UCF Editing

Objectives

After completing this module, you will be able to:

- List the tools available for creating and modifying UCFs
- Describe the behavior of designs with multiple constraint files
- Create groups by using the TNM and TNM_NET attributes
- Write a User Constraint File (UCF) containing the following constraints
  - Grouping constraints
  - Timing constraints
  - Attributes
  - I/O constraints
- Describe constraint priority

Prerequisites

- Timing constraints modules
  - Global Timing Constraints (*Essentials of FPGA Design* course)
    - PERIOD, OFFSET, and PAD-TO-PAD
  - Timing Groups and OFFSET Constraints (*Designing for Performance* course)
    - Creating groups of path endpoints and THRU points
    - Creating pin-specific and group-specific OFFSET constraints
  - Path-Specific Timing Constraints (*Designing for Performance* course)
    - Interclock domain constraints
    - Multicycle path constraints
    - False path constraints
    - Constraint priority

Lessons

- **Overview**
- Grouping Constraints
- Timing Constraints
- Constraint Priority
- Additional Constraints
- Summary

UCF

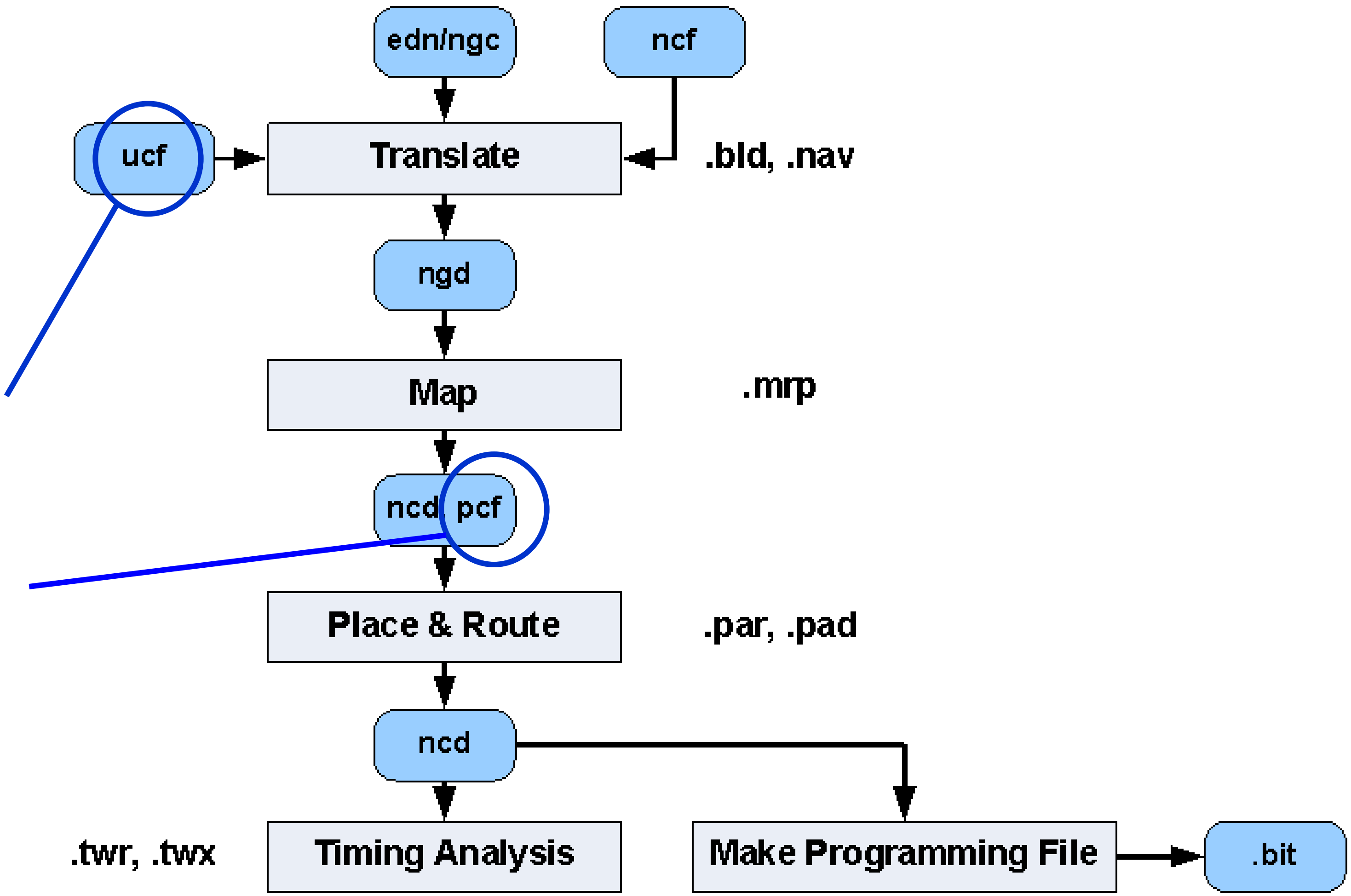
- UCF = User Constraint File
- Plain text file that can be modified in any text editor or in any of the tools that support UCF editing
  - The Constraints Editor does not support all constraints
- Syntax is case sensitive except for Xilinx constraint keywords (for example, PERIOD, HIGH, LOW, ns, or ps)
- Statements must be terminated with a semicolon (;)
- Comments are entered with the pound sign (#)
- Statements do not need to be placed in any particular order

XILINX

Multiple UCFs

- The ISE® tool allows multiple UCFs to be added to a project
- Convenient way to
  - Separate placement constraints from timing constraints
  - Add constraints provided by the tools (from the Architecture Wizard or the CORE Generator™ tool) without using copy & paste
- The Constraints Editor opens the first UCF added to the project
  - Other UCFs can be selected from inside the Constraints Editor
- The PlanAhead tool allows you to select which UCF file to open and write to
- If multiple constraints conflict, then the last constraint takes implicit priority over previous constraints
  - More on constraint priority later

Constraint Flow Review



Classes of FPGA Implementation Constraints

- Grouping: Collects primitives together for later use with other constraints
- Timing: Describes the timing requirements of static timing paths to the implementation tools
- Attribute: Defines the value of a property associated with a primitive
- Placement: Influences the physical placement of primitives by spatial description
- Mapping: Provides specific direction to the mapping tool on an instance-by-instance basis
- Routing: Provides specific direction to the place and route tool on an instance-by-instance basis

Tools for Editing Constraints

Tool	Can Do	Best At
Constraints Editor	Anything related to timing as well as prorating	Advanced constraints (groups, multi-cycle, prorating, …)
PlanAhead™ tool	Virtually everything	Area constraints Pin placement
Text editor	Everything	None

Lessons

- Overview
- **Grouping Constraints**
- Timing Constraints
- Constraint Priority
- Additional Constraints
- Summary

Specifying Groups

- Static timing paths begin and end at I/O pads and internal synchronous points
- To write effective constraints, you must group path endpoints together

User-Created Groups (TNM)

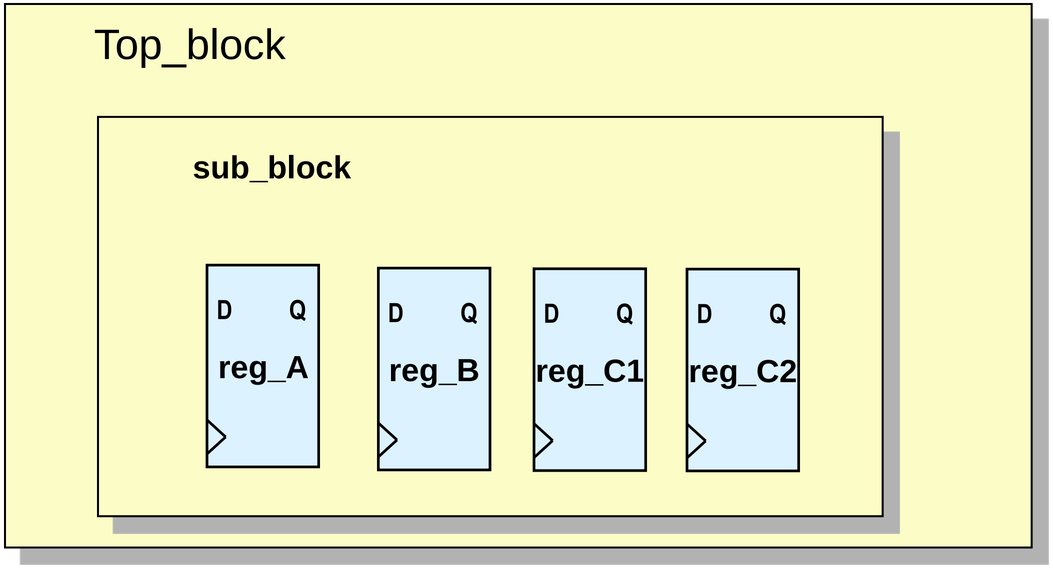
- TNM attribute = Timing NaMe
- TNM creates customized groups of path endpoints
  - All elements tagged with the same TNM are considered a group
  - Elements can be added to the same group using multiple statements
- Basic syntax
  - **INST** <*object_name*> **TNM =** <*identifier*> *;* # or
  - **NET** <*object_name*> **TNM =** <*identifier*> *; # or*
  - **PIN** <*object_name*> **TNM =** <*identifier*> *;*
  - <*object_name*> is the name of an element, net or pin within the design
    - Wildcards are allowed: “*” and “?”
  - <*identifier*> is the name of the time group to create or add elements to
    - can be any combination of letters, numbers, or underscores
    - is case sensitive (TNM=abc $\neq$ TNM=ABC)

Identifying Elements for Group Creation

- Question: How do you determine the instance and signal names?
  - Use names from the RTL that are known to be preserved
  - Look through the netlist
  - Use the Xilinx Constraints Editor
- Xilinx recommendation
  - Use the Constraints Editor to make initial constraints and groups
  - Reduce the number of constraints and add additional constraints with a text editor

TNM on Cells

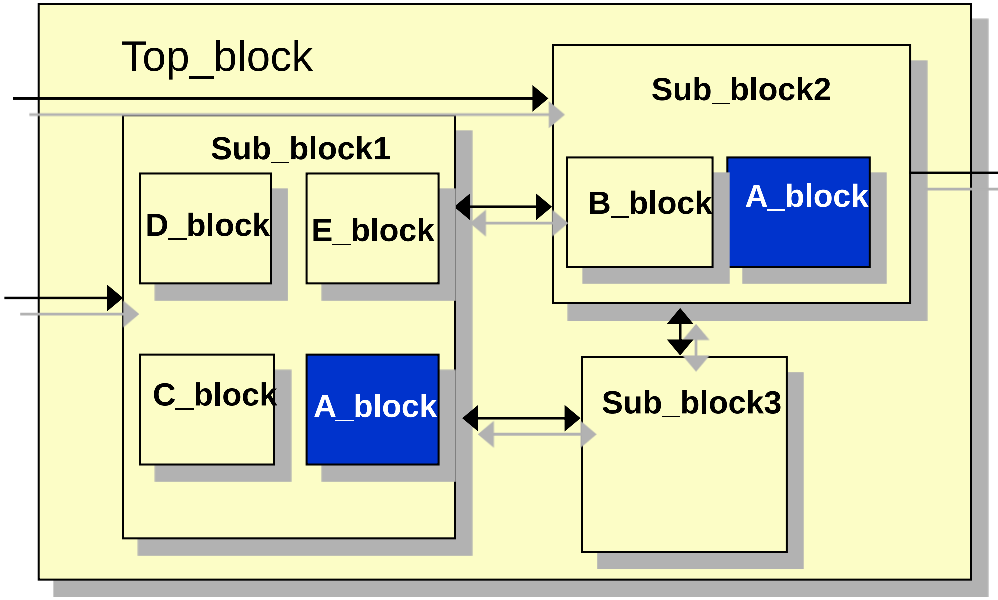
- The TNM attribute on an instance places a single element into the group
  - INST “sub_block1/reg_A” TNM = tgrp_group1;
- Instance names can use wildcards
  - All elements matched by the wildcard are placed in the group
  - INST “sub_block1/reg_C?” TNM = tgrp_group2;



- tgrp_group1 contains:
  - sub_block/reg_A
- tgrp_group2 contains:
  - sub_block/reg_C1
  - sub_block/reg_C2

TNM on Hierarchical Blocks

- The TNM attribute on hierarchical block places all synchronous elements in the hierarchical block and all levels of hierarchy below it in a group
  - INST "Sub-block1/A_block" TNM = tgrp_group3;
- Instance names can use wildcards
  - INST "*/A_block" TNM = tgrp_group4;



- tgrp_group3 contains:
  - All synchronous elements in Sub-block1/A_block
- tgrp_group4 contains:
  - All synchronous elements in Sub-block1/A_block
  - All synchronous elements in Sub_block2/A_block

XILINX

TNM and TNM_NET on Nets or Pins

- Placing TNM on a net groups path endpoints that are driven by the net/pin
  - All synchronous elements that have a combinatorial path from the specified net or pin
- However, TNM will not propagate through IBUF components
  - The TNM will end up on the input pad
  - Use TNM_NET constraint to propagate through IBUF components
    - Syntax: **NET** <*net_name*> **TNM_NET =** <*identifier*> *;*
- Xilinx recommends
  - To group input pads, use a TNM on the net driven by the pad
  - Use TNM_NET to group logic elements driven by a clock net
  - Take care when using TNM_NET on non-clock nets

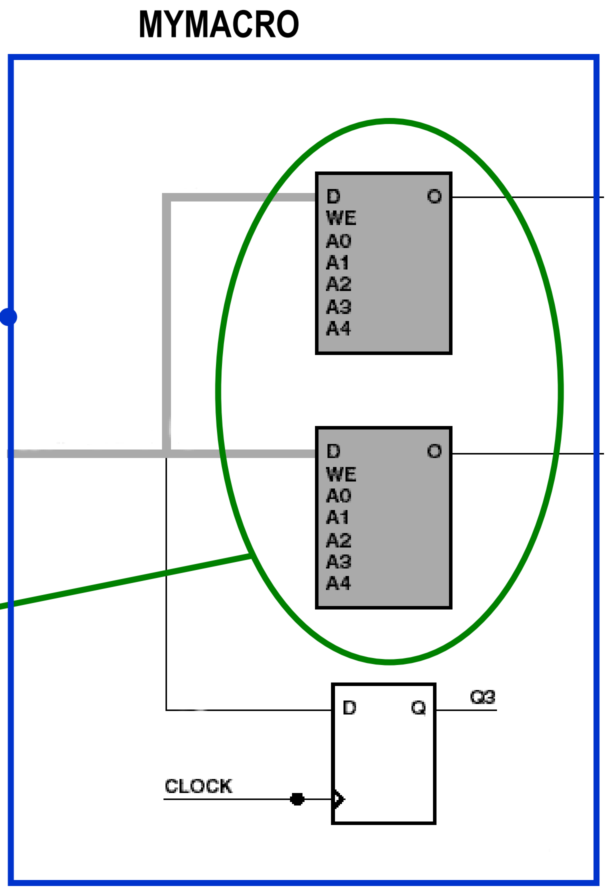
XILINX

Predefined Groups

- You can use a predefined group in place of a user-defined group in any constraint
- Predefined groups include:
  - PADS: All I/O pads
  - FFS: All flip-flops
  - LATCHES: All latches
  - RAMS: All RAM elements (distributed, Block, FIFO)
  - DSPS: All DSP elements
- You can restrict the group by using name qualifiers
  - A colon separated list of element names in brackets; wildcards are allowed
  - Syntax: FFS(reg_A : reg_B : reg_C*)

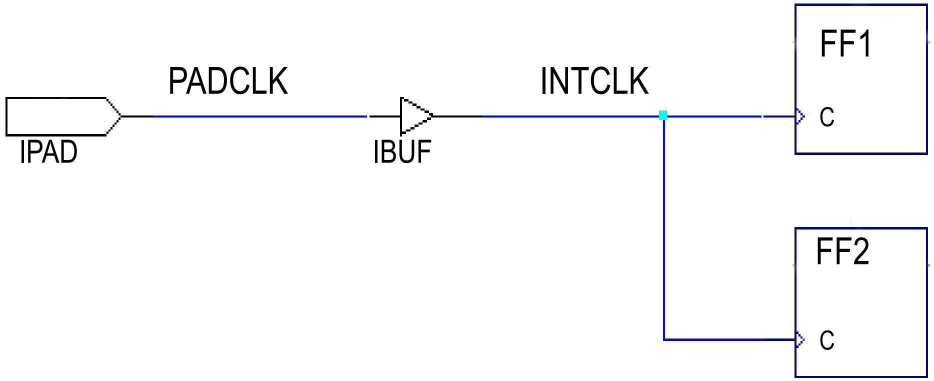
Predefined Groups as Qualifiers

- Use a predefined group keyword to restrict the types of elements that are tagged with the TNM
  - **INST** MYMACRO **TNM =** core;
    - All elements within MYMACRO will be included in the group
    - Includes the two RAM components and one FF
  - **INST** MYMACRO **TNM =** RAMS ram_core;
    - Only the two RAMS are included in this group because of the qualifier



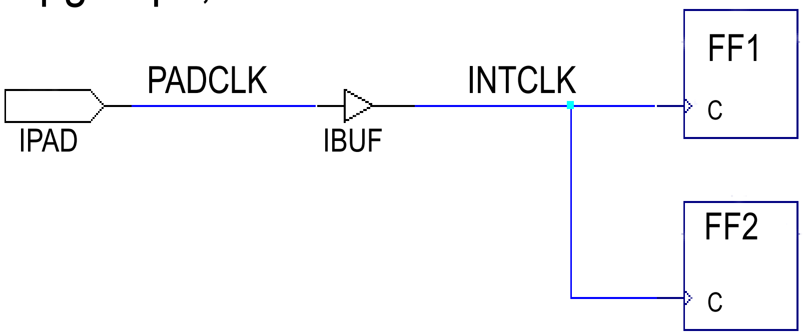
Apply Your Knowledge

1) What elements will the following groups contain?
   - NET PADCLK TNM = PADS padgroup;
   - NET PADCLK TNM = FFS flopgroup1 ;
   - NET INTCLK TNM = FFS flopgroup2;
   - NET PADCLK TNM_NET = FFS flopgroup3;



Answer

1) What elements will the following groups contain?
   - NET PADCLK TNM = PADS padgroup;
     - Contains only the IPAD symbol
   - NET PADCLK TNM = FFS flopgroup1;
     - An empty group which will cause an error during NGDBUILD
   - NET INTCLK TNM = FFS flopgroup2;
     - Includes FF1 and FF2
   - NET PADCLK TNM_NET = FFS flopgroup3;
     - Includes FF1 and FF2



XILINX

Combining Groups

- Use the **TIMEGRP** constraint to
  - Combine multiple groups into one group
  - Create groups by exclusion
  - Define flip-flop subgroups by clock edge
- Syntax to combine groups
  - **TIMEGRP** <*newgroup*> = <*grp1*> <*grp2*> *[grp3...]*;
- Syntax to group by exclusion
  - **TIMEGRP** <*newgroup*> = <*grp1*> *[grp2…]* **EXCEPT** <*grp3*> *[grp4…]*;
- Syntax to group by clock edge
  - **TIMEGRP** <*newgroup*> = **[RISING | FALLING]** <*grp1*>;

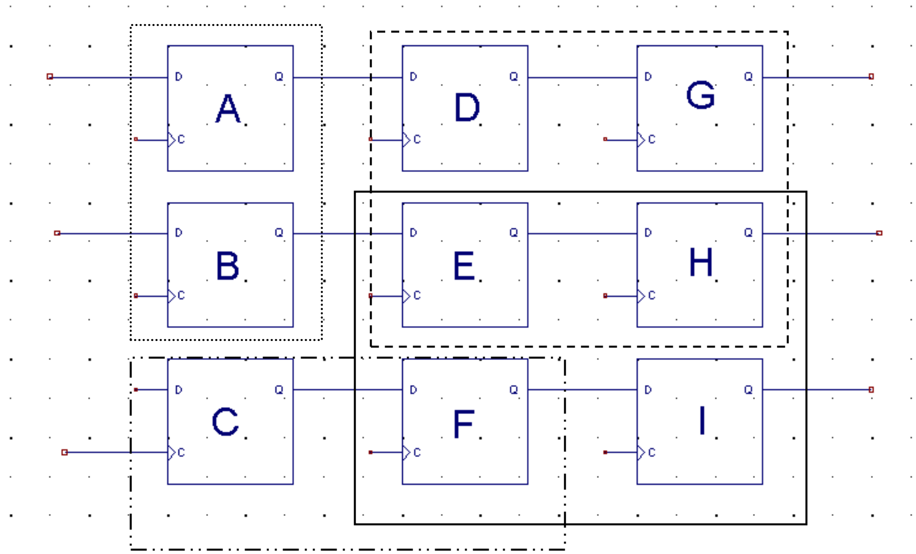
Apply Your Knowledge

grp1 = A, B

grp2 = D, E, G, H

grp3 = C, F

grp4 = E, F, H, I



2) Which flip-flops will each constraint include?

- **TIMEGRP** manyffs = grp1 grp2 grp3 ;
- **TIMEGRP** largeone = grp2 grp3 **EXCEPT** grp4;

XILINX

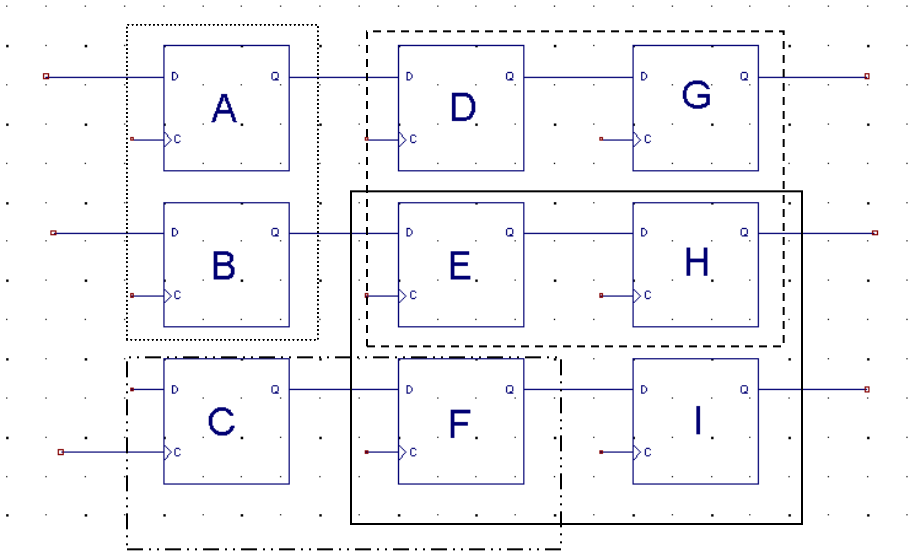
Answer

grp1 = A, B

grp2 = D, E, G, H

grp3 = C, F

grp4 = E, F, H, I



2) Which flip-flops will each constraint include?

– **TIMEGRP** manyffs = grp1 grp2 grp3 ;
  - Includes all flip-flops except I
– **TIMEGRP** largeone = grp2 grp3 **EXCEPT** grp4;
  - Includes D, G, and C

XILINX

Lessons

- Overview
- Grouping Constraints
- **Timing Constraints**
- Constraint Priority
- Additional Constraints
- Summary

Global Timing Constraints

- Square brackets are used for optional parameters or arguments
  - PERIOD
    - **NET** <clk_net_name> **TNM_NET =** <clk_group>**;**
    - **TIMESPEC TS**_<identifier> **= PERIOD** <clk_group> <value> [INPUT_JITTER <value>]**;**
  - OFFSET IN
    - [TIMEGRP <pad_groupname>] **OFFSET = IN** <offset_time> [VALID <datavalid_time>] **{BEFORE|AFTER}** <clk_name> [TIMEGRP <reg_groupname>] [{RISING | FALLING}];
  - OFFSET OUT
    - [TIMEGRP <pad_groupname>] **OFFSET = OUT** [<offset_time>] **{BEFORE|AFTER}** <clk_name> [TIMEGRP <reg_groupname>] [REFERENCE_PIN <ref_pin>] [{RISING | FALLING}];
  - Examples
    - **NET** rd_clk **TNM_NET =** tnm_rd_clk;
    - **TIMESPEC TS**_rd_clk **= PERIOD** tnm_rd_clk 7 ns INPUT_JITTER 100**;** #default ps
    - **OFFSET = IN** 6 ns **BEFORE** rd_clk**;**
    - **OFFSET = OUT** 5 ns **AFTER** rd_clk**;**

Optional OFFSET Parameters

- TIMEGRP
  - Groups of I/O pads or synchronous elements can be identified to create very specific constraints
- VALID
  - Defines the width of the input data window
- RISING, FALLING
  - Specifies which edge of the clock is used to capture the data
- REFERENCE_PIN
  - Used in conjunction with OFFSET OUT
  - Defines the pin against which to report output skew in source-synchronous transmitters

FROM:TO Constraint

- Syntax: **TIMESPEC *TS***<*name*> **= FROM** <*group1*> **TO** <*group2*> <*value*> [DATAPATHONLY];
- ***TS***<*name*> must always start with ***TS***
  - Any alphanumeric character or underscore can follow
- <*group1*> designates the origin of the path
- <*group2*> designates the destination of the path
- <*value*> is in ns by default
  - Other units are ps, ms
  - Can be relative to another timespec constraint, such as TS_C2S/2 or TS_C2S*2
- DATAPATHONLY indicates that the path analysis should not include clock skew or phase information

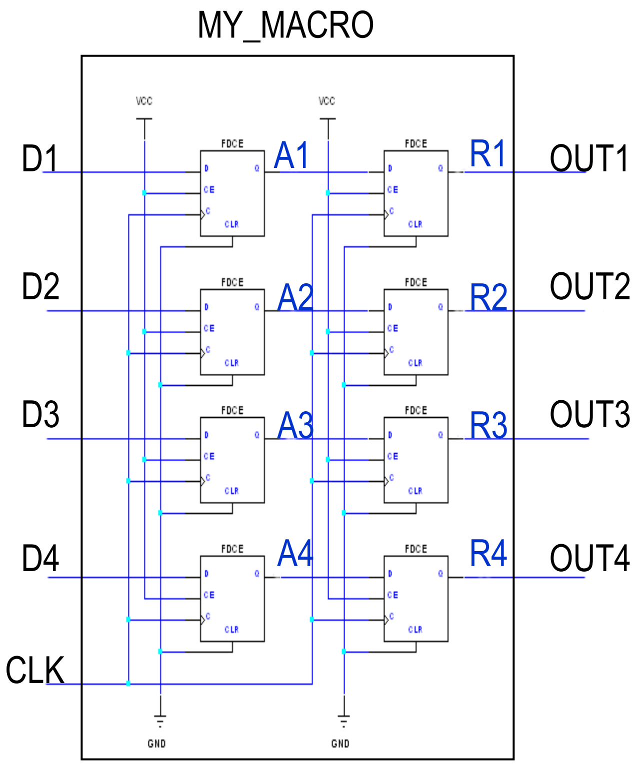
Groups in FROM:TO

- The FROM and TO groups can be any group
- User defined groups
  - Created by TNM or TIMEGRP
- Predefined groups
  - FFS, RAMS, LATCHES…
  - Can use name qualifiers to restrict the group, including wildcards
  - **TIMESPEC** TS_fiforam2reg_a **= FROM RAMS**(*/fifo_ram?) **TO FFS**(reg_a) 10 ns**;**

XILINX

Apply Your Knowledge

3) Write a timing specification to constrain all paths from the A registers to the R registers in the hierarchical block MY_MACRO to a value of 10 ns

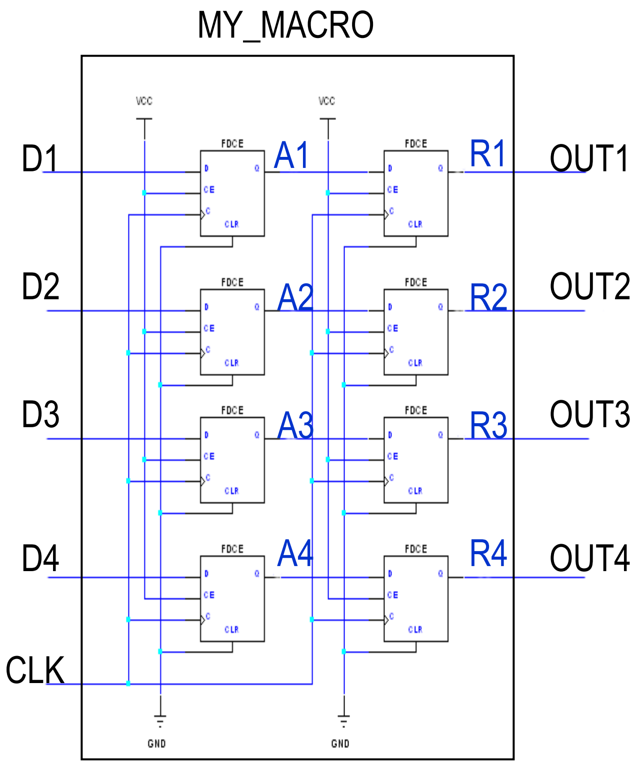


XILINX

Answer

3) Write a timing specification to constrain all paths from the A registers to the R registers in the hierarchical block MY_MACRO to a value of 10 ns

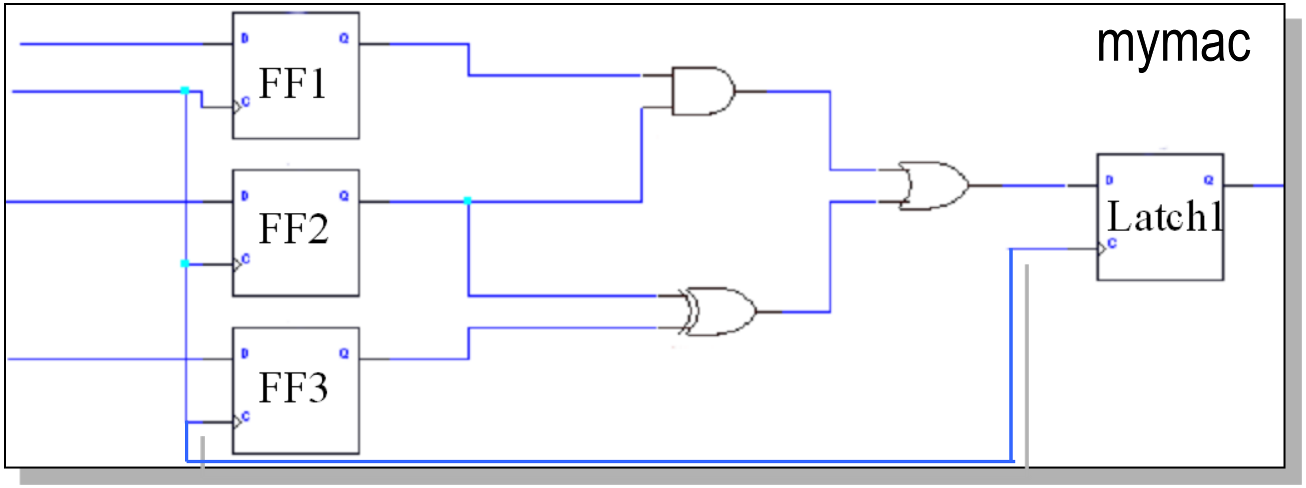
– **TIMESPEC** TS_areg2rreg = **FROM FFS**(MY_MACRO/A?) **TO FFS**(MY_MACRO/R?) 10 ns;



XILINX

Apply Your Knowledge

4) Write constraints to create separate groups for the flip-flops and latches in the instance *mymac*
   - Use TNM on the hierarchy mymac
5) Constrain the path from the flip-flop group to the latch group to TS_sys_clk100 * 2



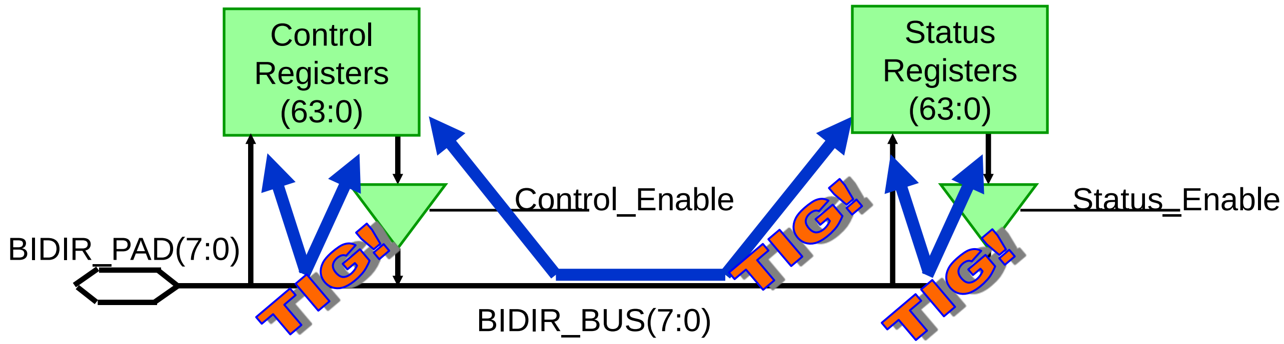
XILINX

Answers

4) Write constraints to create separate groups for the flip-flops and latches in the instance *mymac*
   - **INST** mymac **TNM = FFS** mymac_ffs**;**
   - **INST** mymac **TNM = LATCHES** mymac_latches**;**

5) Constrain the path from the flip-flop group to the latch group to TS_sys_clk100 * 2
   - **TIMESPEC** TS_mymac_ffs2latches **= FROM** mymac_ffs **TO** mymac_latches TS_sys_clk100 * 2**;**

Ignoring Selected Paths (TIG)



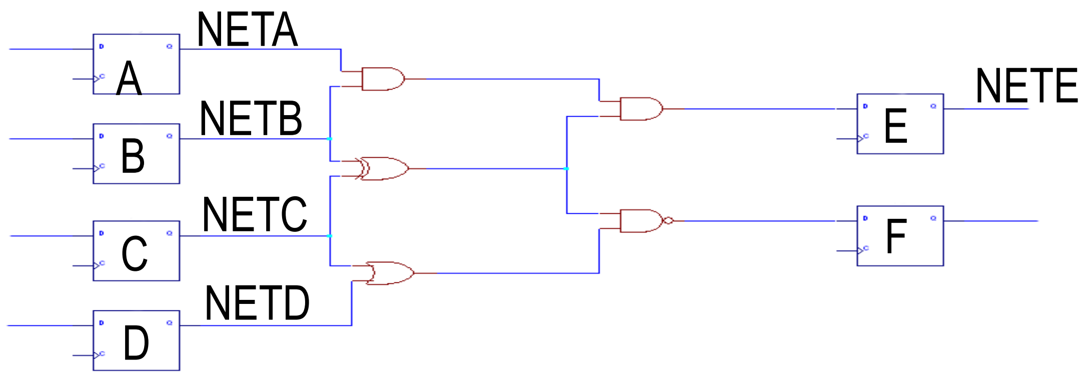
- TIG = Timing IGnore, a.k.a. *false path*
- Why use a TIG?
  - Decreases competition for routing resources
  - False paths through 3-state buffers, static nets, nets, and paths that change infrequently or where a path exists but is never actually used (above example)
- Syntax: **[NET|PIN]** <*object_name*> **TIG** [= *TSid1, TSid2*…];
  - Ignores timing on all paths that contain <*object_name*>
- TIG can also be used as the value in a **FROM:TO** constraint
  - Example: **TIMESPEC** TS_ignore = **FROM** group1 **TO** group2 **TIG**;

XILINX

Apply Your Knowledge

6) Which paths are ignored?

- NET NETC TIG;
- TIMESPEC TS_TIG_A2E = FROM FFS(NETA) TO FFS(NETE) TIG;

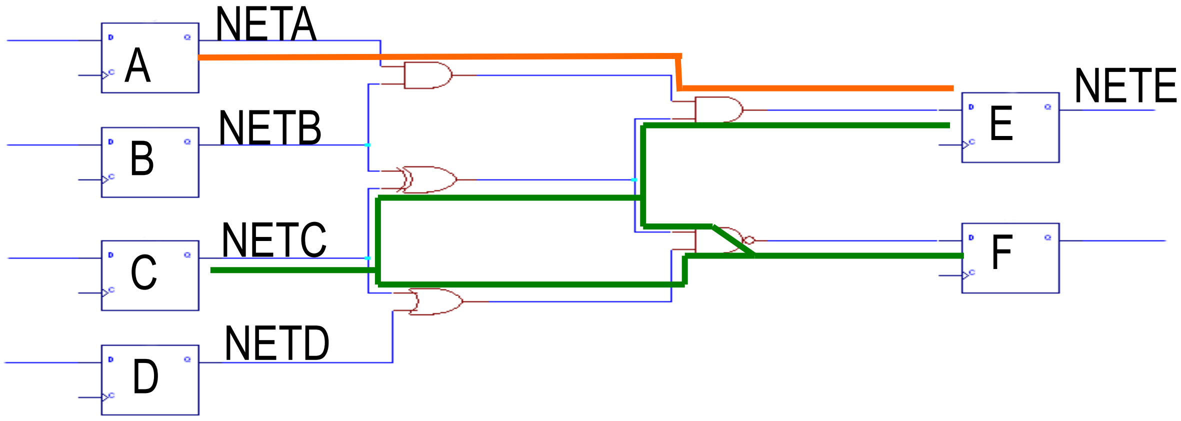


XILINX

Answer

6) Which paths are ignored?

- From register A to register E
- From register C to register E
- From register C to register F



XILINX

Lessons

- Overview
- Grouping Constraints
- Timing Constraints
- **Constraint Priority**
- Additional Constraints
- Summary

Timing Constraint Priority

- False Paths (TIG)
- FROM:THRU:TO
- FROM:TO
- Pin-specific OFFSETs (Net OFFSET)
- Group-specific OFFSETs (Pad/Register OFFSET)
- Global OFFSETs
- PERIOD

XILINX

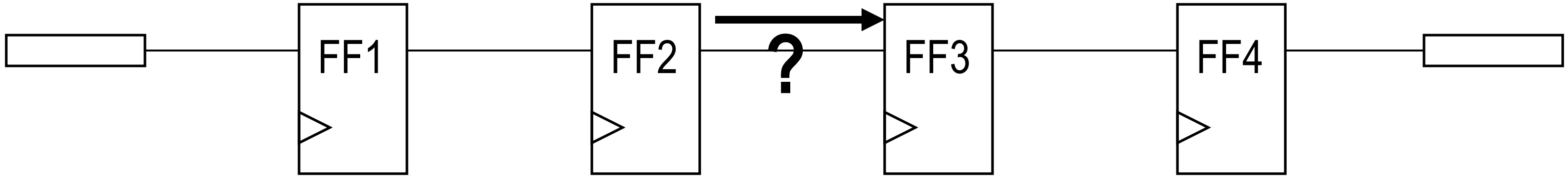
Priority for Conflicting TIMESPECs

- You can assign a priority to timespecs
  - Except MAXDELAY and MAXSKEW
- Used when there are multiple timing constraints of the same type on a delay path
  - Example: More than one FROM TO type constraint
- Syntax: <*constraint_definition*> **PRIORITY** <*value*>;
  - <*value*> represents the priority (between -1000 and 1000)
    - Smaller number = higher priority
  - Strategy: Start by assigning the highest priority = 0
    - As you add additional constraints of higher priority, subtract 1
    - As you add additional constraints of lower priority, add 1
- Example: **TIMESPEC** TS_01 = **FROM** high **TO** low 8 ns **PRIORITY** 3;

Notes on Priority

- The PRIORITY value is only used when conflicting constraints are of the same type
  - A FROM:TO constraint can never have priority over a TIG constraint
- Constraints that have a PRIORITY value defined are given priority over constraints that have no PRIORITY value defined
- If multiple constraints conflict, then the last constraint takes implicit priority over previous constraints

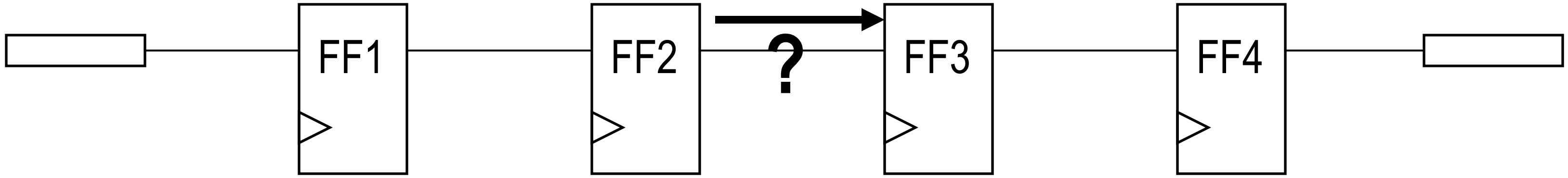
Apply Your Knowledge



- Example
  - FF1, FF2, FF3 are in fastgroup
  - FF2, FF3, FF4 are in slowgroup
  - TIMESPEC ts_fast = FROM fastgroup TO fastgroup 3 ns PRIORITY 1;
  - TIMESPEC ts_slow = FROM slowgroup TO slowgroup 10 ns PRIORITY 0;

7) What happens to the path between FF2 and FF3, which is covered by both timespecs?

Answer



7) What happens to the path between FF2 and FF3, which is covered in both timespecs?

  – Because the PRIORITY attribute is on the constraint, the path would be constrained to 10 ns, rather than 3 ns

- Without the use of PRIORITY, there are several factors to consider when choosing which constraint to apply

  – See ***Constraints Guide*** > **Constraints Entry** > **Constraints Priority**

Lessons

- Overview
- Grouping Constraints
- Timing Constraints
- Constraint Priority
- **Additional Constraints**
- Summary

MAXDELAY and MAXSKEW

- Use these attributes to constrain critical nets
  - Not needed for nets that use global clock buffers
  - Do *not* over constrain, which can adversely affect implementation
  - Do *not* use to constrain clocks that use general routing resources
    - Tools automatically recognize these clocks and use a predefined local clock routing template for balanced skew and limited delay
    - Using these constraints on clocks increases delay and skew

MAXDELAY syntax: **NET** <*net_name*> **MAXDELAY** = <*delay_time*>;

MAXSKEW syntax: **NET** <*net_name*> **MAXSKEW** = <*delay_time*>;

- <*delay_time*> is any numeric value
  - Default unit is ns
- Do *not* set MAXSKEW to 0
  - Can cause a software error

Attributes

- Attribute constraints control functionality or implementation of elements in a design
- Basic syntax
  - **{INST | NET}** <*object_name*> <*attribute_name*> **=** <*attribute_value*>**;**
- Attribute names can be found in the software documentation
  - *Constraints Guide*
  - *Libraries Guides*
- Example
  - **INST** my_block_ram INIT_FILE **=** mem_contents**;**

XILINX

I/O Attributes

- Pin placement
  - **NET** <*net_name*> **LOC =** <*pin number*>**;**
- I/O configuration
  - **NET** <*net_name*> **IOSTANDARD =** <*IO_standard*>**;**
  - **NET** <*net_name*> **SLEW** = {FAST | SLOW}**;**
  - **NET** <*net_name*> **DRIVE =** <*drive strength*>**;**
- Multiple constraints can be defined in one statement
  - Example: **NET** data_in **LOC =** A5 | **SLEW =** FAST**;**
- Using IOB flip-flops
  - Place the attribute **IOB =** {TRUE | FALSE | FORCE} on the flip-flop

XILINX

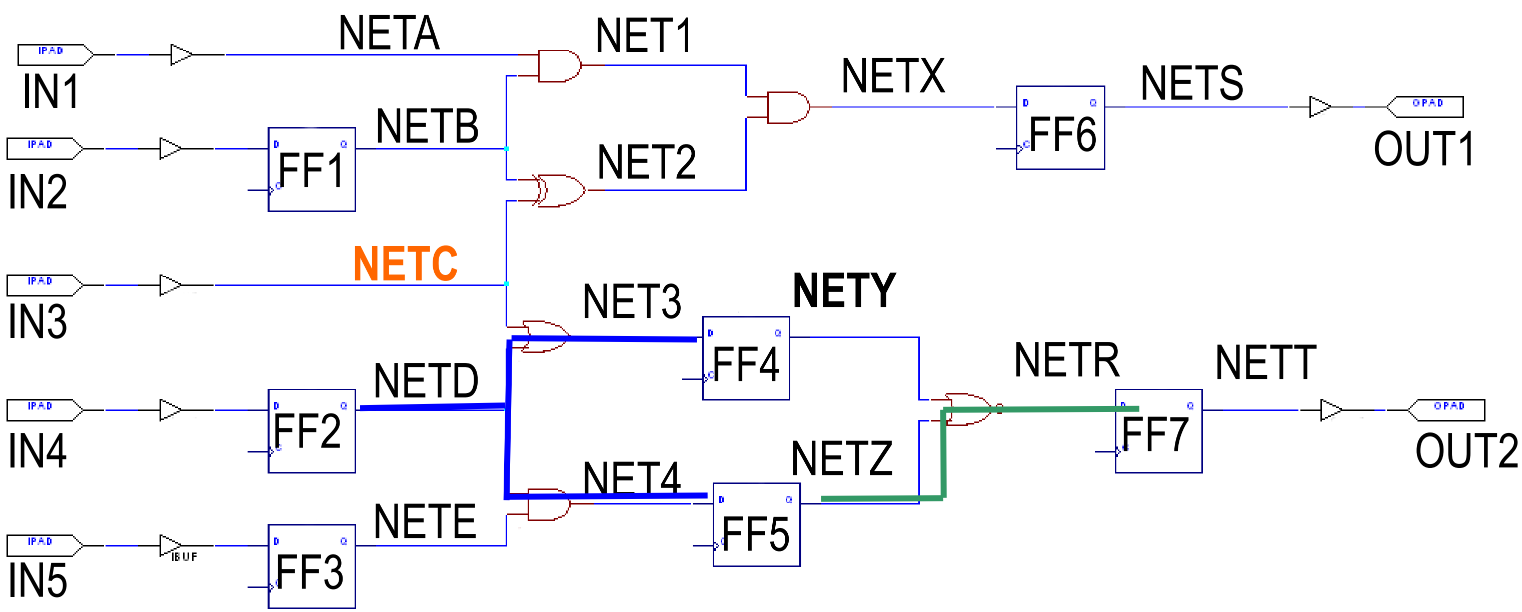
Lessons

- Overview
- Grouping Constraints
- Timing Constraints
- Constraint Priority
- Additional Constraints
- **Summary**

Apply Your Knowledge

8) Write the following constraints

- Multicycle path from FF2 to FF4, FF5 (TS_CLK * 2)
- Timing IGnore on NETZ
- Maxdelay = 2.5 ns, Maxskew = 0.5 ns for NETC



XILINX

Answer

8) Write the following constraints
   - Multicycle path from FF2 to FF4, FF5 (TS_CLK * 2)
     - INST FF2 TNM = tnm_ff2;
     - NET NETD TNM_NET = tnm_ffs4and5;
     - TIMESPEC TS_ff2_to_ffs4and5 = FROM tnm_ff2 TO tnm_ffs4and5 TS_CLK * 2;
     - or
     - TIMESPEC TS_ff2_to_ffs4and5 = FROM FFS(NETD) to FFS (NETY: NETZ) TS_CLK * 2;
   - Timing IGnore on NETZ
     - NET NETZ TIG;
   - Maxdelay = 2.5 ns, Maxskew = 0.5 ns for NETC
     - NET NETC MAXDELAY = 2.5 ns ;
     - NET NETC MAXSKEW = 0.5 ns ;

XILINX

Apply Your Knowledge

9) How do you know if you did a good job constraining your design?

Answer

9) How do you know if you did a good job constraining your design?
   - Are all the paths constrained? Use the Timing Analyzer to locate any unconstrained paths
   - Are failing paths over-constrained?
     - Unidentified multi-cycle paths
     - Unidentified false paths

Summary

- Many tools available for entering, editing, and analyzing constraints
- The ISE software supports multiple UCFs
- Time groups create more specific timing constraints to help you define more accurately your timing requirements to the implementation tools
- The TNM attribute and TIMEGRP create customized groups
- Predefined groups use keywords and qualifiers to define path endpoints
- TIG decreases competition for routing resources by ignoring noncritical paths
- PRIORITY ensures overlapping constraints are handled correctly
- MAXDELAY and MAXSKEW are constraints that can be used on critical nets in a design