Chapter 4 : Synthesis

- Basic Concepts
- Partitioning for Synthesis
- Constraining Designs
- Optimizing Designs

Available Tools from Synopsys

- Library Compiler
- RTL Synthesis
  - Design Compiler, Power Compiler, PrimeTime, PrimeRail, NanoSim
- Design Implementation
  - DFT Compiler, DFT MAX, BSD Compiler, TetraMAX ATPG
- Physical Implementation
  - IC Compiler, JupiterXT
- For more information: http://www.synopsys.com

Need Help?

- Unix command – “sold”
- From Design Vision : Help→On-Line Documentation

File Edit Document View Window Help
100%
SOLD Y-2006.03, Volumes 1, 2, 3
SYNOPSYS
Copyright Notices
Bookmarks
Thumbnails
1 of 5
792 x 522 pt

Using SOLD
Other Sources of Information
Licensing Synopsys Software
Installing Synopsys Software

AMPS®
Astro™
Astro™ Interactive Ultra
Automated Chip Synthesis
Behavioral Compiler™
BSD Compiler
Columbia™
CoreTest
Design Analyzer™
Design Budgeting
Design Compiler®
Design Compiler® FPGA
Design Vision™
DesignWare® Library

DesignWare® Memory Models
DFT Compiler
Encore™ IC Package Designer
ESP
External Interfaces
Floorplan Compiler
Floorplan Manager™
Formality®
JupiterIO
JupiterXT™
Leda®
Library Compiler™
Magellan™
Milkyway™
Module Compiler™
PathMill® and PathMill® Plus
Physical Compiler®
Power Compiler™

PrimePower
PrimeRail
PrimeTime® and PrimeTime® SI
SmartModels and FlexModels
SoCBIST
Star-RCXT™
SystemC™ Compiler
TetraMAX®
VCS®
VCS® MX
Vera®
(V)VHDL Compiler™

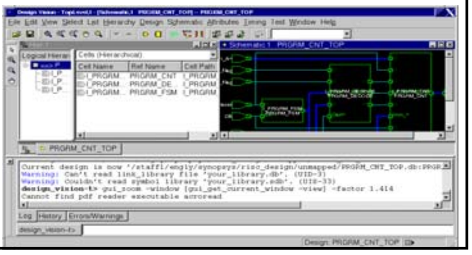
Man Pages and Error Messages

Basic Concepts

DC Interface
Technology Libraries
DC Setup
Design Objects

Design Compiler Interfaces

- Two ways to interface to DC
  - GUI interface: design_vision –xg
  - DC Shell: dc_shell-xg-t (Tcl)


Design Vision - TopLevel.1 - [Schematic.1 PRGRM_CNT_TOP] - PRGRM_CNT_TOP
File Edit View Select List Hierarchy Design Schematic Attributes Timing Test Window Help
Logical Hierar
Cells (Hierarchical)
Cell Name
Ref Name
Cell Path
Schematic.1 PRGRM_CNT_TOP
PRGRM_CNT_TOP
Log
History
Errors/Warnings
design_vision-t>
Design: PRGRM_CNT_TOP

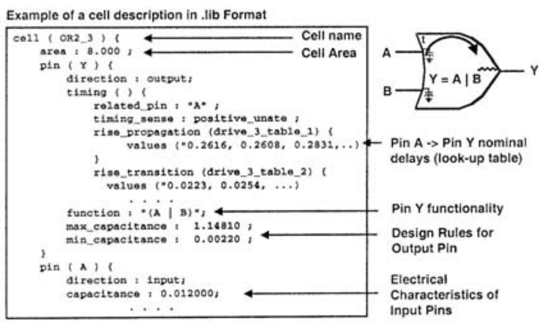
```
Current design is now '/staff1/engly/synopsys/risc_design/unmapped/PRGRM_CNT_TOP.db:PRGR
Warning: Can't read link_library file 'your_library.db'. (UID-3)
Warning: Couldn't read symbol library 'your_library.sdb'. (UID-33)
design_vision-t> gui_zoom -window [gui_get_current_window -view] -factor 1.414
Cannot find pdf reader executable acroread
```

Cell Name	Ref Name	Cell Path
I_PRGRM...	PRGRM_CNT	I_PRGRM
I_PRGRM...	PRGRM_DE...	I_PRGRM
I_PRGRM...	PRGRM_FSM	I_PRGRM

Technology Libraries

- When DC maps a circuit, how will it know which cell library you are using? How will it know the timing of your cells?
- The ASIC vendor must provide a DC-compatible technology library for synthesis.
- Synopsys technology library is a text file(.lib) which is compiled using Library Compiler to generate a binary format with “.db” extension. It includes:
  - Library group – name of the library
  - Library level attributes – contains library features that applies to entire library
  - Environment description– models the variations of temperature, voltage and manufacturing processes, wire-load models.
  - Cell description

An Example of Technology Library


Example of a cell description in .lib Format

```
cell ( OR2_3 ) {
    area : 8.000 ;
    pin ( Y ) {
        direction : output;
        timing ( ) {
            related_pin : "A" ;
            timing_sense : positive_unate ;
            rise_propagation (drive_3_table_1) {
                values ("0.2616, 0.2608, 0.2831,..)
            }
            rise_transition (drive_3_table_2) {
             values ("0.0223, 0.0254, ...)
                  . . . .
        function : "(A | B)";
        max_capacitance :  1.14810 ;
        min_capacitance :  0.00220 ;
    }
    pin ( A ) {
        direction : input;
        capacitance : 0.012000;
                  . . . .
```

Cell name
Cell Area
A
B
Y
Y = A | B
Pin A -> Pin Y nominal delays (look-up table)
Pin Y functionality
Design Rules for Output Pin
Electrical Characteristics of Input Pins

Target Library Variable

- The Target Library is the library used by Design Compiler for building a circuit
- During mapping, DC will
  - Choose functionally-correct gates from this library
  - Calculate the timing of the circuit using vendors-supplied timing data for these gates
- Target_library is a reserved variable in DC

  `set target_library my_tech.db`

  -point to library file provided by ASIC vendor

Link Library Variable

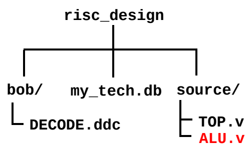
- Used to resolve design references

  `set link_library "* my_tech.db"`

  DC Memory   Target Library

- First DC searches the memory and then the library files specified in the link_library variable
- Second DC searches the all paths defined in the search_path variable

Example

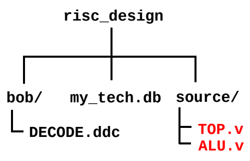


ALU.v

```
module ALU (A,B,OUT1);
  input A, B;
  output [1:0] OUT1;
  always @(A or B)
  begin . . .
```

```
dc_shell-xg-t> set target_library my_tech.db
dc_shell-xg-t> set link_library "* my_tech.db"
dc_shell-xg-t> read_verilog source/ALU.v
  Loading db file standard.sldb
  Loading db file gtech.db
  Loading db file my_tech.db
  Loading verilog file source/ALU.v
  Current design is ALU
```

How to Resolve Design References that is not set in link_library?

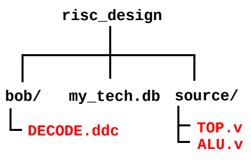


TOP.v

```
module TOP (A,B,OUT1);
  input A, B;
  output [1:0] OUT1;
   ALU U1 (.AIN (A), . .
   DECODE U2 (.A (BUS0), .
```

```
dc_shell-xg-t> set target_library my_tech.db
dc_shell-xg-t> set link_library "* my_tech.db"
dc_shell-xg-t> read_verilog source/ALU.v
dc_shell-xg-t> read_verilog source/TOP.v
dc_shell-xg-t> current_design TOP
dc_shell-xg-t> link
  Unable to resolve reference 'DECODE' in 'TOP'
```

Set the search_path Variable



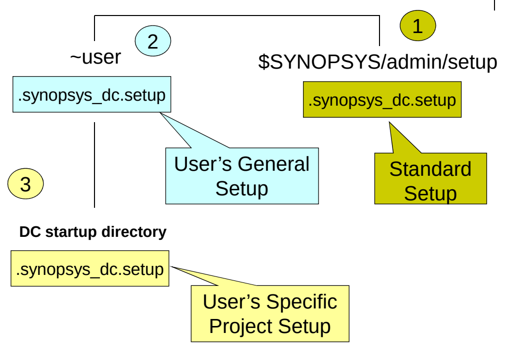
TOP.v

```
module TOP (A,B,OUT1);
  input A, B;
  output [1:0] OUT1;
   ALU U1 (.AIN (A), . .
   DECODE U2 (.A (BUS0), . .
```

```
dc_shell-xg-t> set target_library my_tech.db
dc_shell-xg-t> set link_library "* my_tech.db"
dc_shell-xg-t> lappend search_path ./bob
dc_shell-xg-t> read_verilog source/ALU.v
dc_shell-xg-t> read_verilog source/TOP.v
dc_shell-xg-t> current_design TOP
dc_shell-xg-t> link
  Loading db file bob/DECODE.ddc
```

link only auto-loads ddc files, not Verilog or VHDL files.

DC Setup



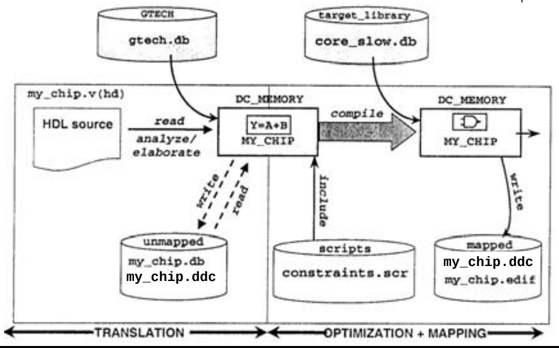
Example: .synopsys_dc.setup

```
# synopsys setup file
set target_library core_slow.db
set link_library "* core_slow.db"

set symbol_library core.sdb
lappend search_path ./unmapped

set sh_enable_line_editing true

alias h history
alias rc "report_constraint -all_violators"
```

Commands in .synopsys_dc.setup are executed upon tool startup.

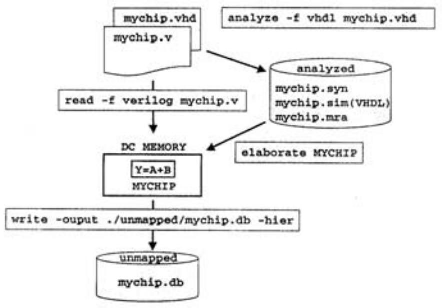
Synthesis Roadmap

Synthesis=Translation+Optimization+Mapping

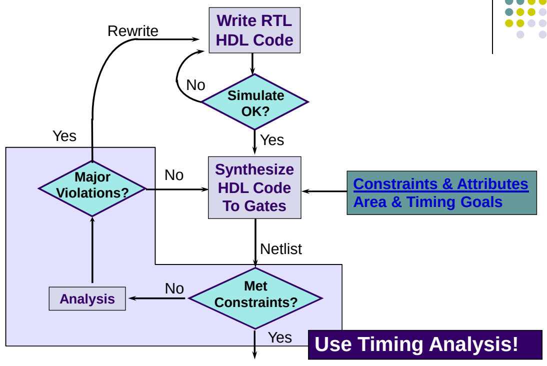


Read the Design into DC

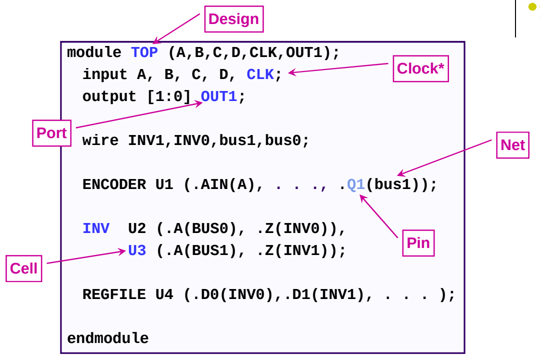
- Analyze, Elaborate - Read



Synthesis Flow

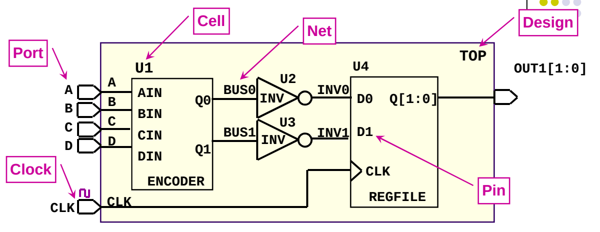


Design Objects: Verilog Perspective



```
module TOP (A,B,C,D,CLK,OUT1);
  input A, B, C, D, CLK;
  output [1:0] OUT1;

  wire INV1,INV0,bus1,bus0;

  ENCODER U1 (.AIN(A), . . ., .Q1(bus1));

  INV  U2 (.A(BUS0), .Z(INV0)),
       U3 (.A(BUS1), .Z(INV1));

  REGFILE U4 (.D0(INV0),.D1(INV1), . . . );

endmodule
```

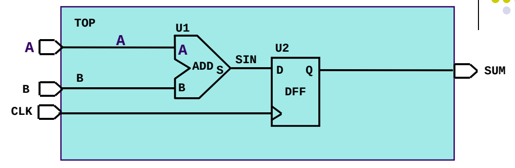
Design Objects: Schematic Perspective



Designs:	{TOP, ENCODER, REGFILE}
Cells:	{U1, U2, U3, U4}

? Why is INV not a design object like ENCODER or REGFILE?

Multiple Objects with the Same Name



```
set_load 5 A
```

? Does "A" refer to a port, net, or pin object?

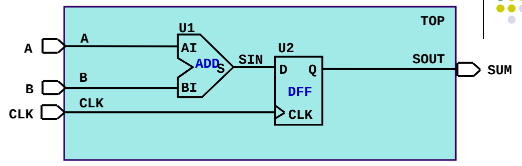
Does it matter onto which object DC places the load?

The "get_*" Command

```
dc_shell-xg-t> set_load 5 [get_nets A]
```

- The "`get_*`" commands return objects in the `current_design`:
  - Can be used stand-alone or composed with other functions
- Objects may be used together with the `*` wildcard:
  ```
  set_load 5 [get_ports addr_bus*]
  set_load 6 [get_ports "A* B*"]
  ```
- "`get_*`" commands return a collection of design objects
  - If no matching objects are found, an empty collection is returned

"get_*" Command Exercise



Write "`get_*`" commands to determine all the:

1. Ports in the design
2. Cells with the letter "U" in their name
2. Nets ending with "CLK"
3. "Q" pins in this design

Other Handy List Commands

- List all input and inout ports of the current design:
  ```
  dc_shell-xg-t> all_inputs
  ```
- List all output and inout ports of the current design:
  ```
  dc_shell-xg-t> all_outputs
  ```
- List all designs in DC memory:
  ```
  dc_shell-xg-t> get_designs *
  ```

Command Summary(1)

`set`	Read and write variables
`echo`	Display a value of a variable
`read_verilog`	Read one or more verilog files
`current_design`	Set the working design in DC
`link`	Resolve design's references
`lappend`	Append list elements onto a variable
`alias`	Create a command which expands to words
`set_load`	Sets load attribute value on specified ports and nets
`get_cells`	Create a collection of cells
`get_clocks`	Create a collection of clocks
`get_designs`	Create a collection of designs
`get_lib`	Create a collection of library
`get_lib_cells`	Create a collection of library cells
`get_lib_pins`	Create a collection of library cell pins
`get_nets`	Create a collection of nets
`get_pins`	Create a collection of pins
`get_ports`	Create a collection of ports

Command Summary(2)

`all_inputs`	Create a collection of input and inout ports
`all_outputs`	Create a collection of output and inout ports
`read_ddc`	Read one or more ddc files
`source`	Apply a Tcl script file
`compile`	Performs logic-level and gate-level synthesis and optimization
`write`	Write a design to a file
`report_constraint`	Display constraint-related information about a design
`remove_design`	Delete designs from DC's memory

Design Partitioning

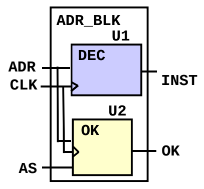
Design Partitioning

- Partitioning is the process of dividing complex designs into smaller parts.
- Ideally, all partitions would be planned prior to writing any HDL.
  - Initial partitions are defined by the HDL.
  - Initial partitions can be modified using Design Compiler.

Partitioning Within the HDL Description

- `module` statements define hierarchical blocks:
  - Instantiation of a module creates a new level of hierarchy
- Inference of Arithmetic Circuits (+, -, *, ..) can create a new level of hierarchy
- Process and Always statements do not create hierarchy

```
module ADR_BLK (...
 DEC  U1(ADR,CLK,INST);
 OK   U2(ADR,CLK,AS,OK);
endmodule
```

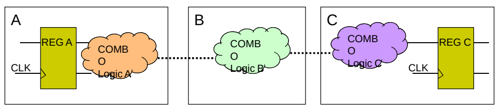


Partitioning for Synthesis

- Keep related combinational logic in the same module.
- Partition for design reuse.
- Separate modules according to their functionality.
- Achieve workable size and complexity.
- Isolate state-machine from other logic.
- Avoid multiple clocks within a block.
- *While partitioning, Think of your layout style.*

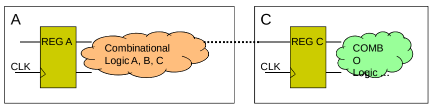
Eliminate Unnecessary Hierarchy

- Design Compiler must preserve port definitions.
- Login optimization does not cross block boundaries.
- An example : path from REG A to REG C may be larger and slower than necessary.



No Hierarchy in Combinational Paths (1)

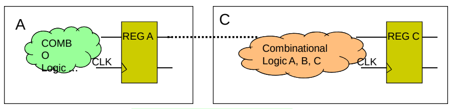
- Related combinational logic is grouped into one block.
  - No hierarchy separates combinational functions A, B, and C.
- Combinational optimization techniques can now be fully exploited.



Better Partitioning

No Hierarchy in Combinational Paths (2)

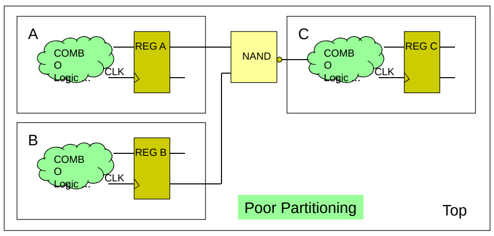
- Related combinational logic is grouped into the same block with the destination register.
  - Combinational optimization techniques can still be fully exploited.
- Sequential optimization may now absorb some of the combinational logic into a more complex Flip-Flop(JK, T, Muxed …)



Best Partitioning

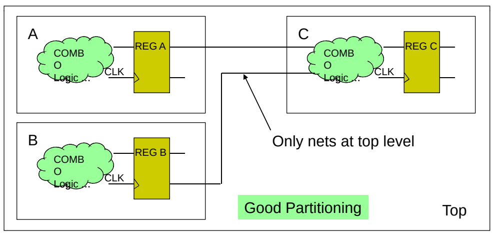
Avoid Glue Logic : Example

- The NAND gate at the top level serves only to "glue" the instantiated cells
  - Optimization is limited because the glue logic cannot be "absorbed".



Remove Glue Logic Between Blocks

- The glue logic can now be optimized with other logic.
- Top-level design is only a structural netlist, doesn't need to be compiled.



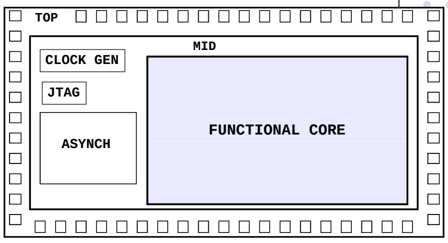
Balance Block Size

- If blocks are too small, the designer may be restricting optimization with artificial boundaries.
- If blocks are too big, compiler run times can be very long.
- For quick turnaround, partition so that each block has 400k – 800k gates.
- Match module size to CPU and memory.

Separate Core Logic, Pads, Clocks, and JTAG

Partition the Top-Level design into at least three levels of hierarchy:

1. Top-level
2. Mid-level
3. Func Core



This partitioning is recommended due to possible:
- **Technology-dependent ("black box") I/O pad cells**
- **Untestable "Divide By" clock generation**
- **Technology-dependent JTAG circuitry**

Partitioning in Design Compiler

- Partitions can be manipulated in two ways:
  - Automatic
    - Synthesis changes partitioning transparently
  - Manual
    - User directs all partitioning changes. "group" and "ungroup" commands provide the designer with the capability of altering the partitions in DC after the design hierarchy has already been defined by the previous written HDL code.
    - "group" creates a new hierarchical block.
    - "ungroup" removes either one or all levels of hierarchy.

Automatic Partitioning

- During synthesis, direct Design Compiler to ungroup small blocks:

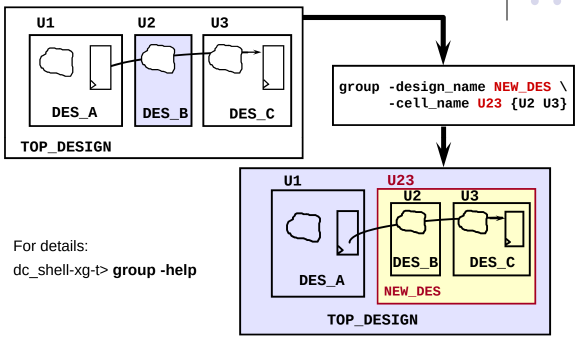
```
compile -auto_ungroup area|delay
```

  - Ungrouping controlled through the variables
    `compile_auto_ungroup_delay_num_cells`
    `compile_auto_ungroup_area_num_cells`
  - Report designs ungrouped during a compile
    `report_auto_ungroup`
- Ungroup the entire hierarchy

```
compile –ungroup_all
```

Manual Partitioning: group

group creates a new hierarchical block

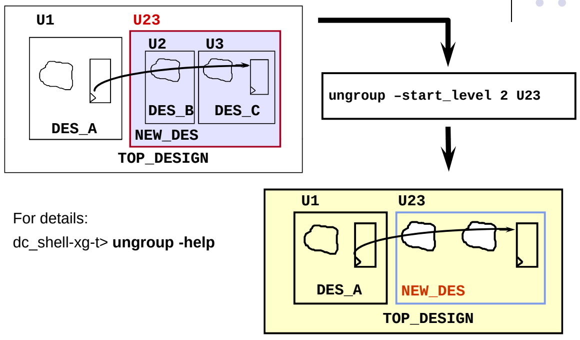


```
group -design_name NEW_DES \
      -cell_name U23 {U2 U3}
```

For details:
dc_shell-xg-t> **group -help**

Manual Partitioning: ungroup

ungroup removes either one or all levels of hierarchy



```
ungroup –start_level 2 U23
```

For details:
dc_shell-xg-t> **ungroup -help**

Partitioning for Synthesis: Summary

What do you gain by "partitioning for synthesis"?

- Better results -- smaller and faster designs
- Easier synthesis process -- simplified constraints and scripts
- Faster compiles -- quicker turnaround

Partitioning Strategies for Synthesis

- Do not separate combinational logic across hierarchical boundaries
- Place hierarchy boundaries at register outputs
- Size blocks for reasonable runtimes
- Separate core logic, pads, clocks, asynchronous logic and JTAG

Design Constraining

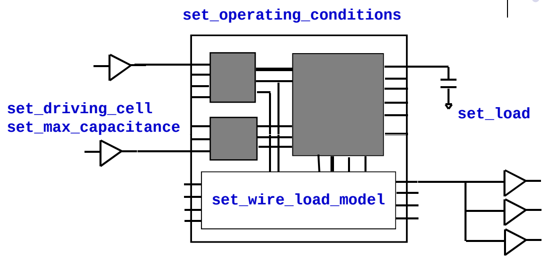
Design environment
Constrain a design for area
Constrain a design for timing
Design rule constrain

Design Constraints

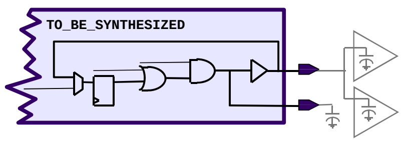
- Design constraints describe the goals for the design. They may consist of environment, timing, area, and design rule constraints. Depending on how the design is constrained, DC tries to meet the set objectives.
- Realistic constraints are expected.

Environment Constrain

Describing Environmental Attributes



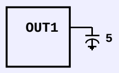
Modeling Capacitive Load



- In order to accurately calculate the timing of an output circuit, DC needs to know the total capacitance driven by the output cells
- `set_load` allows you to specify the external capacitive load on ports (inputs or outputs):
  - By default, DC assumes that the external load on ports is 0
  - You can specify some other constant value
  - The `load_of` command can be used to specify the external load as the pin load of a cell in your technology library

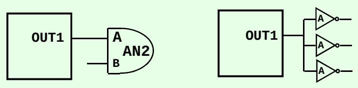
set_load Examples

Use set_load to specify a load *value* on an output port:



```
set_load 5 [get_ports OUT1]
```

Use `load_of` *`lib/cell/pin`* to place the load of a gate from the technology library on the port:

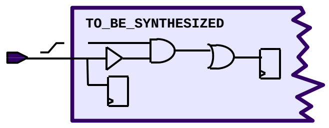


```
set_load [load_of my_lib/AN2/A] [get_ports OUT1]
```

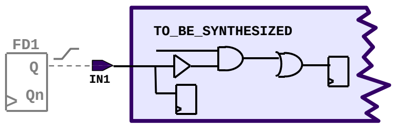
```
set_load [expr [load_of my_lib/inv1a0/A] * 3] \
  [get_ports OUT1]
```

Modeling Input Drive Strength

- In order to accurately calculate the timing of an input circuit, DC needs to know the transition time of the signal arriving at the input port
- `set_driving_cell` allows you to specify a realistic external cell driving the input ports:
  - By default, DC assumes that the external signal has a transition time of 0
  - Placing a driving cell on the input ports causes DC to calculate the actual (non-zero) transition time on the input signal as though the specified library cell was driving it



`set_driving_cell` Examples

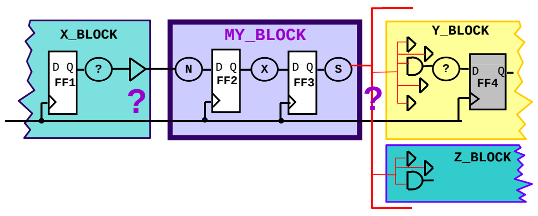


```
set_driving_cell \
                -lib_cell FD1 \
                -pin Q \
                [get_ports IN1]
```

If no pin is given, DC will use first output pin found!

Load Budgeting (1/2)

What if, prior to compiling, the cells driving your inputs, and the loads on your outputs are not known?



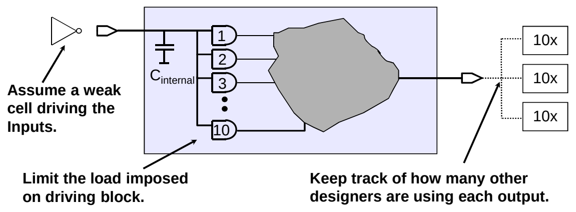
Load Budgeting (2/2)

- Creating a load budget:
  - Assume a weak cell driving the inputs (to be conservative)
  - Limit the input capacitance of each input port
  - Estimate the number of other major blocks your outputs may have to drive

Load Budget Example (1/2)

Example Specification:

1. Inputs of any block shall present no more than the load of 10 **"AND2"** gates to their driving block.
2. Outputs of any blocks will only be allowed to connect to a maximum of 3 other blocks, otherwise, the output port will need to be replicated in code.


Assume a weak cell driving the Inputs.
C internal
1
2
3
10
10x
10x
10x
Limit the load imposed on driving block.
Keep track of how many other designers are using each output.

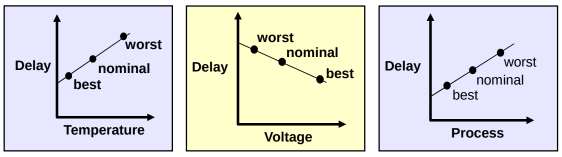
Load Budget Example (2/2)

```
current_design myblock
link
reset_design
source timing_budget.tcl
set all_in_ex_clk [remove_from_collection \
    [all_inputs] [get_ports Clk]]

# Assume a weak driving buffer on the inputs
set_driving_cell -lib_cell inv1a1 $all_in_ex_clk

# Limit the input load
set MAX_INPUT_LOAD [expr \
        [load_of ssc_core_slow/and2a1/A] * 10]
set_max_capacitance $MAX_INPUT_LOAD $all_in_ex_clk

# Model the max possible load on the outputs, assuming
# outputs will only be tied to 3 subsequent blocks
set_load [expr $MAX_INPUT_LOAD * 3] [all_outputs]
```

Script for timing

Operating Conditions

- Library cells are usually characterized using “nominal” voltage and temperature.
- Vendors allow for synthesis of circuits, which will not operate under “nominal” conditions by embedding operating condition models in the technology libraries
- Operating conditions can be placed on your design by using the **`set_operating_conditions`** command:
  - During synthesis “nominal” cell and wire delays will be scaled based on the operating conditions



Specify Operating Condition

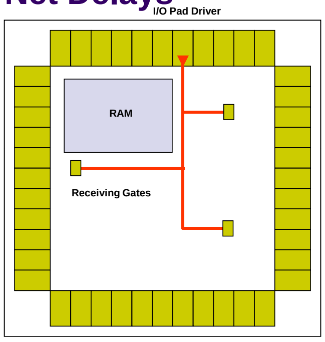
- Usually the library specifies a default operating condition
- Use `report_lib` *`libname`* to list the vendor-supplied operating conditions:

```
Operating Conditions:

    Name            Library        Process    Temp     Volt
    ---------------------------------------------------------
    typ_25_1.80     my_lib          1.00     25.00     1.80
    slow_125_1.62   my_lib          1.05    125.00     1.62
    fast_0_1.98     my_lib          0.93      0.00     1.98
```

- To set operating conditions enter:

```
set_operating_conditions -max "slow_125_1.62"
```

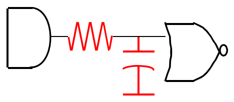
Net Delays



Prior to layout, how can the RC delay of nets be estimated?

What Is a Wire Load Model?

- A wire load model is an estimate of a net’s RC parasitics based on the net’s fanout:
  - Model is created by your vendor
  - Estimates are based on statistics from other designs the vendor has fabricated using this process



Wire Load Model: Standard Format

Example: Standard Format

```
Name          :   160KGATES
Location      :   ssc_core_slow
Resistance    :   0.000271
Capacitance   :   0.00017
Area          :   0
Slope         :   50.3104
Fanout    Length
---------------------------------
      1     31.44
      2     81.75
      3    132.07
      4    182.38
      5    232.68
```

R per unit length

C per unit length

Extrapolation slope

```
Time Unit                : 1ns
Capacitive Load Unit     : 1.000000pf
Pulling Resistance Unit  : 1kilo-ohm
```

Specifying Wire Loads in Design Compiler

- Manual model selection:

```
current_design addtwo
set_wire_load_model -name 160KGATES
```

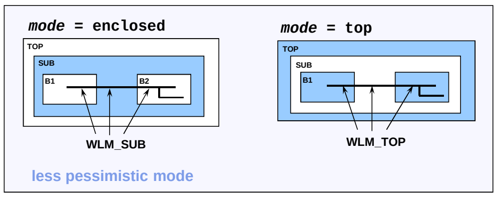
- Automatic model selection (default is TRUE):

```
dc_shell-xg-t> report_lib ssc_core_slow
     Selection                      Wire load name
     min area    max area
   -------------------------------------------
        0.00      43478.00           5KGATES
    43478.00      86956.00          10KGATES
    86956.00     173913.00          20KGATES
   173913.00     347826.00          40KGATES
   347826.00     695652.00          80KGATES
```

To turn off automatic wire load model selection:
`set auto_wire_load_selection false`

Wireload Model Mode

Specifies what wire load model to use for nets that cross hierarchical boundaries.



Example:

```
dc_shell-xg-t> set_wire_load_mode top
```

Summary of Describing Environmental Attributes:

Environmental Attributes:

```
set_driving_cell      set_load
set_wire_load_model
set_operating_conditions
set_wire_load_mode
```

Design Rules:

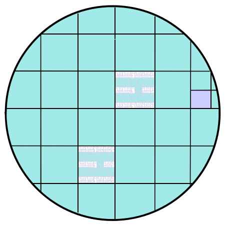
```
set_max_capacitance
```

Area and Time Constrain

Specifying an Area Goal

```
dc_shell-xg-t> current_design PRGRM_CNT_TOP
dc_shell-xg-t> set_max_area 100
```

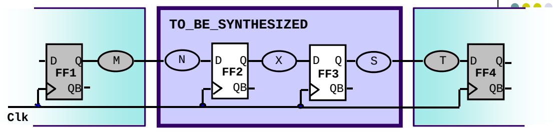
- Units are defined by the target library:
  - 2-input-NAND-gate
  - Transistors
  - Square microns



Timing Goals: Synchronous Designs

- Synchronous Designs:
  - Data arrives from a clocked device
  - Data goes to a clocked device
- Objective:
  - Define the timing constraints for all paths within a design
    - All input logic paths
    - The internal (register to register) paths, and
    - All output paths

Register-to-Register Paths



What information must you provide to constrain all the register-to-register paths in your design?

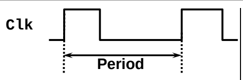
Does the duty cycle of your clock matter?

Example:

Clock Period = 10ns   Setup = 1ns

What is the max delay requirements for the register-to-register paths in the block TO_BE_SYNTHESIZED?

Defining a Clock



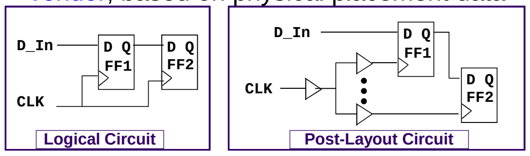
- **You MUST Define:**
  - Clock Source (port or pin), Clock Period

```
create_clock -period <value>  <port list>
example:  create_clock –period 40 [get_ports Clk]
```

- **You *may* also define:** Duty Cycle, Offset/Skew, Clock Name.
- Creating a clock constrains timing paths between registers
- Use report_clock to see defined clocks and their attributes
- By default, DC will not “buffer up” the clock net, even when the flip-flops load to high
  - In other words, DRC checking and optimization is disabled on clock nets

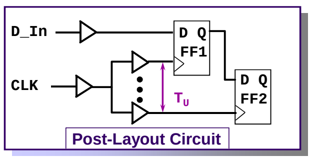
Modeling Clock Trees

- Design Compiler is NOT used for synthesis of the clock tree
- Clock tree synthesis is usually done by the vendor, based on physical placement data



? **What design considerations need to be taken into account by the synthesis tool, prior to layout?**

Modeling Uncertainty on Clock Edges



Uncertainty is the delay difference between the clock network branches (commonly called clock skew).

This may also be used to account for PLL jitter:

```
set_clock_uncertainty –setup TU [get_clocks CLK]
```

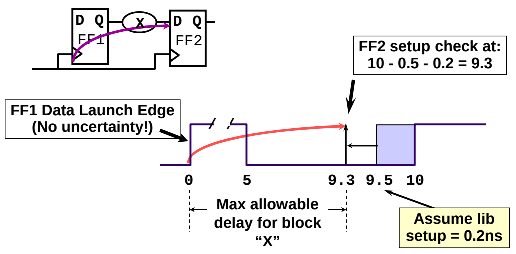
Pre-Layout: clock skew + jitter

placed on clock objects

set_clock_uncertainty and Setup Timing

Example:

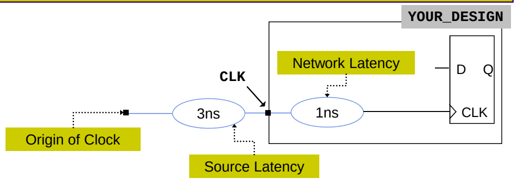
```
create_clock -period 10  [get_ports CLK]
set_clock_uncertainty –setup 0.5 [get_clocks CLK]
```


D Q FF1
X
D Q FF2
FF2 setup check at: 10 - 0.5 - 0.2 = 9.3
FF1 Data Launch Edge (No uncertainty!)
0
5
9.3
9.5
10
Max allowable delay for block “X”
Assume lib setup = 0.2ns

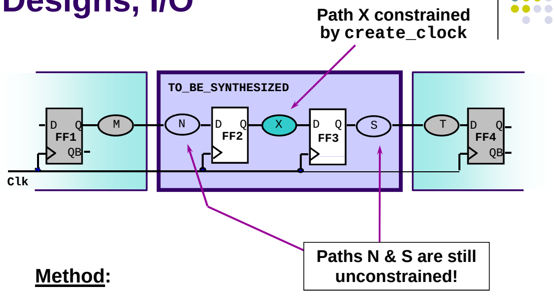
Model Source Latency

- Source latency is the propagation time from the actual clock origin to the clock definition point in the design:
  - Used for either ideal or propagated clocks (post layout)

```
create_clock -period 10 [get_ports CLK]
set_clock_latency -source 3 [get_clocks CLK]
set_clock_latency 1 [get_clocks CLK] ;# pre layout
#set_propagated_clock [get_clocks CLK] ;# post layout
```



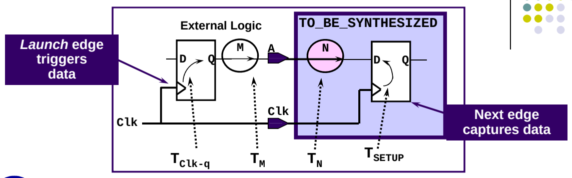
Timing Goals: Synchronous Designs, I/O


Path X constrained by create_clock
TO_BE_SYNTHESIZED
D Q FF1 QB
M
N
D Q FF2
X
D Q FF3
S
T
D Q FF4 QB
Clk
Paths N & S are still unconstrained!

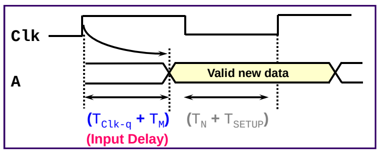
Method:

1. Define the clocks
2. **Define the I/O timing *relative* to the clocks**

Constraining the Input Paths



What information <u>must</u> you provide to constrain the input paths?



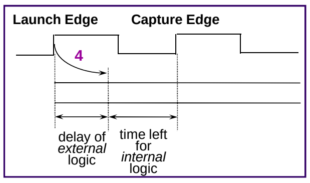
Constraining Input Paths in DC

```
set_input_delay -max 4 -clock Clk [get_ports A]
```

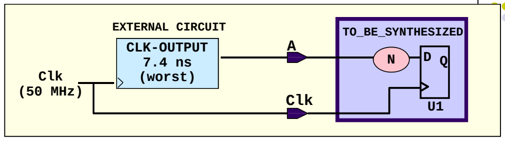
The `set_input_delay` command constrains input paths.

You specify how much time is used by external logic...

DC calculates how much time is left for the internal logic.



`set_input_delay`: Effect on Input Paths

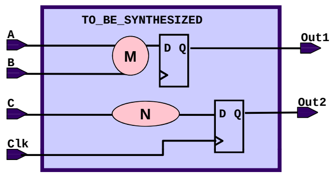


```
create_clock -period 20 [get_ports Clk]
set_input_delay -max 7.4 -clock Clk [get_ports A]
```

If U1 has a 1 ns setup requirement:

What is the maximum delay for T_N? ____________

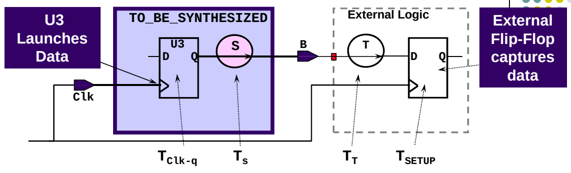
Constraining Example



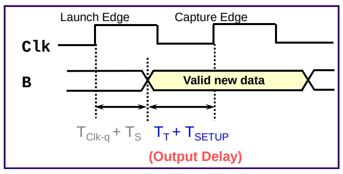
To constrain all inputs except the clock

```
set_input_delay 3.5 –clock Clk –max \
    [remove_from_collection \
    [all_inputs] [get_ports Clk]]
```

Constraining Output Paths of a Design



What information <u>must</u> you provide to constrain the output paths?



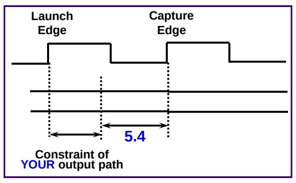
Constraining Output Paths in DC

```
set_output_delay -max 5.4 -clock Clk  [get_ports B]
```

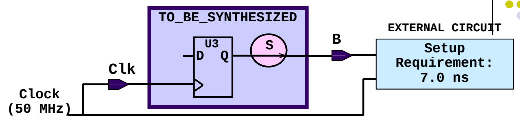
The `set_output_delay` command constrains output paths.

You specify how much time is needed by external logic...

DC calculates how much time is left for internal logic.



set_output_delay: Effect on Output Paths

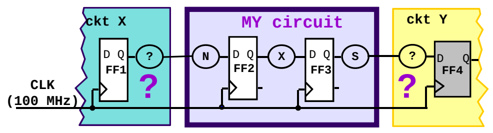


```
create_clock -period 20 [get_ports Clk]
set_output_delay -max 7.0 -clock Clk [get_ports B]
```

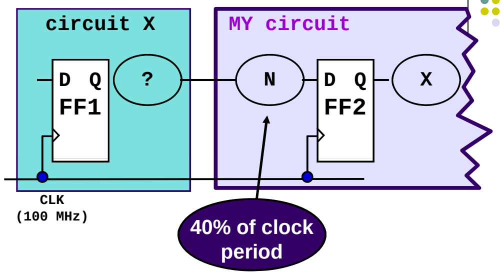
If U3 has T_{CLK-Q} = 1.0ns:
What is the maximum delay for T_S? ________________

Time Budgeting (1/2)

What if you do not know the delays on your inputs or the setup requirements of your outputs?



Time Budgeting (2/2)

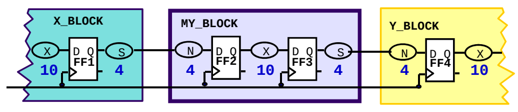


Create a time budget: Better to budget conservatively than to compile with paths unconstrained!

Time Budgeting Example

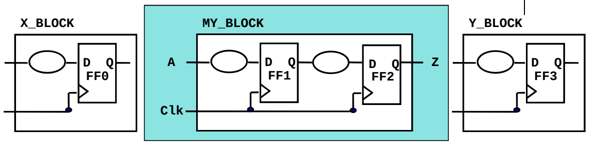
timing_budget.tcl

```
# A generic Time Budgeting script file
# for MY_BLOCK, X_BLOCK and Y_BLOCK
create_clock -period 10 [get_ports CLK]

set_input_delay -max 6 -clock CLK [all_inputs]
remove_input_delay [get_ports CLK]
set_output_delay -max 6 -clock CLK [all_outputs]
```



Would it be easier to specify a time budget if all outputs were registered?

Registered Outputs



```
# Assume every block has registered outputs, 10ns clock:
set all_in_ex_clk [remove_from_collection \
    [all_inputs] [get_ports Clk]]
set_input_delay  -max      $clk_to_q       -clock CLK $all_in_ex_clk
set_output_delay -max [expr 10 - $clk_to_q] -clock CLK [all_outputs]
```

Verify that Constraints are Complete

- After setting constraints, verify that there are no remaining unconstrained paths: **check_timing**
  - Issues warning if unconstrained paths are found

```
dc_shell-xg-t> check_timing
Warning: The following end-points are not constrained for
maximum delay.
End point
--------------
OUT_VALID
PSW[0]
PSW[1]
PSW[2]
...
```

Verify Correctness of Constraints

- Make certain the constraints you applied were applied correctly:

  `report_port –verbose`

  - Reports the constraints set on all ports, compares the numbers to your specification

```
dc_shell-xg-t> report_port –verbose
...
 Output Delay
                     Min              Max       Related  Fanout
Output Port    Rise     Fall     Rise     Fall   Clock     Load
-----------------------------------------------------------------
EndOfInstrn    --       --       3.0      3.0     Clk      0.00
OUT_VALID      --       --       --       --      --       0.00
PSW[0]         --       --       --       --      --       0.00
...
```

Recommended Step in Scripts

Erase all *attributes* and *constraints* from the current design before applying new constraints.

The constraint script should start with:

`reset_design`

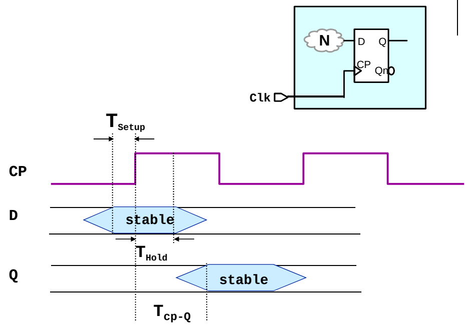
When applying *multiple* constraint scripts, there should only be ONE `reset_design` command.

Command Summary

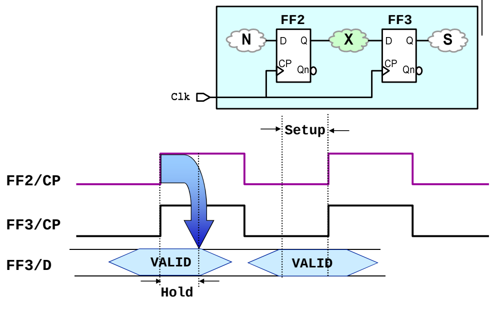
`set_max_area`	Sets the max_area attribute to a specified value on the current design
`create_clock`	Creates a clock and defines its waveform in the current design
`report_clock`	Displays clock-related information on the current design
`set_clock_uncertainty`	Specifies uncertainty (skew) of clock networks
`set_clock_latency`	Specifies clock network and source latency (clock insertion delay)
`set_clock_transition`	Sets clock transition or slope on all clock pins connected to the clock
`set_propagated_clock`	Allows DC to calculate the actual clock latency of a clock tree (post layout)
`set_input_delay`	Specifies the input delay or arrival time at input ports
`set_output_delay`	Specifies the output delay or setup/hold requirement at output ports
`remove_input_delay`	Removes an input delay previously assigned with set_input_delay
`remove_from_collection`	Removes objects from a collection resulting in a new collection. The base collection remains unchanged.
`check_timing`	Warns about possible timing problems in the current design
`report_port`	Displays information on the ports of the current design
`reset_design`	Removes all constraints and attributes from the current design
`list_libs`	Lists the available libraries in memory
`report_lib`	Displays information on technology or symbol libraries
`remove_design`	Removes a list of designs or libraries from DC memory

Design Rule Constrain

Setup and Hold Time (1)



Setup and Hold Time (2)



Checking for Setup and Hold Time

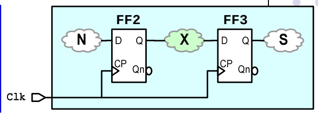
- Checking for setup time
  - dc_shell-xg-t> **report_constraint –all –max_delay**
- Checking for hold time
  - dc_shell-xg-t> **report_constraint –all –min_delay**

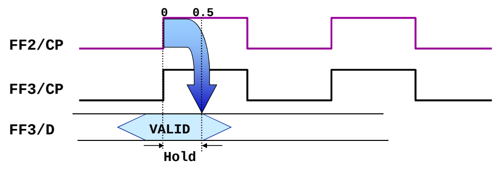
Your design meets timing under setup constraints, but when you check it for hold, you have violations!

Hold Time Violations

? **What is the minimum delay requirement from FF2 to FF3?**

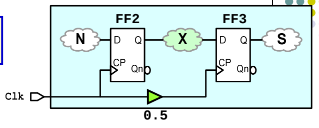
(assume 0.5 ns hold requirement on FF3)

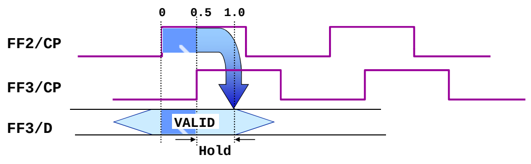




Complicate the Picture

? **What is the minimum delay requirement now?**





- Hold time requirements are affected by:
  - Skew on the clock tree network
  - Operating Conditions
  - FF Hold Time

Fixing Hold Time: Some Considerations

- Should you fix hold violations before layout?
  - Netlist is closer to “final” before going to layout
  - May fix false violations due to inaccuracy of net timing
  - May not fix true violations due to inaccuracy of net timing
- Should you fix hold violations after layout?
  - Fixing only true violations minimizes area impact
  - Could have a large number of ECOs to the layout
- Other considerations:
  - When are you inserting the scan path?
    (Scan paths cause most hold violations)
  - May have to do before and after layout if timing changes a lot

Option: Fix only the BIG violations, which show up under nominal or worst case conditions, before going to layout.

When to Fix Hold Violations

- Small Violations:
  - Fix small hold-time violations post-layout because
    - The clock tree is not even in place until after layout (fixing apparent violations affects speed and area!)
    - They often disappear when net parasitics are annotated
- Large Violations:
  - Fix only the big hold-time violations pre-layout

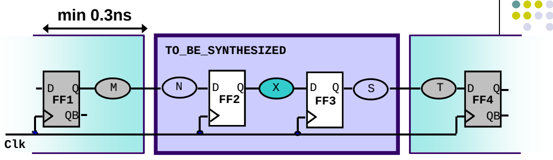
Use Simultaneous Min-Max

- Simultaneous Min-Max Analysis and Optimization:
  - Environment and timing constraints supported for BOTH min and max values
  - Fixes hold time without violating setup time constraints
- What constraints should you specify before analyzing and fixing hold time violations?

```
set_clock_uncertainty -hold
set_input_delay -min
set_output_delay -min
```

- Continue to use your maximum timing library (and operating conditions)

Apply `set_input_delay` for Hold Time

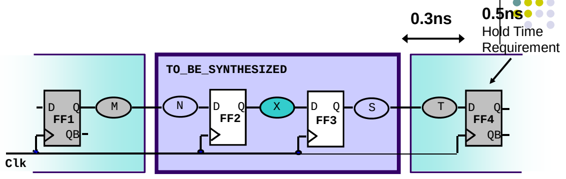


`set_input_delay -min` describes the fastest arrival time of the external logic on the input ports.

```
create_clock -period 10 [get_ports Clk]
set_input_delay -min 0.3 -clock Clk $all_in_ex_clk
```

? **If FF2 has T_{HOLD} = 1ns:**
What is the min delay allowed for N? __________________

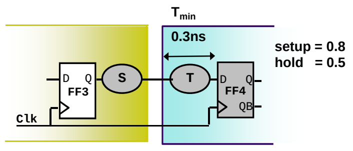
Apply `set_output_delay` for Hold Time



`set_output_delay -min` describes the hold time requirement of the external logic on the output ports.

? **If FF has T_{HOLD} = 0.5ns and T_T = 0.3ns:**
What is the min delay (FF3 + S) requirement? ____________

Calculation of `set_output_delay`



set_output_delay –max (T_{max} + $FF4_{setup}$)
set_output_delay –min (T_{min} - $FF4_{hold}$)

```
create_clock -period 5 [get_ports Clk]
set_output_delay -min [expr 0.3-0.5] -clock Clk   \
     [all_outputs]
```

$\underbrace{0.3-0.5}_{-0.2}$

Reporting Hold Time Violations

```
dc_shell-xg-t>  report_constraint –all –min_delay
 . . .

  min_delay/hold ('Clk' group)
                              Required       Actual
  Endpoint                    Path Delay     Path Delay     Slack
  --------------------------------------------------------------------
  A_reg[2]/D                  0.31           0.22 f         -0.08  (VIOLATED)
  B_reg[0]/D                  0.29           0.22 r         -0.07  (VIOLATED)
```

Fixing Design Rule and Hold Violations

```
set_fix_hold [all_clocks]
compile –incremental -only_design_rule
```

- If you *only* want to fix design rule violations:
  - Do not use **`set_fix_hold`**
- By default, DC does NOT fix hold time violations:
  - Use `set_fix_hold` to tell DC to fix hold time violations
- Use `compile –incr -only_design_rule`:
  - DC only adds buffers or resizes cells
  - DC fixes only design rule violations and may fix hold time violations

Summary: Example Script

```
read_ddc Top_meetsSetup.ddc
source TimingConstraints_max.tcl

set_operating_conditions -max WORST
set ALL_IN_EX_CLOCK [remove_from_collection \
  [all_inputs] [get_ports Clk]]
set_input_delay -min 0.2 -clock Clk $ALL_IN_EX_CLOCK
set_output_delay -min -0.1 -clock Clk [all_outputs]
set_clock_uncertainty -hold 0.5 [get_clocks Clk]

report_timing -delay min

# Fix min timing violations
set_fix_hold [all_clocks]
compile –incremental -only_design_rule

redirect top.rpt {report_constraint -all_violators}
```

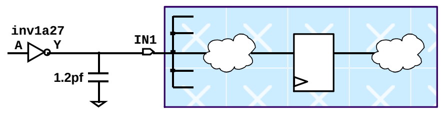
Design Rule Constraints

- Vendors impose design rules that restrict how many cells are connected to one another based on capacitance, transition and fanout
- You may apply more conservative design rules to:
  - Anticipate the interface environment your block will see
  - Prevent the design from operating cells close to their limits, where performance degrades rapidly
- DC respects design rules as highest priority of all in the following order:
  - max_capacitance
  - max_transition
  - max_fanout

set_max_capacitance

my_drc_cons.tcl

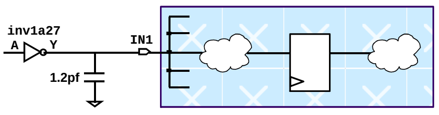
```
# Find the max capacitive load allowed on your expected driver
set DRIVE_PIN TECH_LIB/inv1a27/Y
set MAX_CAP [get_attribute  $DRIVE_PIN  max_capacitance] ;# 3.60
# Add some margin so DC does not fully load the driver
set CONSERVATIVE_MAX_CAP [expr $MAX_CAP / 2.0]  ;# 1.80
set_max_capacitance $CONSERVATIVE_MAX_CAP [get_ports IN1]
set_load 1.2 [get_ports IN1]
# max internal load DC can put on IN1 is [1.8 - 1.2 = 0.6pf]
```



set_max_transition

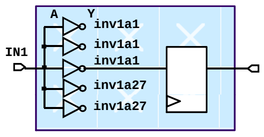
my_drc_cons.tcl

```
# Find the max transition allowed on your expected driver
set DRIVE_PIN TECH_LIB/inv1a27/Y
set MAX_TRANS [get_attribute $DRIVE_PIN max_transition] ;# 0.400
# Add some margin so DC won't fully load the driver
set CONSERVATIVE_MAX_TRANS [expr $MAX_TRANS / 2.0]   ;# 0.200

set_max_transition $CONSERVATIVE_MAX_TRANS [get_ports IN1]
# DC accounts for the driving_cell type and external load on it,
# limits internal loads placed on IN1 to meet your design rule
```



set_max_fanout

```
set_max_fanout 6 [get_ports IN1]
```



Is the max_fanout design rule on port IN1 met?

How many cells might port IN1 have to drive?

Does it matter what the cell type is?

```
get_attribute  TECH_LIB/inv1a1/A  fanout_load
0.25
```

DC might load port IN1 with 6 / 0.25 = 24 inv1a1 cells!

```
get_attribute  TECH_LIB/inv1a27/A  fanout_load
3.00
```

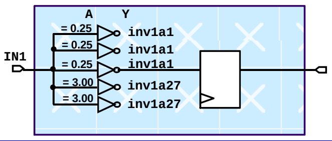
DC can only load port IN1 with 6 / 3.00 = 2 inv1a27 cells!

Fanout Loads

```
set_max_fanout 6 [get_ports IN1]
```

set_max_fanout uses fanout_load, NOT absolute fanout number!

Sum of fanout_load on a port must be less than max_fanout design rule constraint.



Some cell/pins have no `fanout_load` attribute:
- DC checks the library for **`default_fanout_load`** attribute
- If neither exists, DC assumes a value of zero, which is equal to unconstrained

Reporting Design Rule Violations

```
dc_shell-xg-t>  report_constraint –all_violators
...
 max_transition
                          Required     Actual
 Net                     Transition   Transition       Slack
 ---------------------------------------------------------------
 I_PRGRM_CNT/n184           0.50         0.69          -0.19  (VIOLATED)
 I_PRGRM_DECODE/n945        0.50         0.63          -0.13  (VIOLATED)
 Ld_Rtn_Addr                0.50         0.61          -0.11  (VIOLATED)

 max_capacitance
                          Required     Actual
 Net                     Capacitance  Capacitance      Slack
 ---------------------------------------------------------------
 CurrentState[0]            0.20         0.24          -0.04  (VIOLATED)
 PC[0]                      0.20         0.24          -0.04  (VIOLATED)
 CurrentState[1]            0.20         0.24          -0.04  (VIOLATED)
```