constraints*
  – Schematic and Symbol Editors - *IO placement and RLOC constraints*

UCF / NCF

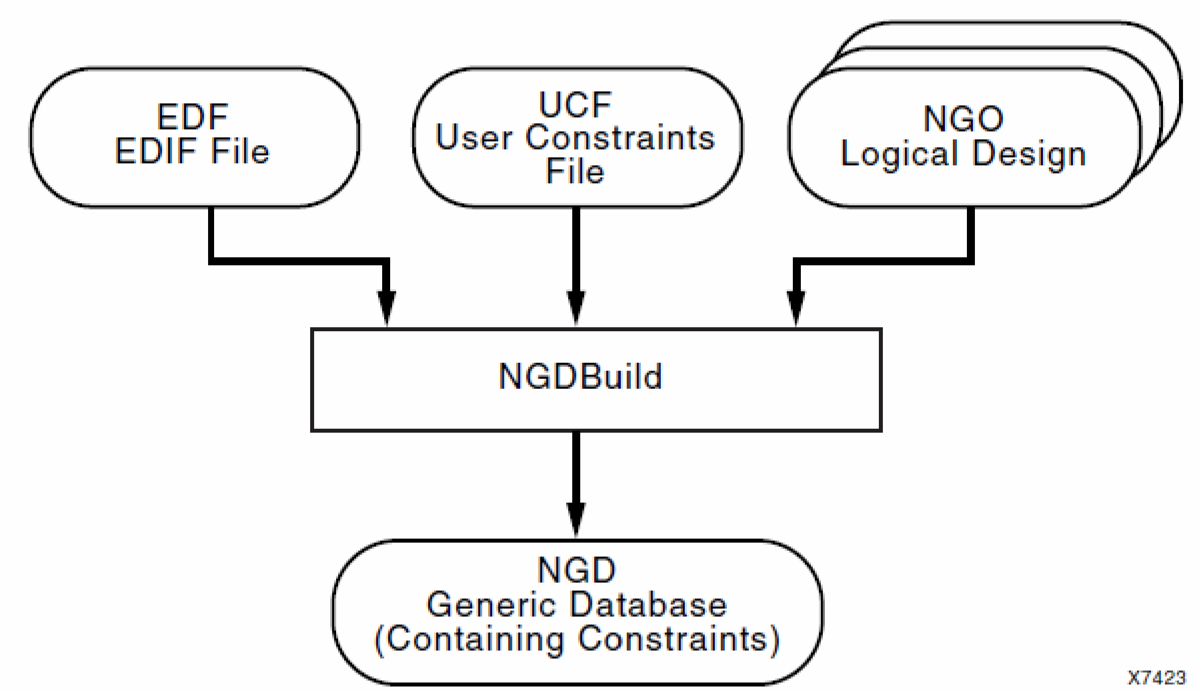
HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

User Constraints File

- The UCF file is an ASCII file specifying constraints on the logical design
- The UCF file can be created with any text editor

- NCF - Netlist Constraint File, an ASCII file generated by synthesis programs

UCF file flow



UCF/NCF rules

- The UCF and NCF files are case sensitive.
- However, any Xilinx constraint keyword (for example, LOC,PERIOD, HIGH, LOW) may be entered in all upper-case, all lower-case, or mixed case.
- Each statement is terminated by a semicolon (;)
- No continuation characters are necessary if a statement exceeds one line, since a semicolon marks the end of the statement

UCF/NCF rules

- To comment a line a pound sign (#) is used
- C and C++ style comments (/* */ and respectively) are supported
- Statements need not be placed in any particular order in the UCF and NCF file
- Enclose inverted signal names that contain a tilde (for example, ~OUTSIG1) in double quotes
- Multiple constraints for a given instance can be entered

UCF/NCF example

- The UCF file supports a basic syntax that can be expressed as:
  - **{NET|INST|PIN}** *“full_name”* *constraint***;**

or as

  - **SET** *set_name set_constraint***;**

- Example:

```
INST DCM_INST CLK_FEEDBACK = NONE;
INST DCM_INST CLKDV_DIVIDE = 2.0;
INST SDA_pin      LOC=P44;
```

UCF/NCF addons

- If using Reserved Words (like „net”) double quote is mandatory
- Wildcard characters, asterisk (*) and question mark (?) can be used in as follows:
  - The asterisk (*) represents any string of zero or more characters.
  - The question mark (?) indicates a single character
- Example:
  - `NET "*AT?" FAST;`

UCF/NCF addons

- If multiple constraints necessary then:
  - **INST** *instanceName constraintName* **=** *constraintValue* | *constraintName* **=***constraintValue***;**
- For example:
  - `INST myInst LOC = P53 | IOSTANDARD = LVPECL33 | SLEW = FAST;`

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

PCF

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

PCF

- The NGD file produced when a design netlist is read into the Xilinx Development System may contain a number of logical constraints from:
  - UCF
  - NCF
  - VHDL code
- Logical constraints in the NGD file are read by MAP. MAP uses some of the constraints to map the design and converts logical constraints to physical constraints.

PCF

- MAP writes these physical constraints into a Physical Constraints File (PCF)
- The PCF file is an ASCII file containing two separate sections:
  - A section for those physical constraints created by the mapper
  - A section for physical constraints entered by the user

PCF

- The structure of the PCF file is as follows.
  - schematic start;
  - *translated schematic and UCF and NCF constraints in PCF format*
  - schematic end;
  - *user-entered physical constraints*
- ***Caution!*** All user-entered physical constraints MUST be put after the "schematic end" statement. *Any constraints preceding this section or within this section may be overwritten or ignored.*

PCF example

- UCF file:
  - INST LED1 LOC=P65;
  - INST LED2 LOC=P64;
- PCF file
  - COMP "LED1" LOCATE = SITE "P65" LEVEL 1;
  - COMP "LED2" LOCATE = SITE "P64" LEVEL 1;

Constraints Editor

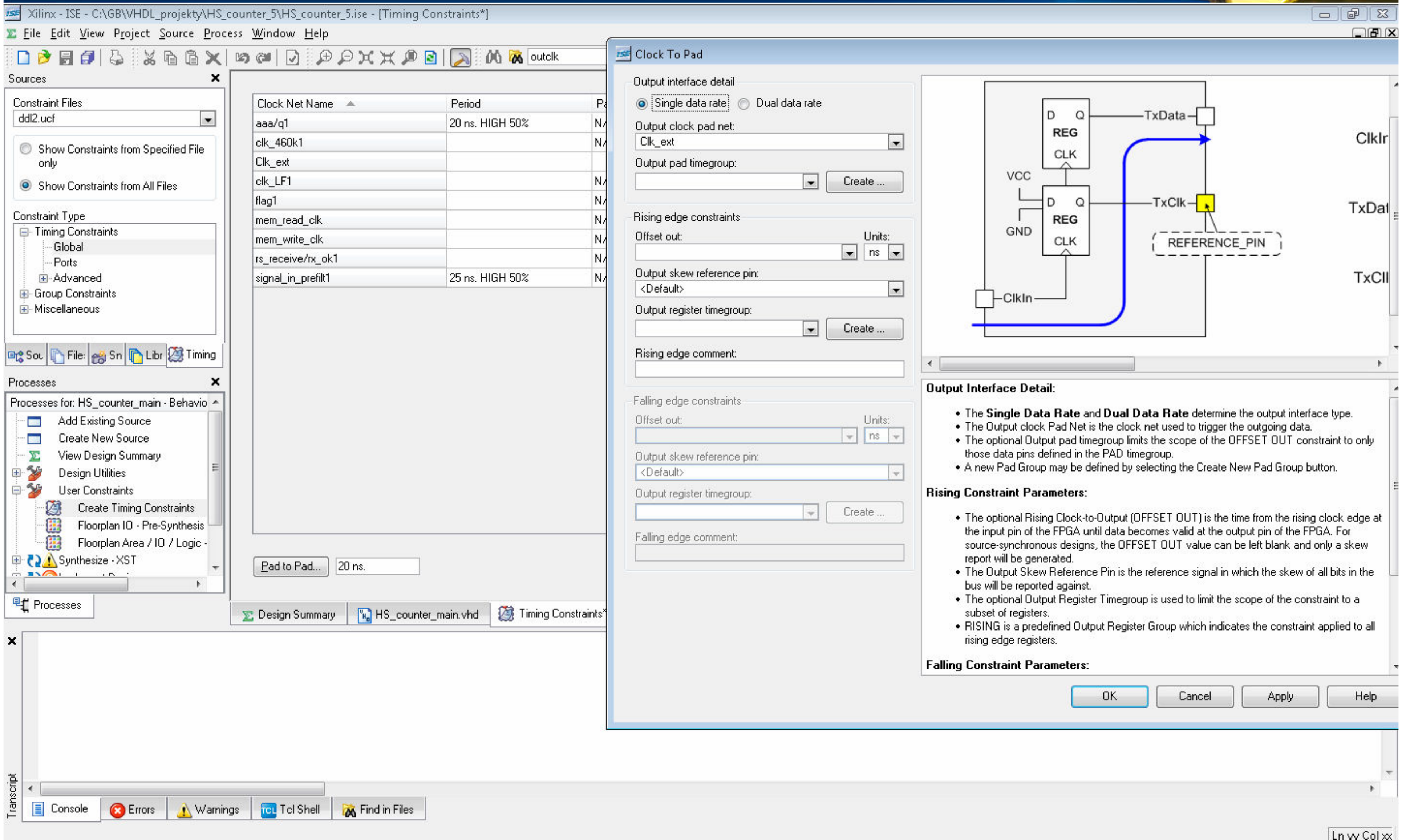
HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

Constraints Editor

- Constraints Editor can be found in the ISE environment
- Constraints Editor is used to enter timing constraints
- The user interface simplifies constraint entry by guiding through constraint creation without needing to understand UCF file syntax.

Constraints Editor

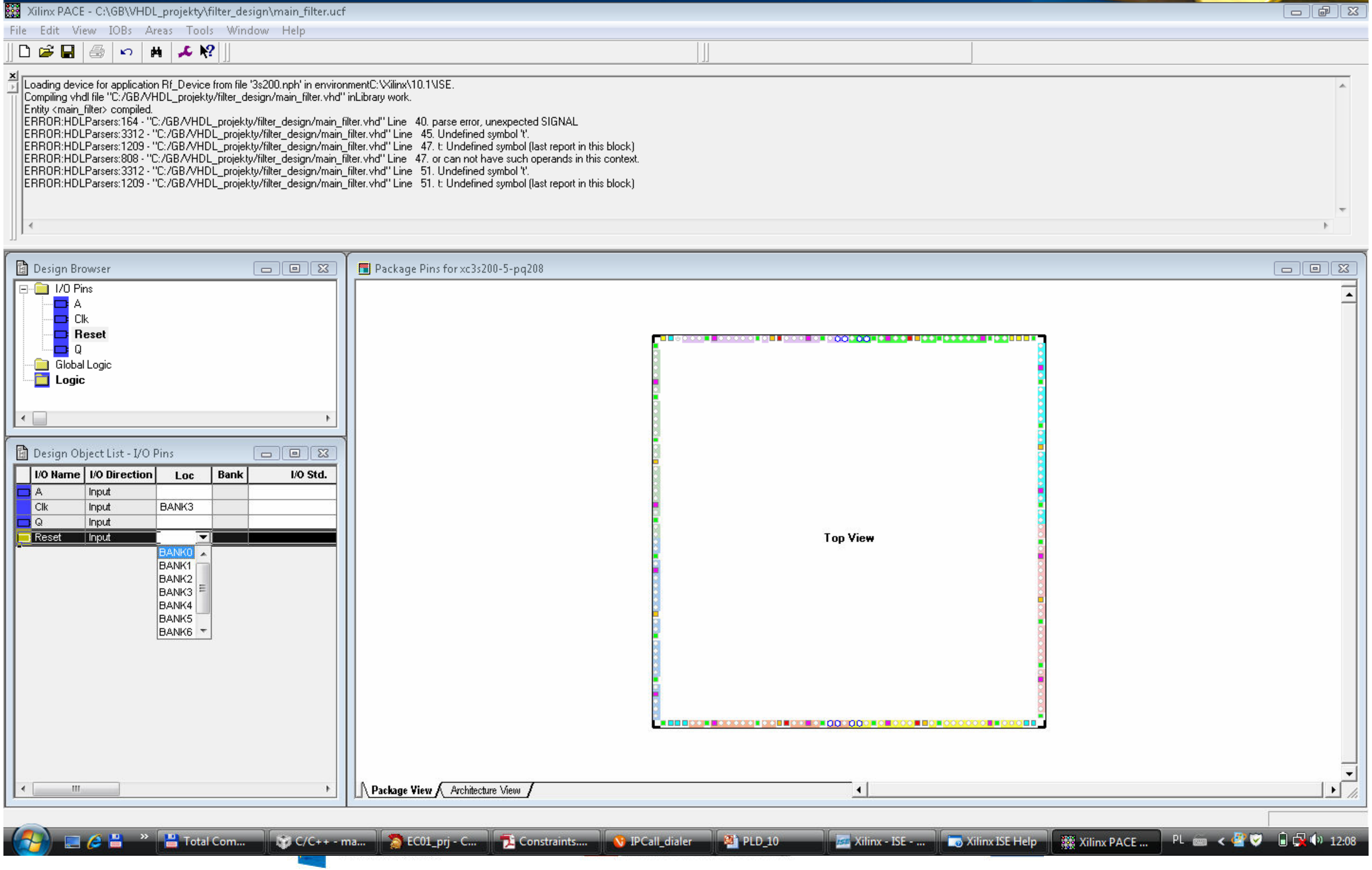
- Constraints Editor requires:
  - A User Constraints File (UCF)
  - A Native Generic Database (NGD) file
- After the constraints are created or modified with Constraints Editor, NGDBuild must be run again, using the new UCF and design source netlist files as input and generating a new NGD file as output


Xilinx - ISE - C:\GB\VHDL_projekty\HS_counter_5\HS_counter_5.ise - [Timing Constraints*]
File Edit View Project Source Process Window Help
Sources
Constraint Files
ddl2.ucf
Show Constraints from Specified File only
Show Constraints from All Files
Constraint Type
Timing Constraints
Global
Ports
Advanced
Group Constraints
Miscellaneous
Processes
Processes for: HS_counter_main - Behavio
Add Existing Source
Create New Source
View Design Summary
Design Utilities
User Constraints
Create Timing Constraints
Floorplan IO - Pre-Synthesis
Floorplan Area / IO / Logic -
Synthesize - XST
Clock Net Name
Period
aaa/q1
20 ns. HIGH 50%
clk_460k1
Clk_ext
clk_LF1
flag1
mem_read_clk
mem_write_clk
rs_receive/rx_ok1
signal_in_prefilt1
25 ns. HIGH 50%
Pad to Pad...
20 ns.
Design Summary
HS_counter_main.vhd
Timing Constraints*
Clock To Pad
Output interface detail
Single data rate
Dual data rate
Output clock pad net:
Clk_ext
Output pad timegroup:
Create ...
Rising edge constraints
Offset out:
Units:
ns
Output skew reference pin:
<Default>
Output register timegroup:
Rising edge comment:
Falling edge constraints
D Q
REG
CLK
TxData
VCC
GND
TxClk
REFERENCE_PIN
ClkIn
Output Interface Detail:
The Single Data Rate and Dual Data Rate determine the output interface type.
The Output clock Pad Net is the clock net used to trigger the outgoing data.
The optional Output pad timegroup limits the scope of the OFFSET OUT constraint to only those data pins defined in the PAD timegroup.
A new Pad Group may be defined by selecting the Create New Pad Group button.
Rising Constraint Parameters:
The optional Rising Clock-to-Output (OFFSET OUT) is the time from the rising clock edge at the input pin of the FPGA until data becomes valid at the output pin of the FPGA. For source-synchronous designs, the OFFSET OUT value can be left blank and only a skew report will be generated.
The Output Skew Reference Pin is the reference signal in which the skew of all bits in the bus will be reported against.
The optional Output Register Timegroup is used to limit the scope of the constraint to a subset of registers.
RISING is a predefined Output Register Group which indicates the constraint applied to all rising edge registers.
Falling Constraint Parameters:
OK
Cancel
Apply
Help
Console
Errors
Warnings
Tcl Shell
Find in Files
Transcript
Ln xx Col xx

Floorplanner

Floorplanner

- Floorplanner is a part of the ISE environment
- It is used for defining parameters of input/output signals and for defining the positioning of the logic nets inside the chip
- **Easy** way of making area constraints
- Floorplanner can be used after translation and/or before mapping
- Floorplanner reads the UCF file constraints


Xilinx PACE - C:\GB\VHDL_projekty\filter_design\main_filter.ucf
File Edit View IOBs Areas Tools Window Help
Loading device for application Rf_Device from file '3s200.nph' in environmentC:\Xilinx\10.1\ISE.
Compiling vhdl file "C:/GB/VHDL_projekty/filter_design/main_filter.vhd" inLibrary work.
Entity <main_filter> compiled.
ERROR:HDLParsers:164 - "C:/GB/VHDL_projekty/filter_design/main_filter.vhd" Line 40. parse error, unexpected SIGNAL
ERROR:HDLParsers:3312 - "C:/GB/VHDL_projekty/filter_design/main_filter.vhd" Line 45. Undefined symbol 't'.
ERROR:HDLParsers:1209 - "C:/GB/VHDL_projekty/filter_design/main_filter.vhd" Line 47. t: Undefined symbol (last report in this block)
ERROR:HDLParsers:808 - "C:/GB/VHDL_projekty/filter_design/main_filter.vhd" Line 47. or can not have such operands in this context.
ERROR:HDLParsers:3312 - "C:/GB/VHDL_projekty/filter_design/main_filter.vhd" Line 51. Undefined symbol 't'.
ERROR:HDLParsers:1209 - "C:/GB/VHDL_projekty/filter_design/main_filter.vhd" Line 51. t: Undefined symbol (last report in this block)
Design Browser
I/O Pins
A
Clk
Reset
Q
Global Logic
Logic
Design Object List - I/O Pins
I/O Name
I/O Direction
Loc
Bank
I/O Std.
A Input
Clk Input BANK3
Q Input
Reset Input
BANK0
BANK1
BANK2
BANK3
BANK4
BANK5
BANK6
Package Pins for xc3s200-5-pq208
Top View
Package View
Architecture View
Total Com...
C/C++ - ma...
EC01_prj - C...
Constraints....
IPCall_dialer
PLD_10
Xilinx - ISE - ...
Xilinx ISE Help
Xilinx PACE ...
PL
12:08

Floorplanner

- Floorplanner™ interactive graphical tool can be used to perform the following functions:
  - Doing detailed-level floorplanning
  - Creating an RPM core that can be used in other designs
  - Viewing and editing location constraints
  - Finding logic or nets by name or connectivity
  - Cross probing from the Timing Analyzer to the Floorplanner
  - Placing ports automatically for modular design

RPM

- RPM – Relationally Placed Macro
- Defines the spatial relationship of the primitives that comprise the RPM
- Can be reused between designs

FPGA Editor

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

FPGA Editor

- In the FPGA Editor certain constraints can be added or deleted from the PCF (Physical Constraints File)
- In the FPGA Editor, net, site, and component constraints are supported as property fields in the individual nets and components

FPGA Editor function 1/2

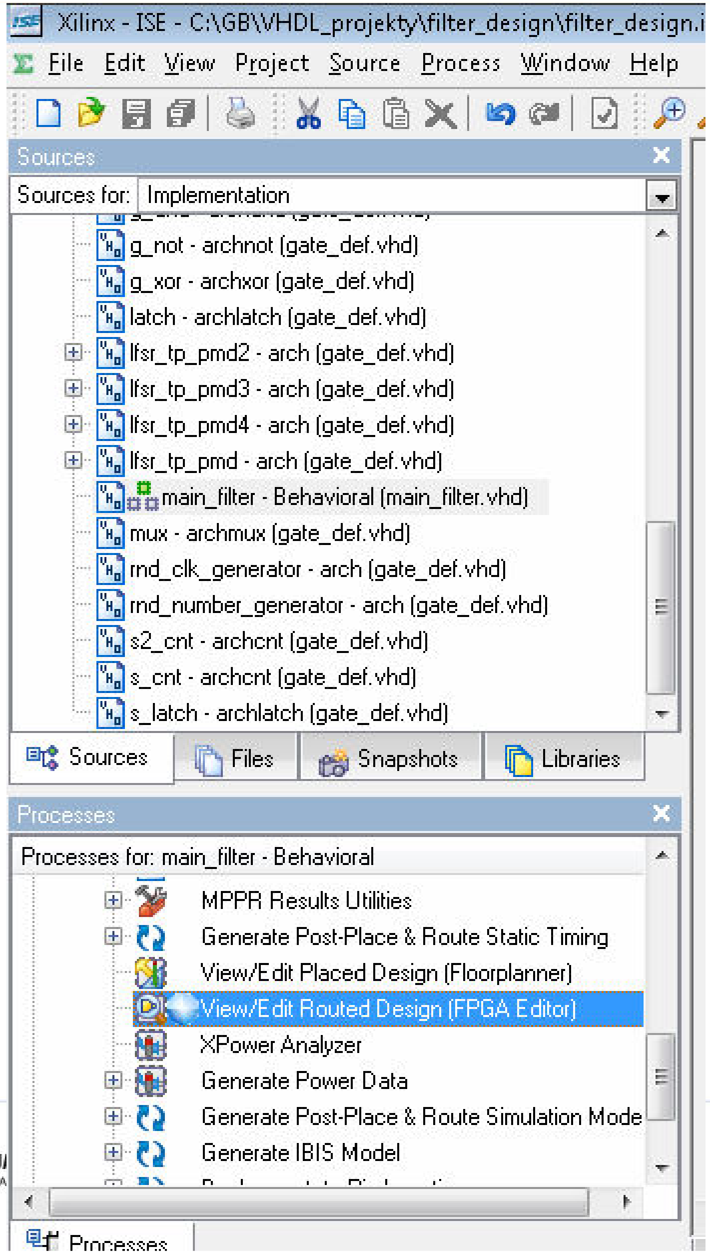
- Functions that can be performed in the FPGA Editor:
  - Placing and routing critical components before running the automatic place and route tools.
  - Finishing placement and routing if the routing program does not completely route the design.
  - Adding probes to the design to examine the signal states of the targeted device. Probes are used to route the value of internal nets to an IOB for analysis during the debugging of a device.
  - Cross-probing the design with Timing Analyzer.

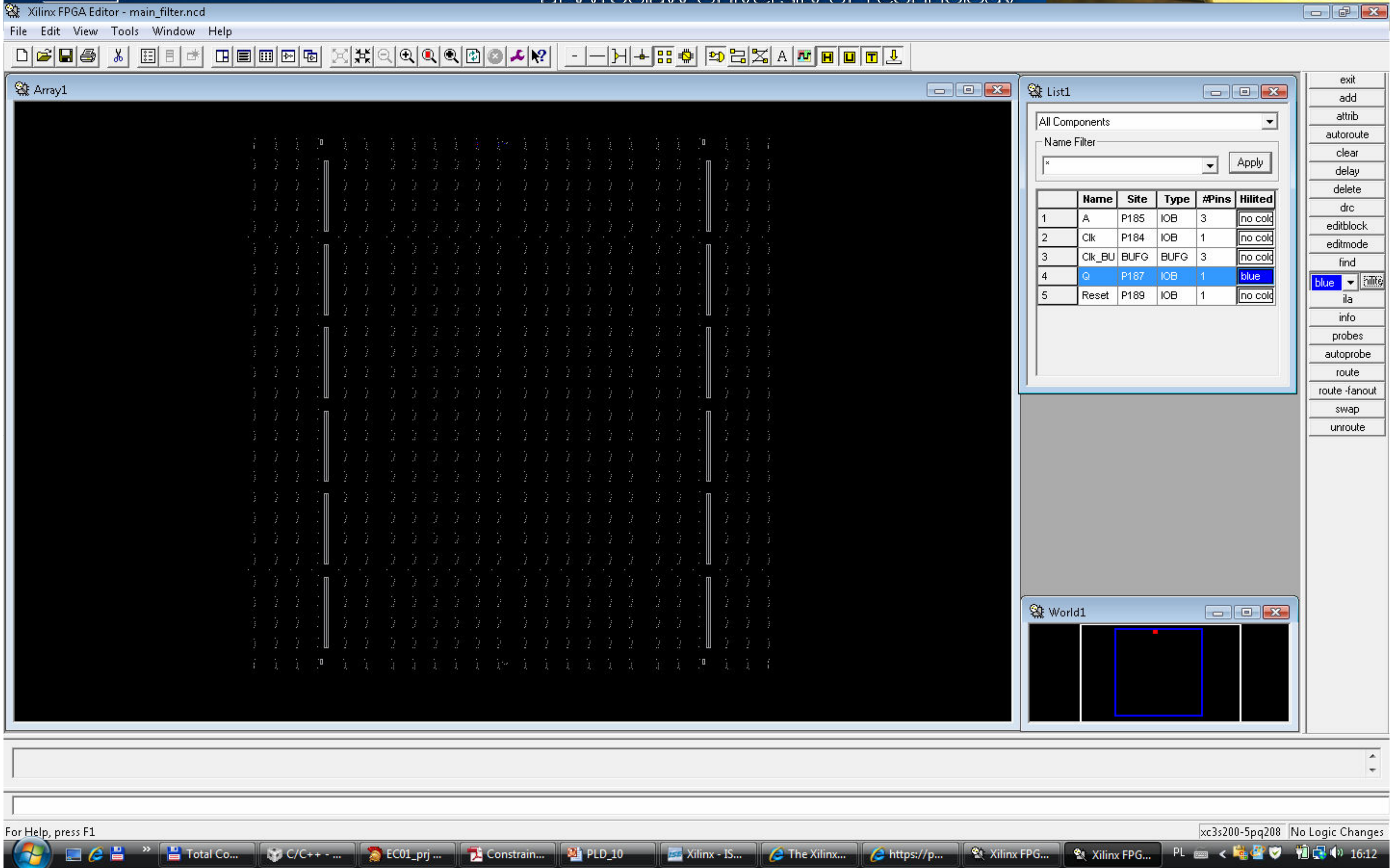
HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

FPGA Editor function 2/2

- Functions that can be performed in the FPGA Editor:
  - Running the BitGen program and downloading the resulting BIT file to the targeted device.
  - Viewing and changing the nets connected to the capture units of an ILA core in your design.
  - Using the ILA command to write a .cdc file
  - Creating an entire design by hand (advanced users)
  - Modifing some constraints

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund


Xilinx - ISE - C:\GB\VHDL_projekty\filter_design\filter_design.i
File Edit View Project Source Process Window Help
Sources
Sources for: Implementation
g_not - archnot (gate_def.vhd)
g_xor - archxor (gate_def.vhd)
latch - archlatch (gate_def.vhd)
lfsr_tp_pmd2 - arch (gate_def.vhd)
lfsr_tp_pmd3 - arch (gate_def.vhd)
lfsr_tp_pmd4 - arch (gate_def.vhd)
lfsr_tp_pmd - arch (gate_def.vhd)
main_filter - Behavioral (main_filter.vhd)
mux - archmux (gate_def.vhd)
rnd_clk_generator - arch (gate_def.vhd)
rnd_number_generator - arch (gate_def.vhd)
s2_cnt - archcnt (gate_def.vhd)
s_cnt - archcnt (gate_def.vhd)
s_latch - archlatch (gate_def.vhd)
Sources
Files
Snapshots
Libraries
Processes
Processes for: main_filter - Behavioral
MPPR Results Utilities
Generate Post-Place & Route Static Timing
View/Edit Placed Design (Floorplanner)
View/Edit Routed Design (FPGA Editor)
XPower Analyzer
Generate Power Data
Generate Post-Place & Route Simulation Mode
Generate IBIS Model
Processes


Xilinx FPGA Editor - main_filter.ncd
File Edit View Tools Window Help
Array1
List1
All Components
Name Filter
Apply
Name Site Type #Pins Hilited
1 A P185 IOB 3 no colo
2 Clk P184 IOB 1 no colo
3 Clk_BU BUFG BUFG 3 no colo
4 Q P187 IOB 1 blue
5 Reset P189 IOB 1 no colo
exit
add
attrib
autoroute
clear
delay
delete
drc
editblock
editmode
find
blue
ila
info
probes
autoprobe
route
route -fanout
swap
unroute
World1
For Help, press F1
xc3s200-5pq208
No Logic Changes

Timing Constraint Strategies

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

Timing Constraints Strategies

- The easiest way to modify time critical path is to use *global coinstraints*
- Main timing constraints methods:
  - Global Timing Assignments
  - Specific Timing Assignments
  - Multi-Cycle and Fast or Slow Timing Assignments

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

Xilinx Constraint

Xilinx Constraints

- FPGA constraints can ge split into:
  - **Grouping Constraints**
  - **Logical Constraints**
  - **Physical Constraints**
  - **Mapping Directives**
  - **Placement Constraints**
  - **Routing Directives**
  - **Synthesis Constraints**
  - **Timing Constraints**
  - **Configuration Constraints**

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

AREA_GROUP constraint

- AREA_GROUP is a design implementation constraint that enables partitioning of the design into physical regions for mapping, packing, placement, and routing
- AREA_GROUP is attached to logical blocks in the design, and the string value of the constraint identifies a named group of logical blocks that are to be packed together by mapper and placed in the ranges if specified by PAR

AREA_GROUP constraint

- Applicable to:
  - Logic groups
  - Timing groups
- Syntax example:
  - **INST “*X*” AREA_GROUP=*groupname*;**
  - AREA_GROUP “groupname” RANGE=range;

BLKNM constraint

- BLKNM (Block Name) assigns block names to qualifying primitives and logic elements
- If the same BLKNM constraint is assigned to more than one instance, the software attempts to map them into the same block
- Placing BLKNM constraints on instances that do not fit within one block creates an error

Project co-financed from the EU European Social Fund

BLKNM constraint

- Applicable to:
  - Flip-flop and latch primitives
  - Any I/O element or pad
  - ROM primitives
  - RAMS and RAMD primitives
  - Carry logic primitives
- Syntax Example:
  - INST “$1I87/block1” BLKNM=U1358;

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

BUFG (CPLD) constraint

- When applied to an input buffer or input pad net, the BUFG attribute maps the tagged signal to a global net.
- When applied to an internal net, the tagged signal is either routed directly to a global net or brought out to a global control pin to drive the global net, as supported by the target device family architecture

BUFG constraint

- Applicable to (CPLD only):
  - Any input buffer (IBUF),
  - input pad net,
  - internal net that drives a CLK, OE, SR,
  - DATA_GATE pin
- Syntax Example:
  - NET “fastclk” BUFG=CLK;

DCI_VALUE constraint

- DCI_VALUE determines which buffer behavioral models are associated with the IOBs of a design in the generation of an IBS file using IBISWriter
- Applicable to:
  - IOB
- Syntax Example:
  - INST pin_name DCI_VALUE = integer;
    - DCI_VALUEare integers 25 through 100 with an implied units of ohms (default 50 ohms)

DRIVE constraint

- DRIVE is a basic mapping directive that selects the output for SPARTAN and VIRTEX devices
- DRIVE selects output drive strength (mA) for the SelectIO buffers that use the:
  - LVTTL, LVCMOS12, LVCMOS15, LVCMOS18, LVCMOS25, or LVCMOS33 interface I/O standard.

FAST constraint

- FAST is a basic mapping constraint. It increases the speed of an IOB output. While FAST produces a faster output, it may increase noise and power consumption.
- Applicable to:
  - Output primitives
  - Output pads
  - Bidirectional pads

FROM TO constraint

- FROM-TO defines a timing constraint between two groups.
- It is associated with the Period constraint of the high or low time.
- A group can be user-defined or predefined.
- From synchronous paths, a FROM-TO constraint controls only the setup path, not the hold path.

FROM TO constraint

- Applicable to:
  – Predefined and user-defined groups
- Syntax Example:
  – **TIMESPEC TS***name***=FROM "***group1***" TO "***group2***"** *value* **{DATAPATHONLY};**
  – **TIMESPEC TS_MY_PathA = FROM "my_src_grp" TO "my_dst_grp" 23.5 ns DATAPATHONLY;**

IBUF_DELAY_VALUE constraint

- The IBUF_DELAY_VALUE constraint is a mapping constraint that adds additional static delay to the input path of the FPGA array
- This constraint can be applied to any input or bidirectional signal that is not directly driving a clock or IOB (Input Output Block) register
- The IBUF_DELAY_VALUE constraint can be set to an integer value from 0-16
- These values do not directly correlate to a unit of time but rather additional buffer delay

Project co-financed from the EU European Social Fund

IBUF_DELAY_VALUE constraint

- Applicable to:
  - Any top-level I/O port
- Syntax Example:
  - **attribute IBUF_DELAY_VALUE : string;**
  - **attribute IBUF_DELAY_VALUE of DataIn1: label is "5";**

IFD_DELAY_VALUE constraint

- The IFD_DELAY_VALUE constraint is a mapping constraint that adds additional static delay to the input path of the FPGA array
- This constraint can be applied to any input or bidirectional signal which drives an IOB (Input Output Block) register
- The IFD_DELAY_VALUE constraint can be set to an integer value from 0-8
- These values do not directly correlate to a unit of time but rather additional buffer delay

Project co-financed from the EU European Social Fund

IFD_DELAY_VALUE constraint

- Applicable to:
  - Any top-level I/O port
- Syntax Example:
  - **attribute IFD_DELAY_VALUE : string;**
  - **attribute IFD_DELAY_VALUE of DataIn1: label is "5";**

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

IOSTANDARD constraint

- IOSTANDARD is a basic mapping constraint and synthesis constraint
- IOSTANDARD is used to assign an I/O standard to an I/O primitive
- IOSTANDARD works differently for FPGA and CPLD devices

IOSTANDARD constraint

- Applicable to:
  – IBUF, IBUFG, OBUF, OBUFT
  – IBUFDS, IBUFGDS, OBUFDS, OBUFTDS
  – Output Voltage Banks
- Syntax Example:
  – **INST "***instance_name***" IOSTANDARD=***iostandard_name***;**
  – **NET "***pad_net_name***" IOSTANDARD=***iostandard_name***;**
    - *Where iostandard_name* is an IO Standard name as specified in the device data sheet

KEEP constraint

- When a design is mapped, some nets may be absorbed into logic blocks.
- When a net is absorbed into a block, it can no longer be seen in the physical design database
- The net may then be absorbed into the block containing the components.
- KEEP prevents this from happening

KEEP constraint

- Applicable to:
  - signals
- Syntax Example:
  - **attribute keep of** ***signal_name*****: signal is "{TRUE|FALSE}";**
  - **NET "$1I3245/$SIG_0" KEEP;**

LOC constraint

- LOC defines where a design element can be placed within an FPGA
- It specifies the absolute placement of a design element on the FPGA die
- It can be a single location, a range of locations, or a list of locations
- LOC can be specified from the design file and also direct placement with statements in a constraints file

NOREDUCE constraint

- NOREDUCE is a fitter and synthesis constraint for CPLD devices
- It prevents minimization of redundant logic terms that are typically included in a design to avoid logic hazards or race conditions
- When constructing combinatorial feedback latches in a design, always apply NOREDUCE to the latch’s output net and include redundant logic terms when necessary to avoid race conditions.

NOREDUCE constraint

- Applicable to:
  - CPLD devices
  - All nets
- Syntax Example:
  - **attribute NOREDUCE of** *signal_name*: **signal is "{TRUE|FALSE}";**
  - **NET "$SIG_12" NOREDUCE;**

PIN constraint

- The PIN constraint in conjunction with LOC defines a net location

- Applicable to:
  - Nets
- Syntax Example:
  - **PIN “***module***.***pin***” LOC=***location***;**

RLOC constraint

- RLOC (Realtive Location) constraints group logic elements into discrete sets and allow to define the location of any element within the set relative to other elements in the set, regardless of eventual placement in the overall design
- The Blocks are set relative to each other to increase speed and use of die resources efficiently

Project co-financed from the EU European Social Fund

RLOC constraint

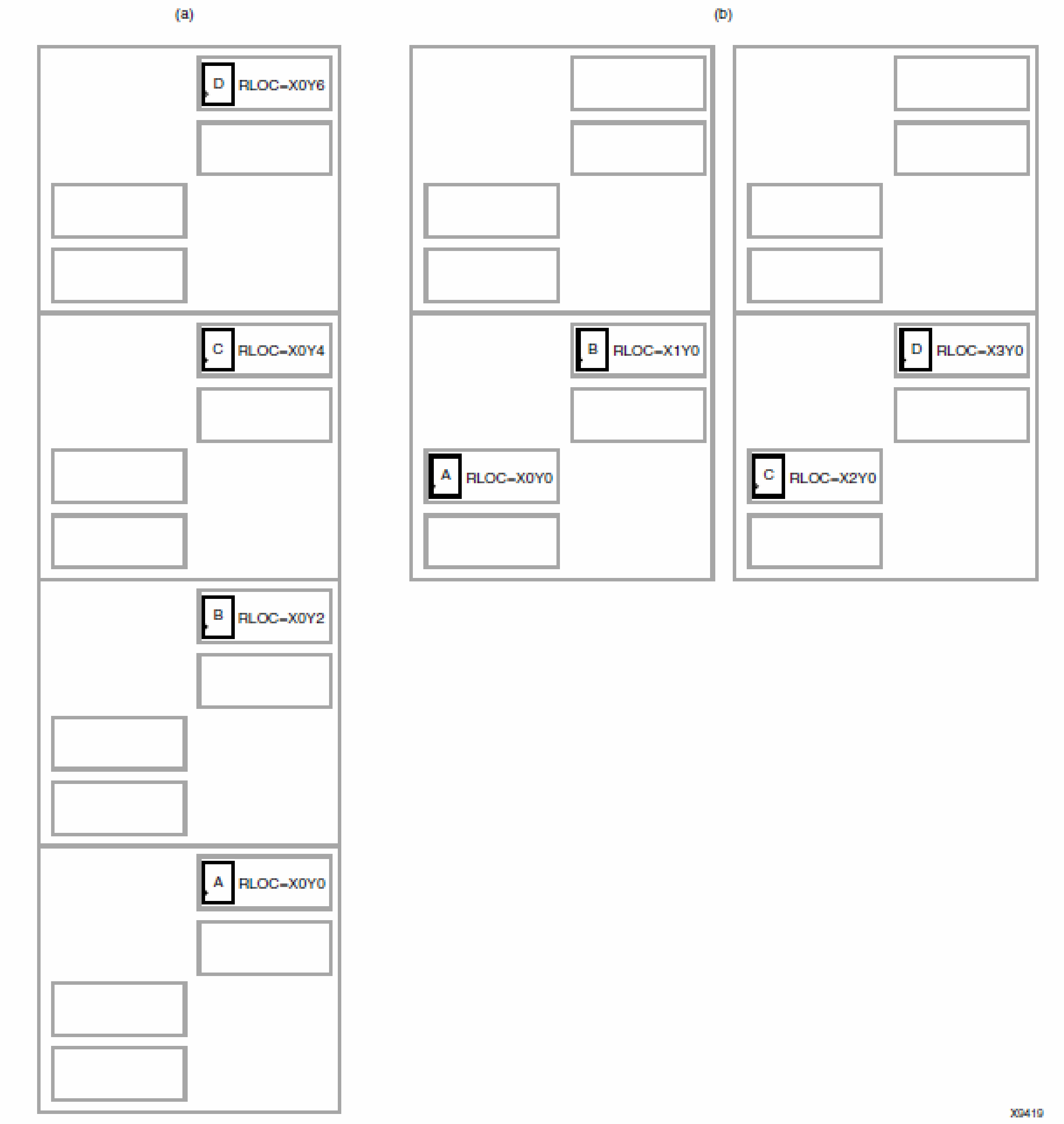
- Applicable to:
  - Registers
  - ROM
  - RAMS, RAMD
  - BUFT
  - LUTs, MUXCY, XORCY, MULT_AND, SRL16, SRL16E
  - DSP48
  - MULT18x18

RLOC constraint

- Syntax Example:
  - Syntax depends on the device
  - For Spartan-3 and most Virtex devices:
    - **RLOC=X*m*Y*n***
      - *where*
      - *m* and *n* are the relative X-axis (left/right) value and the relative Y-axis (up/down) value, respectively
      - the X and Y numbers can be any positive or negative integer including zero

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

- Two RLOC specifications for four flip-flops



HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Project co-financed from the EU European Social Fund

Thank you for your attention

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

References

[1] Spartan-6 family documentation; www.xilinx.com

[2] Constraints Guide, Rev 10.1, www.xilinx.com