Vivado Design Suite Migration Methodology Guide

UG911 (v2012.3) November 15, 2012

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/25/12	2012.2	Initial Xilinx release.
11/15/12	2012.3	Additions are: • Added Table 2-3, page 12. • Added Chapter 6, Migrating Additional Command Line Tools to Vivado.

Table of Contents

Chapter 6: Migrating Additional Command Line Tools to Vivado

Appendix A: Additional Resources

XILINX®

Chapter 1

Introduction to ISE Design Suite Migration

Overview

ISE® Design Suite is an industry-proven solution for all generations of Xilinx® devices, and extends the familiar design flow for projects targeting 7 series and Zynq™-7000 devices.

The Vivado™ Design Suite supports the 7 series devices including Virtex®-7, Kintex™-7 and Artix™-7 and offers enhanced tool performance, especially on large or congested designs.

Because both the ISE Design Suite and the Vivado Design Suite support 7 series devices, you have time to migrate to the Vivado Design Suite.

RECOMMENDED: *If you start a new design for a Kintex K410 or larger device, Xilinx recommends that you contact your local Field Application Engineer to determine if Vivado is right for your design.*

The Vivado Design Suite provides seamless reuse of all or part of a previous design through the ability to import projects and sources into a Vivado Design Suite project, and command mapping to Tcl scripts.

Design Flows

In the ISE Design Suite, there are two ways to run the design tools: use the graphical user interface (GUI), which includes Project Navigator and the point tools; or use batch scripts at the command line.

In the Vivado Design Suite, there are two design flow modes in which you run the tool: Project Mode, or Non-Project Mode.

- **Project Mode**: This mode is the easiest way to get acquainted with the tool behavior and Xilinx recommendations. In this mode, you use the graphical user interface (GUI), or Vivado Integrated Design Environment (IDE), to automatically manage your design.

- **Non-Project Mode**: In this mode, you use Tcl commands to compile a design through the entire flow in memory. Tcl commands provide the flexibility and power to set up and run your designs as well as perform analysis and debugging. Tcl commands can be run in batch mode, from the Tcl Prompt, or through the Vivado IDE Tcl Console. This mode enables you to have full control over each design flow step, but you must manually manage source files, reports, and intermediate results known as "checkpoints."

For more information about the design flow modes, see the *Vivado Design Suite User Guide: Design Flows Overview (UG892)* [Ref 1].

This guide covers migration considerations and procedures for both design flow modes in the Vivado Design Suite.

XILINX®

Chapter 2

Migrating ISE Design Suite Designs to Vivado Design Suite

Importing a Project Navigator Project

You can use the Vivado™ Integrated Design Environment (IDE), which is the GUI to import an XISE project file as follows:

1. Select **File > New Project**.
2. Select Project name and location.
3. In the New Project Wizard, select the **Imported Project** option.
4. Select **ISE** and choose the appropriate `.xise` file for import.

IMPORTANT: *Vivado Design Suite does not support older ISE® Design Suite project (`.ise` extension) files.*

After you have imported the project file:

- Review the Import XISE Summary Report for important information about the imported project.
- Ensure the selected device meets your requirements, and if it does not, select a new device. If the ISE project does not have an equivalent supported device in Vivado, a default device is selected.
- Review the files in the Sources window to ensure all design files were imported properly.

If the design contained a User Constraints File (UCF), the file will appear as an unsupported constraint file. The UCF must be converted to Xilinx® Design Constraints (XDC) format in order to apply any timing or physical constraints in the design. For more information, see Chapter 3, Migrating UCF Constraints to XDC.

For more details on using the Vivado interface for design creation, see *Vivado Design Suite User Guide: System Design Entry (UG895)* [Ref 2].

For information on the next steps in design flow, see the *Vivado Design Suite User Guide: Design Flows Overview (UG892)* [Ref 1].

Converting a PlanAhead Tool Project

To convert a PlanAhead™ tool project to a Vivado IDE project, open the PlanAhead project file (`.ppr` extension) in Vivado IDE. When prompted, type in a new project name and location for the converted project.

The project conversion effects the following changes:

- Targets the Vivado default 7 series device on projects with pre 7 series devices.
- Resets all runs. They are generated after implementing the design in the tool.
- Replaces run strategies with Vivado default strategies.
- Moves UCF files to an Unsupported Constraints Folder because UCF constraints are not supported.

Note: Conversion of designs with Partitions is not supported.

Importing an XST Project File

If you do not have an existing or up-to-date ISE Project Navigator (`.xise`) project file or PlanAhead tool (`.ppr`) project file, a you can use the Xilinx Synthesis Technology (XST) project file to import initial settings for a Vivado project. To import an XST project file:

1. Select **File > New Project**.
2. Select **Project name and location**.
3. In the New Project Wizard, select **Imported Project**.
4. Select **XST**, and select the appropriate project file with a `.xst` extension.

After you have imported the project file:

- Review the Import XST Summary Report for important information about the imported project.
- Ensure the selected device meets your requirements, and if it does not, select a new device. If the XST project does not have an equivalent supported device in Vivado, a default device is selected.
- Review the files in the Sources view to ensure all design files were imported properly.

For more details on using the Vivado interface for design creation, see *Vivado Design Suite User Guide: System Design Entry (UG895)* [Ref 2].

For information on the next steps in design flow, see the *Vivado Design Suite User Guide: Design Flows Overview (UG892)* [Ref 1].

Migrating Source Files

When you import a project or convert a project into the Vivado IDE, as specified above, you can also add all source files that are supported in Vivado Design Suite to the Vivado project.

- **IP**: You can migrate existing ISE Design Suite projects and IP to Vivado Design Suite projects and IP. The Vivado Design Suite can use ISE Design Suite IP during implementation. For more information, see Chapter 4, Migrating Designs with CORE Generator IP to Vivado Design Suite.
- **Source files**: You can add all source files, with the exception of Schematic (SCH) and Architecture Wizard (XAW) source files, from an existing ISE Design Suite project to new project in the Vivado Design Suite. For example, you can add CORE Generator™ tool project files (`.xco` file extension) and netlist files (`.ngc` file extension) as design sources.
- **Constraints**: User Constraint Format (UCF) files used for the design or IP must be converted to Xilinx Design Constraints (XDC) format for use with Vivado Design Suite. For more information about UCF to XDC migration, see Chapter 3, Migrating UCF Constraints to XDC in this guide.

CAUTION! *Do not migrate from ISE Design Suite and Vivado Design Suite in the middle of a current ISE Design Suite project because design constraints and scripts are not compatible between the environments.*

Mapping ISE Design Suite Command Scripts

This section covers the Vivado Design Suite non-project design flow mode only, and is intended for those who plan to use Tcl scripting with the Vivado tool.

You can use Tcl scripts to migrate your ISE Design Suite scripts to implement your FPGA design. Similar to the ISE Design Suite, the compilation flow in the Vivado Design Suite translates the design, maps the translated design to device-specific elements, optimizes the design, places and routes the design, and then generates a BIT file for programming.

Table 2-1 shows the main differences between the two design flows.

Table 2-1: **ISE Design Suite versus Vivado Design Suite Design Flow**

ISE Design Suite	Vivado Design Suite
Separate command line applications	Tcl commands in a shell.
XCF/NCF/UCF/PCF constraints	XDC timing and physical constraints
Design constraints (timing or physical) only applied at beginning of the flow	Constraints (timing or physical) can be applied, changed, or removed at any point in the flow.
Multiple database files (NGC, NGD, NCD) required	A single design database (checkpoint with a `.dcp` extension) written out on-demand at any point in the flow.
Reports generated by applications	Reports generated on-demand at any point in the flow, where applicable.

Table 2-2, page 11 shows the mapping of ISE Design Suite commands to corresponding Vivado Design Suite Tcl commands. You can run the Tcl commands using any of the following:

- In the Vivado IDE Tcl Console
- At the Tcl Prompt (`vivado -mode tcl`)
- Through a batch script (`vivado -mode batch -source my.tcl`)

Table 2-2: **ISE Design Suite Commands and Vivado Design Suite Tcl Commands**

ISE Design Suite Command	Vivado Design Suite Tcl Command
xst	read_verilog read_vhdl read_xdc synth_design ***Note:*** The commands must be run in this order.
ngdbuild	read_edif read_xdc link_design ***Note:*** You need these commands only if you are importing from third-party synthesis. If using synth_design, omit these steps.
map	opt_design power_opt_design (optional) place_design phys_opt_design (optional)
par	route_design
trce	report_timing report_timing_summary
xpwr	read_saif report_power
drc	report_drc
netgen	write_verilog write_vhdl write_sdf
bitgen	write_bitstream

ISE and the Vivado Design Suite use different algorithms; consequently, a one-to-one mapping is not always possible between the two tool flows. Table 2-3 provides a mapping of frequently-used options between the two implementation flows.

Table 2-3: **ISE to Vivado Implementation Flow Mappings**

ISE	Vivado
ngdbuild -p partname	link_design -part
ngdbuild -a (insert pads)	synth_design -no_iobuf (opposite)
ngdbuild -u (unexpanded blocks)	Allowed by default, generates critical warnings.
ngdbuild -quiet	link_design -quiet
map -detail	opt_design -verbose
map -lc auto	happens automatically in place_design
map -logic_opt	opt_design, phys_opt_design
map -mt	place_design automatically runs MT with four cores in Linux or two cores in Windows
map -ntd	place_design -non_timing_driven
map -ol	place_design -effort_level
map -power	power_opt_design
map -u	link_design -mode out_of_context, opt_design -retarget (skip constant propagation and sweep)
par -pl	place_design -effort_level
par -rl	route_design -effort_level
par -mt	route_design automatically runs MT with four cores in Linux or two core in Windows
par -k (keep existing placement and routing)	Default behavior for route_design
par -nopad	report_io generates pad report
par -ntd	route_design -no_timing_driven

Retrieving Tcl Command Information

For information about additional Tcl commands available to use for design implementation and analysis, please refer to the *Vivado Design Suite Tcl Command Reference Guide (UG835)*. To get help at the Tcl command prompt, type:

- `help <command>`
- `<command> -help`

For help with a category of Tcl commands, type:

- `help` (lists the categories)
- `help -category <category>`

Note: The interactive help has an auto-complete feature and is not case-sensitive, so the following categories are the same in Tcl: `end`, `EndGroup`. To save time, you can use all lower-case and auto-complete when getting help on commands or categories (for example, type `end` rather than `endgroup`).

Command Line Examples

Following is an example of a typical ISE Design Suite command line run that can be put in a `run.cmd` file. This is followed by the same run using Vivado Design Suite Tcl commands that can be put into a `run.tcl` file.

Example 1: Mapping ISE Design Suite Commands to Vivado Design Suite Tcl Commands

ISE Design Suite Command Line

```
xst -ifn design_top.xst

#-ifn (input file name with project settings and options)
ngdbuild -sd .. -dd . -p xc7v585tffg1157-2 -uc design_top.ucf design_top.ngd

#-sd (search directory), -dd (destination directory), -p (part), -uc (UCF
#file)
map -xe c -w -pr b -ol high -t 2 design_top_map.ncd design_top.pcf
#-xe c (extra effort), -w (overwrite existing file), -pr b (push registers
#into IOBs), -ol (overall effort), -t (placer cost table)
par -xe c -w -ol high -t 2 design_top_map.ncd design_top.ncd
#-xe c (extra effort), -w (overwrite existing file), -ol (overall effort), -t
#(placer cost table)
trce -u -e 10 design_top.ncd design_top.pcf
#-u (report uncovered paths), -e (generate error report)
bitgen -w design_top.ncd design_top.pcf
```

Equivalent Vivado Design Suite Tcl Command

```
set design_name design_top

#read inputs
read_verilog { $design_name.v source2.v source3.v }
read_vhdl -lib mylib { libsource1.vhdl libsource2.vhdl }
read_xdc $design_name.xdc
#run flow and save the database
synth_design -top $design_name -part xc7v585tffg1157-21
write_checkpoint -force ${design_name}_post_synth.dcp
opt_design
place_design
write_checkpoint -force ${design_name}_post_place.dcp
report_utilization -file post_place_util.txt
route_design
#Reports are not generated by default
report_timing_summary -file post_route_timing.txt
#Save the database after post route
write_checkpoint -force ${design_name}_post_route.dcp
#Check for DRC
report_drc -file post_route_drc.txt
# Write Bitstream
write_bitstream -force ${design_name}.bit
```

Example 2: Vivado Design Suite Tcl Commands for Third-Party Synthesis (starting from EDIF)

```
set design_name design_top

#read inputs
read_edif { source1.edf source2.edf $design_name.edf }
read_xdc $design_name.xdc
link_design -part xc7v585tffg1157-2 -top $design_name
#Reports are not generated by default
report_timing_summary -file post_synth_timing_summ.txt
opt_design
place_design
write_checkpoint -force ${design_name}_post_place.dcp
report_utilization -file post_place_util.txt
route_design
#Reports are not generated by default
report_timing_summary -file post_route_timing.txt _summ.txt
#Save the database after post route
```

```
write_checkpoint -force ${design_name}_post_route.dcp
#Check for DRC
report_drc -file post_route_drc.txt
# Write Bitstream
write_bitstream -force ${design_name}.bit
```

Mapping Makefiles

A makefile is a text file that is referenced by the make command and controls how the make command compiles and links a program. The makefile consists of rules for when to carry out an action and contains information such as source-level and build-order dependencies. To determine the sequence of the compilation commands, the makefile checks the timestamps of dependent files. The following example shows how to write makefiles.

Example: Mapping an ISE Design Suite Makefile to Vivado Design Suite Makefile

Sample Makefile used in the ISE Design Suite

```
DESIGN = test
DEVICE = xc7v585tffg1157-2
UCF_FILE = ../Src/${DESIGN}.ucf
EDIF_FILE = ../Src/${DESIGN}.edf

# Make all runs to place & route
all : place_n_route

# bitstream : Creates device bitstream
bitstream : ./${DESIGN}.bit

# place_n_route: Stops after place and route for analysis prior to bitstream
generation
place_n_route : ${DESIGN}.ncd

# translate: Stops after full design elaboration for analysis and floorplanning prior
to place and route step
translate : ${DESIGN}.ngd

# Following executes the ISE run

${DESIGN}.bit : ${DESIGN}.ncd
bitgen -f ${DESIGN}.ut ${DESIGN}.ncd
```

```
${DESIGN}.ncd : ${DESIGN}_map.ncd
par -w -ol high ${DESIGN}_map.ncd ${DESIGN}.ncd ${DESIGN}.pcf

${DESIGN}_map.ncd : ${DESIGN}.ngd
map -w -ol high -o ${DESIGN}_map.ncd ${DESIGN}.ngd ${DESIGN}.pcf

${DESIGN}.ngd : ${EDIF_FILE} ${UCF_FILE}
ngdbuild -uc ${UCF_FILE} -p ${DEVICE} ${EDIF_FILE} ${DESIGN}.ngd

# Clean up all the files from the Vivado run
clean :
rm -rf *.ncd *.ngd *.bit *.mrp *.map *.par *.bld *.pcf *.xml *.bgn *.html \
           *.lst *.ngo *.xrpt *.unroutes *.xpi *.txt *.pad *.csv *.ngm xlnx_auto* \
            _xmsgs *.ptwx

# Tar and compress all the files
tar :
tar -zcvf ${DESIGN}.tar.gz *.ncd *.ngd *.mrp *.map *.par *.bld *.pcf *.bgn \
              Makefile
```

Equivalent Makefile Used in the Vivado Design Suite

```
DESIGN = test
DEVICE = xc7v585tffg1157-2
XDC_FILE = ../Src/${DESIGN}.xdc
EDIF_FILE = ../Src/${DESIGN}.edf

# Make all runs to place & route
all : place_n_route

# bitstream : Creates device bitstream
bitstream : ./${DESIGN}.bit

# place_n_route: Stops after place and route for analysis prior to bitstream
generation
place_n_route : ./${DESIGN}_route.dcp

# translate: Stops after full design elaboration and initial optimization for
analysis and floorplanning prior to place and route step
translate : ./${DESIGN}_opt.dcp

# Following calls Tcl files for each desired portion of the Vivado run
# Design checkpoint files and bit file used for dependency management
```

```
./${DESIGN}.bit : ./run_vivado_place_n_route.tcl ./${DESIGN}_route.dcp
vivado -mode batch -source run_vivado_bitstream.tcl -tclargs ${DESIGN}

./${DESIGN}_route.dcp : ./run_vivado_place_n_route.tcl ./${DESIGN}_opt.dcp
vivado -mode batch -source run_vivado_place_n_route.tcl -tclargs \
            ${DESIGN}

./${DESIGN}_opt.dcp : ./run_vivado_opt.tcl ${EDIF_FILE} ${XDC_FILE}
vivado -mode batch -source run_vivado_opt.tcl -tclargs ${DESIGN} ${DEVICE}
${EDIF_FILE} ${XDC_FILE}

# Clean up all the files from the Vivado run
clean :
rm -f *.jou *.log *.rpt *.dcp *.bit *.xml *.html

# Tar and compress all the files
tar :
tar -zcvf ${DESIGN}.tar.gz *.jou *.log *.rpt *.dcp *.tcl Makefile
```

Associated Tcl Files for Vivado Makefile

run_vivado_opt.tcl

```
# Gathering TCL Args
set DESIGN [lindex $argv 0]
set DEVICE [lindex $argv 1]
set EDIF_FILE [lindex $argv 2]
set XDC_FILE [lindex $argv 3]

# Reading EDIF/NGC file
read_edif ../Src/${DESIGN}.edf

# Linking Design
link_design -part ${DEVICE} -edif_top_file ../Src/${DESIGN}.edf

# Running Logic Optimization
opt_design

# Adding Constraints
read_xdc ${XDC_FILE}

# Saving Run
write_checkpoint -force ./${DESIGN}_opt.dcp
```

```
# Creating opt reports
report_utilization -file ${DESIGN}_utilization_opt.rpt
report_timing_summary -max_paths 10 -nworst 1 -input_pins -
report_io -file ${DESIGN}_io_opt.rpt
report_clock_interaction -file ${DESIGN}_clock_interaction_opt.rpt

exit
```

run_vivado_place_n_route.tcl

```
# Gathering TCL Arg
set DESIGN [lindex $argv 0]

read_checkpoint ./${DESIGN}_opt.dcp

# Placing Design
place_design
write_checkpoint -force ./${DESIGN}_place.dcp

# Routing Design
route_design

# Saving Run
write_checkpoint -force ./${DESIGN}_route.dcp

# Creating route reports
report_timing_summary -max_paths 10 -nworst 1 -input_pins -
report_drc -file ${DESIGN}_drc_route.rpt

exit
```

run_vivado_bitstream.tcl

```
# Gathering TCL Arg
set DESIGN [lindex $argv 0]

read_checkpoint ./${DESIGN}_route.dcp
# Create bitstream
write_bitstream -force ${DESIGN}.bit

exit
```

Note: This flow exits and reenters Vivado for the defined steps in the makefile. While this allows greater run control from the make infrastructure, it is not the most efficient in execution time because you must exit and restart the software, and the reload the design for each defined step. Building this entire run in Tcl could be more efficient in runtime as the design can remain in memory from step to step if that is more desired than having makefile control.

Understanding the Differences in Messages

The Vivado Design Suite uses the same ISE Design Suite concept of Info, Warning, and Error with unique identifying numbers for messages (for example, ngdbuild:604). Applications have messages with unique identifying numbers (HDL-189). Additionally, the Vivado Design Suite includes two new message types: Status and Critical Warning.

- Status provides information about the current tool process.
- Critical Warnings in Vivado Design Suite are equivalent to Errors in the ISE Design Suite, except the Vivado design process is not stopped. Critical Warnings in the design are upgraded to Error at the bitstream generation (`write_bitstream`) stage, which stops the design process.

RECOMMENDED: *Xilinx recommends that you resolve any Critical Warnings before moving forward with your design.*

Table 2-4 shows the five message types in the Vivado Design Suite, indicates whether use intervention is required, and describes the message purpose.

Table 2-4: **Vivado Design Suite Message Types**

Type	Intervention?	Purpose
STATUS	No	Provides general status of the process and feedback to the user regarding design processing. A *STATUS* message is the same as an *INFO* message, excluding the severity and message ID tag.
INFO	No	Provides general status of the process and feedback to the user regarding design processing. An *INFO* message is the same as a *STATUS* message but includes a severity and message ID tag for filtering or searching.
WARNING	Optional	Indicates that design results may be sub-optimal because constraints or specifications may not be applied as intended. The processing will continue to completion and valid results will be generated.
CRITICAL WARNING	Recommended	Indicates a problem that will likely result in improperly functioning hardware and result in an *ERROR* later in the flow. The processing continues until an *ERROR* is encountered.
ERROR	Yes	Indicates a problem that renders design results unusable and cannot be resolved without user intervention. Further processing is not possible.

Understanding Reporting Differences

In the ISE Design Suite, reports are automatically generated when each application is run through the design flow. This includes but is not limited to the following:

- .syr for xst
- .bld for ngdbuild
- .mrp for map
- .par for par
- .twr for trce
- .pwr for xpwr

In the Vivado Design Suite, you can generate reports at any design stage. The benefits of on-demand report generation are:

- **Better runtime**: You can better manage runtime by generating reports only when needed.
- **More available reports**: At any point in the design flow, you can generate a report, making more reports available to you. For example, you can generate a utilization report for your design post-synthesis, post-optimization or post-route to see current information.

When you use Project Mode in the Vivado IDE, a fixed set of reports is automatically generated and can be accessed from the Reports window.

When you use Non-Project Mode with Tcl commands or a script, you must add Tcl reporting commands to generate reports at stages where you require them while the design is in memory.

Specific `report_*` commands report different types of information, for example, utilization, timing, and DRC results. By default, the report output is sent to the tool transcript and the `vivado.log` file, but it can also be directed to a file. For a list of reports and their descriptions, type `help -category report` at the Tcl Command prompt.

Table 2-5 shows the relationship between ISE Design Suite report information and Vivado Design Suite reporting commands.

Table 2-5: **ISE Design Suite Reports and Vivado Design Suite Reports**

ISE Design Suite Information (Report)	Vivado Design Suite Command
Utilization Information (.syr, .mrp, .par)	`report_utilization`, `report_clock_utilization`
I/O Information (.pad)	`report_io`
Timing Information (.par, .twr)	`report_timing`, `report_timing_summary`
Power Information (.pwr)	`report_power`

Understanding Log File Differences

The ISE Design Suite tools generate status and output information as a part of the individual command log files. For instance, the output status and progress of the Map executable is placed into the `.map` file where the output of PAR is placed in the `.par` file.

The Vivado Design Suite uses a single log file to capture all tool commands and output. The file is named `vivado.log` by default, which you can change by using the vivado `-log` option. The Vivado Design Suite log file displays flow progress using phases. Each phase has a name and number and a single-line performance summary. Following is an example:

```
report_timing: Time (s): cpu = 00:03:57 ; elapsed = 00:03:55 . Memory (MB): peak =
6526.066 ; gain = 64.125
```

Where:

- `cpu` is total run time for all processors.
- `elapsed` is the actual time spent running the process.
- `peak` is the maximum memory usage up to that particular design step.
- `gain` is the incremental contribution of a design step to the peak memory usage. For example, report_timing added 64.125 MB to the peak memory usage in the example above.

Migrating UCF Constraints to XDC

Overview

The Vivado™ Integrated Design Environment (IDE) does not support the User Constraint File (UCF) constraints used in the ISE® Design Suite. You must migrate the designs with UCF constraints to the Xilinx Design Constraint (XDC) format.

- For information on XDC constraints, see the V*ivado Design Suite User Guide: Using Constraints (UG903)* [Ref 3].
- For information on UCF constraints, see the *Constraints Guide (UG625)* [Ref 4].

As with UCF, XDC consists of:

- Timing constraints, for which XDC timing constraints are based on the Synopsys Design Constraint (SDC)
- Physical constraints

Differences Between XDC and UCF Constraints

The fundamental differences between XDC and UCF constraints are:

- XDC is a sequential language, with clear precedence rules.
- UCF constraints are typically applied to nets, for which XDC constraints are typically applied to pins, ports, and cell objects.
- UCF PERIOD constraints and XDC `create_clock` command are not always equivalent and can lead to different timing results.
- UCF by default does not time between asynchronous clock groups, while in XDC, all clocks are considered related and timed unless otherwise constrained (`set_clock_groups`)
- In XDC, multiple clocks can exist on the same object.

UCF to XDC Mapping

Table 3-1, shows the main mapping between UCF constraints to XDC commands.

Table 3-1: **UCF to XDC Mapping**

UCF	SDC
TIMESPEC PERIOD	create_clock create_generated_clock
OFFSET = IN <x> BEFORE <clk>	set_input_delay
OFFSET = OUT <x> BEFORE <clk>	set_output_delay
FROM:TO "TS_"*2	set_multicycle_path
FROM:TO	set_max_delay
TIG	set_false_path
NET "clk_p" LOC = AD12	set_property LOC AD12 [get_ports clk_p]
NET "clk_p" IOSTANDARD = LVDS	set_property IOSTANDARD LVDS [get_ports clk_p]

Constraint Sequence

Whether you use one or more XDC files for your design, Xilinx recommends that you organize your constraints in the following sequence:

```
## Timing Assertions Section
# Primary clocks
# Virtual clocks
# Generated clocks
# Clock Groups
# Input and output delay constraints

## Timing Exceptions Section (sorted by precedence)
# False Paths
# Max Delay / Min Delay
# Multicycle Paths
# Case Analysis
# Disable Timing

## Physical Constraints Section
# located anywhere in the file, preferably before or after the timing constraints
# or stored in a separate XDC file
```

For more information on XDC commands, see:

- *Vivado Design Suite User Guide: Using Constraints (UG903)* [Ref 4]
- *Vivado Design Suite Tcl Command Reference Guide (UG835)* [Ref 5]

Converting UCF to XDC in the PlanAhead Tool

The PlanAhead™ tool assists in converting UCF constraints to XDC when you open an ISE Design Suite or PlanAhead tool project that contains UCF constraints. When a design is loaded into the database, you can use the `write_xdc` command to convert a large percentage of the UCF constraints. You need to manually verify the output file, and you will most likely need to manually convert some constraints to XDC to ensure that all the design constraints are correct.

The Tcl command `write_xdc` requires that a synthesized netlist be open and one or more UCF files loaded. From the PlanAhead tool, do the following:

1. Open your project that contains UCF constraints.
2. Click **Open Synthesized Design**.
3. In the Tcl Console, type:

```
write_xdc filename.xdc.
```

The `write_xdc` command is not a file converter. The command writes out the constraints that were successfully applied to the design as an XDC file. The output XDC file contains:

- A comment with the file and line number from the UCF for each converted UCF.
- A comment for each conversion not done.

IMPORTANT: *Pay attention to Critical Warnings that indicate which constraints were not successfully converted.*

This conversion is only a starting point for migration to XDC-based constraints.

RECOMMENDED: *Xilinx recommends that XDC timing constraints be created without using the conversion process, because fundamental differences between UCF and XDC make automation less than optimal.*

Using the PlanAhead tool for converting UCF is best for physical constraints and basic timing constraints. Timing constraints for simple clock definitions and IO delays typically translate well. Xilinx recommends that you manually convert timing exceptions. Many will not translate and others can produce sub-optimal results. Fundamental differences between the timer in the Vivado IDE (XDC/SDC) and the timer in ISE Design Suite (UCF) make direct translation impossible. For that reason the UCF constraint must be re-evaluated and a new approach might be required with XDC.

Conversion can be done with an elaborated RTL design; however, many objects referenced in a typical UCF will not exist at that stage and thus will not be applied to the database.

Only constraints that are successfully applied to the database can be written out as XDC. Thus simple clock and IO delay constraints typically can be translated from an elaborated RTL design.

Timing Constraints

The following ISE Design Suite timing constraints can be represented as XDC timing constraints in the Vivado Design Suite. Each constraint description contains a UCF example and the equivalent XDC example.

UCF and XDC differ when creating clocks on a net that is not directly connected to the boundary of the design (such as port). In XDC, when defining a primary clock with `create_clock` on a net the source point is the driving pin of the net. The clock insertion delay before that point is ignored. This can be a problem when timing the clock with another related clock; the skew will not be accurate. The `create_clock` must be used where the clock trees originate; not in the middle of the design (for example, input port or GT clock output pin). Only generated clocks should be created in the middle of the design.

Clock Constraints

PERIOD

UCF Example

```
NET "clka" TNM_NET = "clka";
TIMESPEC "TS_clka" = PERIOD "clka" 13.330 ns HIGH 50.00%;
```

XDC Example

```
create_clock -name clka -period 13.330 -waveform {0 6.665} [get_ports clka]
```

PERIOD Constraints with Uneven Duty Cycle

UCF Example

```
NET "clka" TNM_NET = "clka";
TIMESPEC "TS_clka" = PERIOD "clka" 13.330 ns HIGH 40.00%;
```

XDC Example

```
create_clock -name clka -period 13.330 -waveform {0 5.332} [get_ports clka]
```

Generated Clocks Constraints

UCF Example

```
NET "gen_clk" TNM_NET = "gen_clk";
TIMESPEC "TS_gen_clk" = PERIOD "gen_clk" "TS_clka" * 0.500 HIGH 50.00%;
```

XDC Example

```
create_generated_clock -source [get_ports clka] -name gen_clk -multiply_by 2
[get_ports gen_clk]
```

Period Constraints with LOW Keyword

UCF Example

```
NET "clka" TNM_NET = "clka";
TIMESPEC "TS_clka" = PERIOD "clka" 13.330 ns LOW 50.00%;
```

XDC Example

```
create_clock -name clka -period 13.330 -waveform {6.665 13.330} [get_ports clka]
```

Net PERIOD Constraints

UCF Example

```
NET "clk_bufg" PERIOD = 10 ns;
```

XDC Example

```
create_clock -name clk_bufg -period 10 -waveform {0 5} [get_pins clk_bufg/O}
```

Note: Unless there is specific reason to define the clock on `bufg/O`, define it at an upstream top-level port.

OFFSET IN

BEFORE

UCF Example

```
OFFSET = IN 8 BEFORE clka;
```

XDC Example

```
set_input_delay -clock clka 2 [all_inputs]
```

Note: This assumes the clock period is 10 ns.

AFTER

UCF Example

```
OFFSET = IN 2 AFTER clka;
```

XDC Example

```
set_input_delay -clock clka 2 [all_inputs]
```

Note: This assumes the clock period is 10 ns.

BEFORE an Input Port Net

UCF Example

```
NET enable OFFSET = IN 8 BEFORE clka;
```

XDC Example

```
set_input_delay 2 [get_ports enable]
```

Note: This assumes the clock period is 10 ns.

BEFORE an Input Port Bus

UCF Example

```
INST "processor_data_bus[*]" TNM = "processor_bus";
TIMEGRP "processor_bus" OFFSET = IN 8ns BEFORE "clka";
```

XDC Example

```
set_input_delay 2 [get_ports {processor_data_bus[*]}]
```

Note: Offset is applied to ports only.

To TIMEGROUP

UCF Example

```
INST "input_ffs[*]" TNM = "input_ffs";
OFFSET = IN 8ns BEFORE "clka" TIMEGRP "input_ffs";
```

XDC Example

Manual conversion is required. For more information, see Timegroups, page 33.

FALLING/RISING Edge

UCF Example

```
OFFSET = IN 8ns BEFORE "clka" FALLING;
```

XDC Example

```
set_input_delay -clock clka 2 [all_inputs]
```

Note: This assumes the clock period is 10 ns.

LOW/HIGH Keyword

UCF Example

```
OFFSET = IN 8ns BEFORE "clka" HIGH;
```

XDC Example

Manual conversion is required.

Note: HIGH/LOW keywords are precursors to RISING/FALLING. RISING/FALLING is the preferred method.

VALID Keyword

UCF Example

```
OFFSET = IN 1ns VALID 2ns BEFORE clka;
```

XDC Example

```
set_input_delay -clock clka -max 9 [all_inputs]
set_input_delay -clock clka  -min 1[all_inputs]
```

Note: This assumes the clock period is 10 ns.

OFFSET OUT

AFTER

UCF Example

```
OFFSET = OUT 12 AFTER clkc;
```

XDC Example

```
set_output_delay -clock clkc 8 [all_outputs]
```

Note: This assumes the clock period is 20 ns.

BEFORE

UCF Example

```
OFFSET = OUT 8 BEFORE clkc;
```

XDC Example

```
set_output_delay -clock clkc 8 [all_outputs]
```

Note: This assumes the clock period is 20 ns.

Output Net

UCF Example

```
NET out_net OFFSET = OUT 12 AFTER clkc;
```

XDC Example

```
set_output_delay 8 [get_port out_net]
```

Note: This assumes the clock period is 20 ns.

Group of Outputs

UCF Example

```
TIMEGRP outputs OFFSET = OUT 12 AFTER clkc;
```

XDC Example

```
set_output_delay -clock clkc 8 [get_ports outputs*]
```

Note: This assumes the clock period is 20 ns.

From a TIMEGROUP

UCF Example

```
OFFSET = OUT 1.2 AFTER clk TIMEGRP from_ffs;
```

XDC Example

Manual conversion is required.

FALLING/RISING Edges

UCF Example

```
OFFSET = OUT 12 AFTER clkc FALLING;
```

XDC Example

```
set_output_delay -clock clkc -clock_fall 8 [all_outputs]
```

LOW Keyword

UCF Example

```
OFFSET = OUT 12 AFTER clkc LOW;
```

XDC Example

Manual conversion is required.

Note: HIGH/LOW keywords are precursors to RISING/FALLING. RISING/FALLING is the preferred method.

REFERENCE_PIN

UCF Example

```
TIMEGRP mac_ddr_out;
OFFSET = OUT AFTER clk REFERENCE_PIN clk_out RISING;
```

XDC Example

Manual conversion is required.

Note: REFERENCE_PIN acts as a reporting switch to tell TRACE to output a bus skew report. The Vivado Design Suite does not support this feature.

From:To Constraints

In general, UCF From:To constraints are converted to either `set_max_delay` or `set_min_delay` XDC constraints, with the `-from`, `-to` and `-through` design-dependent arguments.

The intent of UCF constraints should be to use the equivalent XDC constraints. While most UCF constraints are net-based, XDC constraints should be constructed to ports and pins.

Some helpful XDC commands for these constraints are `all_fanout`, `get_cells`, and `get_pins` as well as the `-from`, `-to`, and `-through` arguments.

Miscellaneous

Assigning Timing Group to an Area Group

UCF Example

```
TIMEGRP clock_grp = AREA_GROUP clock_ag;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

EXCEPT

UCF Example

```
TIMEGRP my_group = FFS EXCEPT your_group;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

Timing Ignore

Between Groups

UCF Example

```
TIMESPEC TS_TIG = FROM reset_ff TO FFS TIG;
```

XDC Example

Manual conversion is required.

Construct a set_false_path that covers the desired paths.

By Net

UCF Example

```
NET reset TIG;
```

XDC Example

```
set_false_path -through [get_nets reset]
```

Note: A better approach is to find the primary reset port and use:

```
set_false_path -from [get_ports reset_port]
```

By Instance

UCF Example

```
INST reset TIG;
```

XDC Example

```
set_false_path -from [get_cells reset]
set_false_path -through [get_cells reset]
set_false_path -to [get_cells reset]
```

By Pin

UCF Example

```
PIN ff.d TIG;
```

XDC Example

```
set_false_path -to [get_pins ff/d]
set_false_path -from [get_pins ff/q]
set_false_path -through [get_pins lut/i0]
```

Specific Time Constraints

UCF Example

```
NET reset TIG = TS_fast TS_even_faster;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

Note: Constraint-specific TIG tries to disable timing through the net, but only for analysis of the two referenced constraints.

MAXSKEW

UCF Example

```
NET local_clock MAXSKEW = 2ns;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

MAXDELAY

UCF Example

```
NET local_clock MAXDELAY = 2ns;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

Timegroups

You can use Tcl variables with timing exceptions to accomplish the same effect as an INST/TNM and a TIMESPEC. The following example illustrates the point.

UCF Example:

```
INST "DUT/BLOCK_A/data_reg[*]” TNM = "from_data_reg_0";
INST "DUT/BLOCK_A/addr_reg[*]” TNM = "from_data_reg_0";
INST "DUT/BLOCK_B/data_sync[*]” TNM = "to_data_reg_0";
INST "DUT/BLOCK_B/addr_sync[*]” TNM = "to_data_reg_0";
TIMESPEC "TS_MCP" = FROM "from_data_reg_0" TO "to_data_reg_0" TS_FSCLK * 3;
```

Tcl Equivalent:

```
set from_data_reg_0 [get_cells {DUT/BLOCK_A/data_reg[*] DUT/BLOCK_A/addr_reg[*]}];
set to_data_reg_0 [get_cells {DUT/BLOCK_B/data_sync[*] DUT/BLOCK_B/addr_sync[*]}];
set_multicycle_path -setup 3 -from $from_data_reg_0 -to $to_data_reg_0;
set_multicycle_path -hold 2 -from $from_data_reg_0 -to $to_data_reg_0;
```

Physical Constraints

Following are the ISE Design Suite physical constraints that can be represented as XDC constraints in the Vivado Design Suite. Each constraint description contains:

- Target object type
- Constraint value type
- UCF example
- Equivalent XDC example

Placement-Related Constraints

AREA_GROUP

Applied To

Cells

Constraint Values

String

UCF Example

```
INST bmg0 AREA_GROUP = AG1;
```

XDC Example

```
create_pblock ag1; add_cells_to_pblock [get_pblocks ag1]  [get_cells [list bmg0]]
```

AREA_GROUP RANGE

SLICE

Applied To

Area groups and Pblocks

Constraint Values

SLICE_XnYn[:SLICE_XnYn]

UCF Example

```
AREA_GROUP AG1 RANGE = SLICE_X0Y44:SLICE_X27Y20;
```

XDC Example

```
resize_pblock [get_pblocks ag1] -add {SLICE_X0Y44:SLICE_X27Y20}
```

RAMB18

Applied To

Area groups and Pblocks

Constraint Values

RAMB18_XnYn:RAMB18_XnYn

UCF Example

```
AREA_GROUP AG1 RANGE = RAMB18_X0Y86:RAMB18_X3Y95;
```

XDC Example

```
resize_pblock [get_pblocks ag1] -add {RAMB18_X0Y86:RAMB18_X3Y95}
```

RAMB36

Applied To

Area groups and Pblocks

Constraint Values

RAMB36_XnYn:RAMB36_XnYn

UCF Example

```
AREA_GROUP AG1 RANGE = RAMB36_X0Y11:RAMB36_X3Y18;
```

XDC Example

```
resize_pblock [get_pblocks ag1] -add {RAMB36_X0Y11:RAMB36_X3Y18}
```

CLOCKREGION (1)

Applied To

Area groups and Pblocks

Constraint Values

CLOCKREGION_XnYn

UCF Example

```
area_group ag1 range = CLOCKREGION_X0Y0;
```

XDC Example

```
resize_pblock [get_pblocks ag1] -add {CLOCKREGION_X0Y0:CLOCKREGION_X0Y0}
```

CLOCKREGION (2)

Applied To

Area groups and Pblocks

Constraint Values

CLOCKREGION_XnYn[:CLOCKREGION_XnYn]

UCF Example

```
area_group ag1 range = CLOCKREGION_X0Y0:CLOCKREGION_X1Y0;
```

XDC Example

```
resize_pblock [get_pblocks ag1] -add {CLOCKREGION_X0Y0:CLOCKREGION_X0Y0}
```

CLOCKREGION (3)

Applied To

Area groups and Pblocks

Constraint Values

CLOCKREGION_XnYn,CLOCKREGION_XnYn, . . .

UCF Example

```
area_group ag1 range = CLOCKREGION_X0Y0, CLOCKREGION_X1Y0;
```

XDC Example

```
resize_pblock [get_pblocks ag1] -add {CLOCKREGION_X0Y0:CLOCKREGION_X0Y0
CLOCKREGION_X1Y0:CLOCKREGION_X1Y0}
```

DSP48

Applied To

Area groups and Pblocks

Constraint Values

DSP48_XnYn:DSP48_XnYn

UCF Example

```
AREA_GROUP D1 RANGE = DSP48_X2Y0:DSP48_X2Y9;
```

XDC Example

```
resize_pblock [get_pblocks D1] -add {DSP48_X2Y0:DSP48_X2Y9}
```

BUFGCTRL

Applied To

Area groups and Pblocks

Constraint Values

BUFGCTRL_XnYn:BUFGCTRL_XnYn

UCF Example

```
AREA_GROUP ag1 range = BUFGCTRL_X0Y24:BUFGCTRL_X0Y31;
```

XDC Example

```
resize_pblock [get_pblocks ag1] -add {BUFGCTRL_X0Y24:BUFGCTRL_X0Y31}
```

BUFHCE

Applied To

Area groups and Pblocks

Constraint Values

BUFHCE_XnYn:BUFHCE_XnYn

UCF Example

```
AREA_GROUP ag1 range = BUFHCE_X0Y72:BUFHCE_X1Y77;
```

XDC Example

```
resize_pblock [get_pblocks ag1] -add {BUFHCE_X0Y72:BUFHCE_X1Y77}
```

BUFR

Applied To

Area groups and Pblocks

Constraint Values

BUFR_XnYn:BUFR_XnYn

UCF Example

```
AREA_GROUP ag1 range = BUFR_X0Y20:BUFR_X1Y23;
```

XDC Example

```
resize_pblock [get_pblocks ag1] -add {BUFR_X0Y0:BUFR_X1Y2}
```

BUFIO

Applied To

Area groups and Pblocks

Constraint Values

BUFIO_XnYn:BUFIO_XnYn

UCF Example

```
AREA_GROUP ag1 range = BUFIO_X0Y8:BUFIO_X0Y11;
```

XDC Example

```
resize_pblock [get_pblocks ag1] -add {BUFIO_X0Y8:BUFIO_X0Y11}
```

IOB Range

Applied To

Area groups and Pblocks

Constraint Values

IOB_XnYn:IOB_XnYn

UCF Example

```
AREA_GROUP ag1 range = IOB_X0Y341:IOB_X1Y349;
```

XDC Example

```
resize_pblock [get_pblocks ag1] -add {IOB_X0Y341:IOB_X1Y349}
```

IN_FIFO

Applied To

Area groups and Pblocks

Constraint Values

IN_FIFO_XnYn:IN_FIFO_XnYn

UCF Example

```
AREA_GROUP ag1 range = IN_FIFO_X0Y24:IN_FIFO_X1Y27;
```

XDC Example

```
resize_pblock [get_pblocks ag1] -add {IN_FIFO_X0Y24:IN_FIFO_X1Y27}
```

OUT_FIFO

Applied To

Area groups and Pblocks

Constraint Values

OUT_FIFO_XnYn:OUT_FIFO_XnYn

UCF Example

```
AREA_GROUP ag1 range = OUT_FIFO_X0Y24:OUT_FIFO_X1Y27;
```

XDC Example

```
resize_pblock [get_pblocks ag1] -add {OUT_FIFO_X0Y24:OUT_FIFO_X1Y27}
```

ILOGIC

Applied To

Area groups and Pblocks

Constraint Values

ILOGIC_XnYn:ILOGIC_XnYn

UCF Example

```
AREA_GROUP ag1 range = ILOGIC_X0Y76:ILOGIC_X0Y79;
```

XDC Example

```
resize_pblock [get_pblocks ag1] -add {ILOGIC_X0Y76:ILOGIC_X0Y79}
```

OLOGIC

Applied To

Area groups and Pblocks

Constraint Values

OLOGIC_XnYn:OLOGIC_XnYn

UCF Example

```
AREA_GROUP ag1 range = OLOGIC_X0Y76:OLOGIC_X0Y79;
```

XDC Example

```
resize_pblock [get_pblocks ag1] -add {OLOGIC_X0Y76:OLOGIC_X0Y79}
```

LOC

IOB

Applied To

Port nets

Constraint Values

IOB site

UCF Example

```
NET p[0] LOC = H1;
```

XDC Example

```
set_property PACKAGE_PIN H1 [get_ports p[0]]
```

TIP: *To assign pins in the Vivado Design Suite, use the `PACKAGE_PIN` port property, and not the `LOC` property which is used for cells.*

SLICE (1)

Applied To

Cells

Constraint Values

Site range

UCF Example

```
INST a_reg[*] LOC = SLICE_X25Y*;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

SLICE (2)

Applied To

Cells

Constraint Values

SLICE_XnYn

UCF Example

```
INST a_reg[0] LOC = SLICE_X4Y4;
```

XDC Example

```
set_property LOC SLICE_X4Y4 [get_cells a_reg[0]]
```

Applied To

Cells

Constraint Values

RAMB18_XnYn

UCF Example

```
INST ram0 LOC = RAMB18_X0Y5;
```

XDC Example

```
set_property LOC RAMB18_X0Y5 [get_cells ram0]
```

RAMB36

Applied To

Cells

Constraint Values

RAMB36_XnYn

UCF Example

```
INST ram0 LOC = RAMB36_X0Y0;
```

XDC Example

```
set_property LOC RAMB36_X0Y0 [get_cells ram0]
```

DSP48

Applied To

Cells

Constraint Values

DSP48_XnYn

UCF Example

```
INST dsp0 LOC = DSP48_X0Y10;
```

XDC Example

```
set_property LOC DSP48_X0Y10 [get_cells dsp0]
```

BUFGCTRL

Applied To

Cells

Constraint Values

BUFGCTRL_XnYn

UCF Example

```
INST cb[0] LOC = BUFGCTRL_X0Y24;
```

XDC Example

```
set_property LOC BUFGCTRL_X0Y24 [get_cells cb[0]]
```

BUFHCE

Applied To

Cells

Constraint Values

BUFHCE_XnYn

UCF Example

```
INST cb[0] LOC = BUFHCE_X0Y72;
```

XDC Example

```
set_property LOC BUFHCE_X0Y72 [get_cells cb[0]]
```

BUFR

Applied To

Cells

Constraint Values

BUFR_XnYn

UCF Example

```
INST cb[0] LOC = BUFR_X0Y20;
```

XDC Example

```
set_property LOC BUFR_X0Y20 [get_cells cb[0]]
```

BUFIO

Applied To

Cells

Constraint Values

BUFIO_XnYn

UCF Example

```
INST cb[0] LOC = BUFIO_X0Y8;
```

XDC Example

```
set_property LOC BUFIO_X0Y8 [get_cells cb[0]]
```

IOB

Applied To

Cells

Constraint Values

IOB_XnYn

UCF Example

```
INST ib[0] LOC = IOB_X0Y341;
```

XDC Example

```
set_property LOC IOB_X0Y341 [get_cells ib[0]]
```

IN_FIFO

Applied To

Cells

Constraint Values

IN_FIFO_XnYn

UCF Example

```
INST infifo_inst LOC = IN_FIFO_X0Y24;
```

XDC Example

```
set_property LOC IN_FIFO_X0Y24 [get_cells infifo_inst]
```

OUT_FIFO

Applied To

Cells

Constraint Values

OUT_FIFO_XnYn

UCF Example

```
INST outfifo_inst LOC = OUT_FIFO_X0Y24;
```

XDC Example

```
set_property LOC OUT_FIFO_X0Y24 [get_cells outfifo_inst]
```

ILOGIC

Applied To

Cells

Constraint Values

ILOGIC_XnYn

UCF Example

```
INST ireg LOC = ILOGIC_X0Y76;k
```

XDC Example

```
set_property LOC ILOGIC_X0Y76 [get_cells ireg]
```

OLOGIC

Applied To

Cells

Constraint Values

OLOGIC_XnYn

UCF Example

```
INST oreg LOC = OLOGIC_X0Y76;
```

XDC Example

```
set_property LOC OLOGIC_X0Y76 [get_cells oreg]
```

IDELAY

Applied To

Cells

Constraint Values

IDELAY_XnYn

UCF Example

```
INST idelay0 LOC = IDELAY_X0Y21;
```

XDC Example

```
set_property LOC IDELAY_X0Y21 [get_cells idelay0]
```

IDELAYCTRL

Applied To

Cells

Constraint Values

IDELAYCTRL_XnYn

UCF Example

```
INST idelayctrl0 LOC = IDELAYCTRL_X0Y0;
```

XDC Example

```
set_property LOC IDELAYCTRL_X0Y0 [get_cells idelayctrl0]
```

BEL

A5LUT, B5LUT, C5LUT, D5LUT

Applied To

Cells

Constraint Values

A5LUT, B5LUT, C5LUT, D5LUT

UCF Example

```
INST a0 BEL = A5LUT;
```

XDC Example

```
set_property BEL A5LUT [get_cells a0]
```

A6LUT, B6LUT, C6LUT, D6LUT

Applied To

Cells

Constraint Values

A6LUT, B6LUT, C6LUT, D6LUT

UCF Example

```
INST a0 BEL = D6LUT;
```

XDC Example

```
set_property BEL D6LUT [get_cells a0]
```

AFF, BFF, CFF, DFF

Applied To

Cells

Constraint Values

AFF, BFF, CFF, DFF

UCF Example

```
INST a_reg[0] BEL = CFF;
```

XDC Example

```
set_property BEL CFF [get_cells a_reg[0]]
```

A5FF, B5FF, C5FF, D5FF

Applied To

Cells

Constraint Values

A5FF, B5FF, C5FF, D5FF

UCF Example

```
INST a_reg[0] BEL = B5FF;
```

XDC Example

```
set_property BEL B5FF [get_cells a_reg[0]]
```

F7AMUX, F7BMUX

Applied To

Cells

Constraint Values

F7AMUX, F7BMUX

UCF Example

```
INST m0 BEL = F7BMUX;
```

XDC Example

```
set_property BEL F7BMUX [get_cells m0]
```

IOB

TRUE

Applied To

FF cells

Constraint Values

TRUE

UCF Example

```
INST a1_reg[*] IOB = TRUE;
```

XDC Example

```
set_property IOB TRUE [get_cells b1_reg[*]]
```

FALSE

Applied To

FF cells

Constraint Values

FALSE

UCF Example

```
INST b1_reg[*] IOB = FORCE;
```

XDC Example

```
set_property IOB TRUE [get_cells a1_reg[*]]
```

FORCE

Applied To

FF cells

Constraint Values

FORCE

UCF Example

```
INST q_reg[*] IOB = FALSE;
```

XDC Example

```
set_property IOB TRUE [get_cells q_reg[*]]
```

The Vivado Design Suite does not support this constraint in XDC. Use TRUE instead.

H_SET

Applied To

Cells

Constraint Values

Tool-generated string

UCF Example

N/A

XDC Example

N/A

For more information, see Relative Location (RLOC) in the *Constraints Guide (UG625)*.

In the Vivado Design Suite, H_SET cells have a property called RPM.

HU_SET

Applied To

Cells

Constraint Values

String

UCF Example

```
INST u0 HU_SET = h0;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

HU_SET must be placed in HDL code as an attribute.

For more information, see Relative Location (RLOC) in the *Constraints Guide (UG625)*.

U_SET

Applied To

Cells

Constraint Values

String

UCF Example

```
INST u0 U_SET = h0; (usually set in UCF)
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

U_SET must be placed in HDL code as an attribute.

For more information, see Relative Location (RLOC) in the *Constraints Guide (UG625)*.

RLOC

Applied To

Cells

Constraint Values

XnYn

UCF Example

```
INST u0 RLOC = X2Y1;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

RLOC must be placed in HDL code as an attribute.

For more information, see Relative Location (RLOC) in the *Constraints Guide (UG625)*.

RLOC_ORIGIN

Applied To

Cells

Constraint Values

XnYn

UCF Example

```
INST u0 RLOC_ORIGIN = X144Y255;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

RLOC_ORIGIN must be placed in HDL code as an attribute.

For more information, see Relative Location (RLOC) in the *Constraints Guide (UG625)*.

RPM_GRID

Applied To

Cells

Constraint Values

GRID

UCF

```
INST u0 RPM_GRID = GRID;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

RPM_GRID must be placed in HDL code as an attribute.

For more information, see Relative Location (RLOC) in the *Constraints Guide (UG625)*.

USE_RLOC

Applied To

Cells

Constraint Values

TRUE, FALSE

UCF Example

```
INST u0 USE_RLOC = FALSE;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

RLOC_RANGE

Applied To

Cells

Constraint Values

XnYn:XnYn

UCF Example

```
INST u0 RLOC_RANGE = X1Y1:X3Y3;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

Create a Pblock with the desired range, and add the RPM cells to the Pblock.

BLKNM

Applied To

Cells

Constraint Values

String

UCF Example

```
INST u0 BLKNM = blk0;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

Use U_SET or net weight if possible.

Applied To

Nets

Constraint Values

String

UCF Example

```
NET n0 BLKNM = blk0;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

Use U_SET or net weight if possible.

HBLKNM

Applied To

Cells

Constraint Values

String

UCF Example

```
INST u0 HBLKNM = blk0;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

Use H_SET/HU_SET or net weight if possible.

Applied To

Nets

Constraint Values

String

UCF Example

```
NET n0 HBLKNM = blk0;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

Use H_SET/HU_SET or net weight if possible.

XBLKNM

Applied To

Cells

Constraint Values

String

UCF Example

```
INST u0 XBLKNM = blk0;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

Use H_SET/HU_SET or net weight if possible.

Use BEL PROHIBIT to keep out unrelated logic.

Applied To

Nets

Constraint Values

String

UCF Example

```
NET n0 XBLKNM = blk0;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

Use H_SET/HU_SET or net weight if possible.

Use BEL PROHIBIT to keep out unrelated logic.

HLUTNM

Applied To

LUT cells

Constraint Values

String

UCF Example

UCF is not allowed, only HDL.

XDC Example

```
set_property HLUTNM h0 [get_cells {LUT0 LUT1}]
```

LUTNM

Applied To

LUT cells

Constraint Values

String

UCF Example

UCF is not allowed, only HD.L

XDC Example

```
set_property LUTNM h0 [get_cells {LUT0 LUT1}]
```

USE_LUTNM

Applied To

LUT cells

Constraint Values

TRUE, FALSE

UCF Example

```
INST lut0 USE_LUTNM = FALSE;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

CLOCK_DEDICATED_ROUTE

TRUE(1)

Applied To

Nets

Constraint Values

TRUE

UCF Example

```
net clk0 CLOCK_DEDICATED_ROUTE = TRUE;
```

XDC Example

```
set_property CLOCK_DEDICATED_ROUTE TRUE [get_nets clk0]
```

TRUE(1)

Applied To

Pins

Constraint Values

TRUE

UCF Example

```
PIN clkbuf0.O CLOCK_DEDICATED_ROUTE = TRUE;
```

XDC Example

```
set_property CLOCK_DEDICATED_ROUTE TRUE [get_pins clkbuf0/O]
```

FALSE(1)

Applied To

Nets

Constraint Values

FALSE

UCF Example

```
NET clk0 CLOCK_DEDICATED_ROUTE = FALSE;
```

XDC Example

```
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets clk0]
```

FALSE(2)

Applied To

Pins

Constraint Values

FALSE

UCF Example

```
PIN clkbuf0.O CLOCK_DEDICATED_ROUTE = FALSE;
```

XDC Example

```
set_property CLOCK_DEDICATED_ROUTE FALSE [get_pins clkbuf0/O]
```

BACKBONE(1)

Applied To

Nets

Constraint Values

BACKBONE

UCF Example

```
NET clk0 CLOCK_DEDICATED_ROUTE = BACKBONE;
```

XDC Example

```
set_property CLOCK_DEDICATED_ROUTE BACKBONE [get_nets clk0]
```

BACKBONE(2)

Applied To

Pins

Constraint Values

BACKBONE

UCF Example

```
PIN clkbuf0.O CLOCK_DEDICATED_ROUTE = BACKBONE;
```

XDC Example

```
set_property CLOCK_DEDICATED_ROUTE BACKBONE [get_pins clkbuf0/O]
```

I/O-Related Constraints

HIODELAY_GROUP

Applied To

IDELAY and IDELAYCTRL cells

Constraint Values

String

UCF Example

```
INST idelay0 HIODELAY_GROUP = group0;
```

XDC Example

```
set_property HIODELAY_GROUP group0 [get_cells idelay0]
```

IODELAY_GROUP

Applied To

IDELAY and IDELAYCTRL cells

Constraint Values

String

UCF Example

```
INST idelay0 IODELAY_GROUP = group0;
```

XDC Example

```
set_property IODELAY_GROUP group0 [get_cells idelay0]
```

DCI_VALUE

Applied To

I/O buffer cells

Constraint Values

Integer. Resistance in Ohms

UCF Example

```
INST a_IBUF[0]_inst DCI_VALUE = 75;
```

XDC Example

```
set_property DCI_VALUE 75 [get_cells {a_IBUF[0]_inst}]
```

DIFF_TERM

Applied To

I/O buffer cells

Constraint Values

Boolean

UCF Example

```
INST a_IBUF[0]_inst DIFF_TERM = TRUE;
```

XDC Example

```
set_property DIFF_TERM true [get_cells {a_IBUF[0]_inst}]
```

DRIVE

Applied To

Inout and output buffer cells

Constraint Values

Integer: 2, 4, 6, 8, 12, 16, 24

UCF Example

```
INST q_OBUF[0]_inst DRIVE = 24;
```

XDC Example

```
set_property DRIVE 24 [get_ports q[0]]
```

LVTTL allows a value of 24.

IOSTANDARD

Applied To

I/O buffer cells

Constraint Values

I/O standard string

UCF Example

```
INST q_OBUF[0]_inst IOSTANDARD = LVCMOS25;
```

XDC Example

```
set_property IOSTANDARD LVCMOS25 [get_ports q[0]]
```

For more information, see the *Constraints Guide (UG625)*.

SLEW

Applied To

Inout and output buffer cells

Constraint Values

SLOW or FAST

UCF Example

```
INST q_OBUF[0]_inst SLEW = FAST;
```

XDC Example

```
set_property SLEW FAST [get_ports q[0]]
```

FAST

Applied To

Inout and output buffer cells

Constraint Values

N/A

UCF Example

```
INST q_OBUF[0]_inst FAST;
```

XDC Example

```
set_property SLEW FAST [get_ports q[0]]
```

SLOW

Applied To

Inout and output buffer cells

Constraint Values

N/A

UCF Example

```
INST q_OBUF[0]_inst SLOW;
```

XDC Example

```
set_property SLEW SLOW [get_ports q[0]]
```

IN_TERM

Applied To

Ports

Constraint Values

- NONE
- UNTUNED_SPLIT_40
- UNTUNED_SPLIT_50
- UNTUNED_SPLIT_60

UCF Example

```
NET a[0] IN_TERM = UNTUNED_SPLIT_50;
```

XDC Example

```
set_property IN_TERM UNTUNED_SPLIT_50 [get_ports [list clk]]
```

OUT_TERM

Applied To

Ports

Constraint Values

- NONE
- UNTUNED_25
- UNTUNED_50
- UNTUNED_75

UCF Example

```
net q[0] OUT_TERM = UNTUNED_50;
```

XDC Example

```
set_property OUT_TERM UNTUNED_50 [get_ports q[0]]
```

IOBDELAY

NONE

Applied To

Port nets

Constraint Values

NONE

UCF Example

```
net b[0] IOBDELAY = NONE;
```

XDC Example

```
set_property IOBDELAY NONE [get_nets b[0]]
```

Note: You cannot set IOBDELAY on ports. However, you can set IOBDELAY on cells such as input buffers.

BOTH

Applied To

Port nets

Constraint Values

BOTH

UCF Example

```
net b[0] IOBDELAY = BOTH;
```

XDC Example

```
set_property IOBDELAY BOTH [get_nets b[0]]
```

Note: You cannot set IOBDELAY on ports. However, you can set IOBDELAY on cells such as input buffers.

IBUF

Applied To

Port nets

Constraint Values

IBUF

UCF Example

```
net b[0] IOBDELAY = IBUF;
```

XDC Example

```
set_property IOBDELAY IBUF [get_nets b[0]]
```

Note: You cannot set IOBDELAY on ports. However, you can set IOBDELAY on cells such as input buffers.

IFD

Applied To

Port nets

Constraint Values

IFD

UCF Example

```
net b[0] IOBDELAY = IFD;
```

XDC Example

```
set_property IOBDELAY IFD [get_nets b[0]]
```

Note: You cannot set IOBDELAY on ports. However, you can set IOBDELAY on cells such as input buffers.

KEEPER

Applied To

Port nets

Constraint Values

- TRUE
- FALSE
- YES
- NO

UCF Example

```
NET n1 KEEPER = TRUE;
```

XDC Example

```
set_property KEEPER true [get_ports n1]
```

PULLDOWN

Applied To

Port nets

Constraint Values

- TRUE
- FALSE
- YES
- NO

UCF Example

```
NET n1 PULLDOWN = TRUE;
```

XDC Example

```
set_property PULLDOWN true [get_ports n1]
```

PULLUP

Applied To

Port nets

Constraint Values

- TRUE
- FALSE
- YES
- NO

UCF Example

```
NET n1 PULLUP = TRUE;
```

XDC Example

```
set_property PULLUP true [get_ports n1]
```

VCCAUX_IO

Applied To

Ports

Constraint Values

- NORMAL
- HIGH
- DONTCARE

UCF Example

```
NET d[0] VCCAUX_IO = HIGH;
```

XDC Example

```
set_property VCCAUX_IO HIGH [get_ports d[0]]
```

Miscellaneous Net-Related Constraints

KEEP

Applied To

Nets

Constraint Values

- TRUE
- FALSE

UCF Example

```
net x_int KEEP = TRUE;
```

XDC Example

```
set_property DONT_TOUCH true [get_nets x_int]
```

SAVE NET FLAG

Applied To

Nets

Constraint Values

N/A

UCF Example

```
net x_int S;
```

XDC Example

```
set_property DONT_TOUC true [get_nets x_int]
```

KEEP_HIERARCHY

Applied To

Cells

Constraint Values

- TRUE
- FALSE
- YES
- NO

UCF Example

```
INST u1 KEEP_HIERARCHY = TRUE;
```

XDC Example

```
set_property DONT_TOUCH true [get_cells u1]
```

LOCK_PINS

Applied To

LUT cell

Constraint Values

CSV string:

```
I[0-5]:A[6-1]
```

UCF Example

```
INST LUT1 LOCK_PINS = I3:A6, I2:A5;
```

XDC Example

```
set_property LOCK_PINS {I3:A6 I2:A5} [get_cells LUT1]
```

ROUTE

Applied To

Nets

Constraint Values

Directed Routing String (DIRT)

UCF Example

```
NET n85 ROUTE={2;1;-4!-1;-53320; . . .16;-8!};
```

XDC Example

```
set_property FIXED_ROUTE {EE2BEG0 NR1BEG0 CLBLL_LL_AX} [get_nets n85]
```

Note: ISE Design Suite directed routing strings and Vivado Design Suite net route properties are incompatible. Vivado uses a unique, un-encoded format.

Configuration-Related Constraints

CONFIG PROHIBIT

Pin site

Applied To

Sites

Constraint Values

Pin site

UCF Example

```
CONFIG PROHIBIT = K24, K26, K27, K28;
```

XDC Example

```
set_property PROHIBIT true [get_sites {K24 K26 K27 K28}]
```

Bank number

Applied To

Sites

Constraint Values

Bank number

UCF Example

```
CONFIG PROHIBIT = BANK34, BANK35, BANK36;
```

XDC Example

```
set_property PROHIBIT true [get_sites -of [get_iobanks 34 35 36]]
```

RAM(1)

Applied To

Sites

Constraint Values

RAMs

UCF Example

```
CONFIG PROHIBIT = RAMB18_X0Y0;
```

XDC Example

```
set_property PROHIBIT true [get_sites RAMB18_X0Y0]
```

RAM(2)

Applied To

Sites

Constraint Values

RAMs

UCF Example

```
CONFIG PROHIBIT = RAMB18_X0Y1, RAMB18_X0Y3, RAMB18_X0Y5;
```

XDC Example

```
set_property PROHIBIT true [get_sites {RAMB18_X0Y1 RAMB18_X0Y3 RAMB18_X0Y5}]
```

Note: The comma-separated list shown here uses RAM sites but can use any supported site type.

RAM(3)

Applied To

Sites

Constraint Values

RAMs

UCF Example

```
CONFIG PROHIBIT = RAMB36_X1Y1:RAMB36_X2Y2;
```

XDC Example

```
set_property PROHIBIT true [get_sites -range {RAMB36_X1Y1 RAMB36_X2Y2}]
```

RAM(4)

Applied To

Sites

Constraint Values

RAMs

UCF Example

```
CONFIG PROHIBIT = RAMB36_X3Y*;
```

XDC Example

```
set_property PROHIBIT true [get_sites RAMB36_X3Y*]
```

DSP48

Applied To

Sites

Constraint Values

DSP48s

UCF Example

```
CONFIG PROHIBIT = DSP48_X0Y*;
```

XDC Example

```
set_property PROHIBIT true [get_sites DSP48_X0Y*]
```

SLICE

Applied To

Sites

Constraint Values

Slices

UCF Example

```
CONFIG PROHIBIT = SLICE_X0Y0:SLICE_X47Y49;
```

XDC Example

```
set_property PROHIBIT true [get_sites -range {SLICE_X0Y0 SLICE_X47Y49}]
```

ILOGIC

Applied To

Sites

Constraint Values

ILOGIC

UCF Example

```
CONFIG PROHIBIT = ILOGIC_X0Y0:ILOGIC_X0Y49;
```

XDC Example

```
set_property PROHIBIT true [get_sites -range {ILOGIC_X0Y0 ILOGIC_X0Y49}]
```

OLOGIC

Applied To

Sites

Constraint Values

OLOGIC

UCF Example

```
CONFIG PROHIBIT = OLOGIC_X0Y0:OLOGIC_X0Y49;
```

XDC Example

```
set_property PROHIBIT true [get_sites -range {OLOGIC_X0Y0 OLOGIC_X0Y49}]
```

BUFGCTRL

Applied To

Sites

Constraint Values

BUFGCTRL

UCF Example

```
CONFIG PROHIBIT = BUFGCTRL_X0Y0:BUFGCTRL_X0Y15;
```

XDC Example

```
set_property PROHIBIT true [get_sites -range {BUFGCTRL_X0Y0 BUFGCTRL_X0Y15}]
```

BUFR

Applied To

Sites

Constraint Values

BUFR

UCF Example

```
CONFIG PROHIBIT = BUFR_X0Y0:BUFR_X0Y3;
```

XDC Example

```
set_property PROHIBIT true [get_sites -range {BUFR_X0Y0 BUFR_X0Y3}]
```

BUFIO

Applied To

Sites

Constraint Values

BUFIO

UCF Example

```
CONFIG PROHIBIT = BUFIO_X0Y0:BUFIO_X0Y3;
```

XDC Example

```
set_property PROHIBIT true [get_sites -range {BUFIO_X0Y0 BUFIO_X0Y3}]
```

BUFHCE

Applied To

Sites

Constraint Values

BUFHCE

UCF Example

```
CONFIG PROHIBIT = BUFHCE_X0Y0:BUFHCE_X1Y11;
```

XDC Example

```
set_property PROHIBIT true [get_sites -range {BUFHCE_X0Y0 BUFHCE_X1Y11}]
```

CONFIG INTERNAL_VREF_BANK

Voltage

Applied To

I/O bank

Constraint Values

Voltage

UCF Example

```
CONFIG INTERNAL_VREF_BANK14 = 0.75;
```

XDC Example

```
set_property INTERNAL_VREF 0.75 [get_iobanks 14]
```

NONE

Applied To

I/O bank

Constraint Values

NONE

UCF Example

```
CONFIG INTERNAL_VREF_BANK0 = NONE;
```

XDC Example

```
reset_property INTERNAL_VREF  [get_iobanks 0]
```

CONFIG DCI_CASCADE

Applied To

I/O banks

Constraint Values

Bank sequence

UCF Example

```
CONFIG DCI_CASCADE = 17 15 14;
```

XDC Example

```
set_property DCI_CASCADE {15 14} [get_iobanks 17]
```

CONFIG CONFIG_MODE

M_SERIAL

Applied To

Global

Constraint Values

M_SERIAL

UCF Example

```
CONFIG CONFIG_MODE = M_SERIAL;
```

XDC Example

```
set_property CONFIG_MODE M_SERIAL [current_design]
```

S_SERIAL

Applied To

Global

Constraint Values

S_SERIAL

UCF Example

```
CONFIG CONFIG_MODE = S_SERIAL;
```

XDC Example

```
set_property CONFIG_MODE S_SERIAL [current_design]
```

B_SCAN

Applied To

Global

Constraint Values

B_SCAN

UCF Example

```
CONFIG CONFIG_MODE = B_SCAN;
```

XDC Example

```
set_property CONFIG_MODE B_SCAN [current_design]
```

B_SCAN+READBACK

Applied To

Global

Constraint Values

B_SCAN+READBACK

UCF Example

```
CONFIG CONFIG_MODE = B_SCAN+READBACK;
```

XDC Example

```
set_property CONFIG_MODE B_SCAN+READBACK [current_design]
```

M_SELECTMAP

Applied To

Global

Constraint Values

M_SELECTMAP

UCF Example

```
CONFIG CONFIG_MODE = M_SELECTMAP;
```

XDC Example

```
set_property CONFIG_MODE M_SELECTMAP [current_design]
```

M_SELECTMAP+READBACK

Applied To

Global

Constraint Values

M_SELECTMAP+READBACK

UCF Example

```
CONFIG CONFIG_MODE = M_SELECTMAP+READBACK;
```

XDC Example

```
set_property CONFIG_MODE M_SELECTMAP+READBACK [current_design]
```

S_SELECTMAP

Applied To

Global

Constraint Values

S_SELECTMAP

UCF Example

```
CONFIG CONFIG_MODE = S_SELECTMAP;
```

XDC Example

```
set_property CONFIG_MODE S_SELECTMAP [current_design]
```

S_SELECTMAP+READBACK

Applied To

Global

Constraint Values

S_SELECTMAP+READBACK

UCF Example

```
CONFIG CONFIG_MODE = S_SELECTMAP+READBACK;
```

XDC Example

```
set_property CONFIG_MODE S_SELECTMAP+READBACK [current_design]
```

S_SELECTMAP16

Applied To

Global

Constraint Values

S_SELECTMAP16

UCF Example

```
CONFIG CONFIG_MODE = S_SELECTMAP16;
```

XDC Example

```
set_property CONFIG_MODE S_SELECTMAP16 [current_design]
```

S_SELECTMAP16+READBACK

Applied To

Global

Constraint Values

S_SELECTMAP16+READBACK

UCF Example

```
CONFIG CONFIG_MODE = S_SELECTMAP16+READBACK;
```

XDC Example

```
set_property CONFIG_MODE S_SELECTMAP16+READBACK [current_design]
```

S_SELECTMAP32

Applied To

Global

Constraint Values

S_SELECTMAP32

UCF Example

```
CONFIG CONFIG_MODE = S_SELECTMAP32;
```

XDC Example

```
set_property CONFIG_MODE S_SELECTMAP32 [current_design]
```

S_SELECTMAP32+READBACK

Applied To

Global

Constraint Values

S_SELECTMAP32+READBACK

UCF Example

```
CONFIG CONFIG_MODE = S_SELECTMAP32+READBACK;
```

XDC Example

```
set_property CONFIG_MODE S_SELECTMAP32+READBACK [current_design]
```

SPIx1

Applied To

Global

Constraint Values

SPIx1

UCF Example

```
CONFIG CONFIG_MODE = SPIx1;
```

XDC Example

```
set_property CONFIG_MODE SPIx1 [current_design]
```

SPIx2

Applied To

Global

Constraint Values

SPIx2

UCF Example

```
CONFIG CONFIG_MODE = SPIx2;
```

XDC Example

```
set_property CONFIG_MODE SPIx2 [current_design]
```

SPIx4

Applied To

Global

Constraint Values

SPIx4

UCF Example

```
CONFIG CONFIG_MODE = SPIx4;
```

XDC Example

```
set_property CONFIG_MODE SPIx4 [current_design]
```

BPI8

Applied To

Global

Constraint Values

BPI8

UCF Example

```
CONFIG CONFIG_MODE = BPI8 ;
```

XDC Example

```
set_property CONFIG_MODE BPI8  [current_design]
```

BPI16

Applied To

Global

Constraint Values

BPI16

UCF Example

```
CONFIG CONFIG_MODE = BPI16;
```

XDC Example

```
set_property CONFIG_MODE BPI16 [current_design]
```

CONFIG POST_CRC

ENABLE

Applied To

Global

Constraint Values

ENABLE

UCF Example

```
CONFIG POST_CRC = ENABLE;
```

XDC Example

```
set_property POST_CRC ENABLE [current_design]
```

DISABLE

Applied To

Global

Constraint Values

DISABLE

UCF Example

```
CONFIG POST_CRC = DISABLE;
```

XDC Example

```
set_property POST_CRC DISABLE [current_design]
```

CONFIG POST_CRC_ACTION

HALT

Applied To

Global

Constraint Values

HALT

UCF Example

```
CONFIG POST_CRC_ACTION = HALT;
```

XDC Example

```
set_property POST_CRC_ACTION HALT [current_design]
```

CONTINUE

Applied To

Global

Constraint Values

CONTINUE

UCF Example

```
CONFIG POST_CRC_ACTION = CONTINUE;
```

XDC Example

```
set_property POST_CRC_ACTION CONTINUE [current_design]
```

CORRECT_AND_CONTINUE

Applied To

Global

Constraint Values

CORRECT_AND_CONTINUE

UCF Example

```
CONFIG POST_CRC_ACTION = CORRECT_AND_CONTINUE;
```

XDC Example

```
set_property POST_CRC_ACTION CORRECT_AND_CONTINUE [current_design]
```

CORRECT_AND_HALT

Applied To

Global

Constraint Values

CORRECT_AND_HALT

UCF Example

```
CONFIG POST_CRC_ACTION = CORRECT_AND_HALT;
```

XDC Example

```
set_property POST_CRC_ACTION correct_and_halt [current_design]
```

CONFIG POST_CRC_FREQ

Applied To

Global

Constraint Values

Integer; frequency in MHz

UCF Example

```
CONFIG POST_CRC_FREQ = 50;
```

XDC Example

```
set_property POST_CRC_FREQ 50 [current_design]
```

CONFIG POST_CRC_INIT_FLAG

ENABLE

Applied To

Global

Constraint Values

ENABLE

UCF Example

```
CONFIG POST_CRC_INIT_FLAG = ENABLE;
```

XDC Example

```
set_property POST_CRC_INIT_FLAG ENABLE [current_design]
```

DISABLE

Applied To

Global

Constraint Values

DISABLE

UCF Example

```
CONFIG POST_CRC_INIT_FLAG = DISABLE;
```

XDC Example

```
set_property POST_CRC_INIT_FLAG DISABLE [current_design]
```

CONFIG POST_CRC_SOURCE

FIRST_READBACK

Applied To

Global

Constraint Values

FIRST_READBACK

UCF Example

```
CONFIG POST_CRC_SOURCE = FIRST_READBACK;
```

XDC Example

```
set_property POST_CRC_SOURCE FIRST_READBACK [current_design]
```

PRE_COMPUTED

Applied To

Global

Constraint Values

PRE_COMPUTED

UCF Example

```
CONFIG POST_CRC_SOURCE = PRE_COMPUTED;
```

XDC Example

```
set_property POST_CRC_SOURCE PRE_COMPUTED [current_design]
```

CONFIG VCCOSENSEMODE*n*

Applied To

I/O bank

Constraint Values

OFF, ALWAYSACTIVE, FREEZE

UCF Example

```
CONFIG VCCOSENSEMODE15 = ALWAYSACTIVE;
```

XDC Example

```
set_property VCCOSENSEMODE ALWAYSACTIVE [get_iobanks 15]
```

CONFIG VREF

Applied To

Global

Constraint Values

Pin site

UCF Example

```
CONFIG VREF = E11, F11;
```

XDC Example

```
set_property VREF {E11 F11} [current_design]
```

DEFAULT FLOAT

Applied To

Global

Constraint Values

Boolean

UCF Example

```
DEFAULT FLOAT = TRUE;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

Note: DEFAULT is not supported. I/O ports must be individually configured.

DEFAULT KEEPER

Applied To

Global

Constraint Values

Boolean

UCF Example

```
DEFAULT KEEPER = TRUE;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

Note: DEFAULT is not supported. I/O ports must be individually configured.

DEFAULT PULLDOWN

Applied To

Global

Constraint Values

Boolean

UCF Example

```
DEFAULT PULLDOWN = TRUE;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

Note: DEFAULT is not supported. I/O ports must be individually configured.

DEFAULT PULLUP

Applied To

Global

Constraint Values

Boolean

UCF Example

```
DEFAULT PULLUP = TRUE;
```

XDC Example

The Vivado Design Suite does not support this constraint in XDC.

Note: DEFAULT is not supported. I/O ports must be individually configured.

Migrating Designs with CORE Generator IP to Vivado Design Suite

Overview

Vivado™ Design Suite enables you to migrate designs using CORE Generator™ IP. You can reuse IP in the Vivado Design Suite from the following sources:

- ISE Design Suite project using CORE Generator IP
- PlanAhead tool project using CORE Generator IP
- IP from a CORE Generator project

IMPORTANT: *Before migrating your design to Vivado Design Suite, make sure that your design uses the latest version of IP available in the CORE Generator IP catalog.*

Migrating CORE Generator IP to the Vivado Design Suite

CORE Generator IP can be migrated to Vivado Design Suite in two steps:

1. Migrating a design using CORE Generator IP
2. Migrating IP to the latest version

Step 1: Migrating Design Using CORE Generator IP Sources

You can migrate a project with IP to Vivado Design Suite. To do so, you can do one of the following:

- Import an ISE Design Suite project into Vivado Design Suite project (see Importing a Project Navigator Project)
- Convert PlanAhead tool project to Vivado Design Suite project (see Converting a PlanAhead Tool Project)
- Add the IP core source file from a CORE Generator project to a Vivado Design Suite project

Step 2: Migrating IP to Latest Version

You should also use the latest version of IP in your design. To migrate IP, you can either update your current IP, or recustomize IP.

IMPORTANT: *For IP that is no longer available in the IP catalog, you can continue to reuse existing IP netlists and sources, such as NGC netlist, simulation files, with Vivado synthesis and implementation flows.*

Updating IP

To update existing IP:

1. In the Sources window, click the **IP Sources** tab.
2. Right-click on an IP core source.
3. Select **Upgrade IP** from the popup menu.

Recustomizing IP

To manually recustomize IP to match existing customization parameters:

1. In the Sources window, click the **IP Sources** tab.
2. Right-click on an IP core source.
3. Select **Re-customize IP** from the popup menu.

Update the options in the dialog box as necessary.

Migrating ISE Simulator Tcl to Vivado Simulator Tcl

Tcl Command Migration

The following table lists the ISE Simulator (ISim) Tcl commands and the equivalent Tcl commands in the Vivado™ simulator.

Table 5-1: **ISE Simulator (ISim) Tcl to Vivado Tcl Mappings**

ISim Tcl	Vivado Design Suite Tcl
bp add <file_name> <line_number>	add_bp file_name line_number
bp clear	remove_bps
bp del <index> [<index>…]	remove_bp indexlist…
bp list	report_bps
bp remove <file_name> <line_number>	remove_bps [get_bps -filter {file_name==<file_name> && line_number == <line_number>}]
describe <name>	describe name
dump	report_values
dump -p <process_scope_name>	report_values process_scope_name/*
isim condition add <condition_expression> <command> [-label <label_name>]	add_condition [-label name] <condition_expression> <command>
isim condition remove [<label_names>…] [<indexlist>…] [-all]	remove_conditions [names_indices_objects…]
isim condition list	report_conditions
isim force add <object_name> <value> [-radix <radix>] [-time <time_offset>] { [-value <value> [-radix <radix>] -time <time_offset>] } <[-cancel <time_offset>] [-repeat <time_offset>]	add_force [-radix radix] [-cancel_after <time_offset>] [-repeat_every <time_duration>] <object_name> {<value> [<time>] } [{ <value> <time>}…]
isim force remove	remove_force

Table 5-1: **ISE Simulator (ISim) Tcl to Vivado Tcl Mappings *(Cont'd)***

ISim Tcl	Vivado Design Suite Tcl
`isim get <property> Properties: arraydisplaylength, radix, userunit, maxtraceablesize, ltrace, ptrace`	`get_property property_name [current_sim] Properties: array_display_limit, radix, time_unit, trace_limit, line_tracing, process_tracing`
`isim set <property> <value> properties: arraydisplaylength, radix, userunit, maxtraceablesize, ltrace, ptrace`	`set_property property_name property_value [current_sim] Properties: array_display_limit, radix, time_unit, trace_limit, line_tracing, process_tracing`
`onerror {tcl_commands}`	`onerror {tcl_commands}`
`put [-radix <radix>] name value`	`set_value [-radix radix] Design_object value`
`quit [-f|-force] [-s|-sim]`	`quit [-f|-force]`
`restart`	`restart`
`resume`	`resume`
`run [all | continue | <time> <unit>]`	`run [-all] [time unit]`
`saif open [-scope <path_name>] [-file <file_name>] [-level <nesting_level>] [-allnets]`	`open_saif file_name; log_saif hdl_objects`
`saif_close`	`close_saif [SaifObj]`
`scope [<path>]`	`current_scope hdl_scope`
`sdfanno`	SDF annotation an option for the simulation xelab (elaborator) command. `sdfanno` is no longer supported.
`show time`	`current_time`
`show port`	`report_objects [get_objects * -filter {type == port}]`
`show scope`	`report_scope`
`show signal`	`report_objects [get_objects * -filter {type == signal}]`
`show variable`	`report_objects [get_objects * -filter {type == variable}]`
`show constant`	`report_objects [get_objects * -filter {type == constant}]`
`show child [-r]`	`report_scopes [get scopes -r *]`
`show driver <hdl_object_name>`	`report_drivers hdl_object` (not supported)
`show load <hdl_object_name>`	`report_readers hdl_object` (not supported)
`show value [-radix <radix>] <hdl_object_name>`	`report_value [-radix radix] hdl_object`
`step`	`step [-over]`
`test [-radix radix] <hdl_object_name> <test_value>`	No longer supported. Use Tcl built-in command as follows: `expr {[get_value -radix radix hdl_object] == test_value}`

Table 5-1: **ISE Simulator (ISim) Tcl to Vivado Tcl Mappings *(Cont'd)***

ISim Tcl	Vivado Design Suite Tcl
`vcd dumpfile <file_name>`	`open_vcd file_name`
`vcd dumpvars -m <hdl_scope_name> [-l <level>]`	`log_vcd hdl_objects`
`vcd dumplimit <size>`	`limit_vcd [VCDObject] filesize`
`vcd dumpon`	`start_vcd [VCDObject]`
`vcd dumpoff`	`stop_vcd [VCDObject]`
`vcd dumpflush`	`flush_vcd [VCDObject]`
`wave log [-r] name`	`log_wave hdl_objects`

Migrating Additional Command Line Tools to Vivado

Introduction

This chapter describes how to migrate the use of various Xilinx® command line tools for use in the Vivado™ Integrated Design Suite environment.

Migrating the ISE Partgen Command Line Tool

The ISE® Design Suite Partgen tool obtains:

- Information on all the devices installed on the system
- Detailed package information

You can retrieve the same type of information in the Vivado Design Suite using Tools Command Language (Tcl) commands. Table 6-1 lists the Vivado Tcl commands that retrieve the equivalent information that is stored in the Partgen Partlist file (`.xct`).

Partlist File Contents

Table 6-1: **Tcl Command to Partgen Partlist Content Mapping**

Partlist Content	Tcl Command
Device	`get_parts`
Package	`get_property PACKAGE [get_parts <part_name>]`
Speedgrade	`get_property SPEED [get_parts <part_name>]`
NBIOBS	`llength [get_sites -filter {IS_BONDED==1 && SITE_TYPE =~ IOB*}`
SLICES_PER_CLB	`[llength [get_sites -of_objects [lindex [get_tiles CLBLM_L_*] 0] -filter {NAME=~SLICE*}]]`
NUM_BLK_RAMS	`llength [get_sites RAMB36*]`

Table 6-1: **Tcl Command to Partgen Partlist Content Mapping *(Cont'd)***

Partlist Content	Tcl Command

NUM_BLK_RAM_COLS	`set looplimit[llength [get_sites RAMB36*]];`
	`for {set i 0} {$i <= $looplimit} {incr i} {`
	`set BLK_PER_COL [llength [get_sites RAMB36_X${i}Y*]]`
	`if {$BLK_PER_COL > 0} {`
	`puts "Number of BlockRAM per Column for RAMB36_X${i}, $BLK_PER_COL"}}`
	`for {set x 0} {$x <= $looplimit} {incr x} {`
	`set BLK_COLS [llength [get_sites RAMB36_X*Y$x]]`
	`if {$BLK_COLS > 0 } {`
	`puts "Number of BlockRAM Columns for RAMB36_Y$x, $BLK_COLS"}}`

FF_PER_SLICE	`[llength [get_bels -of [get_sites SLICE_X0Y0] -filter {NAME=~*FF*}]]`
NUM_MMCM	`llength [get_sites MMCM*]`
NUM_LUTS_PER_SLICE	`llength [get_bels -of [get_sites SLICE_X0Y0] -filter {TYPE=~LUT_OR_MEM*}]`

LUT_NAME ENUMERATION and LUT_SIZE_ENUMERATION	`foreach bel [get_bels -of [get_sites SLICE_X0Y0] -filter "TYPE=~LUT_OR_MEM*"] {`
	`set name [split $bel /]`
	`set type [get_property TYPE $bel]`
	`set fields [split $type "M"]`
	`lappend newlist "LUT_NAME=[lindex $name 1] and LUT_SIZE=[lindex $fields 2]"}`
	`foreach line $newlist {puts "$line"}`

NUM_GLOBAL_BUFFERS	`llength [get_sites BUFGCTRL*]`
GLOBAL_BUFFERS ENUMERATION	`get_sites BUFGCTRL`
GLOBAL_BUFFER IOBS ENUMERATION	`[get_sites -of [get_package_pins -filter {IS_CLK_CAPABLE==1 && IS_MASTER==1}]]`
NUM_BUFIO_BUFFERS	`llength [get_sites BUFIO*]`
BUFIO_BUFFERS ENUMERATION	`get_sites BUFIO`
NUM_DSP	`llength [get_sites DSP*]`
NUM_PCIE	`llength [get_sites PCIE*]`
NUM_PLL	`llength [get_sites PLL*]`
NUM_CLB	`llength [get_tiles CLB*]`
CLKRGN ENUMERATION	`get_clock_regions`
NUM_OF_SLR	`llength [get_slrs]`
NUM_DSP_COLUMNS	`llength [get_sites DSP48_X*Y1]`
NUM_DSP_PER_COLUMN	`llength [get_sites DSP48_X1Y*]`

Table 6-1: **Tcl Command to Partgen Partlist Content Mapping *(Cont'd)***

Partlist Content	Tcl Command

NUM_BRAM_PER_COLUMN	set looplimit [llength [get_sites RAMB36*]]
	for {set i 0} {$i <= $looplimit} {incr i} {
	set BLK_PER_COL [llength [get_sites RAMB36_X${i}Y*]]
	if {$BLK_PER_COL > 0} {
	puts "Number of BlockRAM per Column for RAMB36_X${i}, $BLK_PER_COL"}}
	for {set x 0} {$x <= $looplimit} {incr x} {
	set BLK_COLS [llength [get_sites RAMB36_X*Y$x]]
	if {$BLK_COLS > 0 } {
	puts "Number of BlockRAM Columns for RAMB36_Y$x, $BLK_COLS"}}

HEIGHT_OF_DSP	foreach region [get_clock_regions] { puts "Height of DSP48 in $region, [llength [get_sites -filter "CLOCK_REGION==$region" DSP48*]]" }
SLR ENUMERATION	get_slrs

Package Information

Table 6-2 lists the Partgen Package file content to Vivado Tcl commands.

Table 6-2: **Tcl Command to Partgen Package File Content Mapping**

Package File	Tcl Command
Pin Type	*foreach pin [get_package_pins] {puts "Pin Type = [get_property CLASS* [get_package_pins $pin|get_package_pins $pin]_]"}_
Pin Name	*foreach pin [get_package_pins] {puts "Pin Name = $pin"}*
Pin Function	*foreach pin [get_package_pins] {puts "Pin Function = [get_property PIN_FUNC* [get_package_pins $pin|get_package_pins $pin]_]"}_
PAD Name	*foreach pin [get_package_pins] {puts "PAD Name = [get_property NAME* [get_sites $pin|get_sites $pin]_]"}_
Bank Number of Pin	*foreach pin [get_package_pins] {puts "Bank Number = [get_property BANK* [get_package_pins $pin|get_package_pins $pin]_]"}_
Differential Pair	*foreach pin [get_package_pins] {puts "DIff Pair = [get_property DIFF_PAIR_PIN* [get_package_pins $pin|get_package_pins $pin]_]"}_
IO Bank Type	*foreach pin [get_package_pins] {puts "Bank Type = [get_property BANK_TYPE [get_iobanks [get_property BANK* [get_package_pins $pin|get_package_pins $pin]_]]]"}_

ISE Bitgen Command Line Tool

The ISE Design Suite `Bitgen` tool generates bitstreams. In Vivado, you can use the `write_bitstream` Tcl command. For more information, see:

- *Vivado Design Suite: Tcl Command Reference Guide (UG835)* [Ref 5]
- *Vivado Design Suite: Programming and Debugging User Guide (UG908)* [Ref 6]

Note: The Bitgen command options are Tcl properties in the Vivado Design Suite. See "Appendix A Device Configuration Bitstream Settings" in the *Vivado Design Suite: Programming and Debugging User Guide (UG908)* for details on the new properties and values.

ISE Speedprint Command Line Tool

The ISE Design Suite `Speedprint` tool generates speed data for all device components.

The Vivado `get_delays` Tcl command provides speed data. The *Vivado Design Suite: Tcl Command Reference Guide (UG835)* [Ref 5] describes the command details.

ISE PROMGen Command Line Tool

The ISE `PROMGen` tool creates PROM files for programming.

IMPORTANT: *The Vivado Design Suite does not support this capability. Use the ISE Design Suite PROMGen tool to create PROM files.*

ISE BSDLAnno Command Line Tool

The ISE BSDLAnno tool creates post-configuration Boundary Scan Description Language (BSDL) files.

IMPORTANT: *The Vivado Design Suite does not support this capability. Use the ISE Design Suite BSDLAnno tool to create BSDL files.*

XILINX

Appendix A

Additional Resources

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx Support website at:

www.xilinx.com/support.

For a glossary of technical terms used in Xilinx documentation, see:

www.xilinx.com/company/terms.htm.

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

References

The following documents are cited within this guide:

- *Vivado Design Suite 2012.3 Documentation* (www.xilinx.com/support/documentation/dt_vivado_vivado2012-3.htm)

1. *Vivado Design Suite User Guide: Design Flows Overview (UG892)*
2. *Vivado Design Suite User Guide: System Design Entry (UG895)*
3. *Vivado Design Suite User Guide: Using Constraints (UG903)*
4. *Constraints Guide (UG625)* (www.xilinx.com/support/documentation/sw_manuals/xilinx14_3/cgd.pdf)
5. *Vivado Design Suite Tcl Command Reference Guide (UG835)*
6. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)