EEE4120F

High Performance Embedded Systems

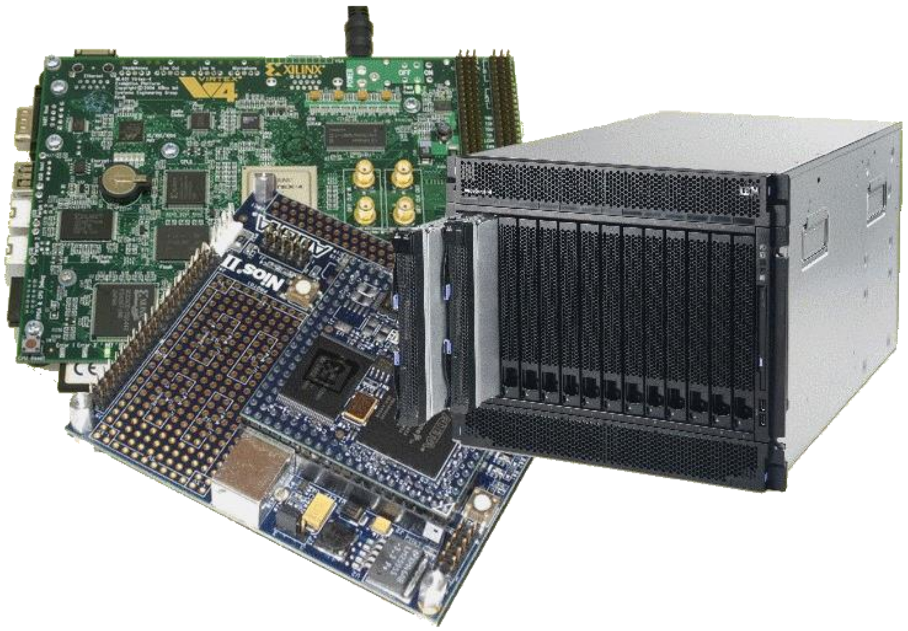
Lecture 17

Attributes, Constraints and UCF files

Lecturer:
Simon Winberg

Lecture Overview

- Attributes
- Constraints
- UCF Files
- Xilinx Design Constraints



Attributes and Constraints

A Xilinx Perspective

Difference between Attributes and Constraints

- Two types of attributes:
  - Predefined attributes (a part of the 1076 standard*)
  - Those outside the standard provided by the designer or by the design tool supplier (such as Xilinx or Altera)

* http://vhdl-manual.narod.ru/books/ieee_manual.pdf

Design Constraints

- Keep in mind that you are essentially describing hardware… it is *not* a program running on a CPU.
- Issues of design constraints are very necessary for efficient implementations, and so important and prevalent (for large systems) … that it's turned into a type of specialized programming of its own*.
- These are leading to 'constraint languages' (as yet largely unstandardized between tools), e.g.:
  - Xilinx: .ucf User Constraint File, or .xdc Xilinx Constraint File
  - Intel / Altera: .qsf Quartus II Setting File
  - Synopsis: .sdc Synopsis Design Constraints (de facto standard)
- Two main types of constraints:
  - Placement Constraints (the geographical position of pins and related aspects)
  - Timing constraints (clock and timing related aspects of the design)
- These are others types of constrains (see next slide) but we will look into those later in the course

* Indeed if you were an expert in using constrains languages you'd probably be in high demand by companies using FPGAs in their designs, it can probably make for a challenging and rewarding career. If you like Sudoku you might light constrains programming ☺

Classes of FPGA constraints

- Grouping Constraints
- Timing Constraints
- Logical Constraints
- Physical Constraints
- Mapping Directives
- Placement Constraints
- Routing Directives
- Synthesis Constraints
- Configuration Constraints

Source: http://xilinx.eetrend.com/files-eetrend-xilinx/forum/201103/1746-3205-02a_ucf_editing_12.pdf

Where constrains can be defined

- Depending on a attribute they can be defined in various places:
  - **Constraints Editor** for Timing Constraints
  - **Floorplanner** for Non-timing placement constraints
  - **PACE** for IO placement and area constraints
  - **Floorplan Editor** for IO placement and area constraints
  - **Schematic and Symbol Editors** for IO placement and RLOC constraints
- The UCF (User Constrains File) is where you can manually (and quickly!!) define constraints, we will focus on this option.

UCF

Xilinx User Constraints Files

We will look at “Xilinx Design Constraints” (XDC), which is the more recant version of specifying Xilinx constrains editor later. However, I like using UCFs if possible as the syntax is quicker and easier to work with.

UCF Files

- UCF files are in ASCII
- They are used to specify constraints on the logical design
- You can use your favorite text editor to edit them UCF (can edit in ISE)
- NCF file:
  - this often goes with UCT file.
  - The Netlist Constraint File (NCF) is also an ASCII file that is usually generated by some synthesis programs

UCF Files Syntax

- UCF files are generally case sensitive
- However any Xilinx constraint keyword (e.g., LOC, PERIOD, LOW, HIGH) may be entered in all uppercase, in all lower-case, or mixed case.
- Each statement is terminated by a semicolon (;) - the errors are sometimes cryptic, not obviously indicating a missing ';'.
- Keep a blank line at the end of the file
- Comments are started with # and end at the end of the line. The C or C++ comments // and /**/ are also supported

UCF Files Syntax

- Each line has a structure such as:

```
{NET|INST|PIN} "name" constraint;
```

- Particular bus lines on a bus are indicated by using *name*<*n*> where n is the bit (counted from 0) that you want to refer to
- Typical usage: setting **location of pins**

```
NET reset LOC=P16;
NET leds<0> LOC=P23;
NET leds<1> LOC=P24;
```

UCF quick features

- Wildcards can be used for the naming, such as
  - The asterisk (*): represents any string of zero or more characters
  - The question mark (?): represents a single character
- Examples:
  - `NET "leds*" SLOW;`
  - `NET "CLK?" FAST;`

XDC

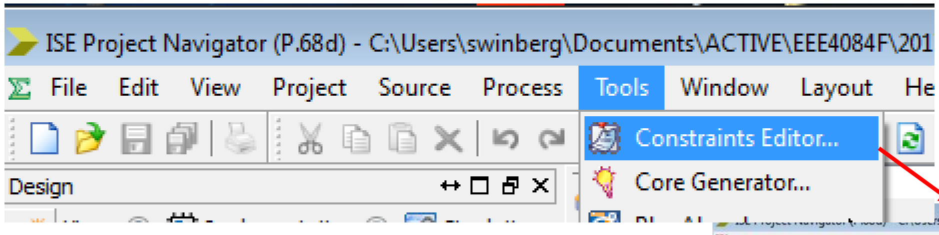
Xilinx Design Constraints Files

The Xilinx Design Constraints (XDC) are now more mainstream for use of Xilinx Vivado (although Vivado still has compatibility with the more classical and simpler UCFs files).

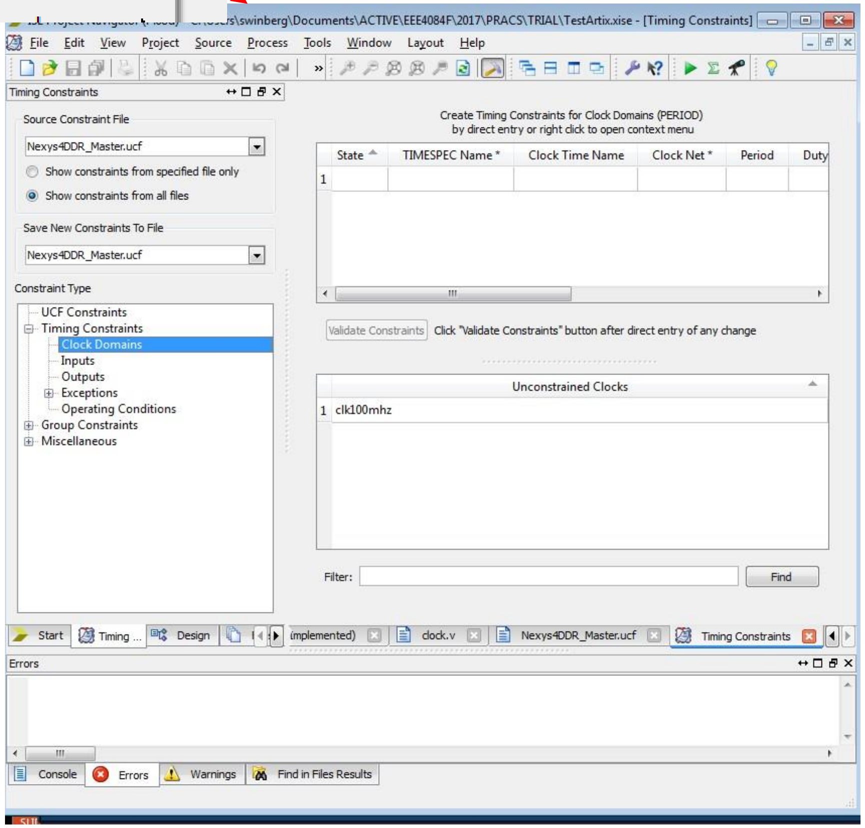
Xilinx Constraints Editor

- The Xilinx Constraints Editor is useful for enter timing constraints
- The GUI simplifies constraint entry by using wizards to guide the user through constraint creation, without needing to understand UCF file syntax.
- Although it is useful to see what kinds of constrains result in the file as this could help speed up constrains entry/reuse in future projects

Constraints Editor



Runs synthesize etc.


C:\Users\swinberg\Documents\ACTIVE\EEE4084F\2017\PRACS\TRIAL\TestArtix.xise - [Timing Constraints]
File
Edit
View
Project
Source
Process
Tools
Window
Layout
Help
Timing Constraints

Source Constraint File: Nexys4DDR_Master.ucf

( ) Show constraints from specified file only
(●) Show constraints from all files

Save New Constraints To File: Nexys4DDR_Master.ucf

Constraint Type
- UCF Constraints
- Timing Constraints
  - Clock Domains
  - Inputs
  - Outputs
  - Exceptions
  - Operating Conditions
- Group Constraints
- Miscellaneous

Create Timing Constraints for Clock Domains (PERIOD)
by direct entry or right click to open context menu

	State	TIMESPEC Name *	Clock Time Name	Clock Net *	Period	Duty
1						

Validate Constraints — Click "Validate Constraints" button after direct entry of any change

	Unconstrained Clocks
1	clk100mhz

Filter: Find

Provides functions to find pins, to assign wires to groups, to structure clock domains.

Can view the UCF file lines in the “UCF Constraints” tab (this includes the comment lines).

PERIOD constraints

- The PERIOD constraint provide accurate timing requirements, so it indicates…
  - Clock skew between the source and destination registers / flip-flops
  - Synchronous elements clocked on edge
  - Unequal clock duty cycles
  - Clock jitter
- ISE uses this information to place and route elements of your design in an attempt to satisfy these requirements (and thus make the design work on hardware)

Example PERIOD constraint

The PERIOD statement covers all timing paths that start or end at a register, latch, or synchronous RAM that are clocked by the reference net

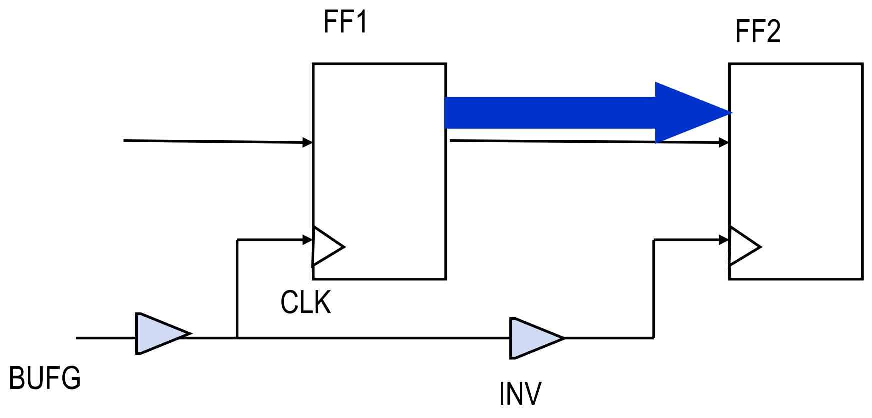
Assumptions…

- 50% duty cycle on CLK line
- PERIOD constraint of 10 ns

Clocking

- Because reg2 will be clocked on the falling edge of CLK, the link between the flip-flops will be constrained to 50% of 10 ns = 5 ns

The implementation tools will automatically account for triggering the one flip-flop on the rising edge and another on the falling edge



NET clock PERIOD = 10ns ;

Tutorial on using UCF files

- Recommended tutorial by Xilinx on understanding and using UCF files
  - http://xilinx.eetrend.com/files-eetrend-xilinx/forum/201103/1746-3205-02a_ucf_editing_12.pdf

Exam hint: statemachines are highly likely to come up in the exam or class test, building statemachines is an essential skill for HDL developers.

Disclaimers and copyright/licensing details

References: Verilog code adapted from
http://www.asic-world.com/examples/verilog