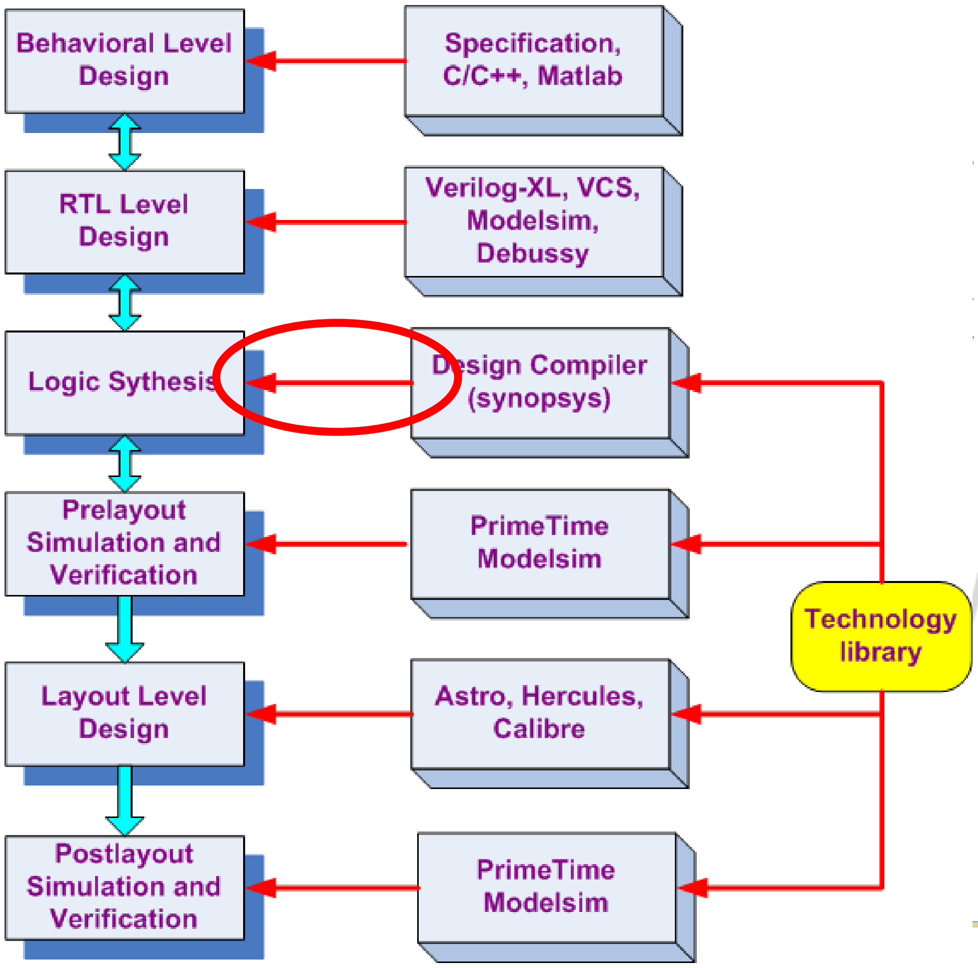
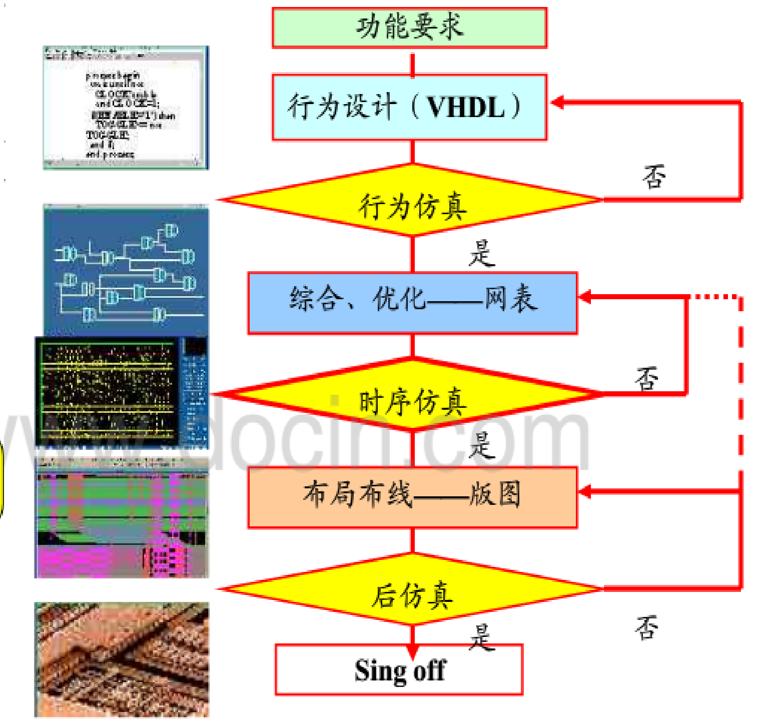
Design Compiler

JoyShockley
School of Microelectronics
Fudan University


Behavioral Level Design
Specification, C/C++, Matlab
RTL Level Design
Verilog-XL, VCS, Modelsim, Debussy
Logic Sythesis
Design Compiler (synopsys)
Prelayout Simulation and Verification
PrimeTime Modelsim
Layout Level Design
Astro, Hercules, Calibre
Postlayout Simulation and Verification
PrimeTime Modelsim
Technology library

设计过程：



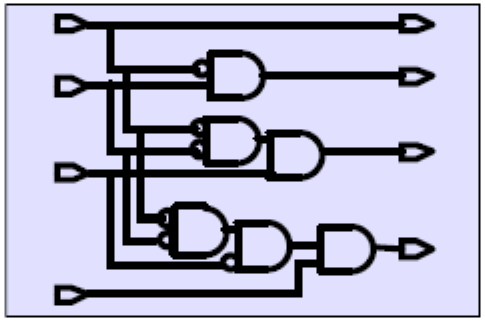
集成电路芯片设计过程框架

Synthesis = Translation + Optimization + Mapping

```
residue = 16'h0000;
if (high_bits == 2'b10)
   residue = state_table[index];
else
   state_table[index] = 16'h0000;
```

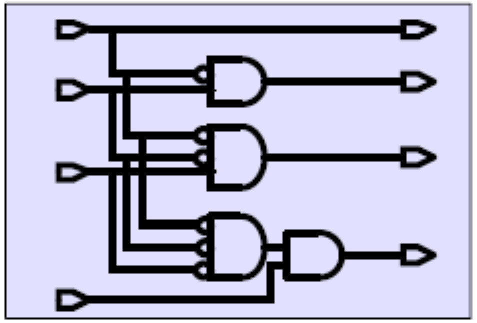
HDL Source

Translate



Generic Boolean (GTECH)

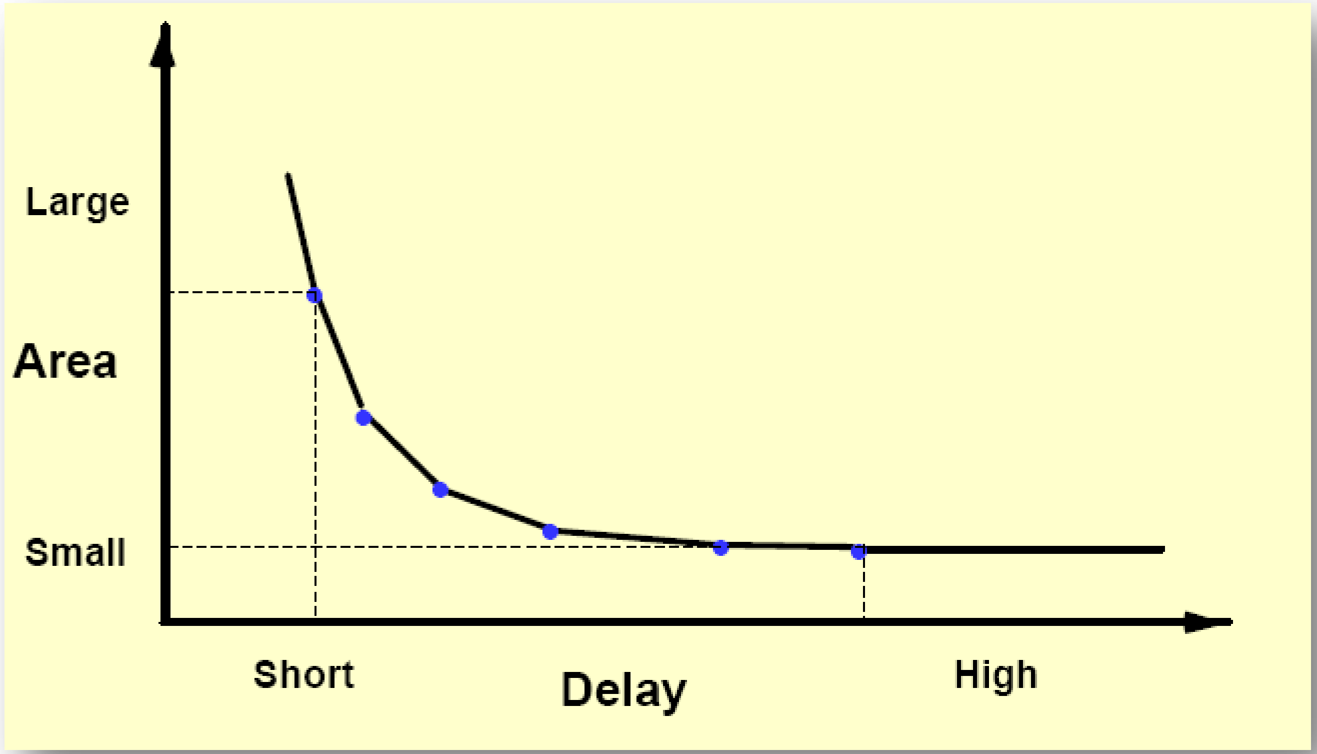
Optimize + Map



Target Technology

Map to lib

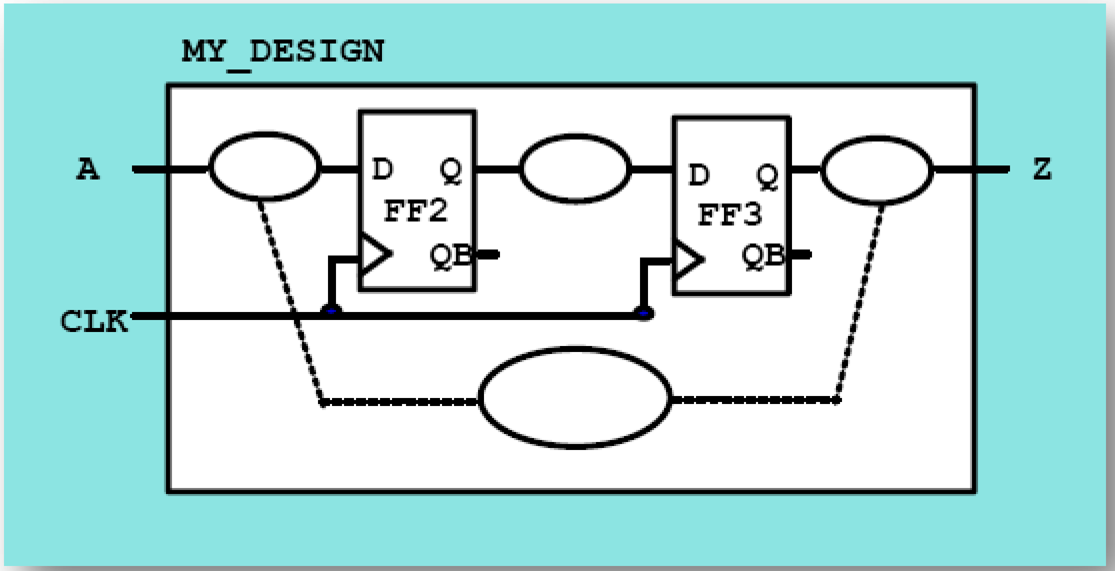
Synthesis Is Constraint-Driven



You set the goals (through constraints)

Design Compiler optimizes the design to meet your goals

Synthesis Is Path-Based



Design Compiler uses Static Timing Analysis (STA) to calculate the timing of the paths in the design.

Agenda

1. Setup, Libraries and Objects
2. Set Design Environment
3. Set Design Constraints
4. Compile Design
5. Static Timing Analysis

Agenda

1. **Setup, Libraries and Objects**
2. Set Design Environment
3. Set Design Constraints
4. Compile Design
5. Static Timing Analysis

Recall the 3 steps involved in synthesis:

- Translation
- Optimization
- Mapping

When DC maps a circuit, how will it know which cell library you are using?
How will it know the timing of your cells?

Your ASIC vendor must provide a DC-compatible *technology library* for synthesis!

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

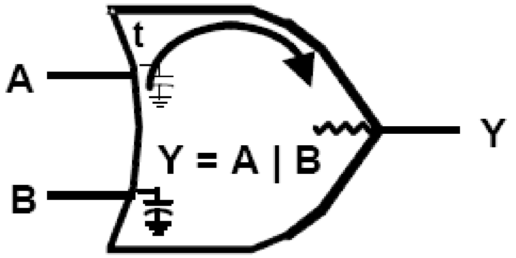
Cell Area

Pin A -> Pin Y nominal delays (look-up table)

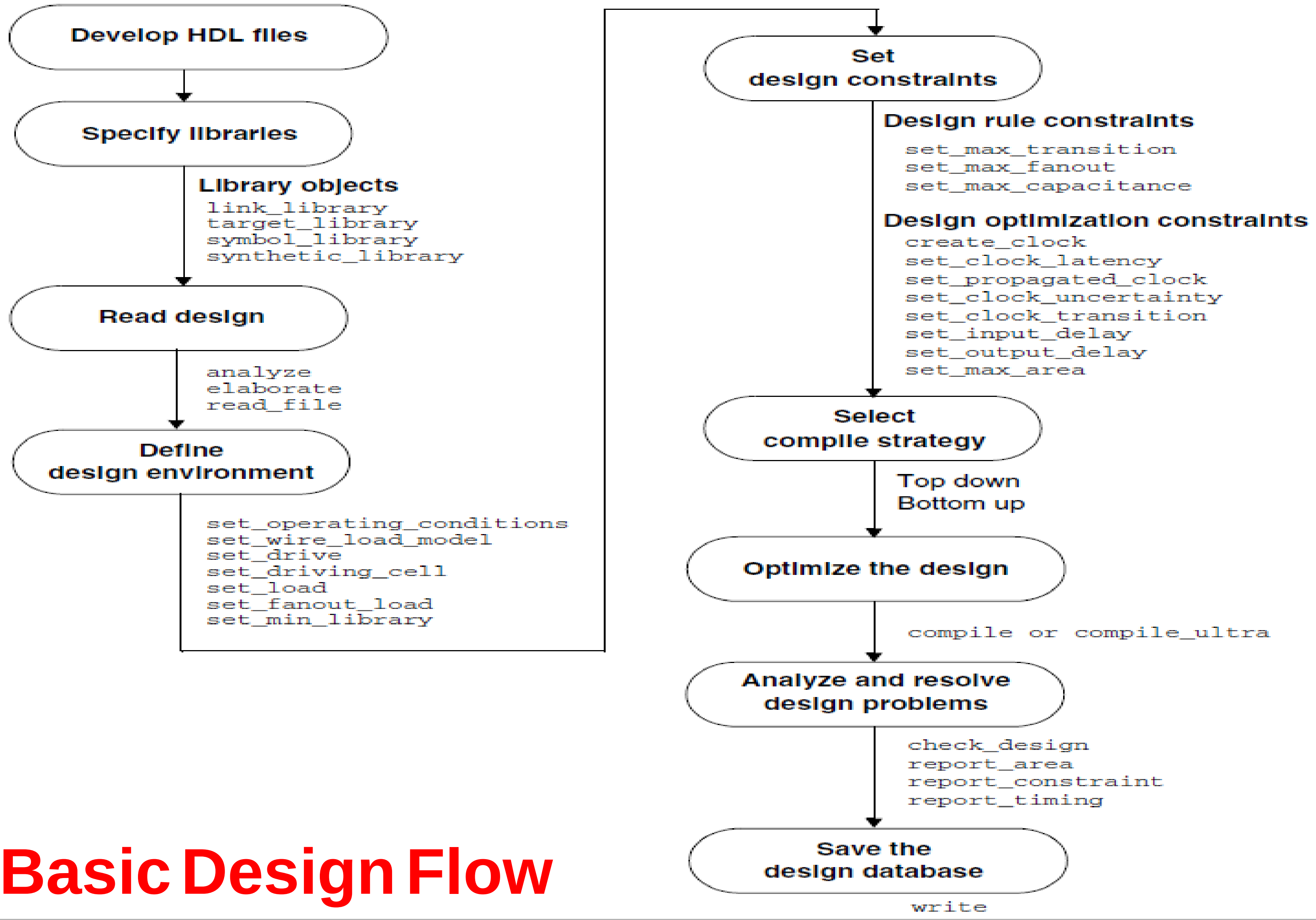
Pin Y functionality

Design Rules for Output Pin

Electrical Characteristics of Input Pins



Basic Design Flow



Library Specification

- *search_path: the path to search for unsolved reference library or design.*
- *link_library: the library used for interpreting input description, any cells instantiated in your HDL code, Wire Load or Operating Condition models used during synthesis.*
- *target_library: the ASIC technology that the design map to.*
- *symbol_library: Used during schematic generation.*
- *synthetic_library: designware library to be used. (IP)*

You can write them to .synopsys_dc.setup file.

.synopsys_dc.setup example

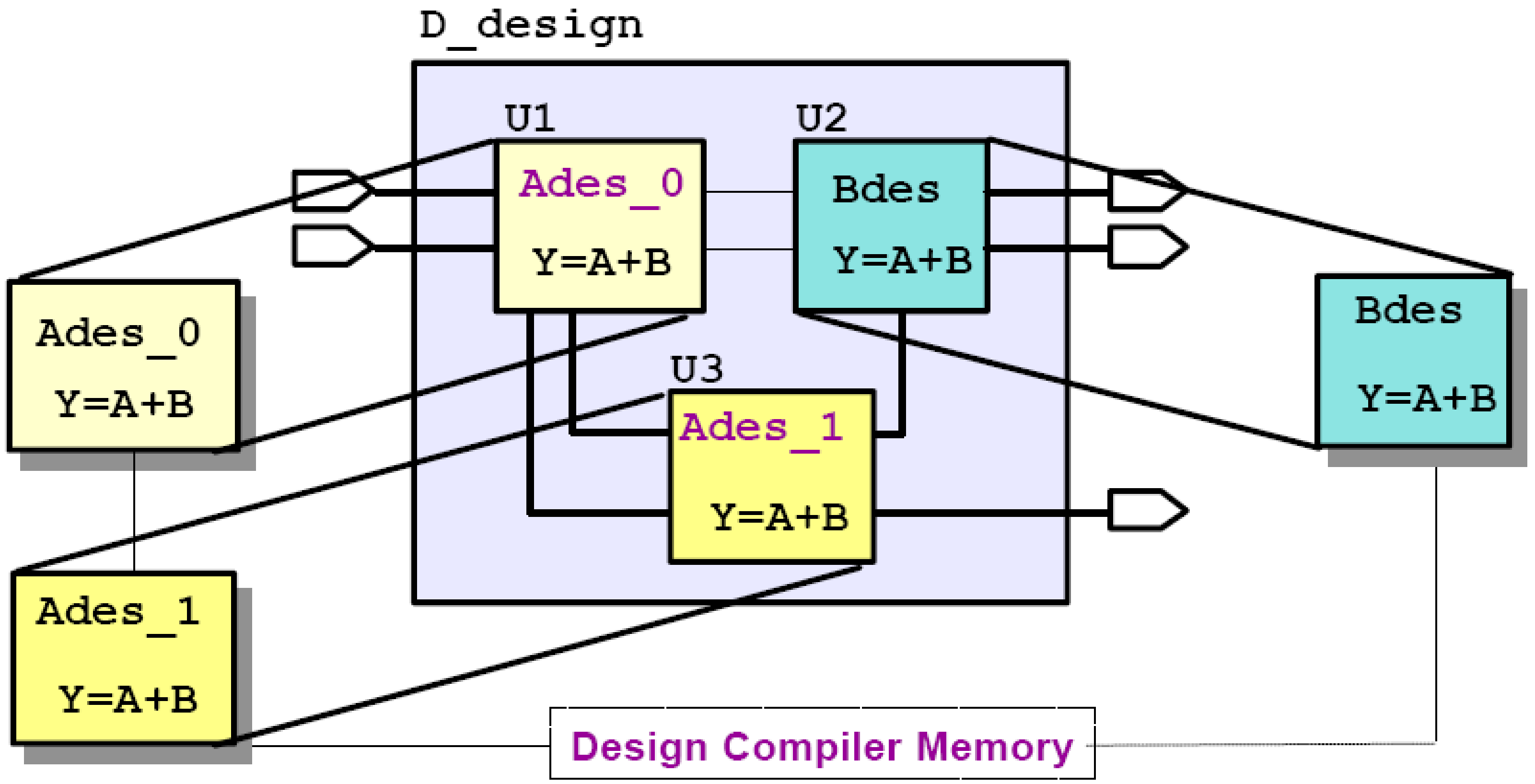
```
lappend search_path [list  ./src ./scripts \
    /net/home/EDAs/synopsys/syn/libraries/syn\
    /net/sunb2i/export/home/xyzeng/xiaohao/fft_2k_4k_8k/dc/lib\
    ]
set target_library  "smic18_tt.db"
set link_library "  *  $target_library smic18IO_line_tt.db\
                       smic18IO_stagger_tt.db dw_foundation.sldb"
set symbol_library " smic18.sdb "
set synthetic_library " dw_foundation.sldb “
……
```

- To specify technology libraries, you must specify the *target library* and *link library*.
  - Design Compiler uses the target library to build a circuit. During mapping, Design Compiler selects functionally correct gates from the target library. It also calculates the timing of the circuit, using the vendor-supplied timing data for these gates.
  - Design Compiler uses the link library to resolve references. For a design to be complete, it must connect to all the library components and designs it references. This process is called linking the design or resolving references.

Read Design

- ***step1:*** ***read, read_verilog, read_vhdl***
  或 analyze + elaborate (analyze HDL code and establish Boolean functions)
  - Difference?
- ***step2:*** ***link (resolve design reference)***
- ***step3:*** ***uniquify (removes multiply-instantiated hierarchy in the current design by creating a unique design for each cell instance)***

- `uniquify` **makes a copy of each multiply-instantiated design** (one copy for each instance)
  - Each instance gets a unique design name
  - DC can now map each instance to its own specific environment



Example

```
1 dc_shell-t> read_verilog top.v
  dc_shell-t> read_verilog sub_design.v
  dc_shell-t> current_design top
  dc_shell-t> link

2 dc_shell-t> analyze -f verilog sub_design.v
  dc_shell-t> elaborate sub_design
  dc_shell-t> analyze -f verilog top.v
  dc_shell-t> elaborate top
  dc_shell-t> current_design top
  dc_shell-t> link
```

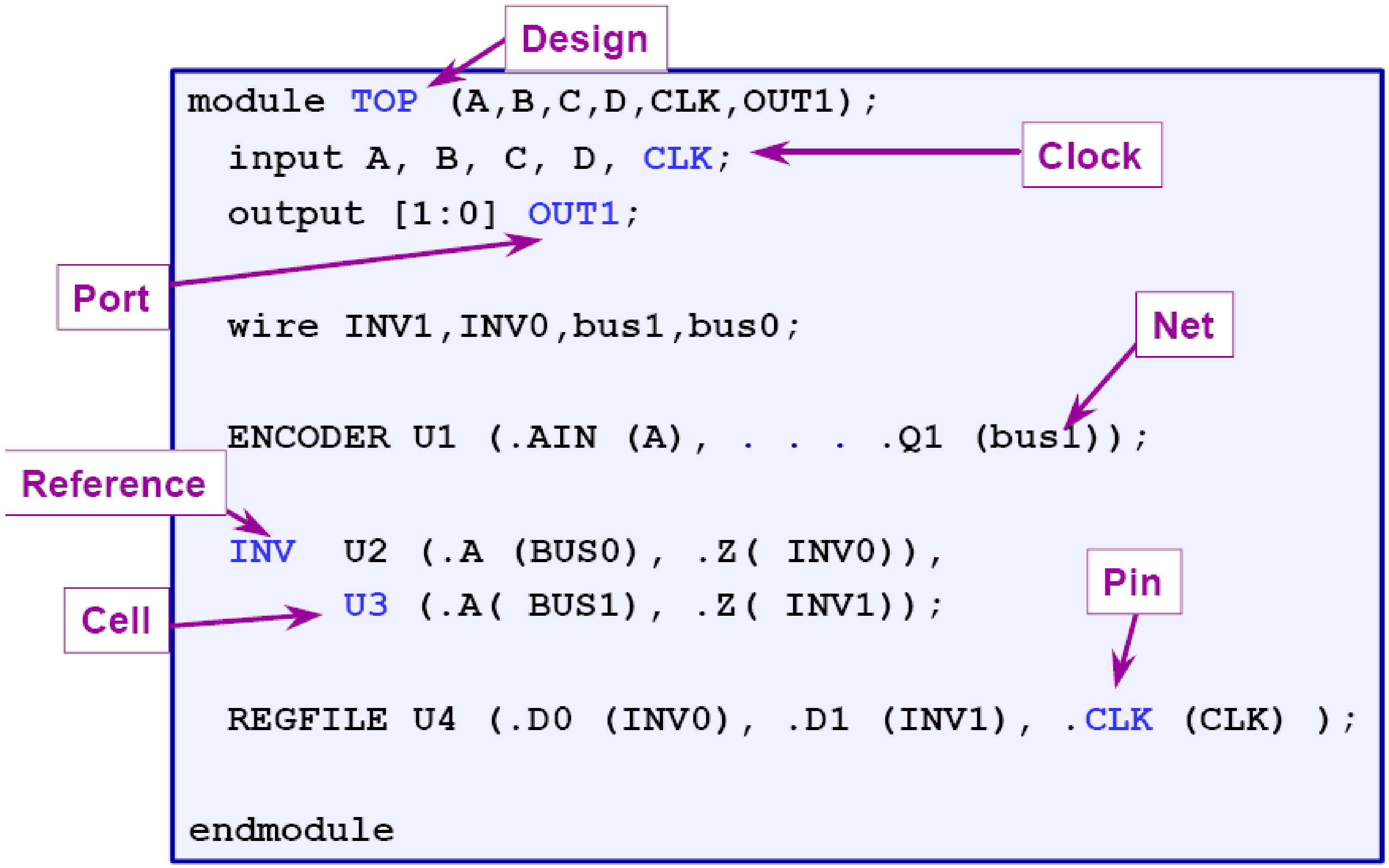
2 is preferred!

Design Objects In Synthesis

Seven Types of Design Objects:

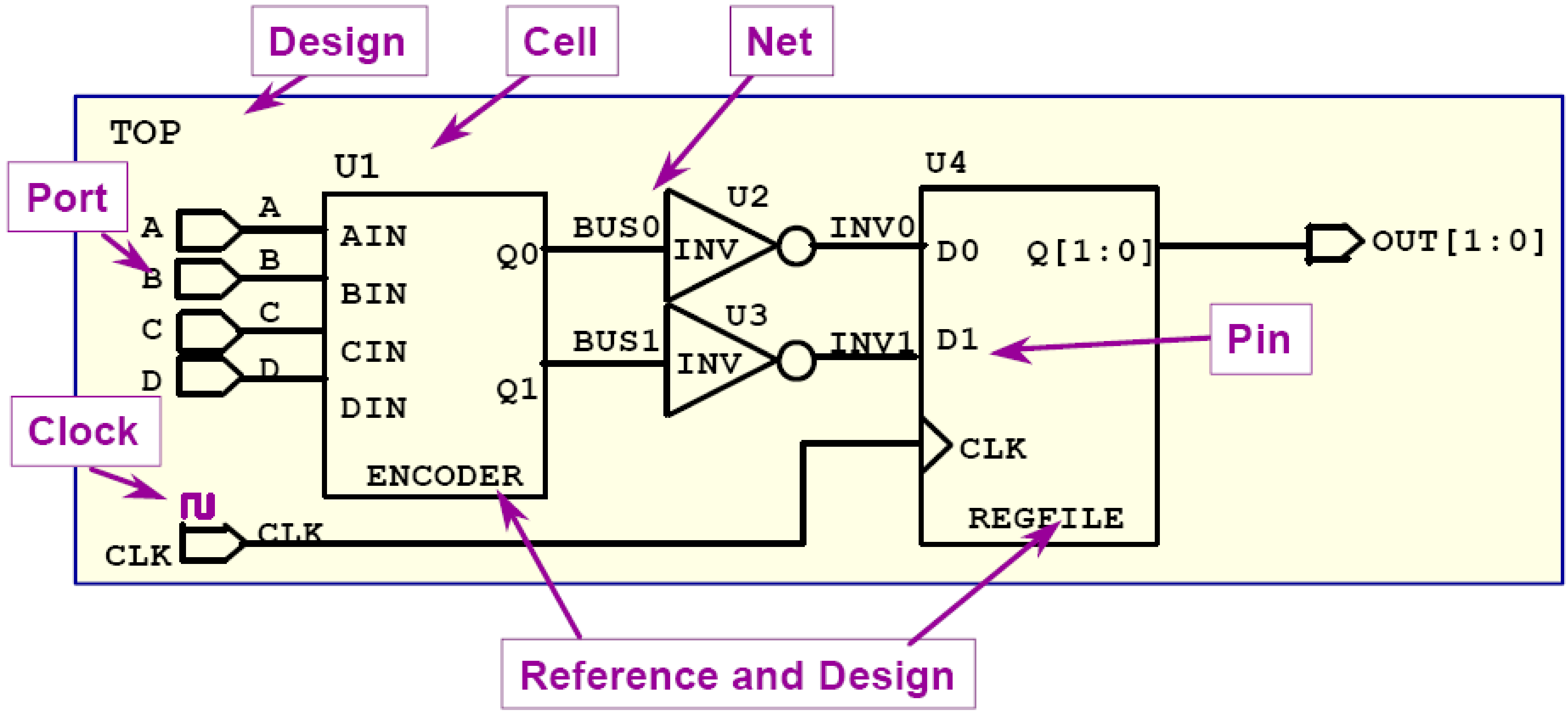
Design:	A circuit that performs one or more logical functions
Cell:	An *instance* of a design or library primitive within a design
Reference:	The name of the original design that a cell instance “points to”
Port:	The input or output of a *design*
Pin:	The input or output of a *cell*
Net:	The wire that connects ports to pins and/or pins to each other
Clock:	A timing reference object in DC memory which describes a waveform for timing analysis

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), .CLK (CLK) );

endmodule
```

Design Objects: Schematic Perspective

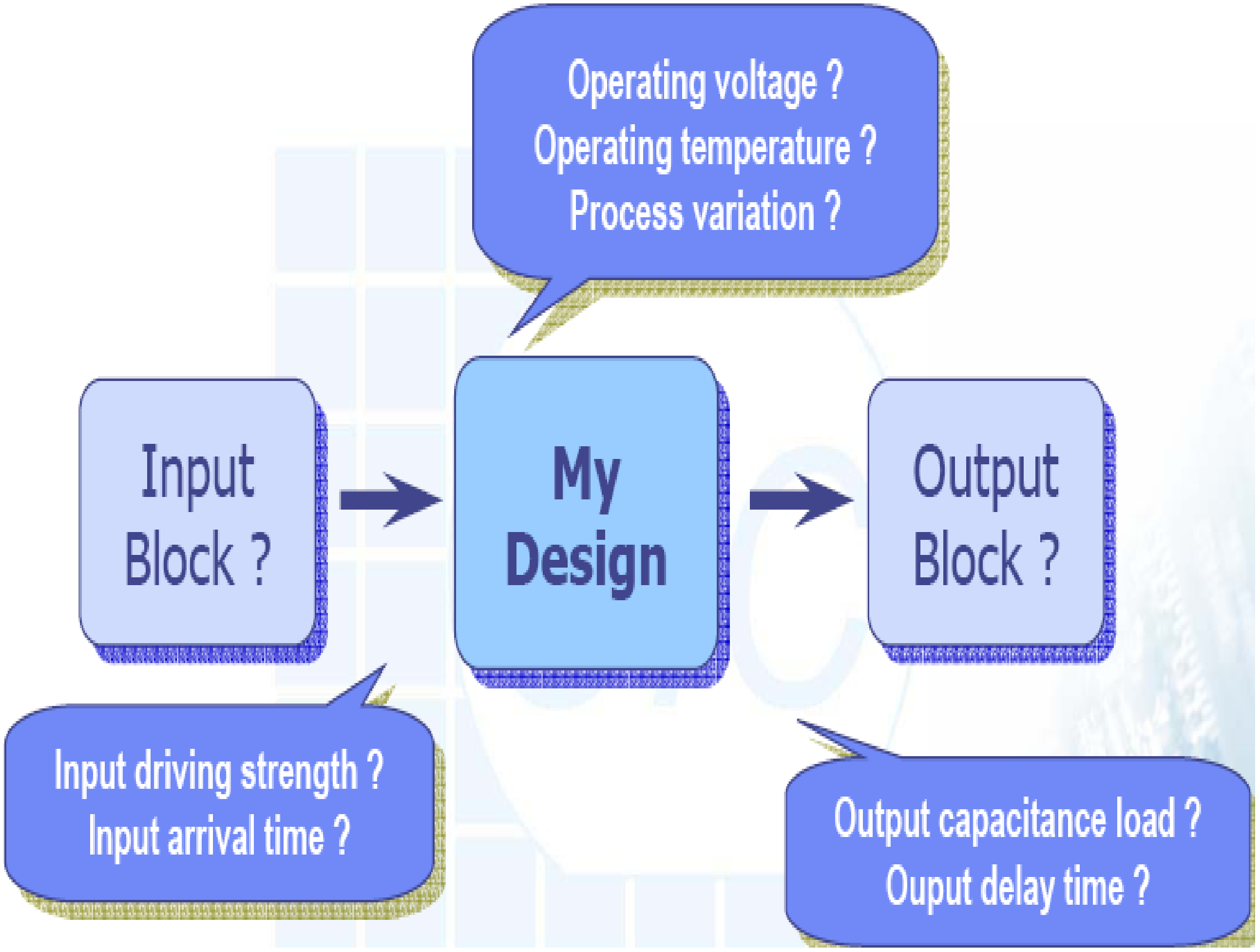


Unit in Standard Library

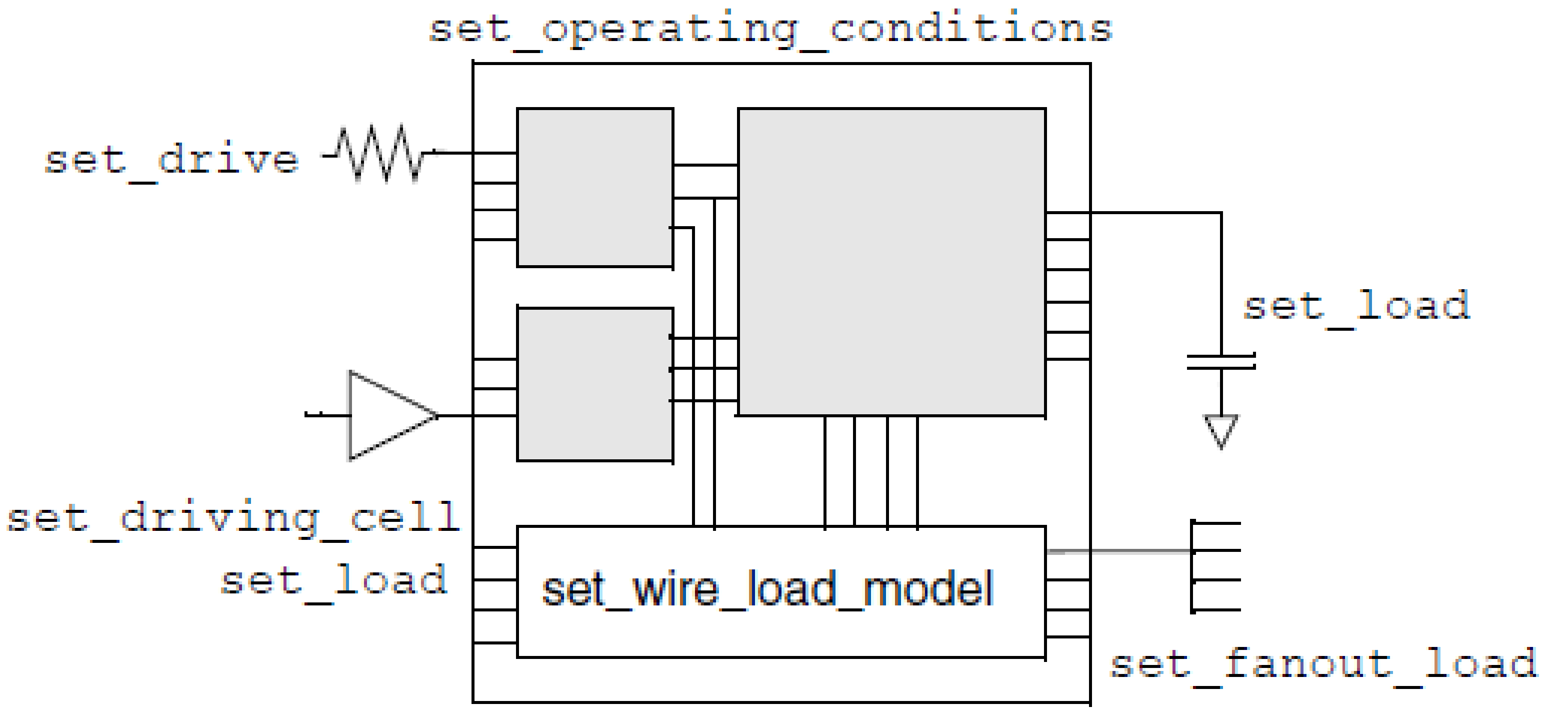
time_unit:	"1ns"
voltage_unit:	"1V"
Current_unit:	"1mA"
pulling_resistance_unit:	"1kohm"
Leakage_power_unit:	"1pW"
Capacitive_load_unit:	"1pF"

1. Setup, Libraries and Objects
2. **Set Design Environment**
3. Set Design Constraints
4. Compile Design
5. Static Timing Analysis

2. Define Design Environment

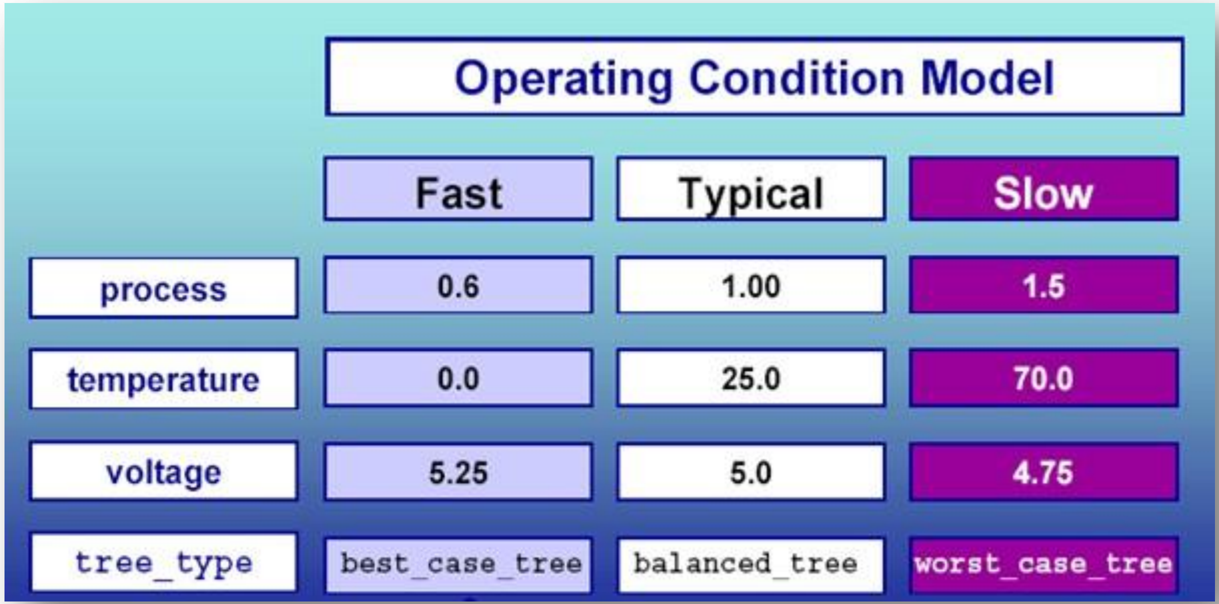


Conmands used to Define the Design Environment



Set Operating Conditions

- ***Operating condition model scales components delay, directs the optimizer to simulate variations in process, temperature and voltage***
  - set_operating_conditions



Operating Condition Model

	Fast	Typical	Slow
process	0.6	1.00	1.5
temperature	0.0	25.0	70.0
voltage	5.25	5.0	4.75
tree_type	best_case_tree	balanced_tree	worst_case_tree

Set Input Pulling Resistance (khoms) - set_drive

- set_drive : ***Sets the rise_drive or fall_drive attributes to specified resistance values on specified input and inout ports.***
  - *set_drive 2 .0 {A B C};* or *set_drive 2 .0 “A B C”;*
  - *set_drive 2 [all_inputs];*
  - *set_drive -rise 1 [get_ports B];*
  - *set_drive -rise drive_of(-rise TECH_LIBRARY/INVERTER/OUT) \ [get_ports C]*

- drive_of : ***Returns the drive resistance value of the specified library cell pin.***

Set Input Pulling Resistance (khoms) - set_drive

- **set_drive_cell :** ***sets attributes on input or inout ports of the current design that specify that a library cell or output pin of a library cell drives the specified ports.***
  - *set_driving_cell -lib_cell AND2 {IN1}*
  - *set_driving_cell -lib_cell INV -pin Z \ -library* tech_lib *[all_inputs]*
  - *set_driving_cell -lib_cell INV -don't_scale {IN1}*
  - *set_driving_cell -rise -lib_cell BUF1_TS -pin Z {IN1}*
  - *set_driving_cell -fall -lib_cell DFF_TS -pin Q {IN1}*

Setting Output Loading Capacitance (pF) -set_load

- ***set_load :*** ***Sets the load attribute to a specified value on specified ports and nets.***
- ***load_of:*** ***Returns the capacitance of the specified library cell pin.***

- ***Example:***

```
set MAX_LOAD [load_of slow/AND2X1/A]
set_load [expr $MAX_LOAD*15] [all_outputs]
```

Setting Output Loading Capacitance (pF) -set_load

- *set _load 2 in1*
- *set port_load [expr 2.5+3*[load_of tech_lib/IV/A]]*
- *set_load $port_load [all_outputs]*
- *set_load [load_of tech_lib/IV/Z] {input_1 inoput_2}*

set_fanout_load

- You can model the **external fanout effects** by specifying the expected fanout load values on **output ports** with the set_fanout_load command.
  - set_fanout_load 4 {out1}

- Design Compiler tries to ensure that the sum of the **fanout load on the output port** plus **the fanout load of cells connected to the output port driver** is less than **the maximum fanout** limit of the library, library cell, and design.

set_load V.S. set_fanout_load

- fanout load is not the same as load.
- fanout load is a **unitless value** that represents a numerical contribution to the total fanout. Load is a capacitance value.
- Design Compiler uses fanout load primarily to measure the fanout presented by each input pin. **An input pin normally has a fanout load of 1, but it can have a higher value.**

Design Constraints: set_max_fanout

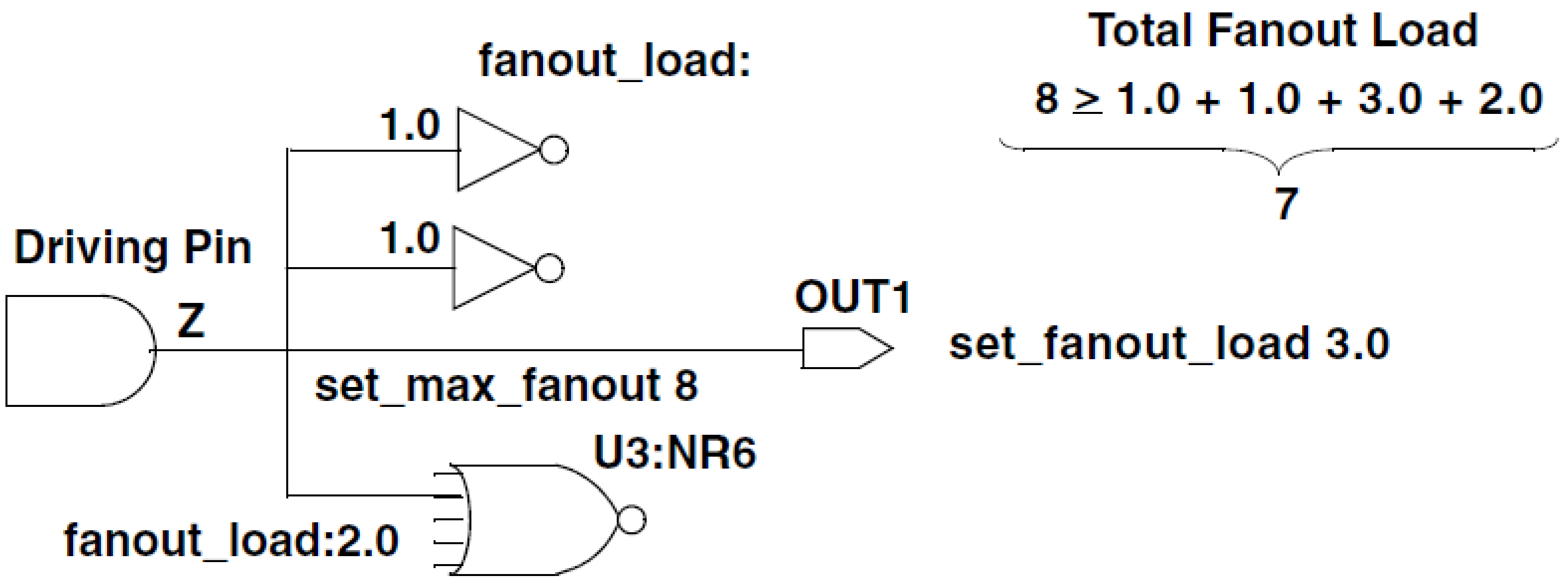
- set_max_fanout is a **design constraints.**
- set_max_fanout: set the max_fanout attribute to a specified value on **input ports** and **designs**
  - set_max_fanout 20.0 going_places
  - set_max_fanout 18.5 TEST

set_fanout_load V.S. set_max_fanout

- set_fanout_load is **design envoronment**; set_max_fanout is a **design constraint**.
- Design Compiler models fanout restrictions by associating **a fanout_load** attribute with **each input pin** and **a max_fanout** attribute with **each output (driving) pin on a cell**.
- Design Compiler tries to ensure that the sum of the **fanout load on the output port** plus **the fanout load of cells connected to the output port driver** is less than **the maximum fanout** limit of the library, library cell, and design.

Case Study: fanout_load & max_fanout_load

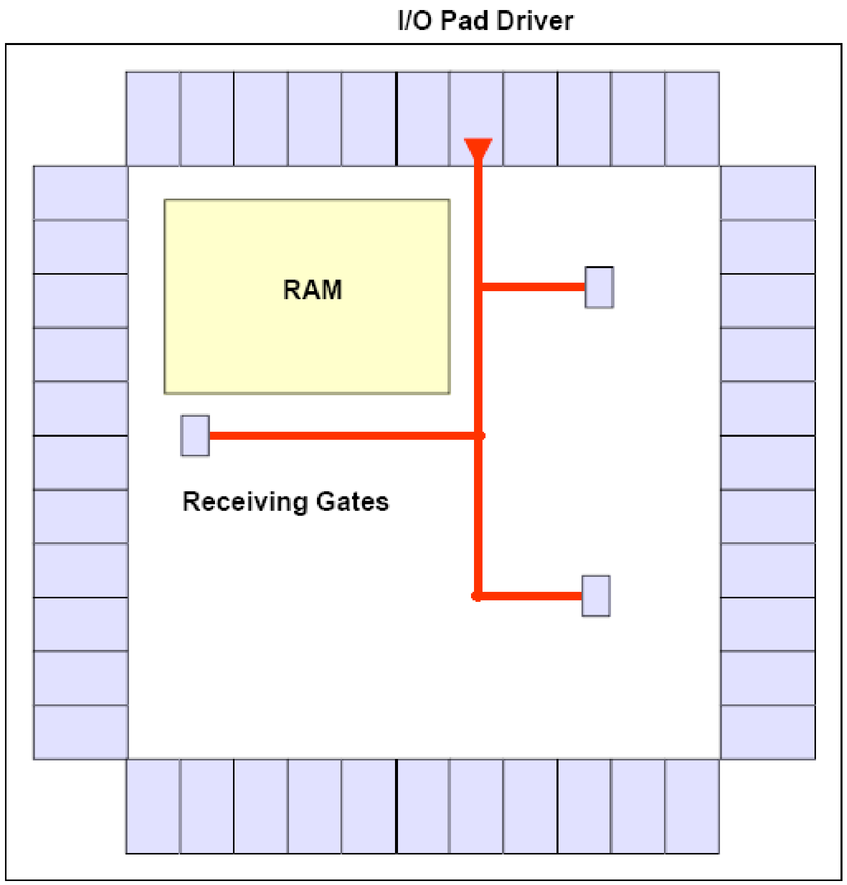
```
>>set_fanout_load    3.0    OUT1
>>set_max_fanout     8    [get_designs     ADDER]
```



Case Study

- An input pin normally has a fanout load of 1, but it can have a higher value.
- The fanout load imposed by a driven cell (U3) is not necessarily 1.0. Library developers can assign higher fanout loads (for example, 2.0) to **model internal cell fanout effects.**
- You can also set a fanout load on an output port (OUT1) to **model external fanout effects.**

Net Delay

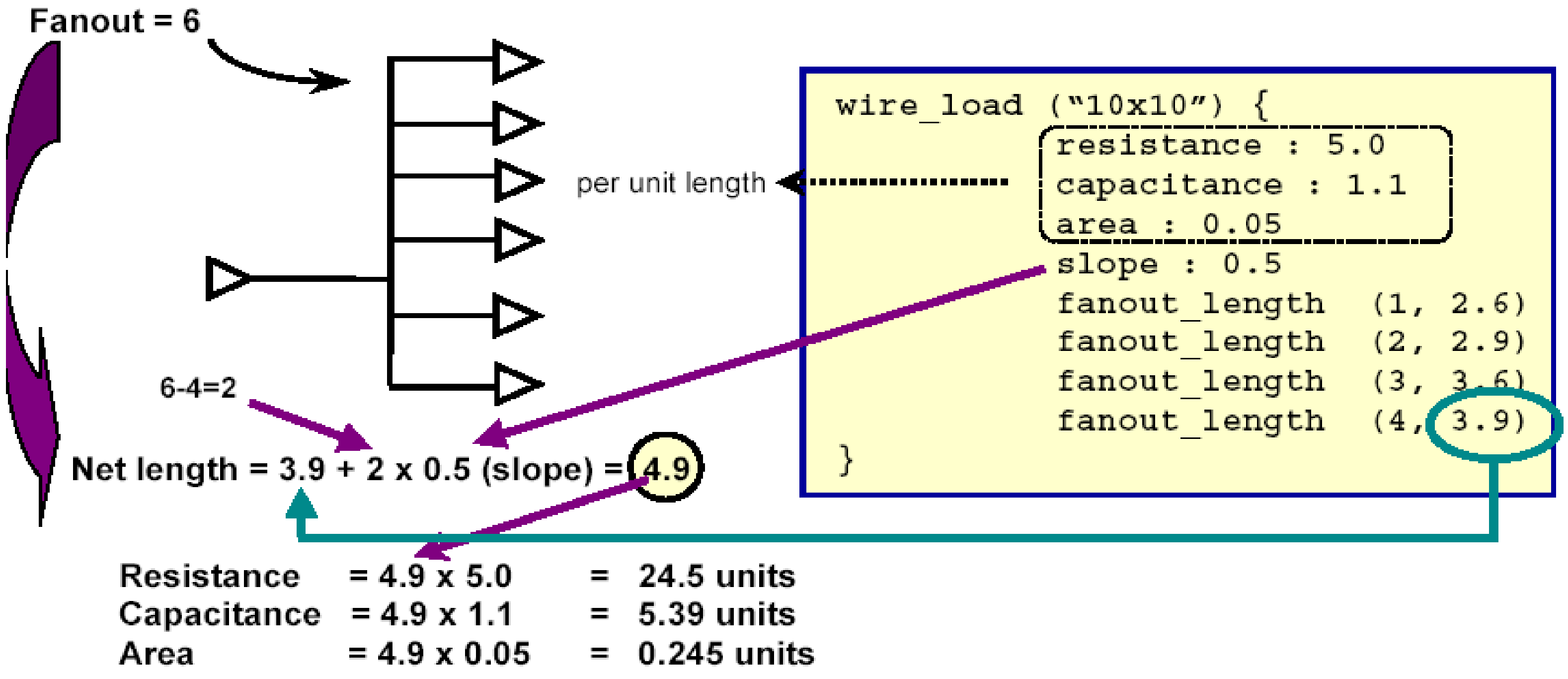


Prior to layout, how can the RC delay of nets be estimated?

Wire Load Model

- ***A wire load model is an estimate of a net's RC parasitics based on the net's fanout.***

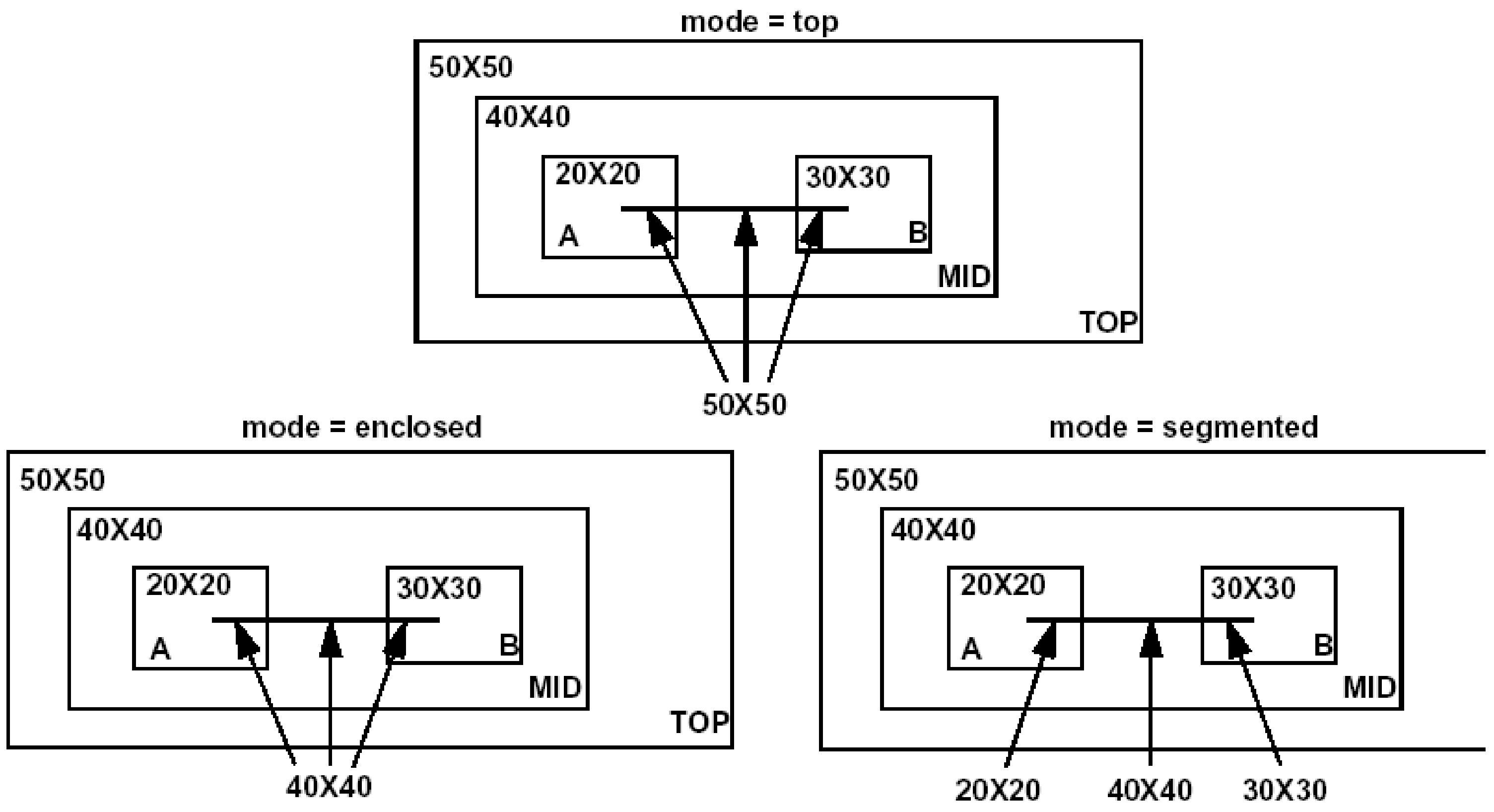
Example: dc_shell-t> set_wire_load_model 10x10



```
wire_load ("10x10") {
        resistance : 5.0
        capacitance : 1.1
        area : 0.05
        slope : 0.5
        fanout_length   (1, 2.6)
        fanout_length   (2, 2.9)
        fanout_length   (3, 3.6)
        fanout_length   (4, 3.9)
}
```

Resistance	= 4.9 x 5.0	=	24.5 units
Capacitance	= 4.9 x 1.1	=	5.39 units
Area	= 4.9 x 0.05	=	0.245 units

Setting Wire Load Mode



dc_shell-t> set_wire_load_mode <top|enclosed|segmented>

Hierarchical Wire Load Models

- DC supports three modes for determining which wire load model to use for nets that cross hierarchical boundaries:
  - *Top*: ***DC models nets as if the design has no hierarchy and uses the wire load model specified for the top level of the design hierarchy for all nets in a design and its sub-designs.***
  - ***Enclosed:*** …
  - ***Segmented:*** …

Hierarchical Wire Load Models

- **Enclose:** ***DC uses the wire load model of the smallest design that fully encloseds the net. If the design enclosing the net has no wire load model, the tool traverses the design hierearchy upward until it finds a wire load model. Enclosed mode is more accurate than top mode when cells in the same design are placed in a contiguous region during layout.***

- **Segmented:** ***DC determines the wire load models of each segment of a net by the design encompassing the segment. Nets crossing hierarchical boundaries are divided into segments.***

- set _wire_load_model -name “10*10” -library my_lib.db

- current_design LOW
- set_wire_load_model -name “10*10”
- current_design TOP
- set_wire_load_mode enclosed
- set_wire_load_model -name “20*20MIN” -min

1 Setup, Libraries and Objects

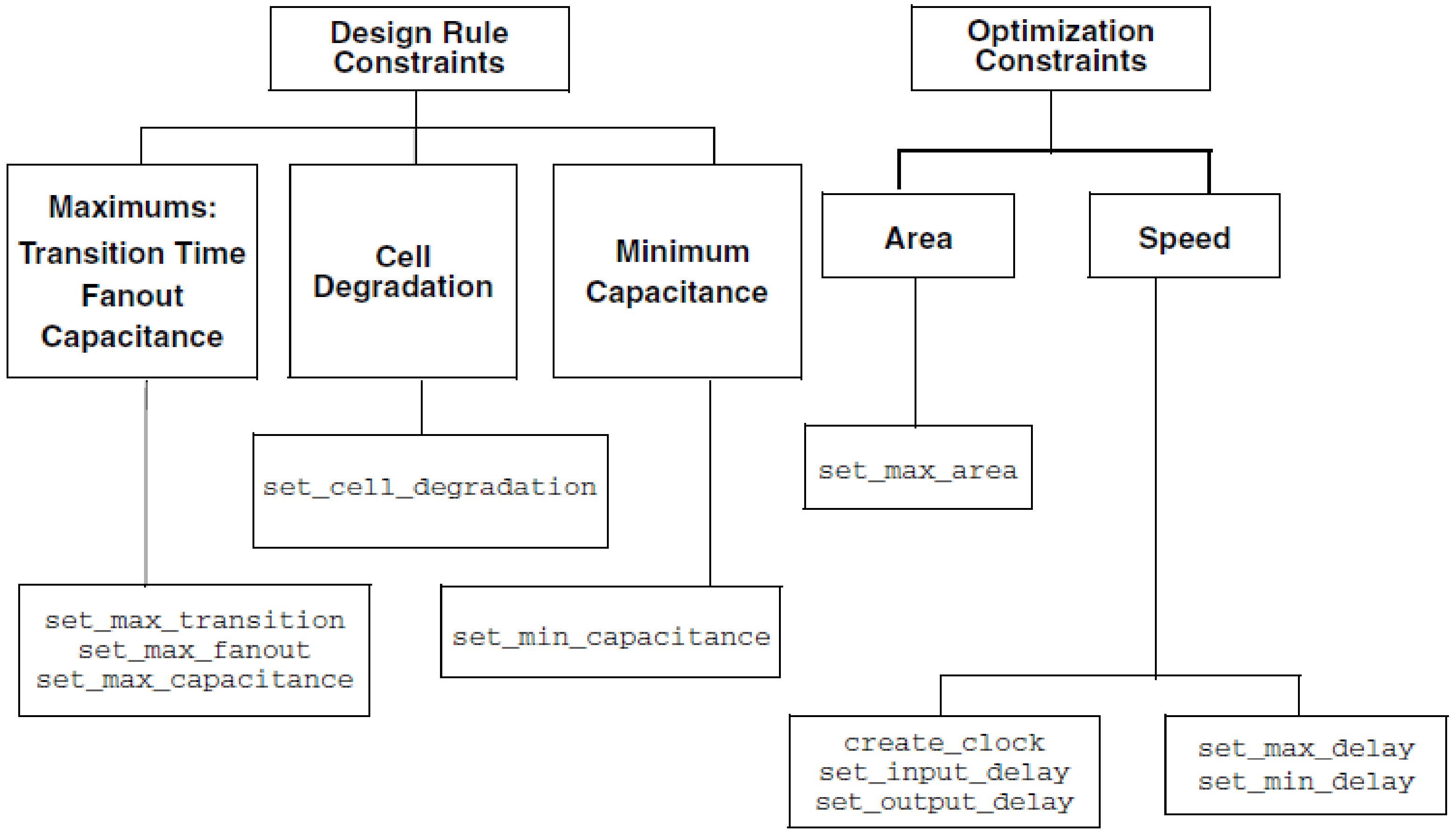
2 Set Design Environment

3 **Set Design Constraints**

4 Compile Design

5 Static Timing Analysis

Design Constraints: Design Rule Constraints and Optimization Constraints


Design Rule Constraints
Optimization Constraints
Maximums: Transition Time Fanout Capacitance
Cell Degradation
Minimum Capacitance
Area
Speed
set_cell_degradation
set_max_area
set_max_transition set_max_fanout set_max_capacitance
set_min_capacitance
create_clock set_input_delay set_output_delay
set_max_delay set_min_delay

Design Rule Constraints and Optimization Constraints

- ***Design Rule Constraints:*** ***technology-specific restriction.***
- ***Optimization Constraints:*** ***design goals and requirements.***
- ***During compiling, Design Compiler attempts to meet all constraints.***

Design Rule Constraints

- ***Design rules can not be violated at any cost, even if it will violate the timing and area goal.***

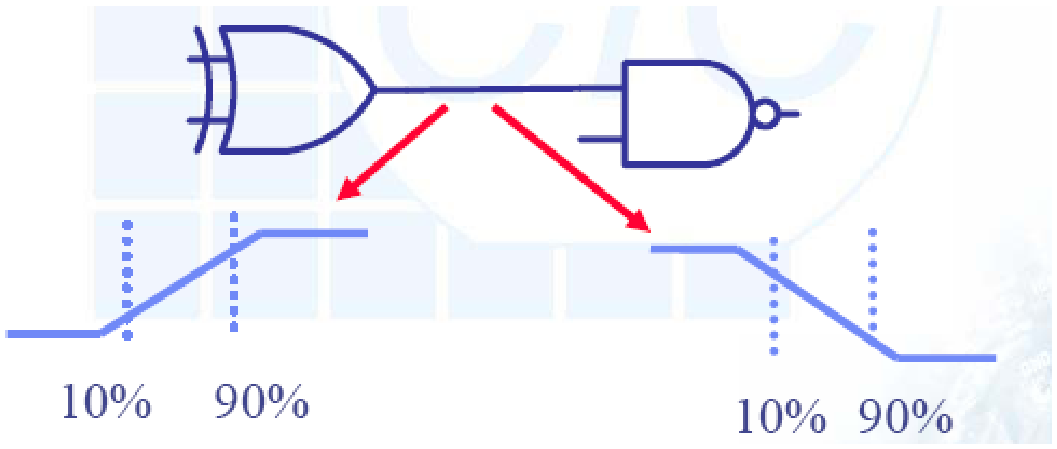
- ***Rule Constraints Demand:***
  - ***set_max_transition***
  - ***set_max_fanout** (explained in last chapter)*
  - ***set_max_capacitance***

  - ***set_cell_degradation***
  - ***set_min_capacitance***

set_max_transition (ns)

- *set_max_transition: set the maximum transition time for specified clocks, ports, or designs.*

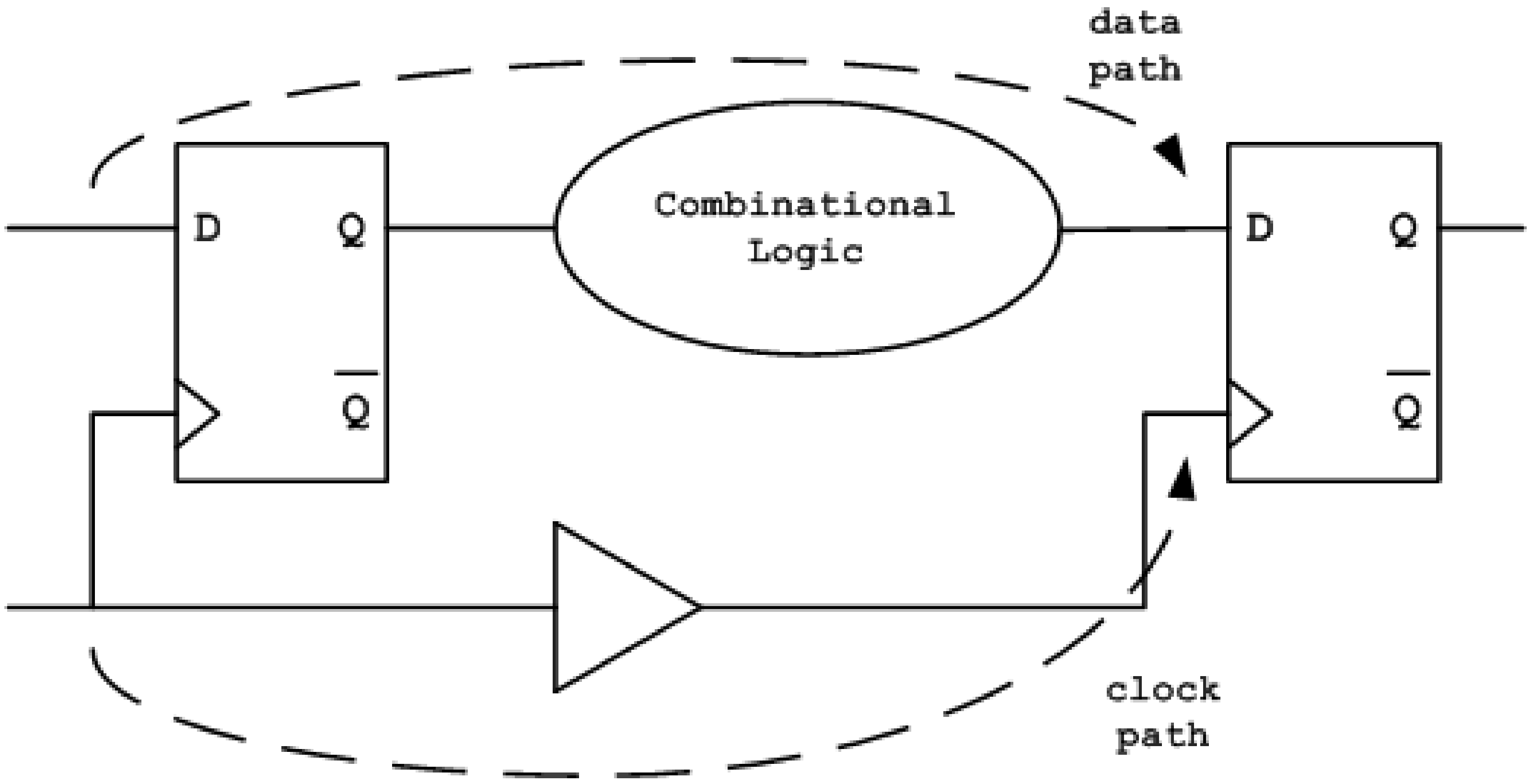
  set_max_transition 1.5 all_inputs



set_max_transition (ns)

- Port: late_riser
  - set_max_transition 2.0 late_riser
- Design: TEST
  - set_max_transition 2.0 TEST
- Clock Path: clk1
  - set_max_transition 2.0 [get_clocks clk1]
- Data Path: clk1
  - set_max_transition 2.0 -datapath [get_clocks clk1]

clock path & data path



set_max_capacitance

- It is set as a **pin-level** attribute that defines the maximum total capacitive load that an **output pin** can drive.
- The **max_capacitance** design rule constraint allows you to control the capacitance of nets directly. (The design rule constraints **max_fanout** and **max_transition** limit the actual capacitance of nets indirectly.)
- The **max_capacitance** attribute functions independently, so you can use it with **max_fanout** and **max_transition**.

set_max_capacitance

- set_max_capacitance 2.0 [get_ports late_riser]
- set_max_capacitance 2.0 [current_design]
- set_max_capacitance 2.0 [get_clocks clk1]
- set_max_capacitance 2.0 -datapath [get_clocks clk1]

Summary of Design Rule Commands and Objects

Command	Object
`set_max_fanout`	Input ports or designs
`set_fanout_load`	Output ports
`set_load`	Ports or nets
`set_max_transition`	Ports or designs
`set_cell_degradation`	Input ports
`set_min_capacitance`	Input ports

Optimization Constraints

The optimization constraints comprise

- Timing constraints (performance and speed)
  - Input and output delays (synchronous paths)
  - Minimum and maximum delay (asynchronous paths)
- Maximum area (number of gates)

Timing Constraints

- Use the create_clock command to specify a clock.
- Use the set_input_delay and set_output_delay commands to specify the input and output port timing specifications.
- Use the set_max_delay and set_min_delay commands to specify these point-to-point delays.

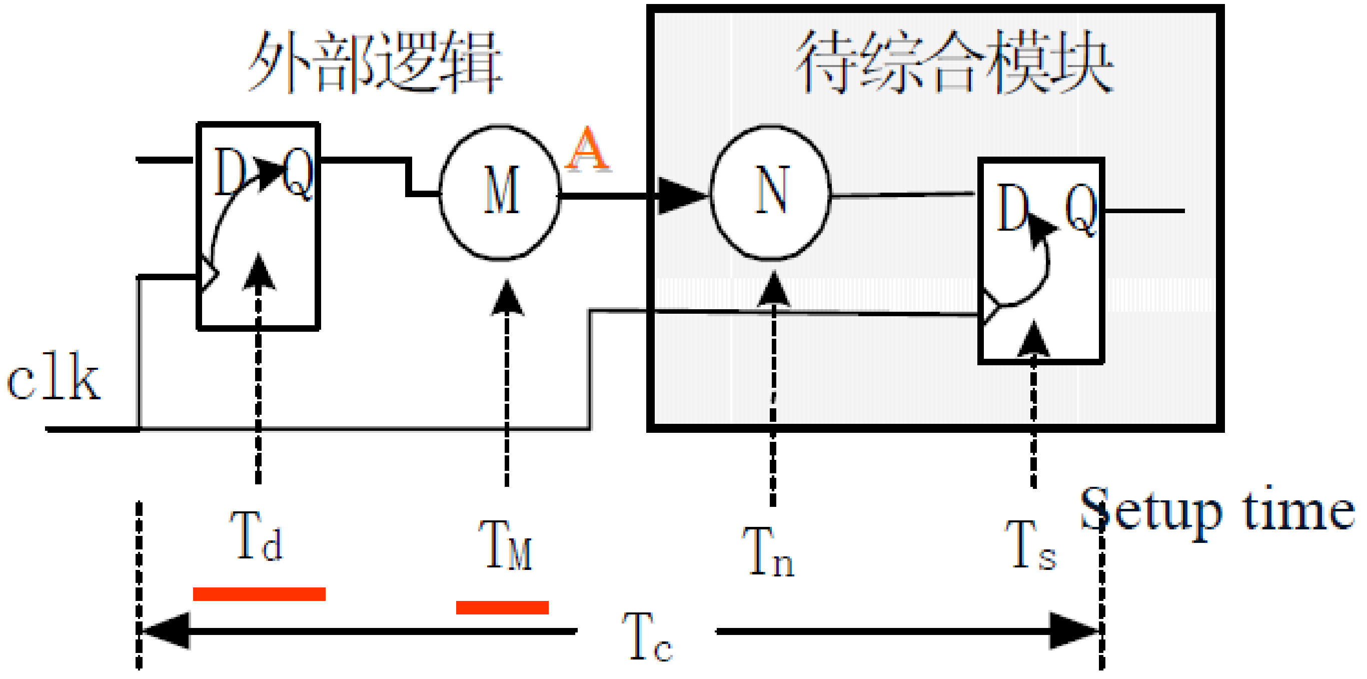
create_clock

- create_clock: creates a clock object and defines its waveform in the current design.

- create_clock “PHI1” –period 10 –waveform {5.0 9.5}

set_input_delay

输入信号A在时钟有效沿后多长时间后才到达或有效，DC会用它来计算内部逻辑的延迟时间。

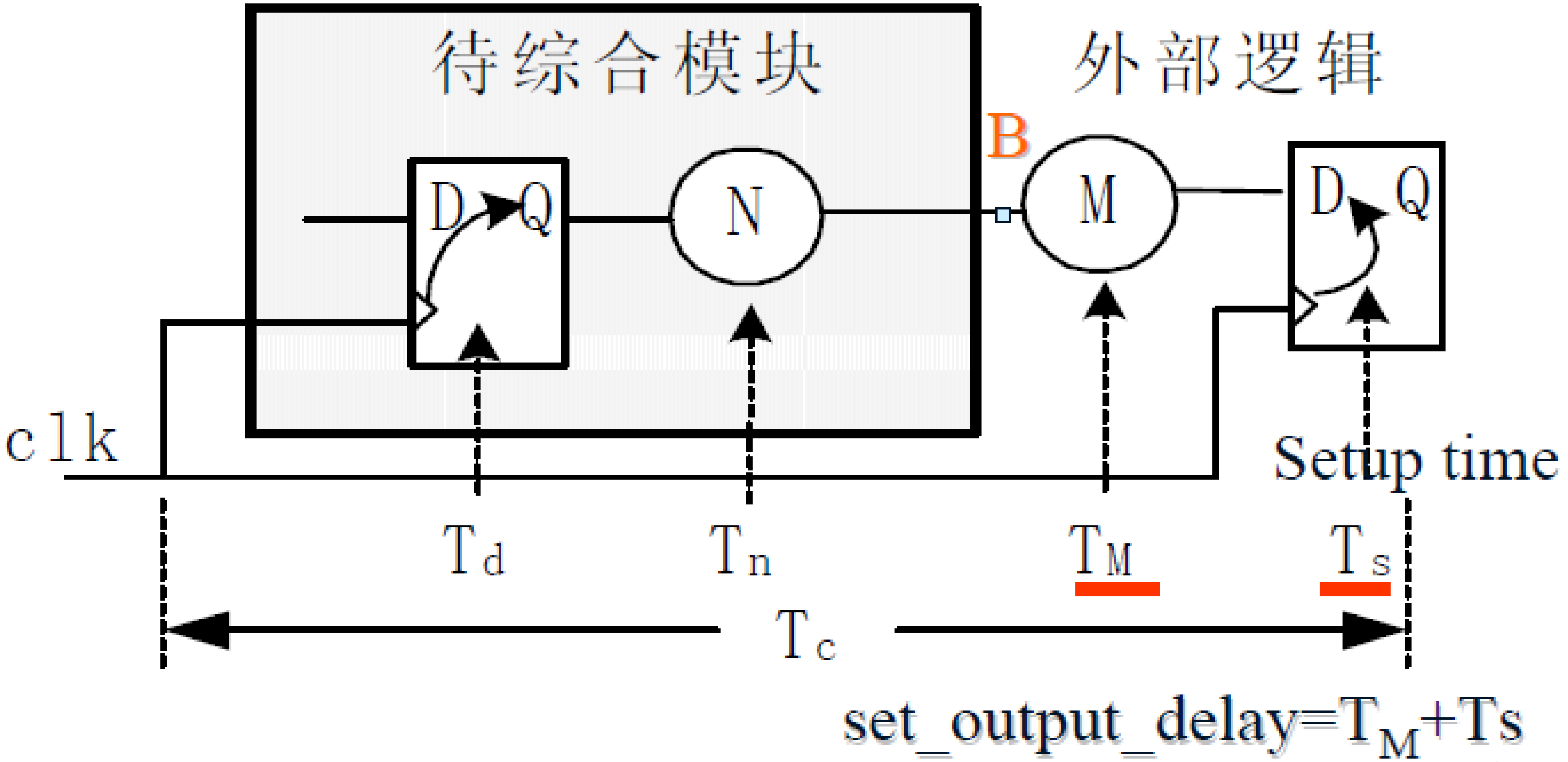


set_input_delay=T_d+T_M

University, Jinmei Lai

set_output_delay

输出信号B在时钟有效沿前多长时间数据有效。



set_max_delay & set_min_delay

- set_max_delay: Specifies a maximum delay target for path in the current design.
  - set_max_delay 10.0 -to {Y}
  - set_max_delay 15.0 -from {ff1a ff1b} \
    -through {u1} -to {ff2e}
  - set_max_delay 8.5 -to [get_clocks PHI2]
  - set_min_delay 12.5 –through U1 –to Y
  - set_min_delay 4.0 –from {A1 A2} –to Z5

Optimization Constraints

- *Optimization constraints, in order of attention are*
  - ***Maximum delay***
  - ***Minimum delay***
  - ***(Maximum power)***
  - ***Minimum area***

- ***About combinational circuit,* we only set *maximum delay & minimum delay* for timing constraint.**

  set_max_delay 5.0 –from port_A –to port_B

  set_min_delay 2.0 –from port_A –to port_B

Timing Constraints for Sequential Circuit

- ***create_clock:* define your clock's waveform & respect the set-up time requirements of all clocked flip-flops.**

  create_clock -period 12.5 -waveform {0 6.25} [get_ports clkM]
- ***set_dont_touch_network:* do not re-bufer the clock network.**
- ***set_dont_touch_network [get_clocks clkM]***

Setting Area Constraint

- ***Area Unit:***
  - ***Equivalent gate counts***
  - ***um^2***
  - ***Transistors***
- ***To reduce the area as much as possible***

  set max_area 0

  The area violation will disclose the total area of your design after synthesis.

Constraints Priority

- *During the optimization ,there exists a constraint priority relationship*
  - ***Design Rule Constraint***
  - ***Timing constraint***
  - ***Power constraint***
  - ***Area constraint***

- ***Using set_cost_priority command to modify the order***

  set_cost_priority [-default] [-delay] [-cost_list]

1 Setup, Libraries and Objects

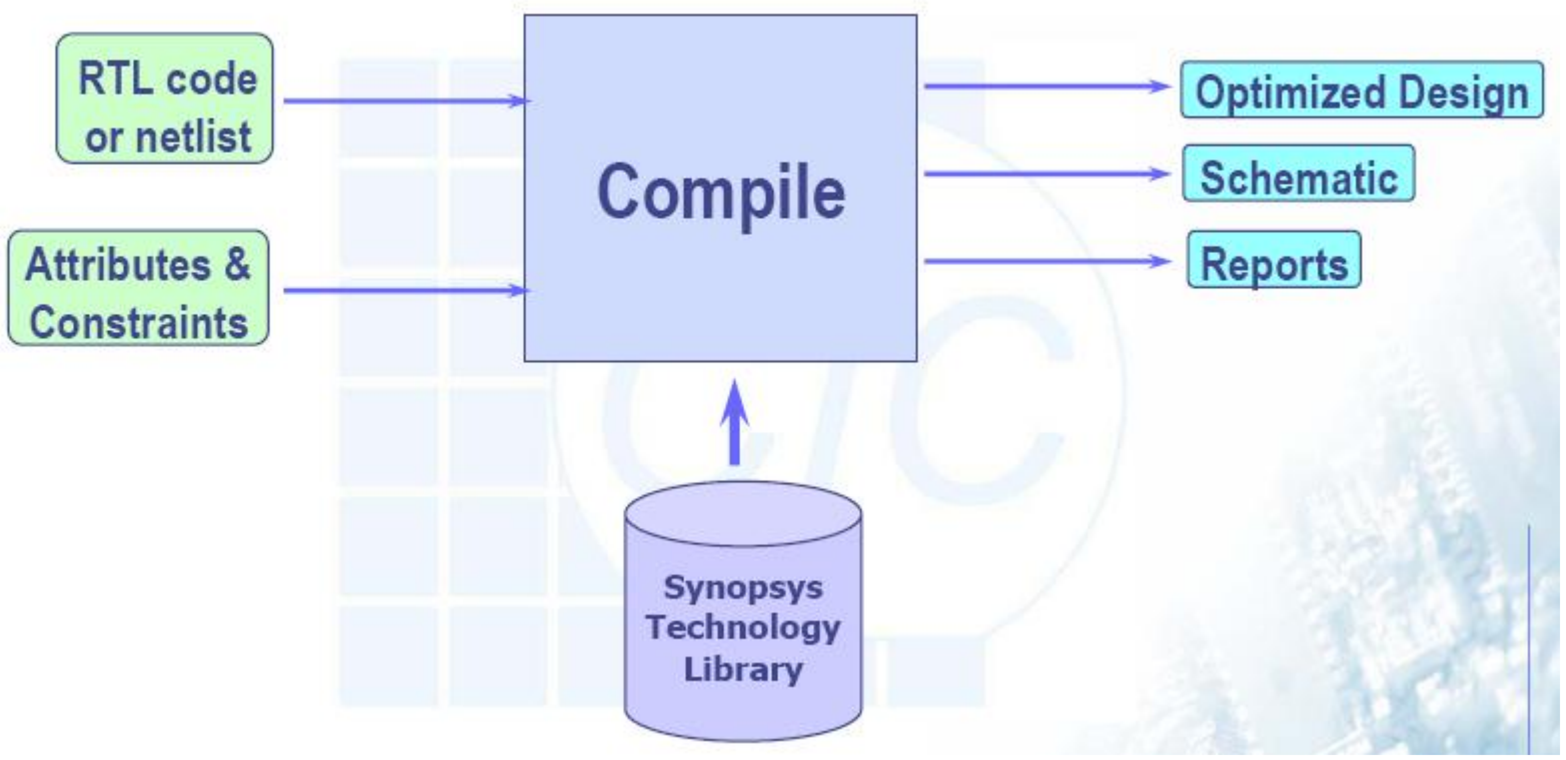
2 Set Design Environment

3 Set Design Constraints

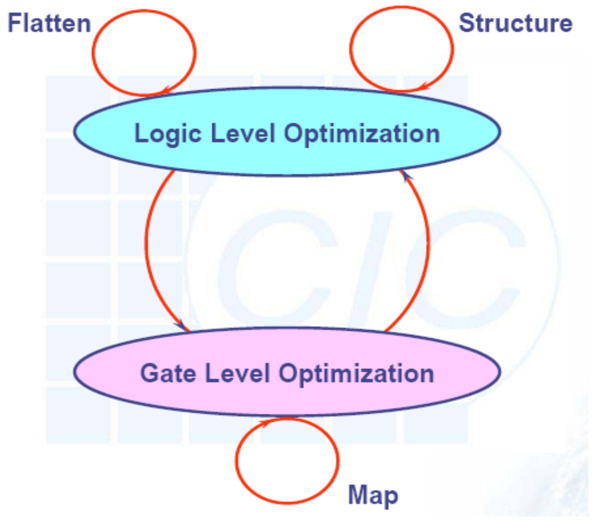
4 **Compile Design**

5 Static Timing Analysis

How to map and optimize a design



Compile is the “art” of synthesis



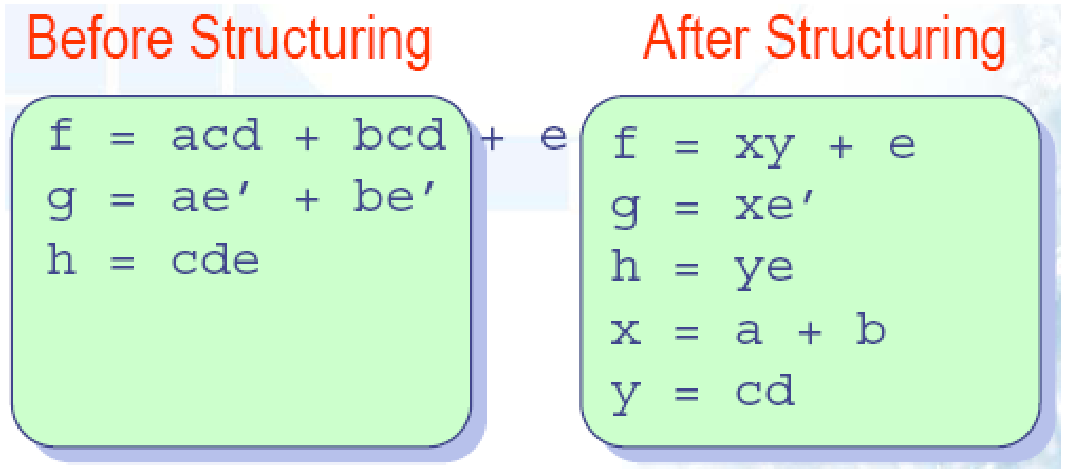
- ***Logical level optimization***
  - ***flatten :** remove structure*
  - ***structure:** minimize generic logic*
- ***Gate level optimization***
  - ***map:** make design technology dependent*

Logical Level Optimization

- ***Operate with boolean representation of circuit .***
- ***Has a global effect on the overall area/speed characteristic of a design.***
- ***Strategy: structure or flatten, if both are true, the design is first flatten and then structured .***

Structure

- *Factors out common sub-expression as intermediate variable*
- *Useful for speed optimization as well as area optimization*
- *Example: set_structure true|false*



Before Structuring

```
f = acd + bcd + e
g = ae' + be'
h = cde
```

After Structuring

```
f = xy + e
g = xe'
h = ye
x = a + b
y = cd
```

Flatten

- *Flatten is default OFF*
- *Remove all intermediate variable*
- *Result a two-level sum-of-product form*
- *Example: set_flatten true|false*

Before Flattening

$f0 = a \cdot t$

$f1 = d + t$

$f2 = t' \cdot e$

$t = b + c$

After Flattening

$f0 = a \cdot b + a \cdot c$

$f1 = d + b + c$

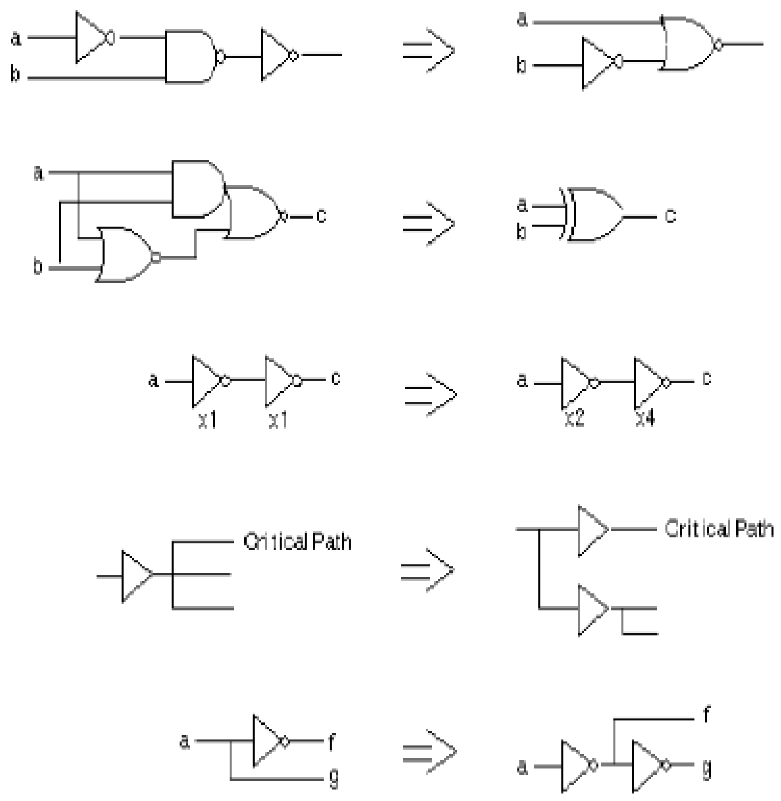
$f2 = b' \cdot c' \cdot e$

Gate Level Optimization

- ***Select components to meet timing, design rule & area goals specified for the circuit.***
- ***Has a local effect on the area/speed characteristic of a design.***
- ***Strategy: combinational mapping & sequential mapping.***

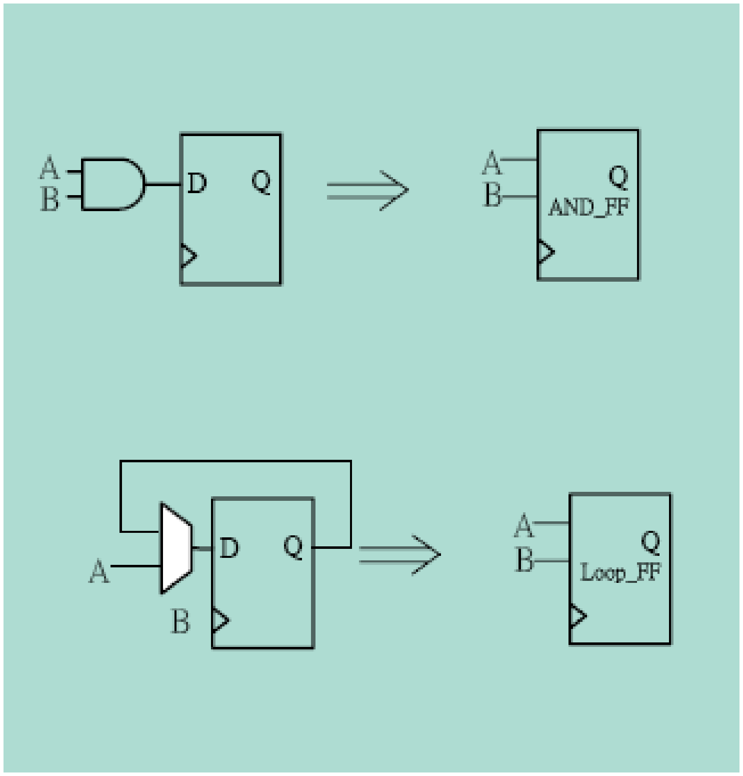
Combinational Mapping

- ***Mapping* rearranges components, *combining* and *re-combining* logic into different components.**
- ***May use different algorithms such as cloning, resizing or buffering.***
- ***Try to meet the design rule constraints and timing area goals.***



Sequential Mapping

- *Optimize the mapping to sequential cells from technology library.*
- *Analyze combinational surrounding a sequential cell to see if it can absorb the logic attribute with HDL.*
- *Try to save speed and area by using a more complex sequential cell.*



Compile Command and Options

- ***compile –map_effort medium | high***
  - ***high**, it does critical path re-synthesis, but it will use more CPU time, in some case the action of compile will not terminate.*
  - ***medium**, by default, good for many design.*

- ***Compile –incremental_mapping***
  - *Perform incremental gate level optimization but no logical level optimization*

- ***Compile –only_design_rule***
  - *Perform only design rule fixing, take less time than regular compile because it is incremental.*

How to generate results and report attributes for synthesis

- ***Attributes to Report***

  report_design, report_clock, report_port, report_net, report_hierarch, report_area, report_reference, report_constraint

- ***Results to generate***

  write –output –format ddc mapped.ddc

  write –output –format verilog mapped.v

  write_sdc mapped.sdc

  write_sdf mapped.sdf

1 Setup, Libraries and Objects

2 Set Design Environment

3 Set Design Constraints

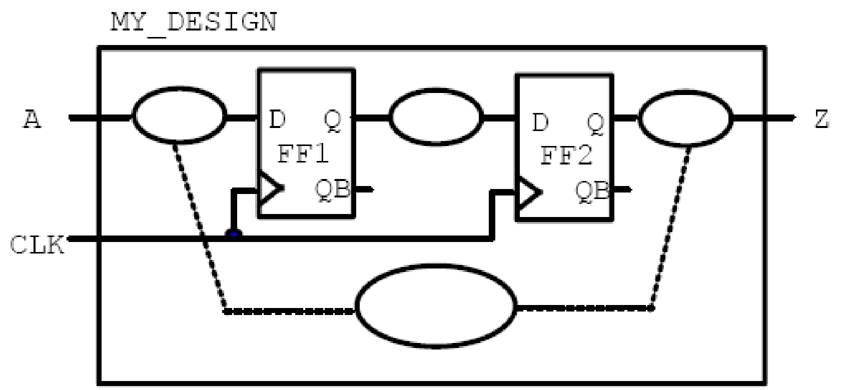
4 Compile Design

5 Static Timing Analysis

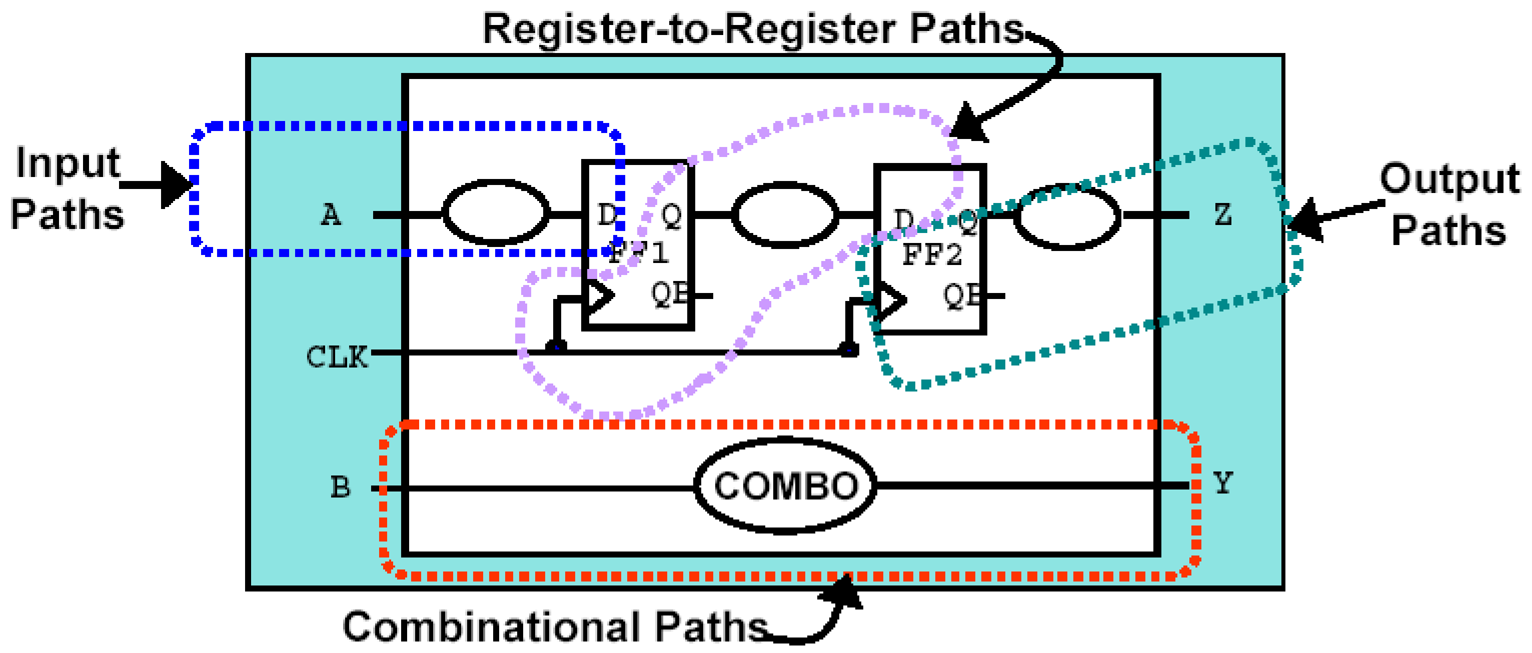
Basic Conceptions of STA

- ***Timing paths and groups***
- ***Delay calculation***
- ***Setup and hold time check***
- ***Gated clocks check***
- ***Removal and Recovery time check***
- ***Clock tree modeling***
- ***Input delay and output delay***

Timing Paths in Design Compiler



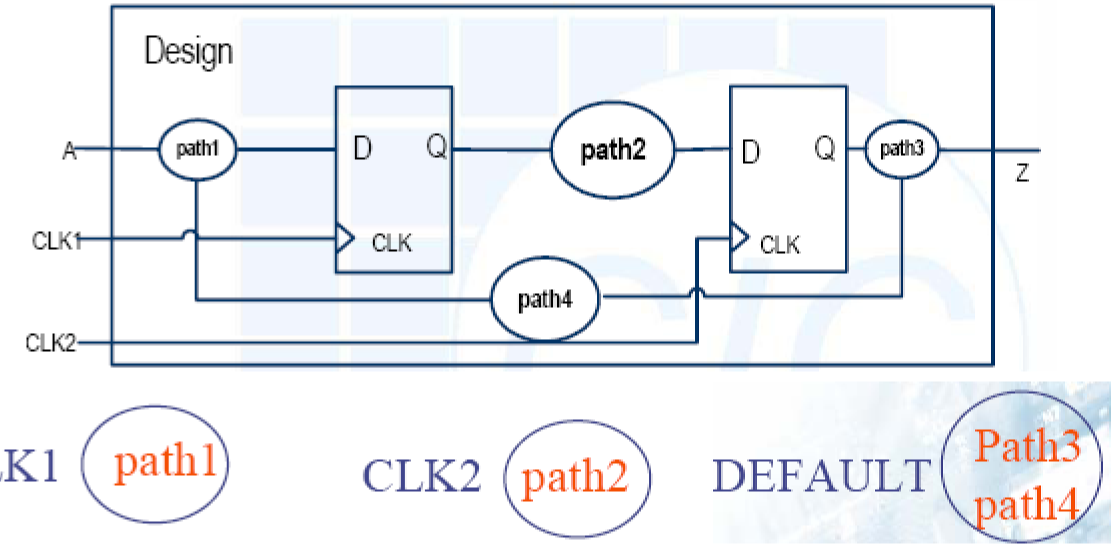
- DesignTime breaks designs into sets of signal paths
- Each path has a startpoint and an endpoint:
  - Startpoints:
    - Input ports
    - Clock pins of Flip-Flops or registers
  - Endpoints:
    - Output ports
    - Data input pins of sequential devices



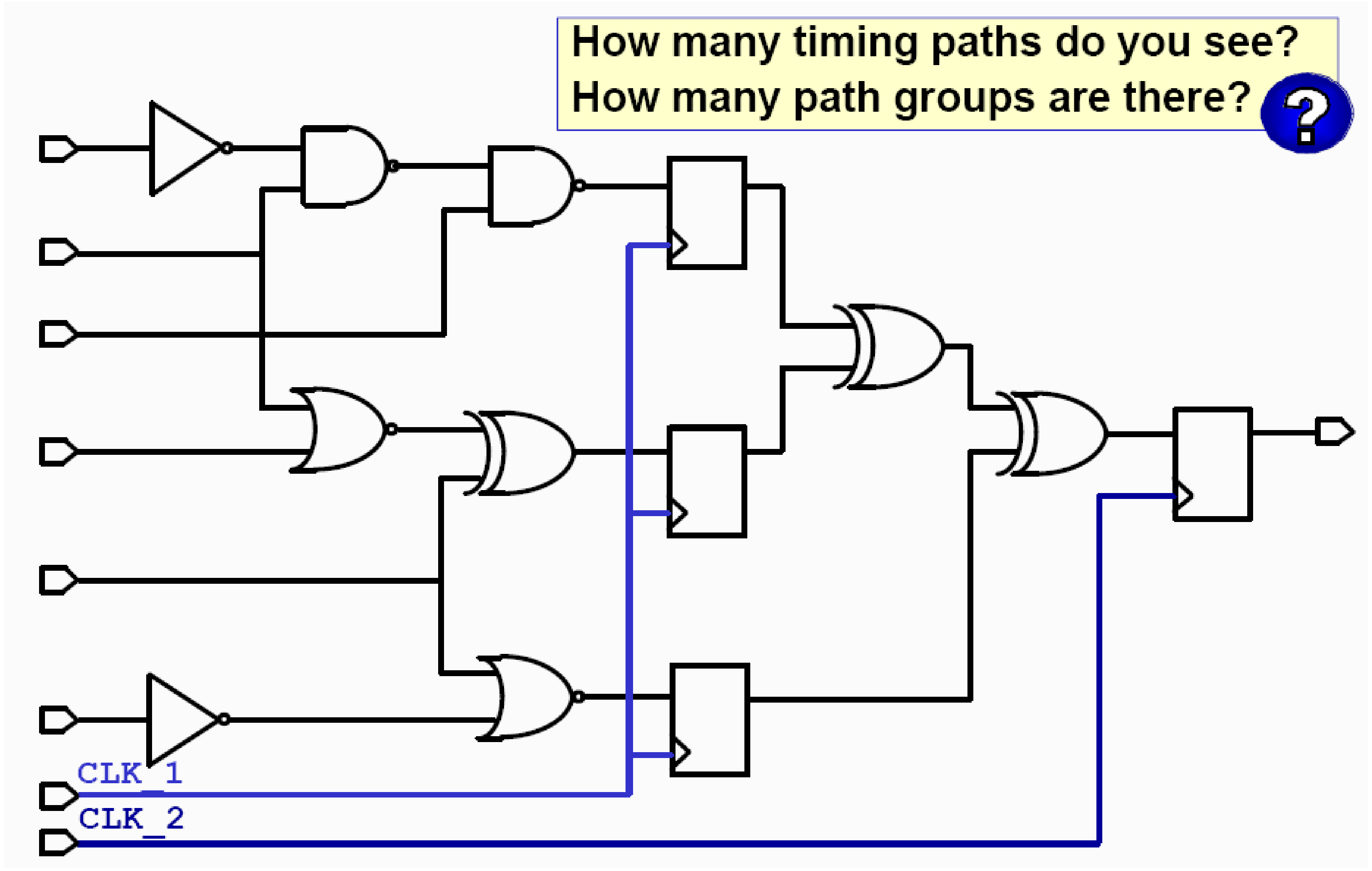
- *Input Paths*
- *Output Paths*
- *Register-to-Register Paths*
- *Combinational Paths*

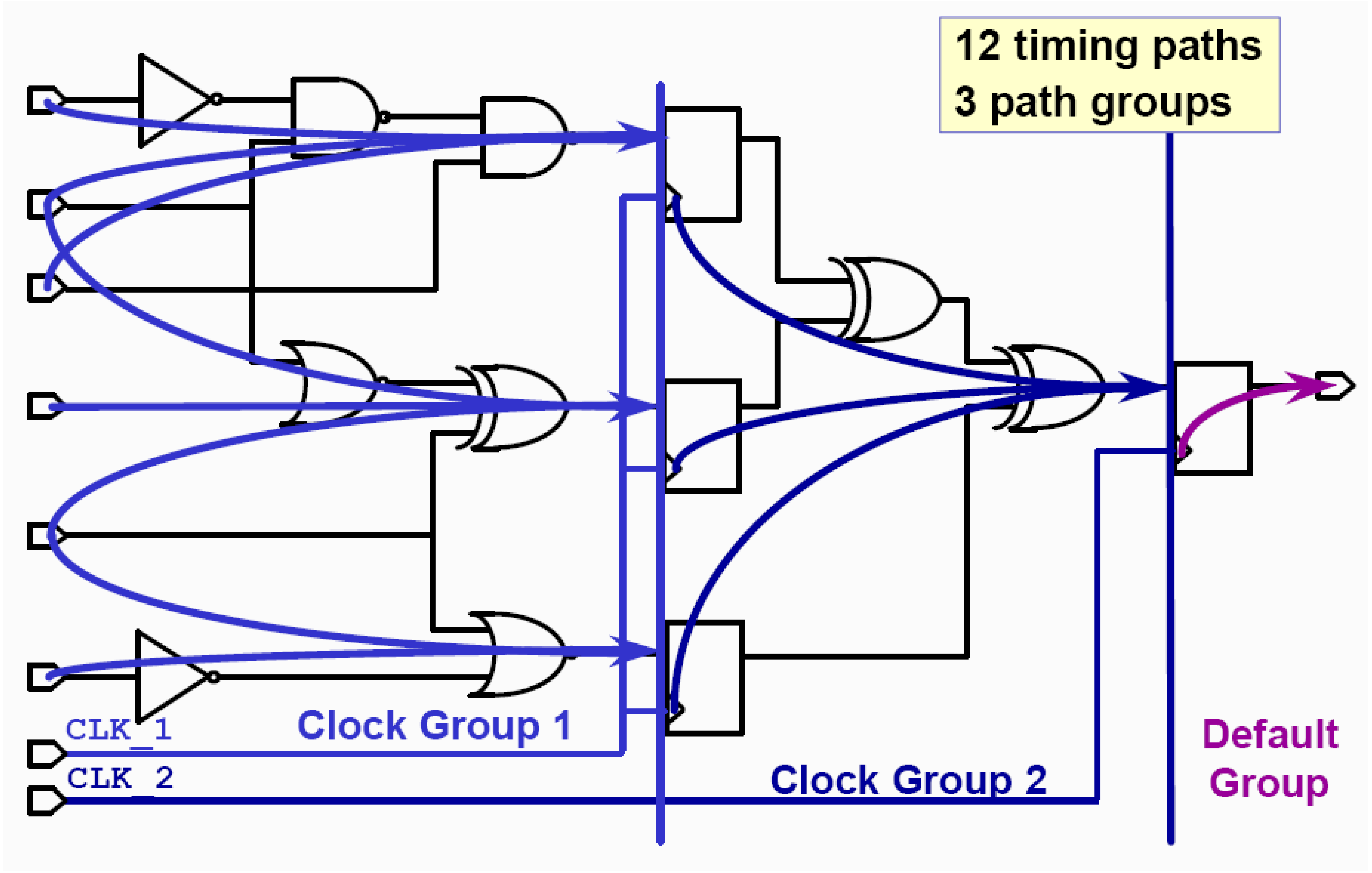
Timing Groups

- *How to organize timing paths into group?*
  - ***Paths are grouped according to the clocks controlling their endpoints.***
  - ***Each clock will be associated with a set paths called a path group.***
  - ***The default path group comprises all paths not associated with a clock.***



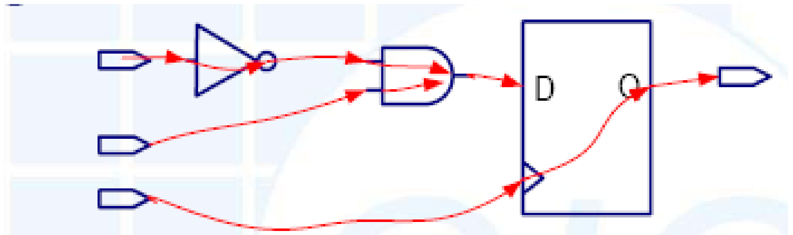
Timing Paths Exercise


How many timing paths do you see?
How many path groups are there?

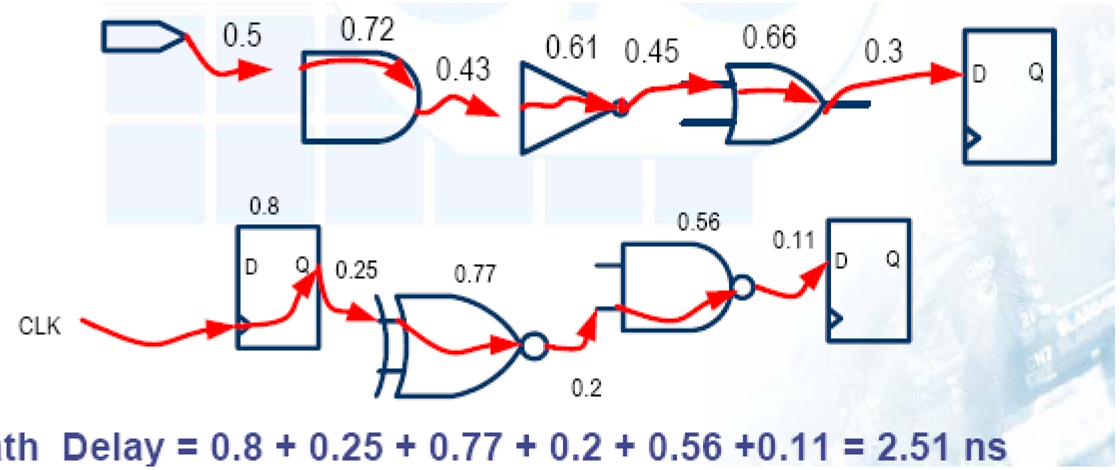

12 timing paths
3 path groups
CLK_1
CLK_2
Clock Group 1
Clock Group 2
Default Group

Path Delay Calculation

- *To calculate total delay, each path is broken into timing arcs.*
- ***Each timing arc contributes either a net delay or cell delay***



- **All the net and cell timing arcs along the path are added together.**



Path Delay = 0.8 + 0.25 + 0.77 + 0.2 + 0.56 +0.11 = 2.51 ns

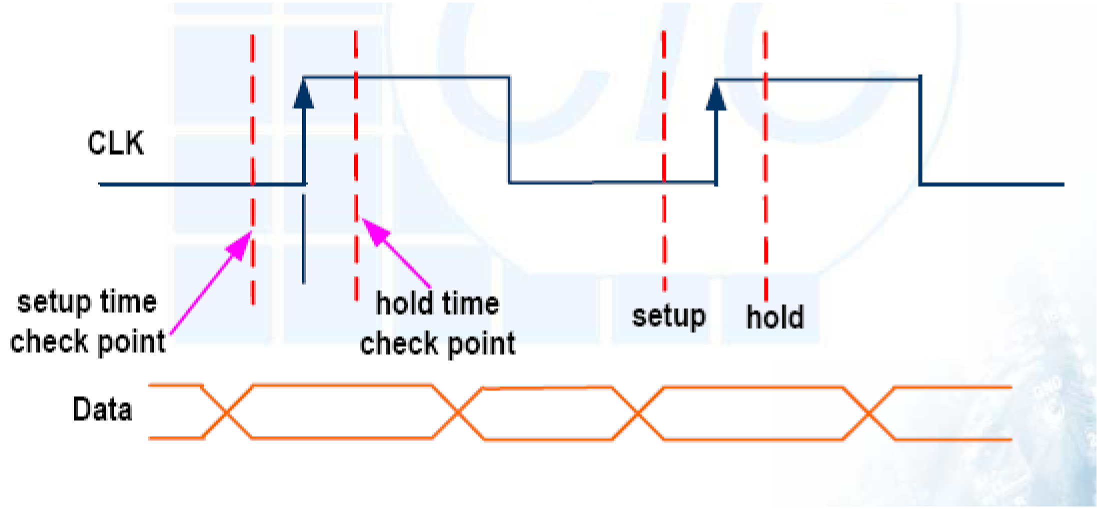
Setup and Hold Time Check

- *Setup Time: The length of time that data must stabilize before the clock transition.*

  The maximum data path is used to determine if setup constraint is met.

- *Hold time: The length of time that data must remain stable at the input pin after the active clock transition.*

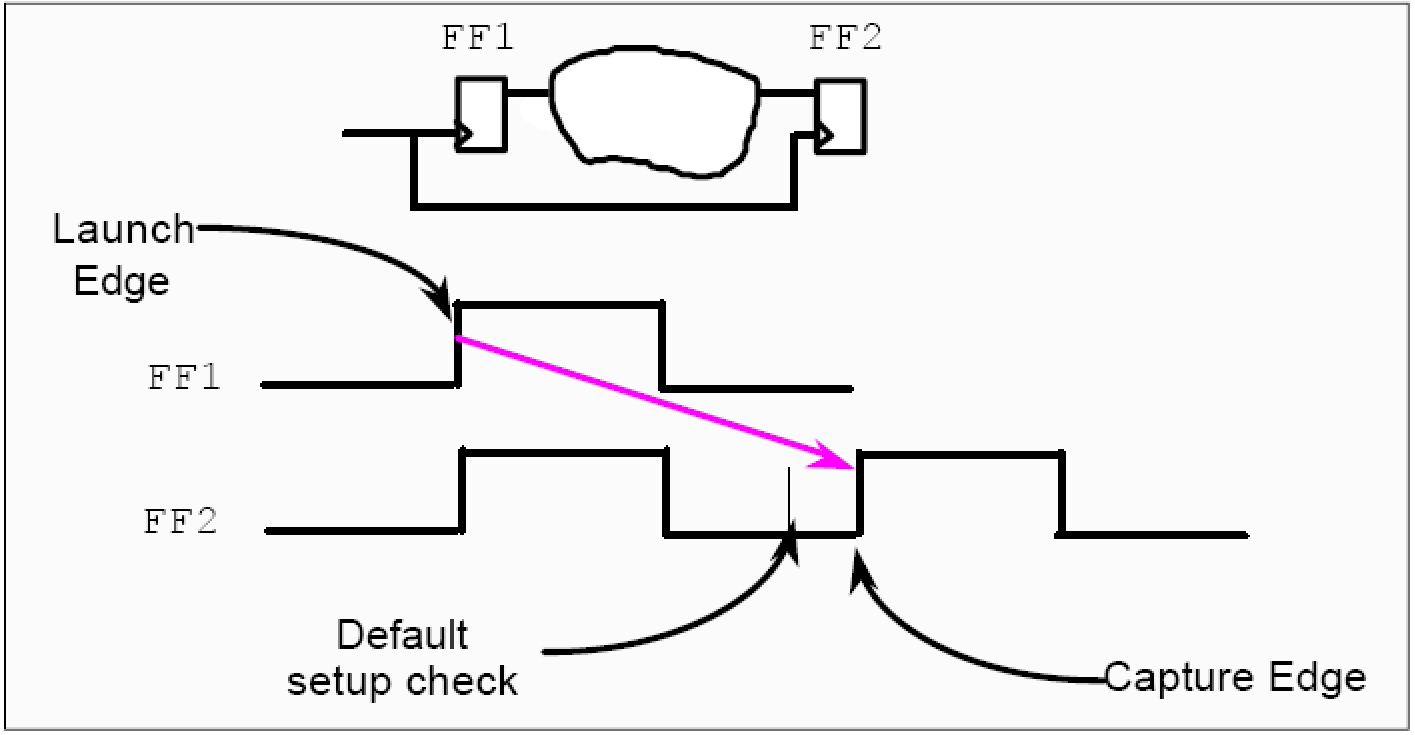
  The minimum data path is used to determine if hold time is met.



The default behavior of Design Compiler is to assume that all data must go from launch to capture edge in one clock cycle.

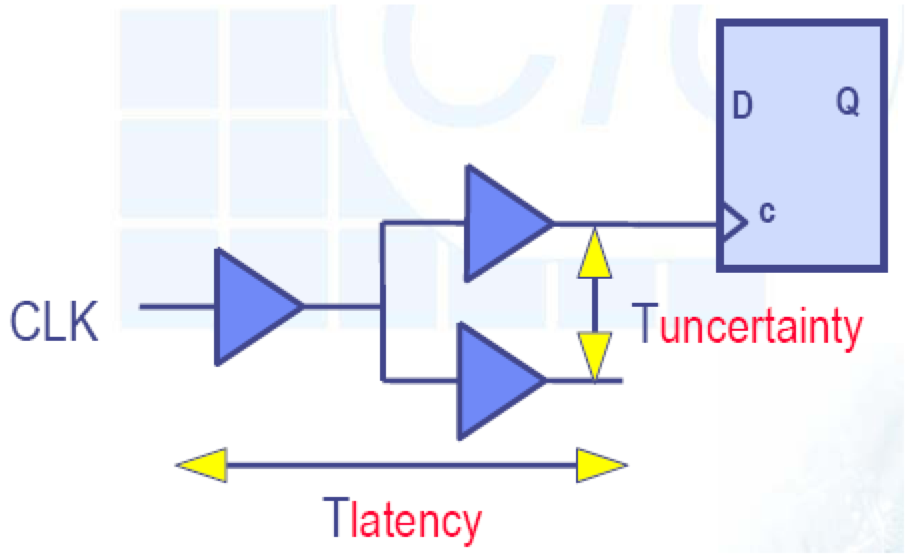
The path between FF1 and FF2 has a max delay constraint of

$$T_{CLK} - FF2_{libSetup}$$



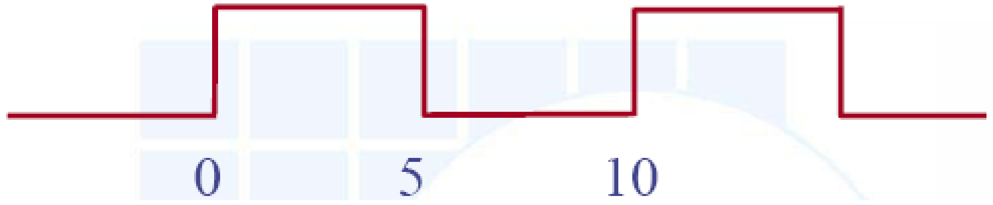
Clock Tree Modeling

- ***Two parameter to model:***
  - ***Specify clock network latency***
    - ***set_clock_latency –rise tr –fall tf [get_ports TCK]***
  - ***Specify uncertainty(skew) of clock network***
    - ***set_clock_uncertainty –rise tp –fall tm [get_ports TCK]***

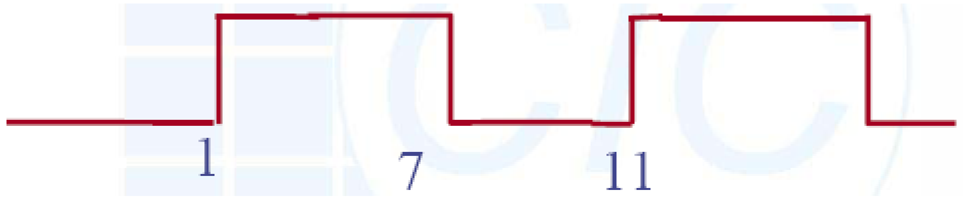


Clock Tree Modeling Example

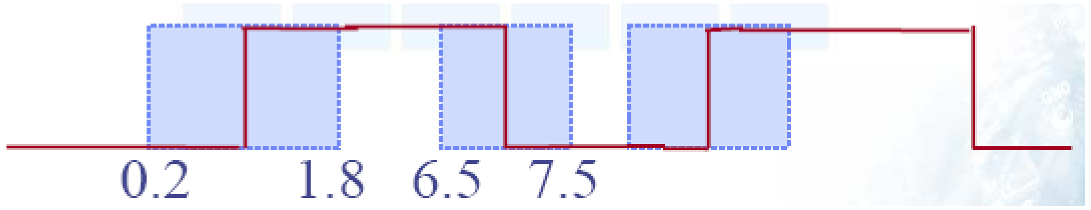
- create_clock –period 10 –waveform {0 5} [get_ports TCK]



- set_clock_latency –rise 1 –fall 2 [get_ports TCK]



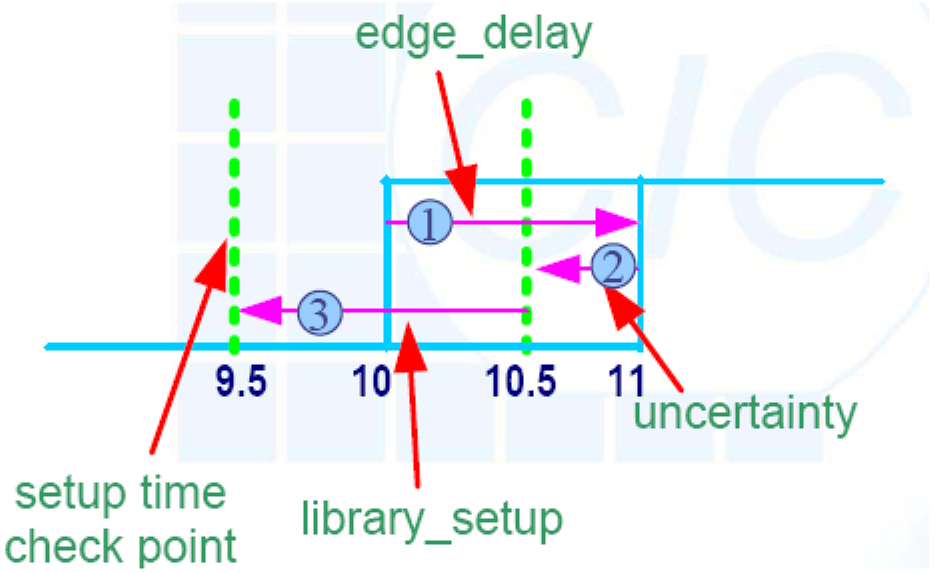
- set_clock_uncertainty –rise 0.8 –fall 0.5 [get_ports TCK]



Effect of Clock Tree Modeling on Setup Time

- *Assumed library (Flip Flop) setup time requirement = 1ns*

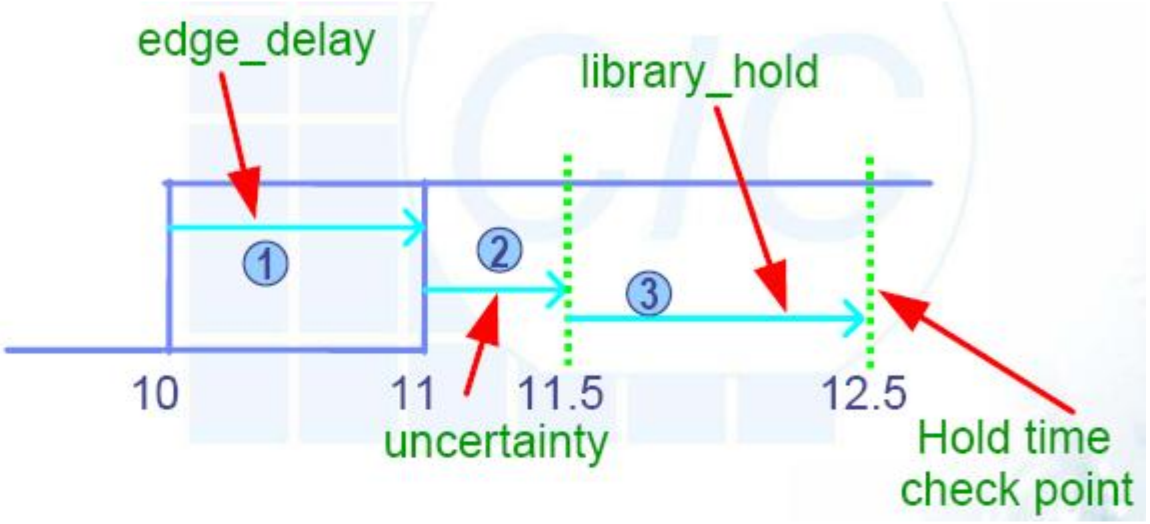
```
create_clock –period 10 –waveform {0 5} [get_ports TCK]
set_clock_latency –rise 1–fall 2 [get_ ports TCK]
set_clock_uncertainty –rise 0.5 –fall 0.7 [get_ ports TCK]
```



- **Setup time check = (clock_edge + edge_delay -uncertainty – lib_setup)**

Effect of Clock Tree Modeling on Hold Time

- *Assumed library (Flip Flop) hold time requirement = 1ns*
  create_clock –period 10 –waveform {0 5} [get_ports TCK]
  set_clock_latency –rise 1 –fall 2 [get_ports TCK]
  set_clock_uncertainty –rise 0.5 –fall 0.8 [get_ports TCK]

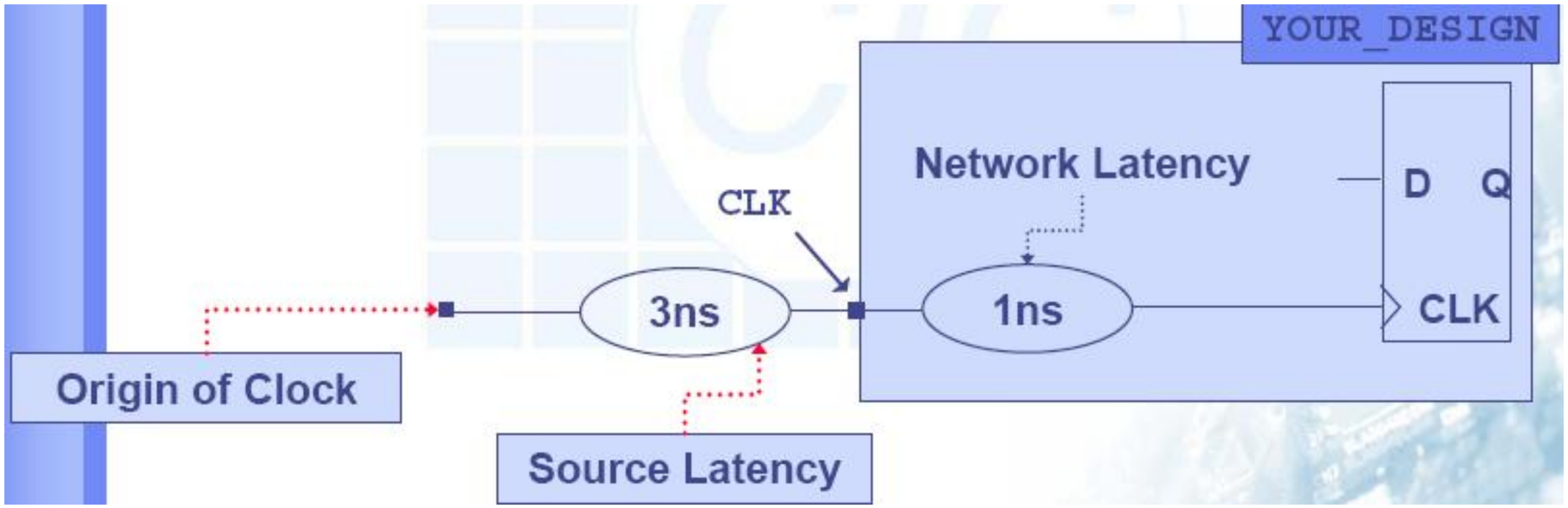


- **Hold time check = (clock_edge + edge_delay +uncertainty + lib_hold)**

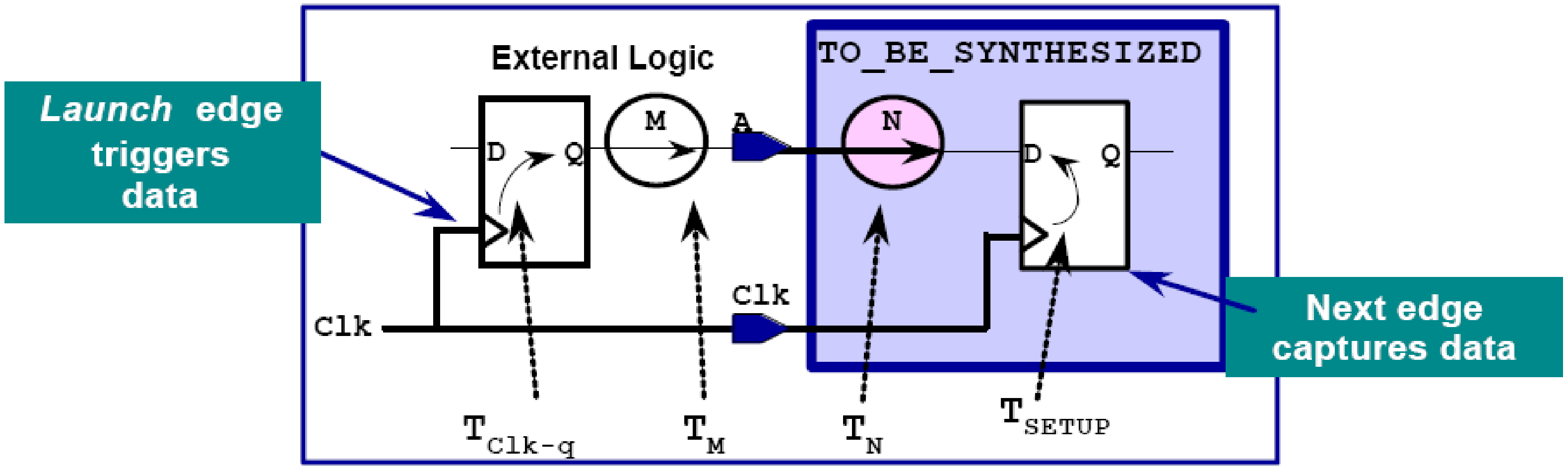
Modeling Source Latency

- *Source latency is the propagation time from the actual clock origin to the clock definition point in the design*

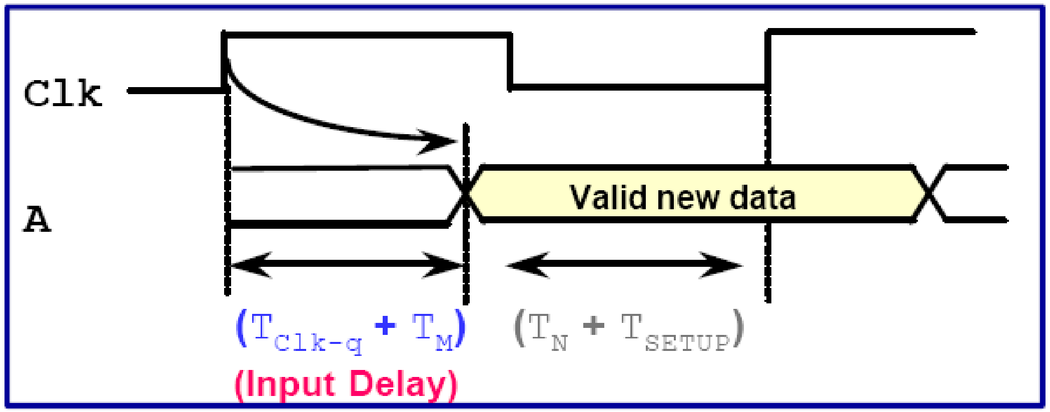
```
create_clock –period 10 –waveform {0 5} [get_ports TCK]
set_clock_latency –source 3 [get_clocks TCK]
set_clock_latency 1 [get_clocks TCK]
```



Constraining the Input Paths



What information must you provide to constrain the input paths?



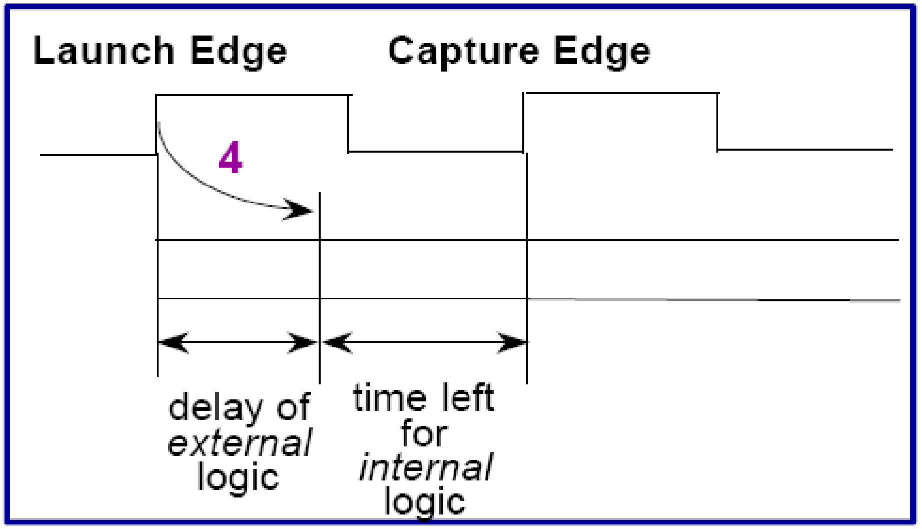
Constraining the Input Paths

```
dc_shell-t> set_input_delay -max 4 -clock Clk [get_ports A]
```

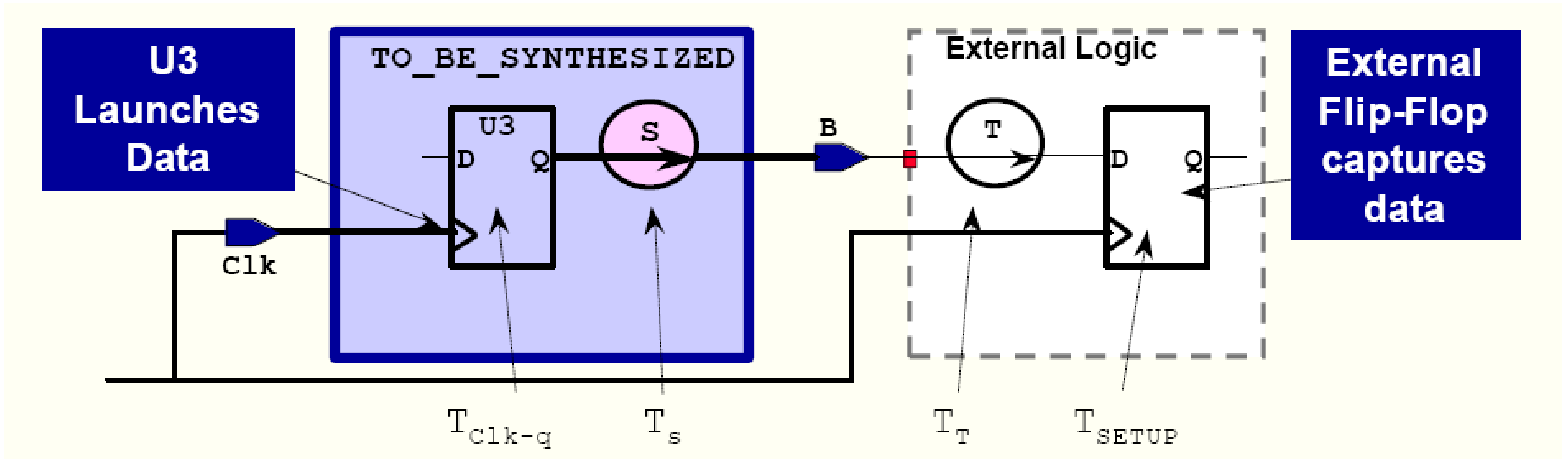
The `set_input_delay` command constrains input paths.

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

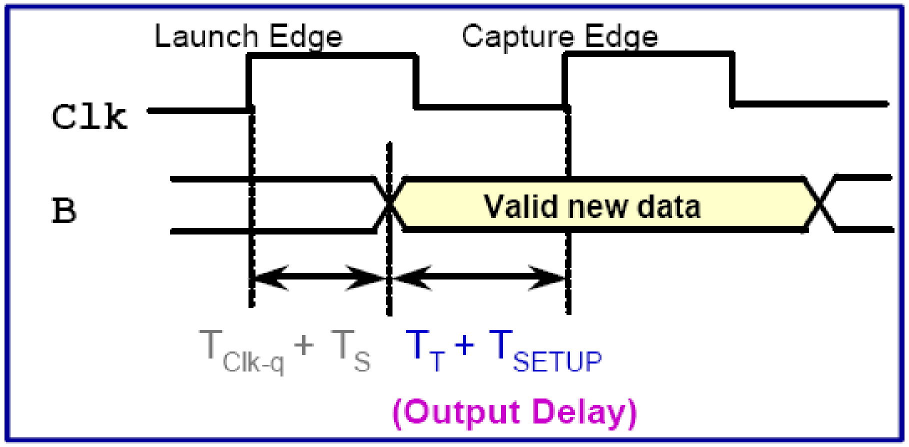
DC calculates how much time is left for the internal logic



Constraining the Output Paths



What information must you provide to constrain the output paths?



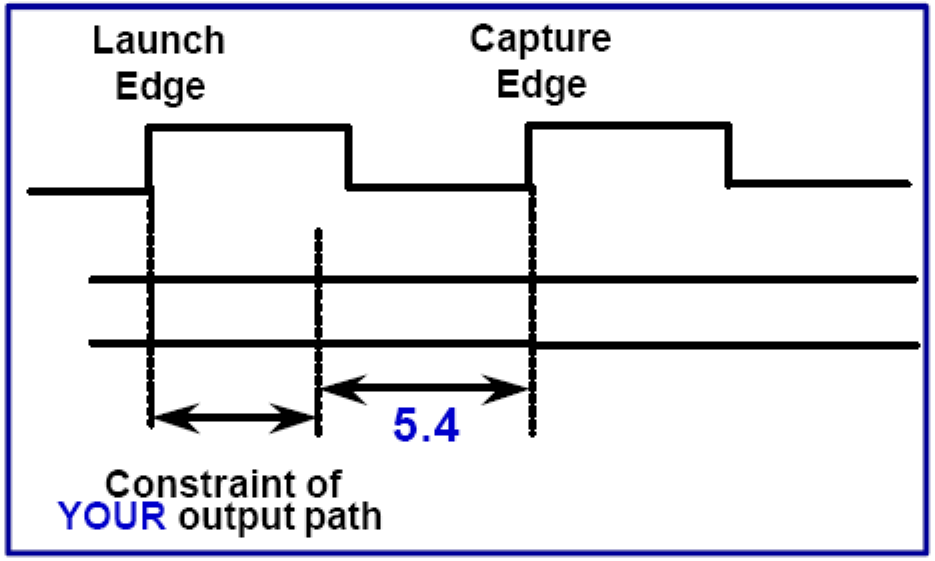
Constraining the Output Paths

```
dc_shell-t> 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



DC Timing Reports

- *Use the report_timing command to access the timing reports.*

```
report_timing
                [ -delay max/min ]
                [ -to name_list ]
                [ -from name_list ]
                [ -through name_list ]
                [ -input_pins ]
                [ -max_paths path_count ]
                [ -nets ]
                [ -capacitance ]
                [ -path full_clock ]
```

How to read timing report?

```
Startpoint: ARM7TDMI/debuger/tms_latch_reg/CKN
           (internal path startpoint clocked by HCLK)
Endpoint: ARM7TDMI/debuger/tms_s_reg
           (falling edge-triggered flip-flop clocked by HCLK)
Path Group: HCLK
Path Type: min

Point                                              Incr       Path
-----------------------------------------------------------------------------
clock HCLK (fall edge)                             6.00       6.00
clock network delay (propagated)                   0.87       6.87
input external delay                               0.00       6.87 f
ARM7TDMI/debuger/tms_latch_reg/CKN (FFDNHD2X)      0.00       6.87 f
ARM7TDMI/debuger/tms_latch_reg/Q (FFDNHD2X) <-     0.18 &     7.05 r
ARM7TDMI/debuger/tms_s_reg/D (FFDNHD2X)            0.00 &     7.05 r
data arrival time                                             7.05

clock HCLK (fall edge)                             6.00       6.00
clock network delay (propagated)                   0.87       6.87
clock uncertainty                                  0.10       6.97
ARM7TDMI/debuger/tms_s_reg/CKN (FFDNHD2X)                     6.97 f
library hold time                                  0.08       7.05
data required time                                            7.05
-----------------------------------------------------------------------------
data required time                                            7.05
data arrival time                                            -7.05
-----------------------------------------------------------------------------
slack (MET)                                                   0.00
```

Timing Report: Path Information Section

```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : TT
Version: 2000.05
Date   : Tue Aug 29 18:22:38 2000
****************************************

Operating Conditions: slow_125_1.62   Library: ssc_core_slow
Wire Load Model Mode: enclosed

  Startpoint: data1 (input port clocked by clk)
  Endpoint: u4 (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  TT                 5KGATES               ssc_core_slow
```

Timing Report: Path Delay Section

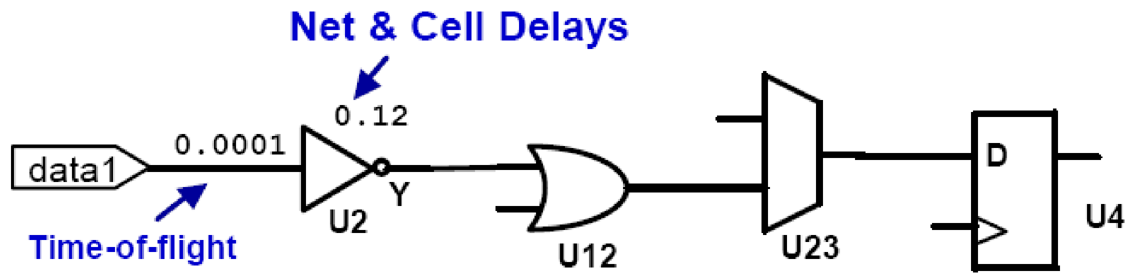
Individual Contribution to Path Delay

Running Total of the Path Delay

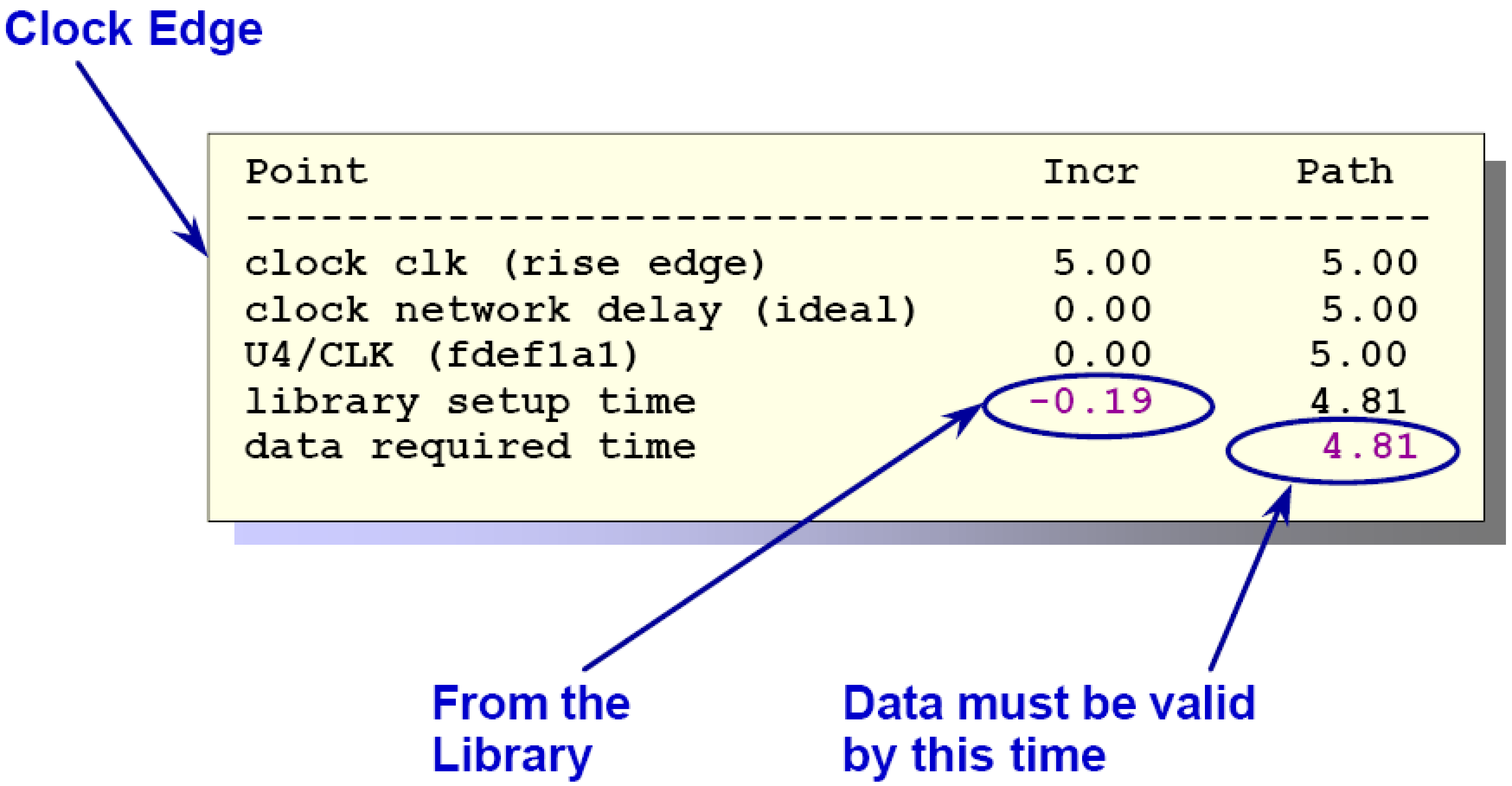
```
Point                                   Incr       Path
---------------------------------------------------------
clock clk (rise edge)                   0.00       0.00
clock network delay (ideal)             0.00       0.00
input external delay                    1.00       1.00 f
data1 (in)                              0.00       1.00 f
u2/Y (inv1a1)                           0.12       1.12 r
u12/Y (or2a1)                           0.26       1.38 r
u23/Y (mx2d2)                           0.23       1.61 f
u4/D (fdef1a1)                          0.00       1.61 f
data arrival time                                  1.61
```

Signal Transition

Total Delay



Timing Report: Path Required Section



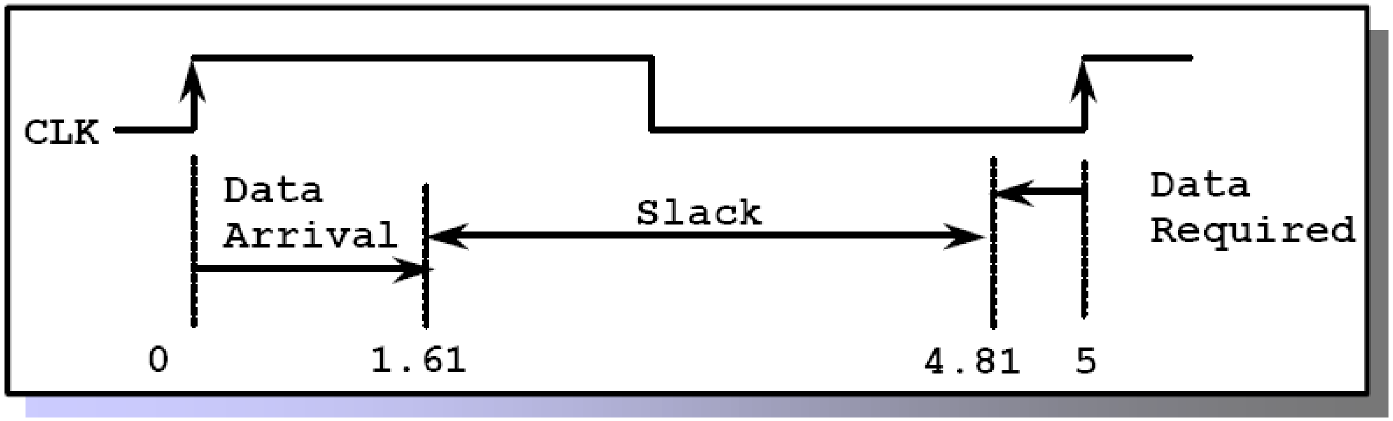
```
Point                                    Incr       Path
----------------------------------------------------------
clock clk (rise edge)                    5.00       5.00
clock network delay (ideal)              0.00       5.00
U4/CLK (fdef1a1)                         0.00      5.00
library setup time                      -0.19      4.81
data required time                                  4.81
```

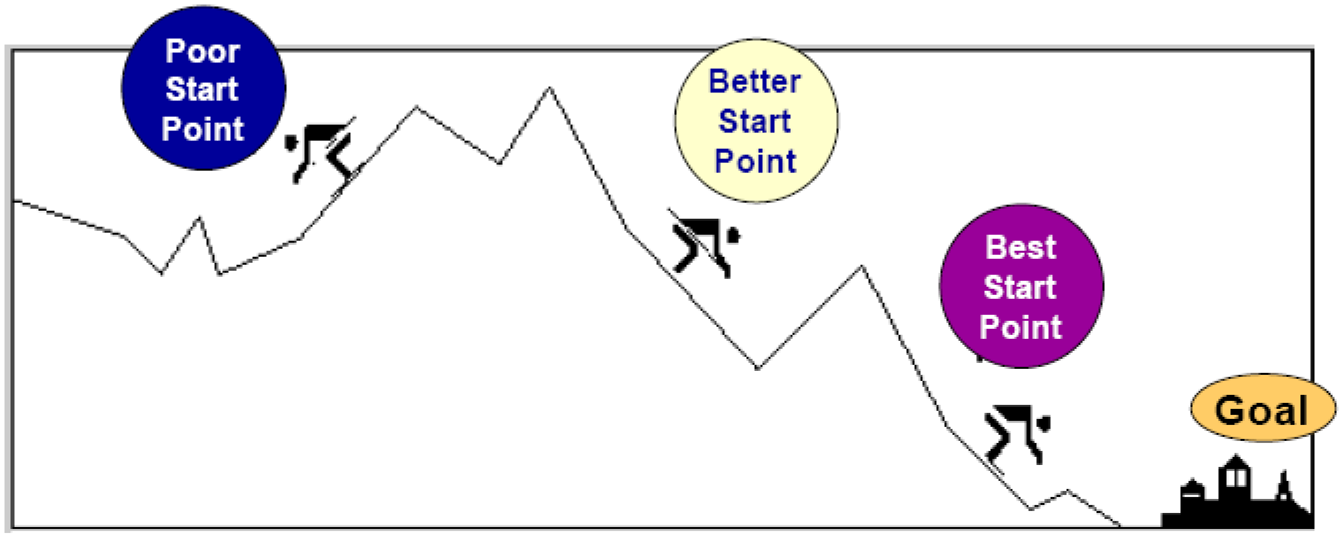
Timing Report: Summary Section

```
---------------------------------------------------
data required time                          4.81
data arrival time                          -1.61
---------------------------------------------------
slack (MET)                                 3.20
```

Either (MET) or (VIOLATED)

Timing margin (slack): negative indicates constraint violation



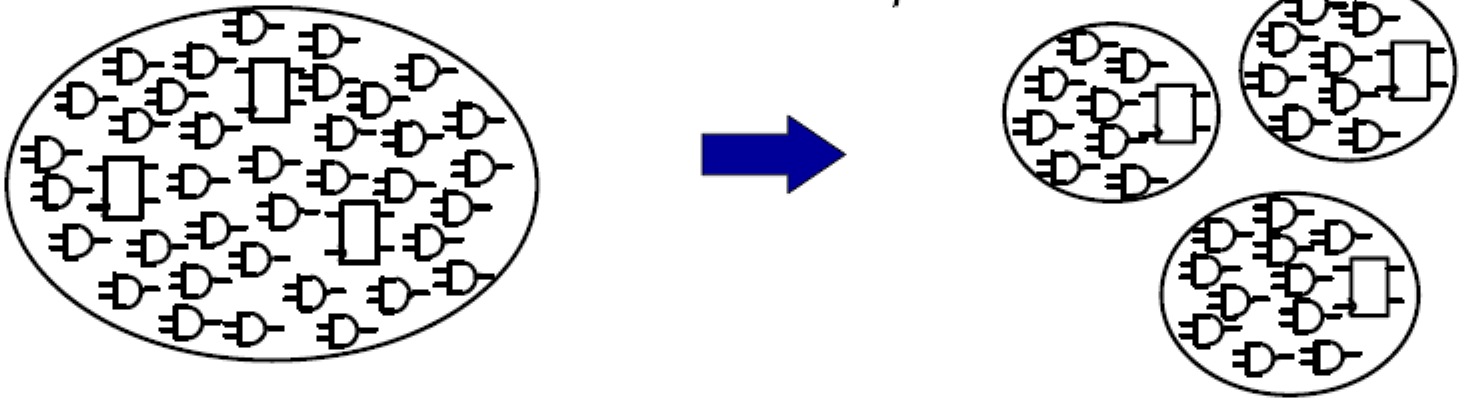


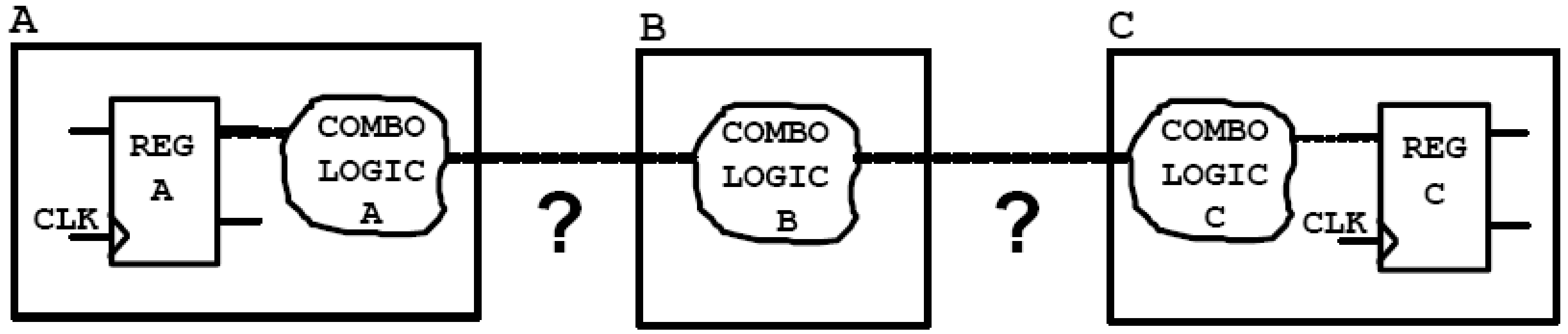
- Codes that are functionally equivalent, but coded differently, will give different synthesis results
- You cannot rely solely on Design Compiler to “fix” a poorly coded design!
- Try to understand the “hardware” you are describing, to give DC the best possible starting point

What is partitioning?

Partitioning is the process of dividing complex designs into smaller parts

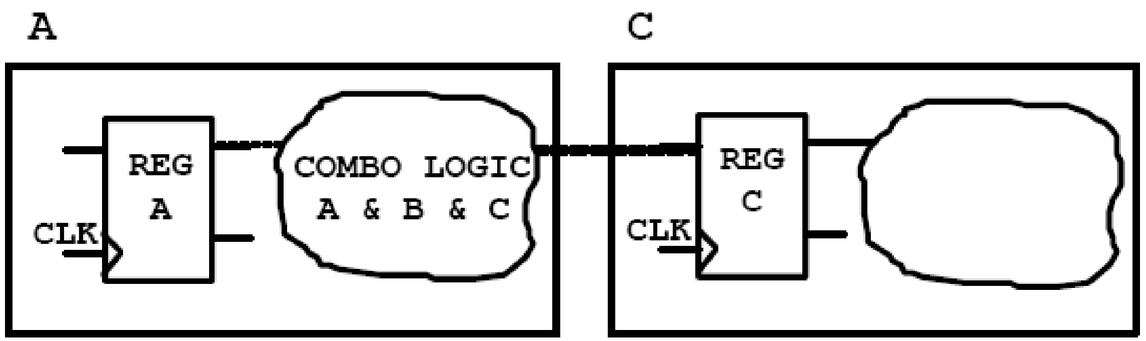
"Divide and conquer!"





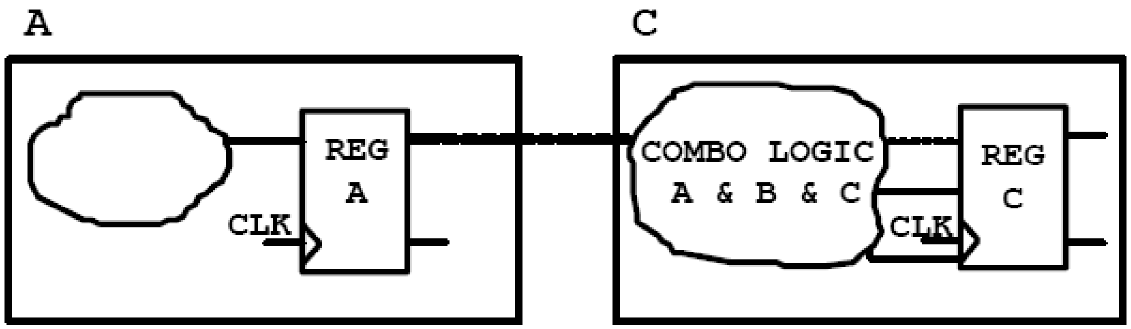
Are these partitions truly needed?

- **Design Compiler must preserve port definitions**
- **Logic optimization does not cross block boundaries**
  - Adjacent blocks of combinational logic cannot be merged
- **Path from REG A to REG C may be larger and slower than necessary!**



Better Partitioning

- 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

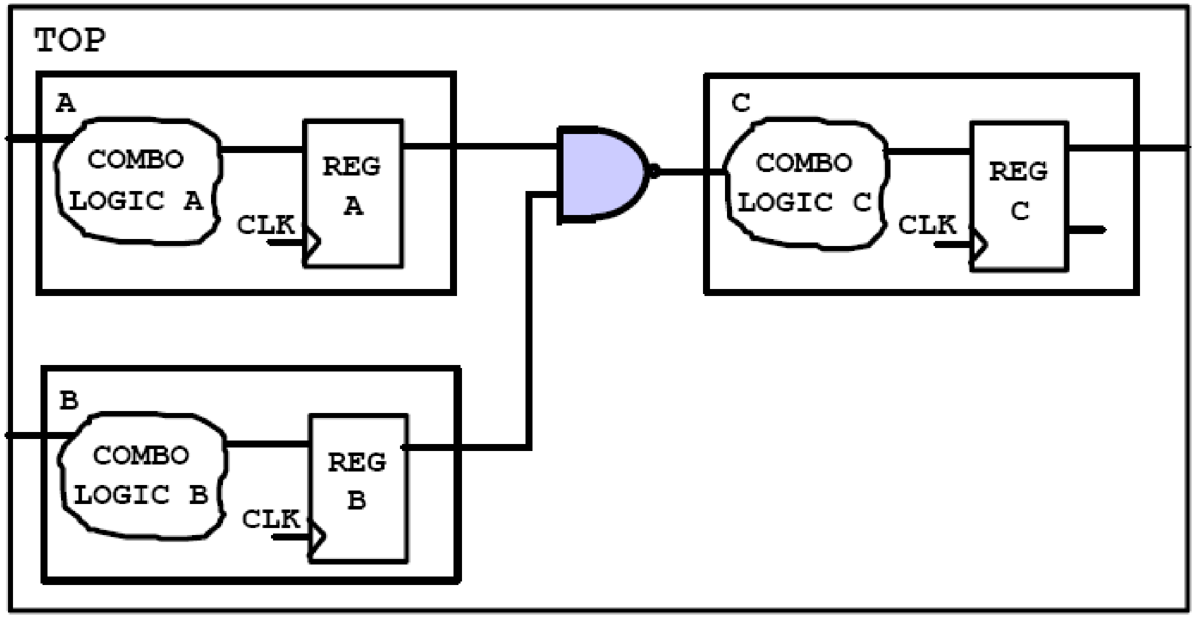


Best Partitioning

- **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, Clock-enabled…)

Avoid Glue Logic

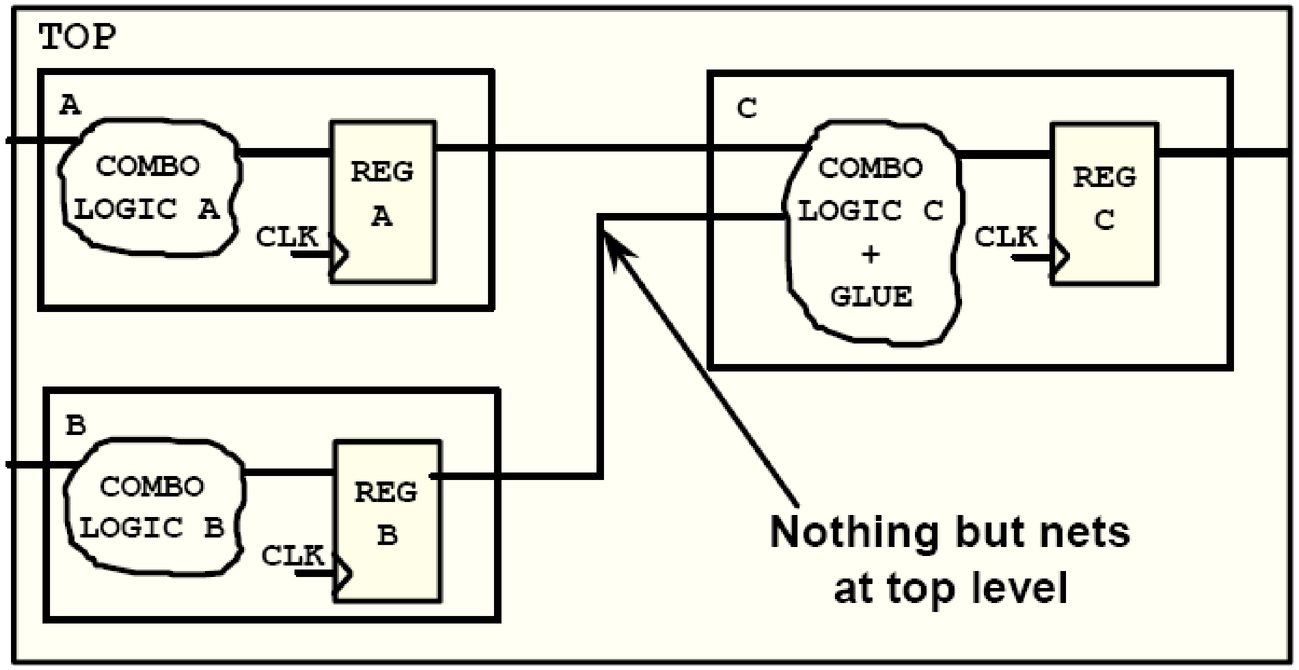


Poor Partitioning

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

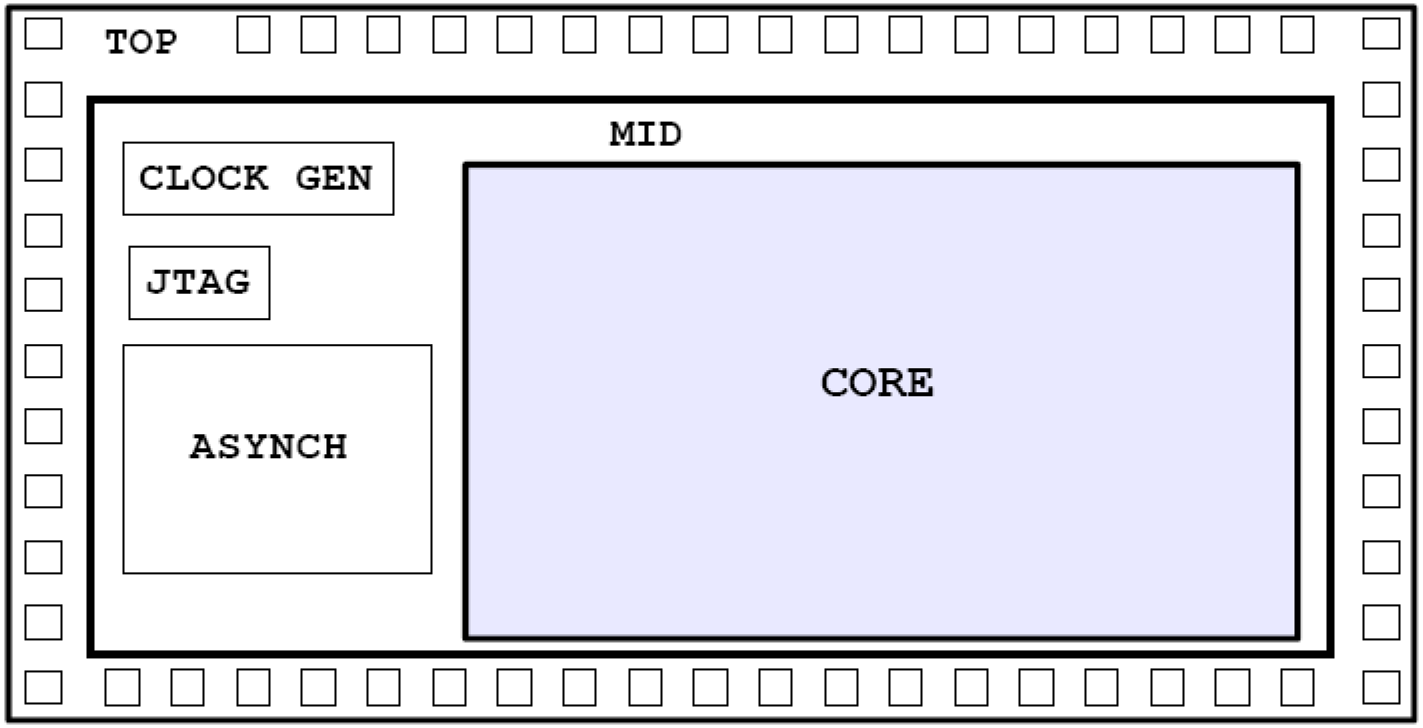


Good Partitioning

- The glue logic can now be optimized with other logic
- Top-level design is only a structural netlist, it does not need to be compiled

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

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



This partitioning is recommended due to:

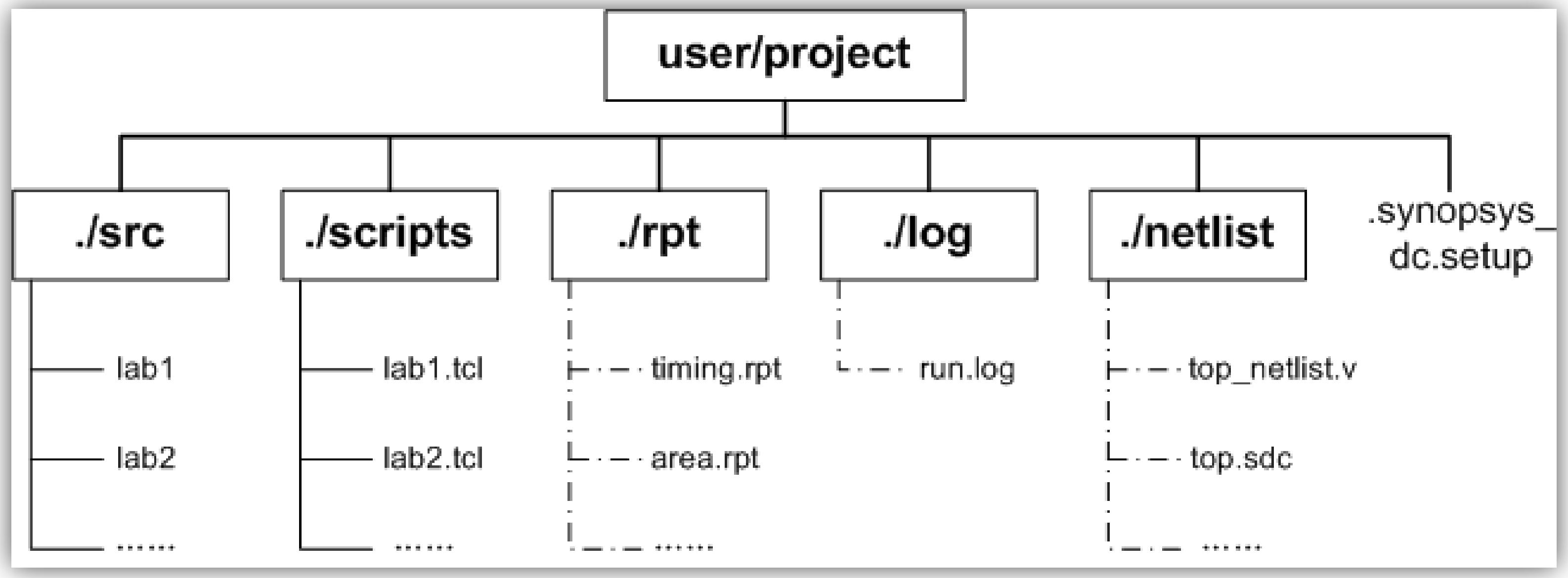
- ***Possible technology-dependent ("black box") I/O pad cells***
- ***Possible untestable "Divide By" clock generation***
- ***Possible technology-dependent JTAG circuitry***

Q&A!

How to start DC

```
1 unix/linux> dc_shell-t
  dc_shell> source scripts/xx.tcl -e -v

2 unix/linux> dc_shell –f scripts/xx.tcl (|tee ./log/run.log)

3 unix/linux> design_vision
  design_vision> source scripts/xx.tcl -e -v
```

Project Directory



How Get Help?

For example:

- *dc_shell> man set_input_delay*
- *dc_shell> set_input_delay -help*