Chip Synthesis

Wang Jinxiang

Microelectronics Center of HIT

Agenda

1. Introduction to Synthesis
2. Partitioning for Synthesis
3. Coding for Synthesis
4. Timing and Area
5. Environmental Attributes
6. Time and Load Budgeting
7. Timing Analysis
8. Compiling a Hierarchical Design
9. Compiling a Large Design
10. Conclusion

What is "Synthesis"?

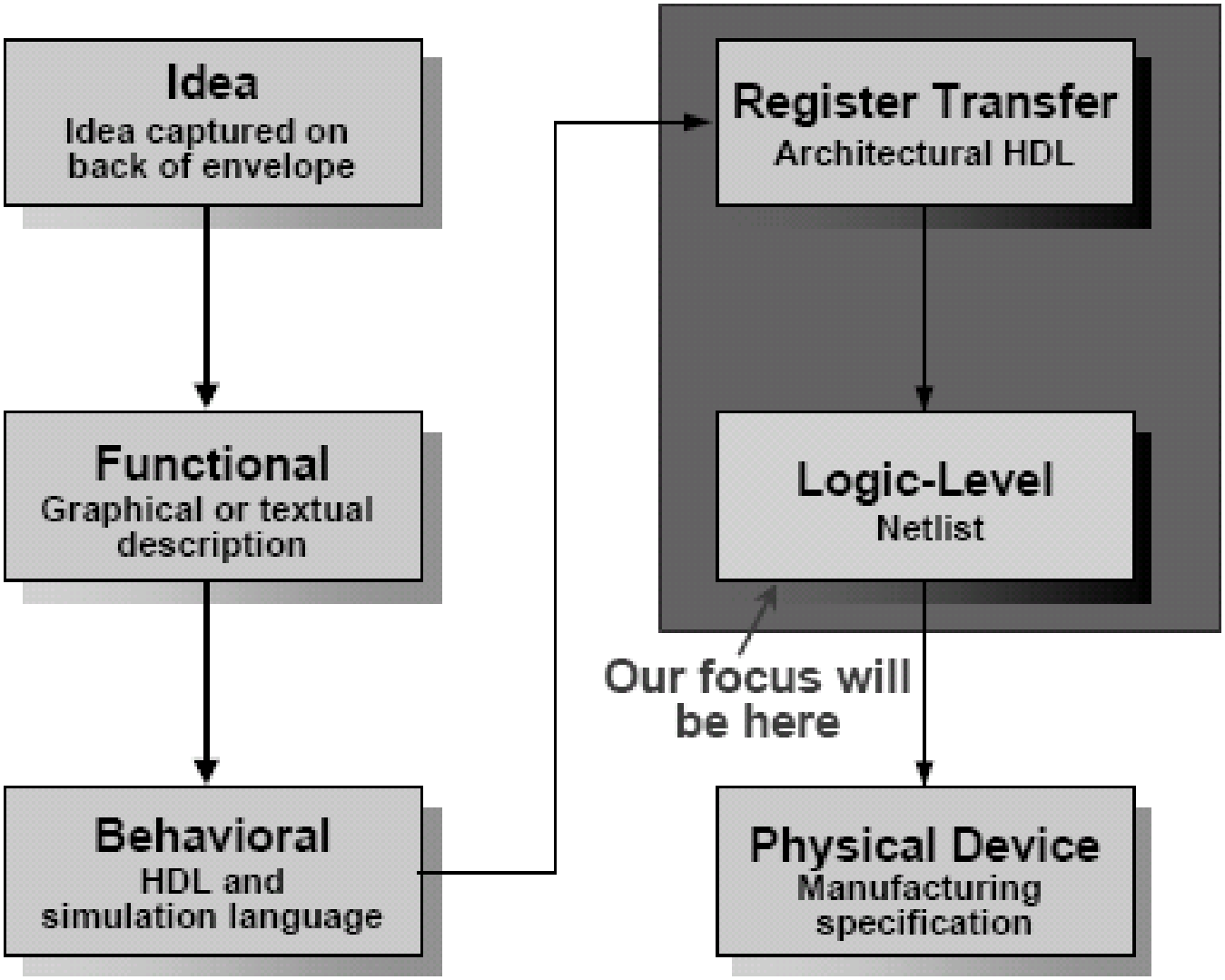
Synthesis is the ***transformation*** of an idea into a manufacturable device to carry out an intended function

- Designers have synthesized circuits for years--

  -by hand

- Transformation from abstract to concrete

Levels of Abstraction



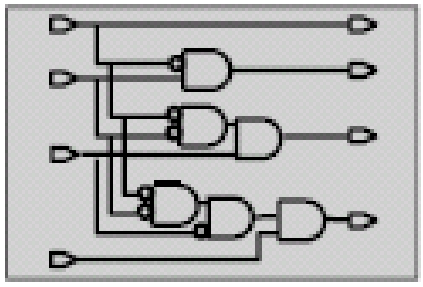
For EDA, Synthesis is …

Synthesis = Translation + Optimization + Mapping

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

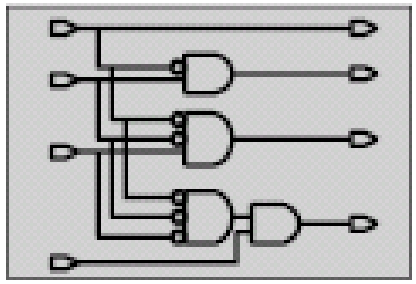
HDL Source

Translate



Generic Boolean
(GTECH)

Optimize + Map



Target Technology

Why Synthesis?

Reusability
Parameterized code; Building-block approach; Retarget new libraries;

Design Tricks
DC knows plenty, tries them in context of loads, fanouts, library limitations

Verifiable
Validate, implement, & verify in same language, Less error-prone entry

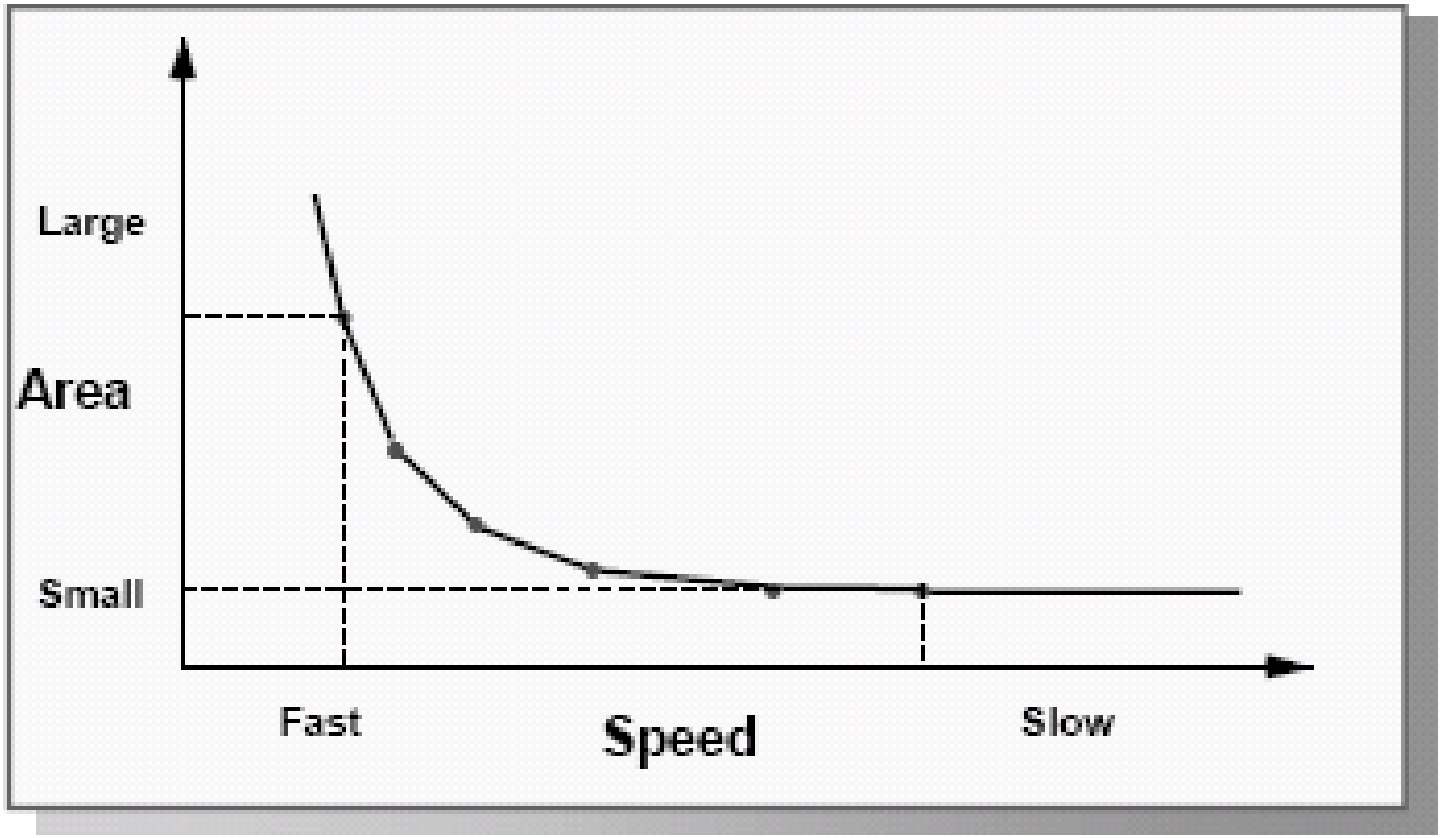
Abstraction
Focus on high-level issues; tool & computer do dirty work to meet constraints

Portability
IEEE standards; HDL portable across tools; technology-independent designs

Productivity
How else to do 10^6 gates in 6 months?

Prestige
Impress friends; hot skill on resume; job security; wealth and fame

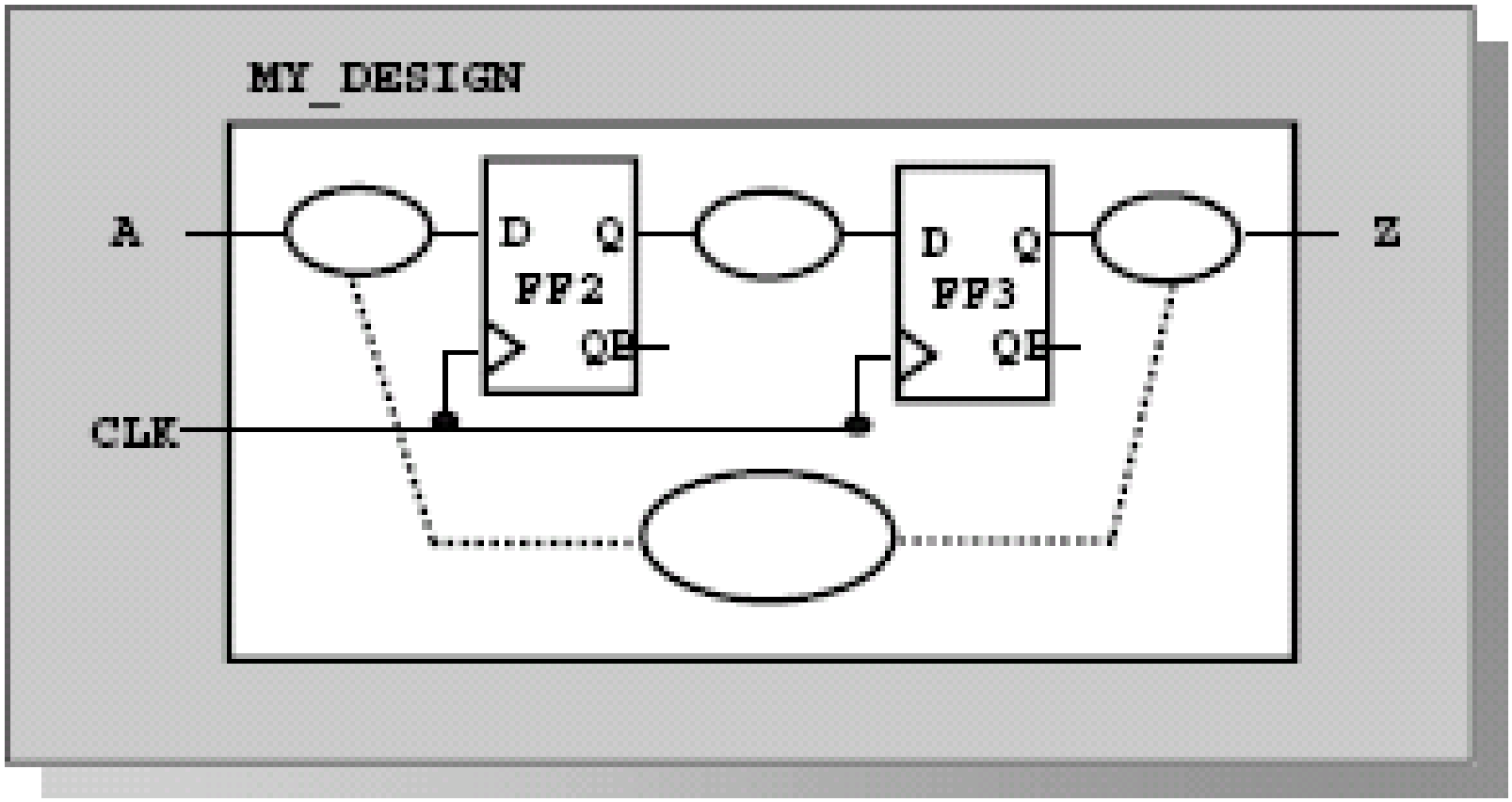
Synthesis is Constraint-Driven



You set the goals(through constraints)

Design Compiler optimizes the design to meet your goals

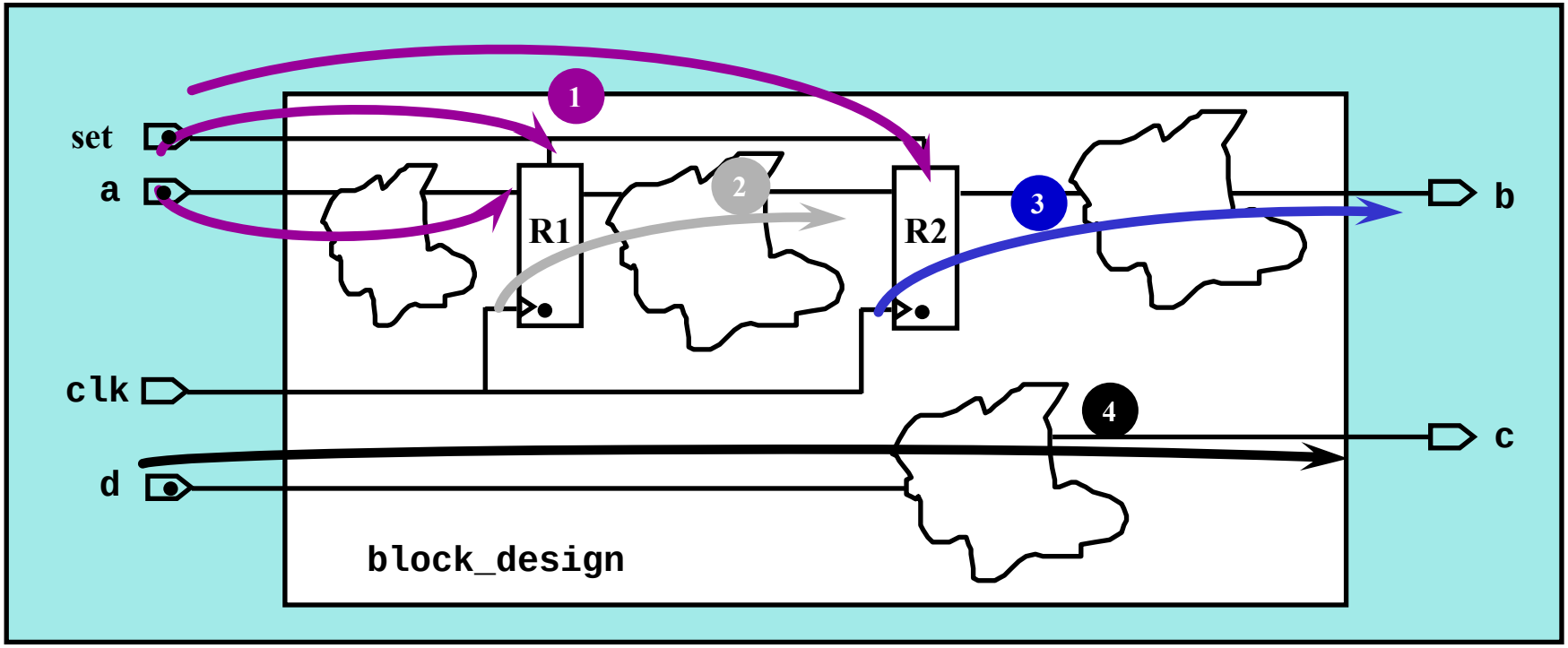
Synthesis is Path-based



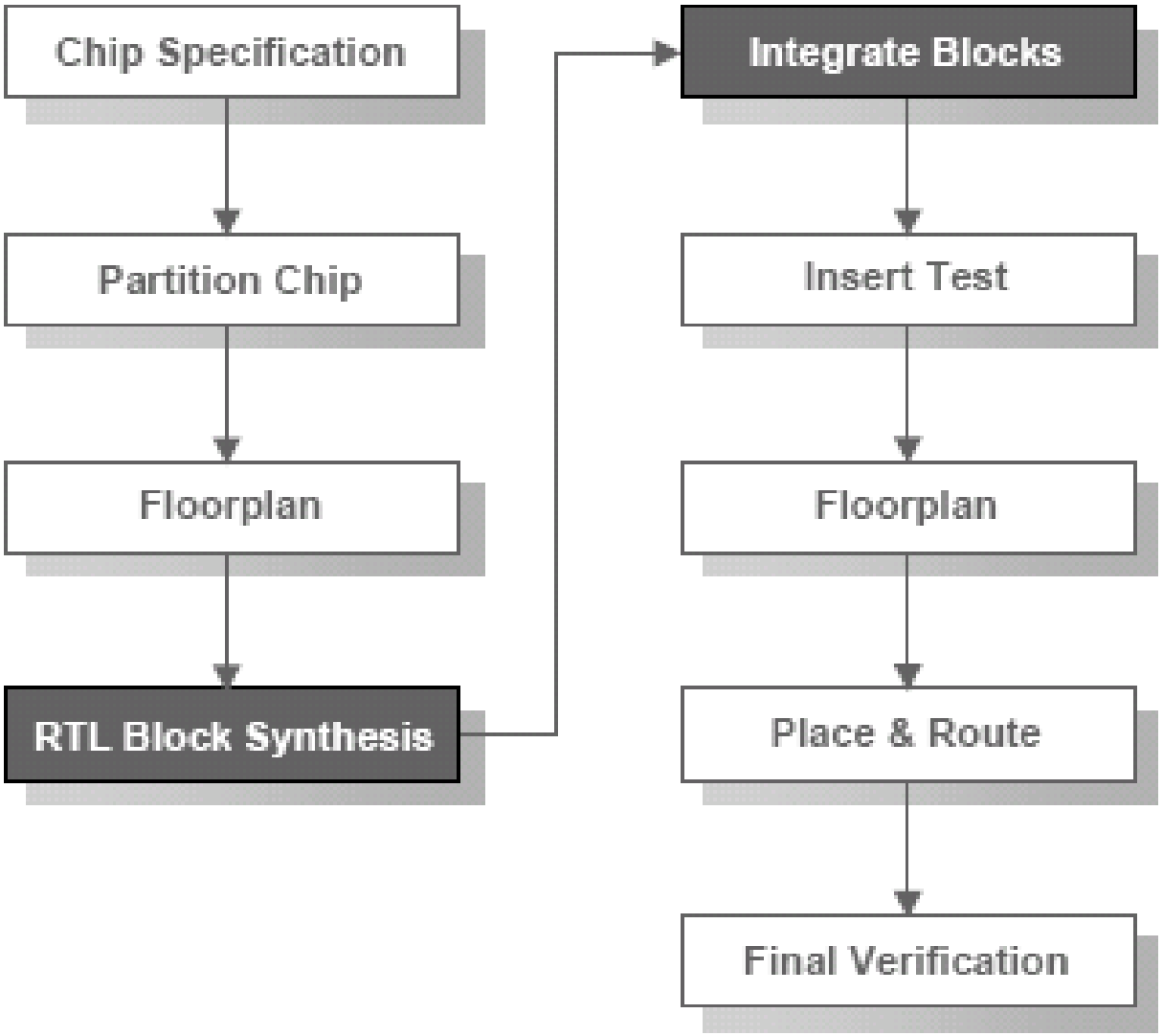
How many timing paths do you see in MY_DESIGNS?

Four Type of Timing Paths

1. Input ports to registers
2. Register to register
3. Register to output ports
4. Input ports to output ports



Chip Synthesis Process



Tools for Chip Synthesis

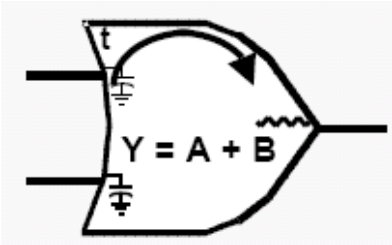
Task	Tool
Chip Specification	
Draw Block Diagram / Partition Chip / Floorplan Top Level	Chip Architect
Code Each Block • Simulate • Check Functional Circuit ◆ Fix Problems -> Recode	VSS, Cyclone, VCS, Vera RTL Analyzer
Synthesize Blocks	Design Compiler
Check Each Block For Testability	Test Compiler
Integrate Blocks to Build Chip Hierarchy -> Check Timing -> Re-optimize	Design Compiler

Tools for Chip Synthesis – cont.

Task	Tool
Check Chip For Testability	Design Compiler
Insert Test Structures	Test Compiler, TestGen
Insert I/O Pads -> Check Timing -> Re-optimize	Design Compiler, DesignTime & PrimeTime
Floorplan Chip -> Back Annotate -> Check Constraints -> Reoptimize	Floorplan Manager
Place and Route (Layout) Chip -> Back Annotate -> Check -> Reoptimize	Floorplan Manager
Run ATPG -> Fault Coverage	Test Compiler, TestGen
Perform Final Verification (throughout) • Timing or Simulation • Functional	PrimeTime, VSS, VCS, Vera Formality

Technology Library

- A technology library contains
  - a set of primitive cells which be used by DC to build a circuit
    - The timing and electrical characteristics of these cells
    - Net delay and net parasitic information
    - Definition of capacitance, time and resistance units
- Technology libraries are created by vendor



Environment Setting

- Target Library

  set target_library {smic_018.db}

- Link Library

  set link_library {* smic_018.db dw_foundation.sldb}

- Synthetic Library

  set synthetic_library { dw_foundation.sldb}

- Symbol Library

  set symbol_library { smic018.sdb}

- Search Path

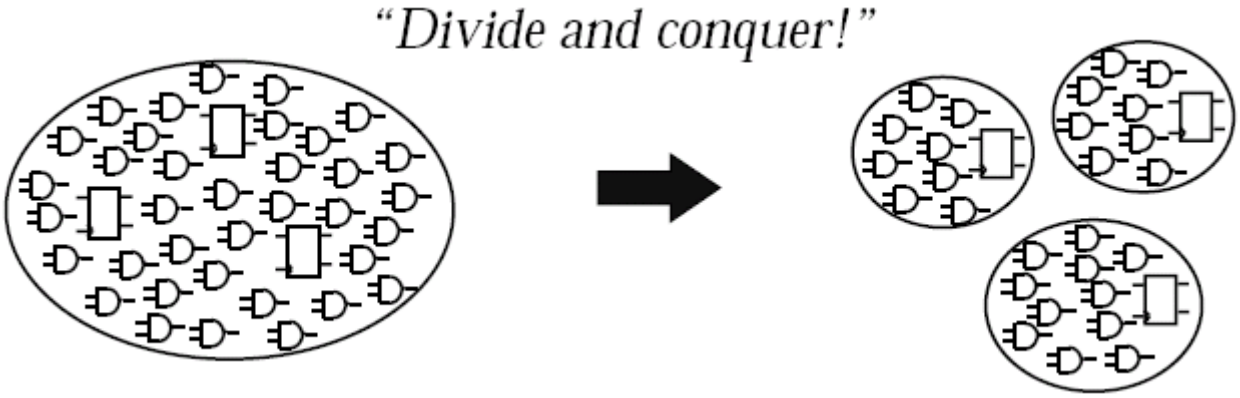
  set search_path [list . ${smic_std_cell}]

Agenda

1. Introduction to Synthesis
2. Partitioning for Synthesis
3. Coding for Synthesis
4. Timing and Area
5. Environmental Attributes
6. Time and Load Budgeting
7. Timing Analysis
8. Compiling a Hierarchical Design
9. Compiling a Large Design
10. Conclusion

What is 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 by DC

Why Partition a Design?

Partitioning is driven by many needs:

- Separate distinct functions
- Achieve workable size and complexity
- Manage project in team environment
- Design Reuse
- Meet physical constraints
- And many, many others….

We will focus on partitioning for synthesis

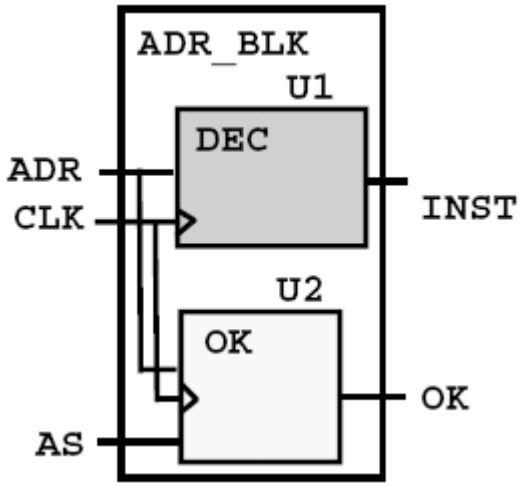
Partitioning for Synthesis

What do we gain by “partitioning for synthesis”:

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

Partitioning within the HDL Description

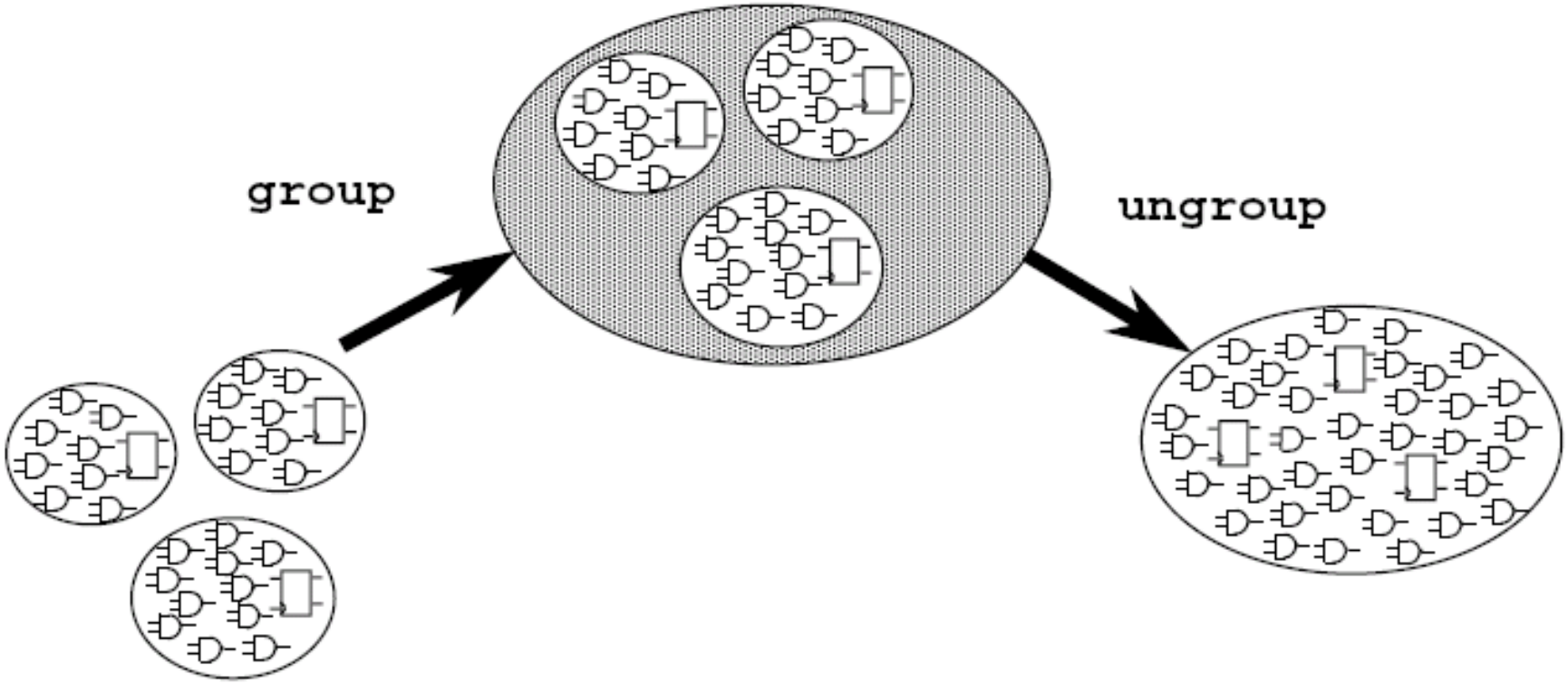
```
module ADR_BLK (…);
   DEC U1 (ADR, CLK, INST);
    OK U2 (ADR, CLK, AS, OK1);
endmodule
```



- entity and module statement define hierarchical blocks
- Inference of Arithmetic Circuits (+, -, *, …) can create a new level of hierarchy
- process and always statements do not create hierarchy

Partitioning within DC

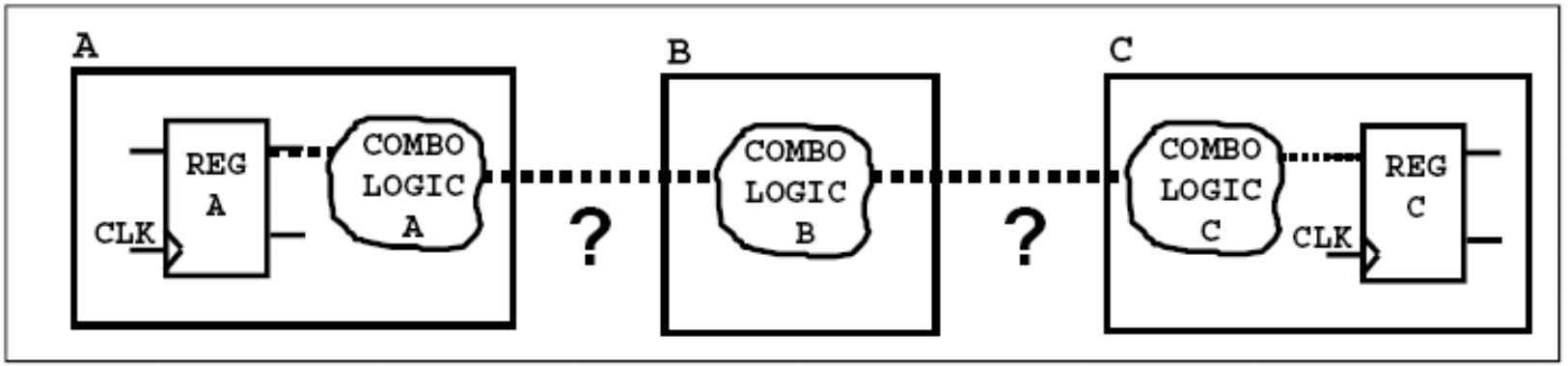
The group and ungroup commands modify the partitions in a design



Partitioning Strategies for Synthesis

- No combinational path crossing hierarchy boundaries
- Place hierarchy boundaries at register outputs
- Limit block size for reasonable runtimes
- Separate core logic, pads, clocks, and JTAG

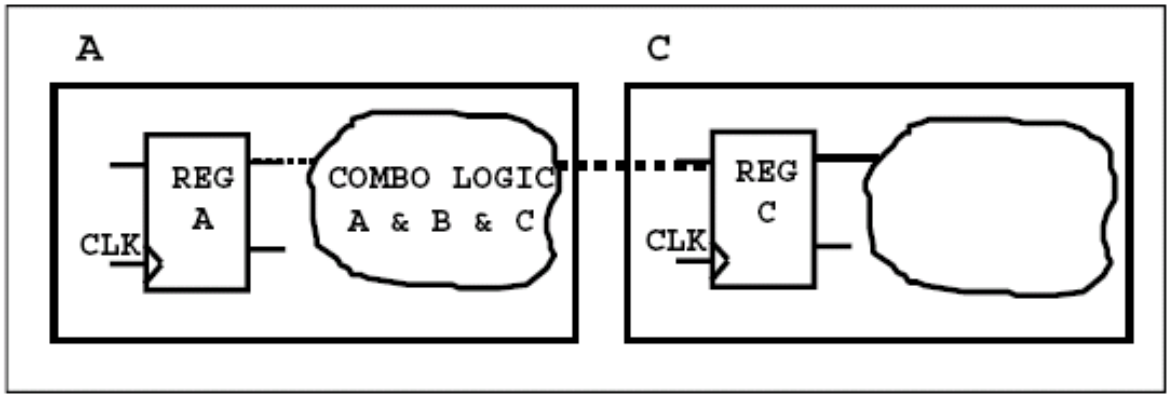
Eliminate unnecessary Hierarchy



Are These Partitions Truly needed?

- Design Compiler must preserve port definitions
- Logic optimization cannot 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!

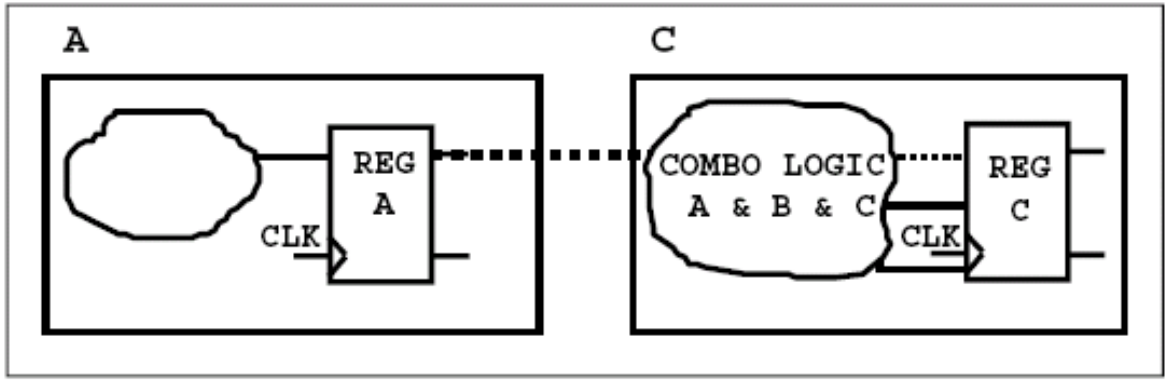
No Hierarchy Dividing Combinational Paths



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

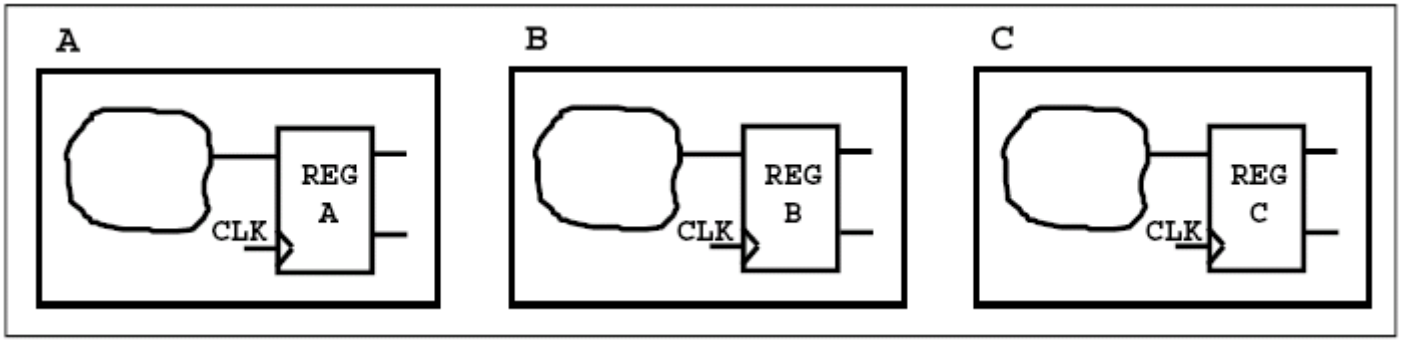
No Hierarchy Dividing Combinational Paths cont.



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 filp-flop: JK, T, Muxed, Clock-enabled …

Partition at Register Boundaries

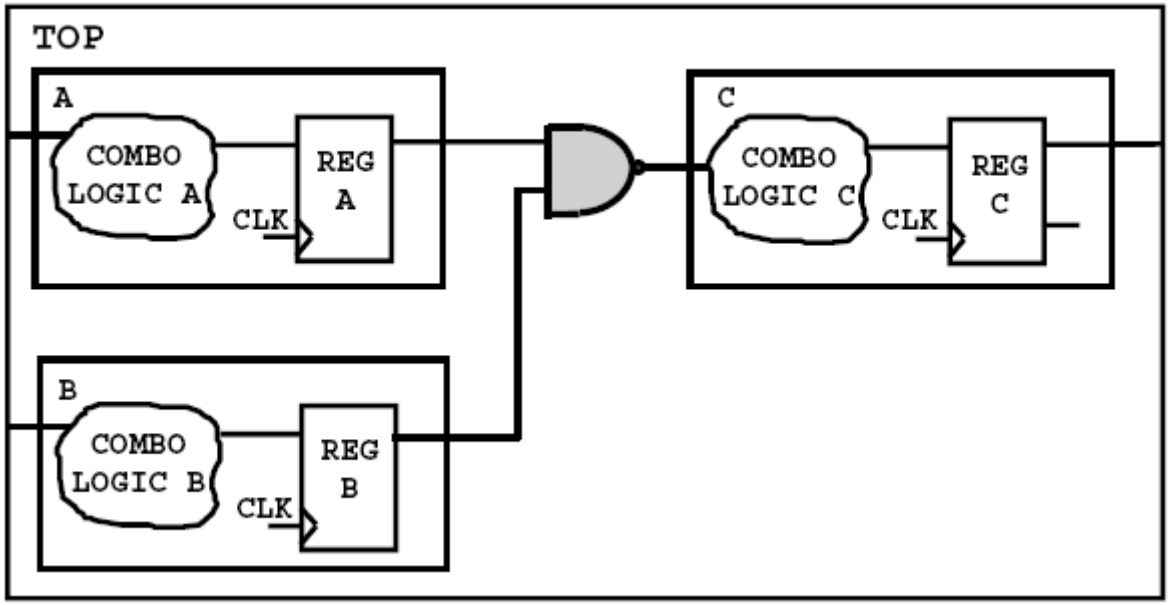


Good Partitioning

- Try to design so hierarchy boundaries follow register outputs

- Simplifies timing constraints:
  - all inputs to each block arrive with the same relative delay

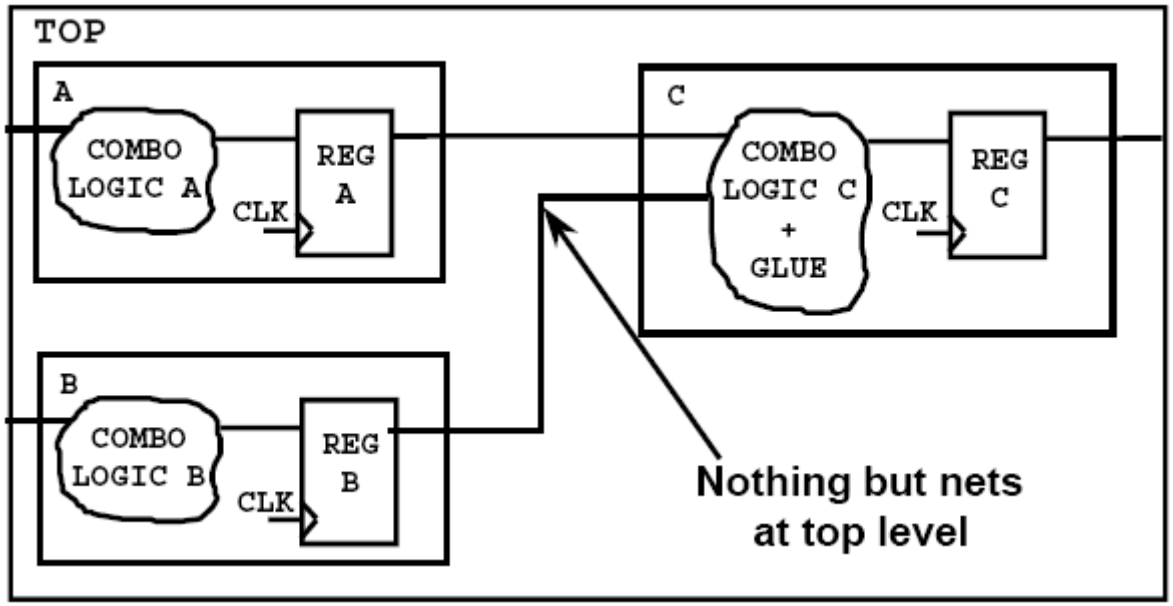
Avoid Glue Logic



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"

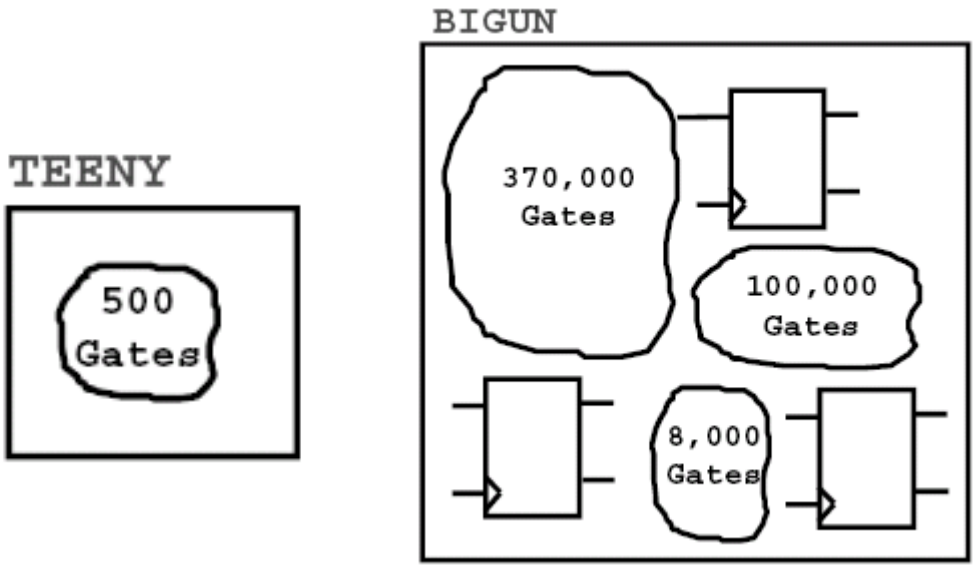
Remove Glue Logic Between Block



Good Partitioning

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

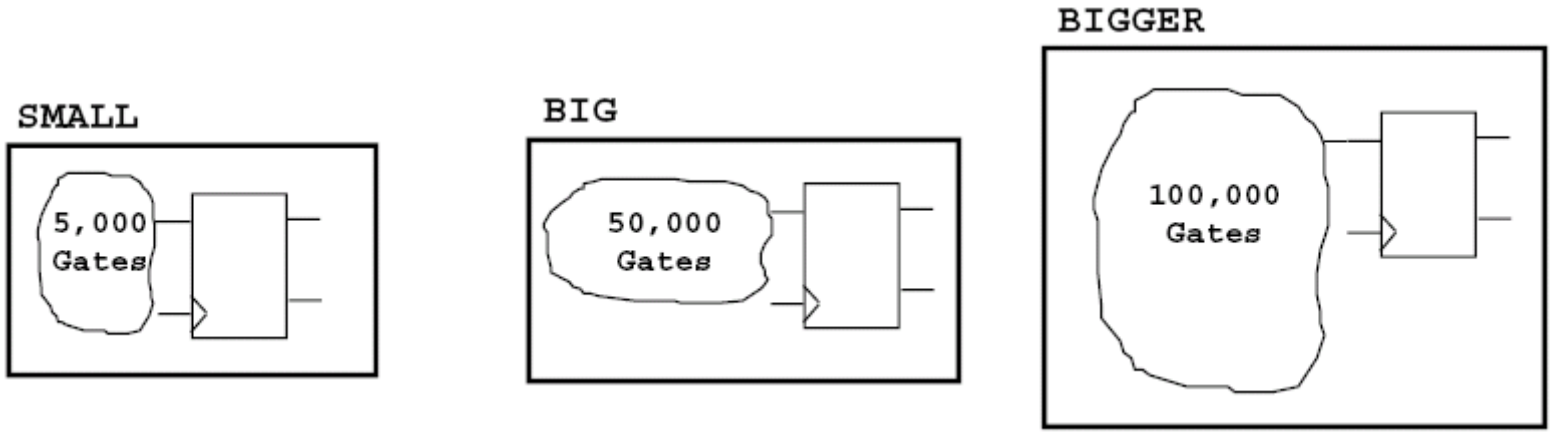
Balance Block Size With Run Times



Poor Partitioning

- If blocks are too small, the designer may be restricting optimization with artificial boundaries
- If blocks are too big, compile run times can be very long

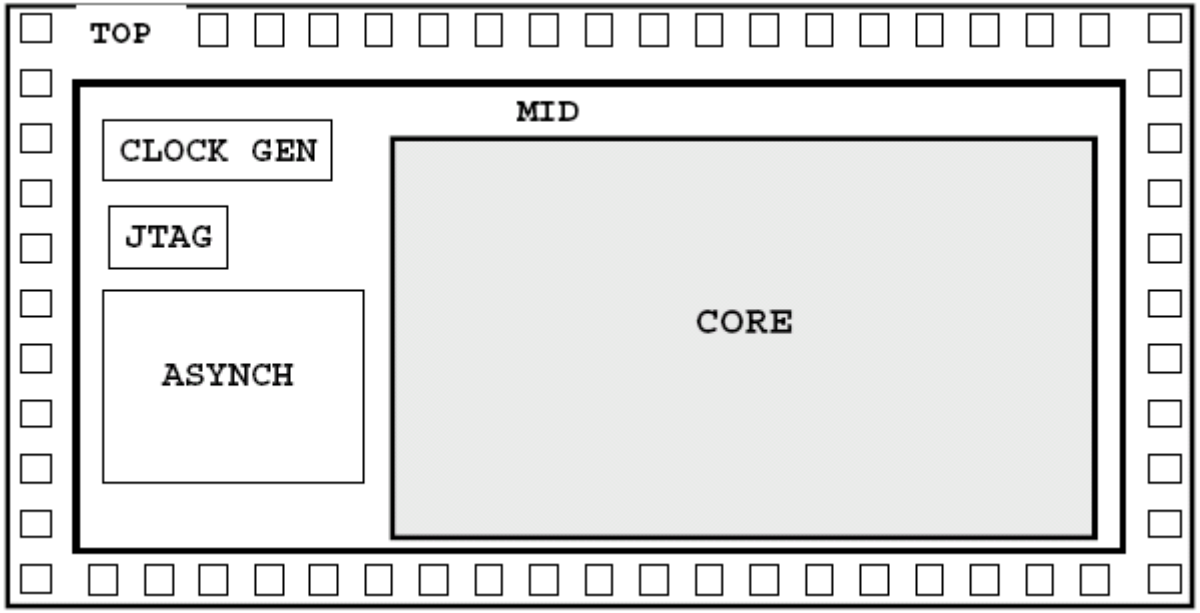
Balance Block Size With Run Times



Good Partitioning

- For quick turnaround, partition so that each block has 5000 – 100,000 gates
- Match module size to CPU and memory:
  - Larger modules are fine if sufficient resources are available
  - Choose smaller sizes when workstation power is limited

Separate Core Logic, Pads, Clocks and JTAG



- This partitioning is recommended due to:
  - Possible technology-dependent I/O pad cells
  - Possible untestable “Divide By” clock generation
  - Possible technology-dependent JTAG circuit

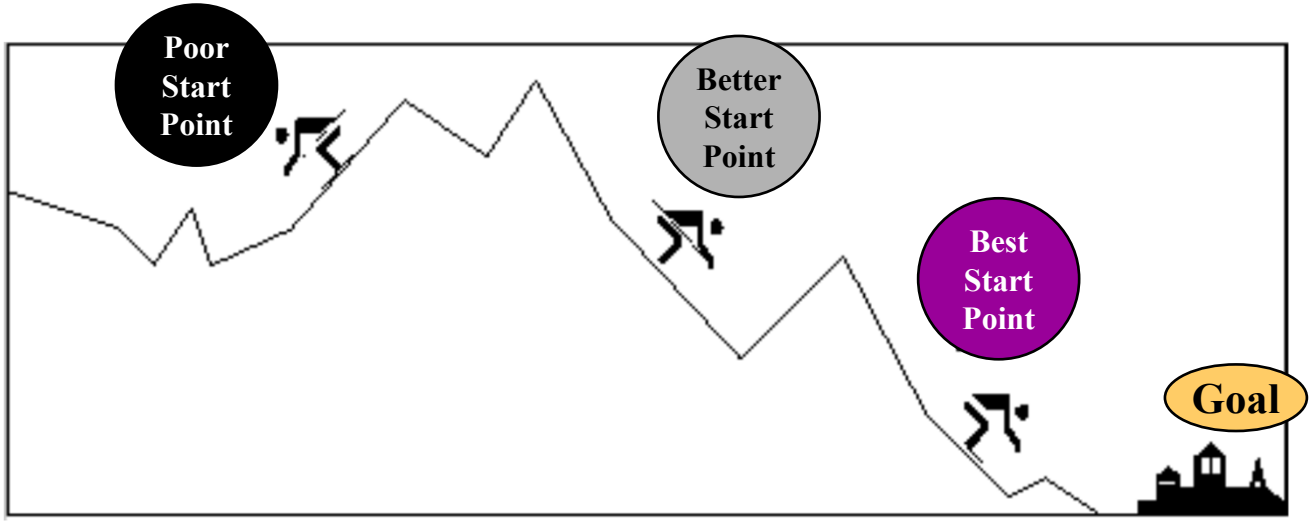
Question & Answer

- List Two effects of partitioning a circuit through combinational logic
- State the main guideline for partitioning for synthesis
- State how partitions are created in HDL code
- List two DC commands for modifying partitions

Agenda

1. Introduction to Synthesis
2. Partitioning for Synthesis
3. Coding for Synthesis
4. Timing and Area
5. Environmental Attributes
6. Time and Load Budgeting
7. Timing Analysis
8. Compiling a Hierarchical Design
9. Compiling a Large Design
10. Conclusion

The Importance of Quality of Source

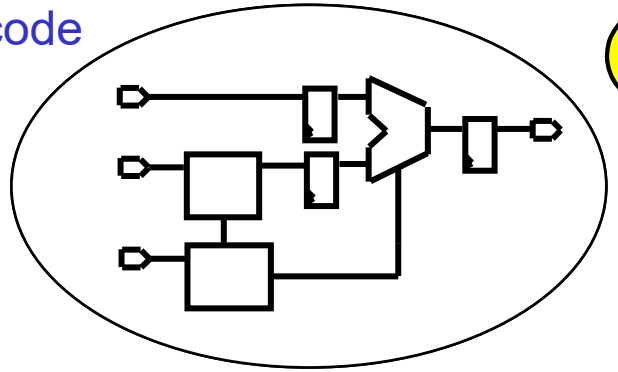


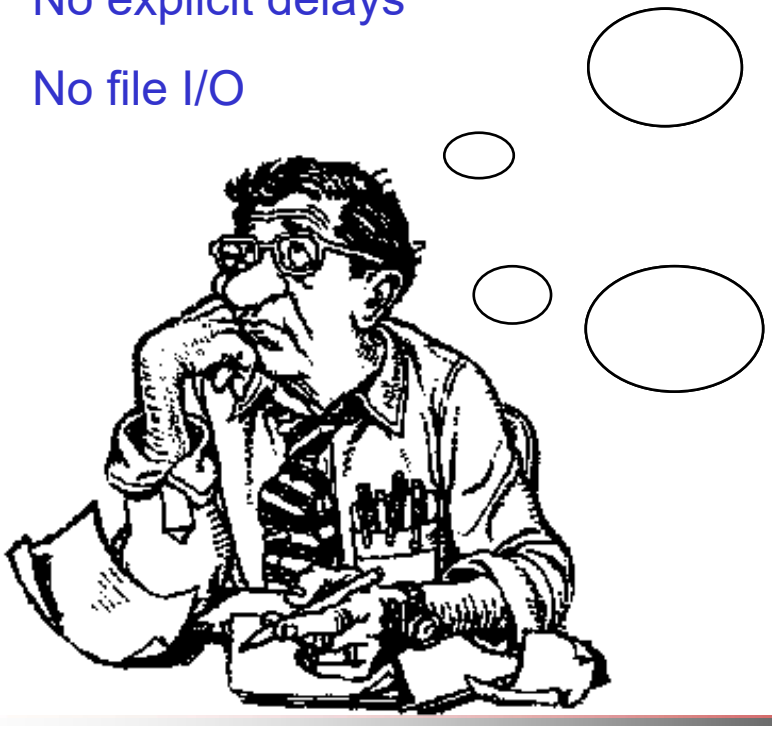
- Code that is functionally equivalent, but coded differently, will give different synthesis results
- You cannot rely solely on DC to “fix” a poorly coded design
- Try to understand the “hardware” you are describing, so that DC has the best possible starting point

Picture 1: Think Hardware

- Write HDL Hardware descriptions
  - Think of the topology implied by the code
- Do not write HDL simulation models
  - No explicit delays
  - No file I/O

Yes!



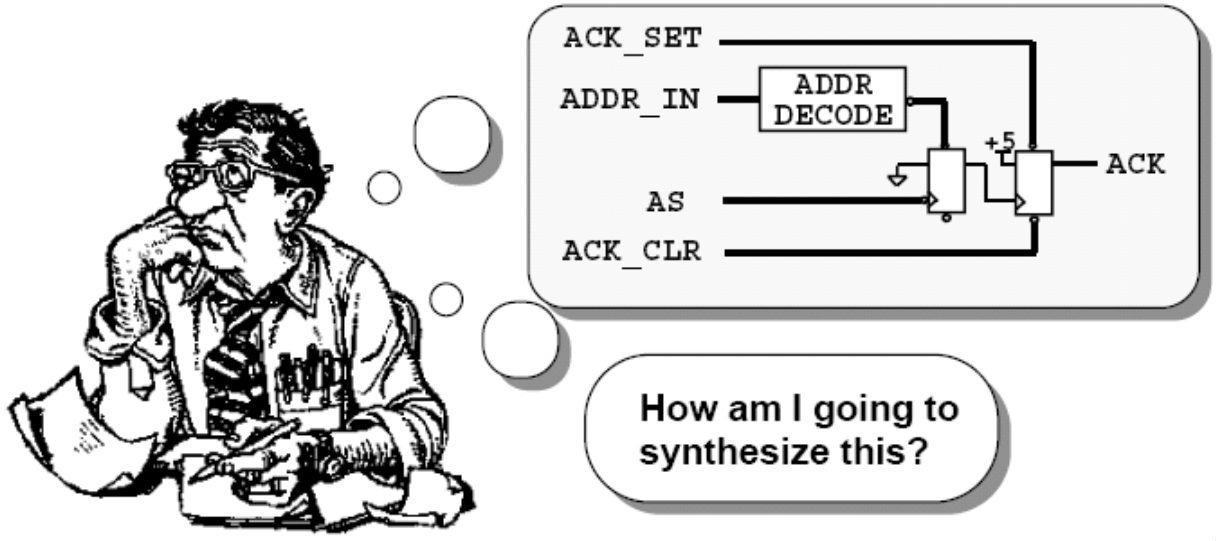


No!

```
after 20 ns and
     2 clock cycles
     OUTPUT <= IN1 + RAM1;
     wait 20 ns;
     ...
```

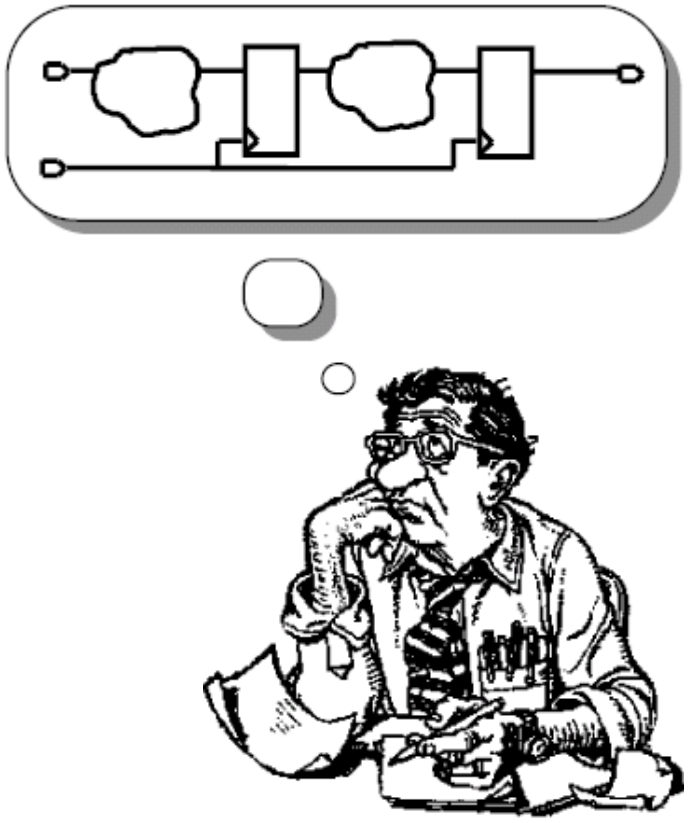
Picture 2: Think Synchronous

- Synchronous designs run smoothly through synthesis, test, simulation and layout
- Asynchronous designs may require hand instantiation and extensive simulation to verify
  - Isolate asynchronous logic into separately compiled blocks



Picture 3: Think RTL

RTL = Register Transfer Level



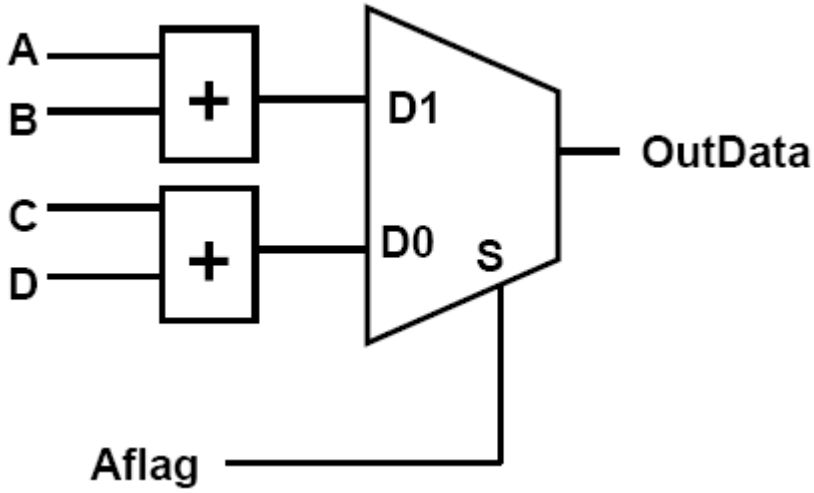
- Writing in an RTL coding style means describing
  - the register architecture
  - the circuit topology, and
  - the functionality between registers
  - DC optimizes logic between registers

Coding Cookbook

- Synthesis of if statements
- Synthesis of case statements
- Synthesis of loop statements
- Synthesis of flip-flop
- Synthesis of arithmetic circuits

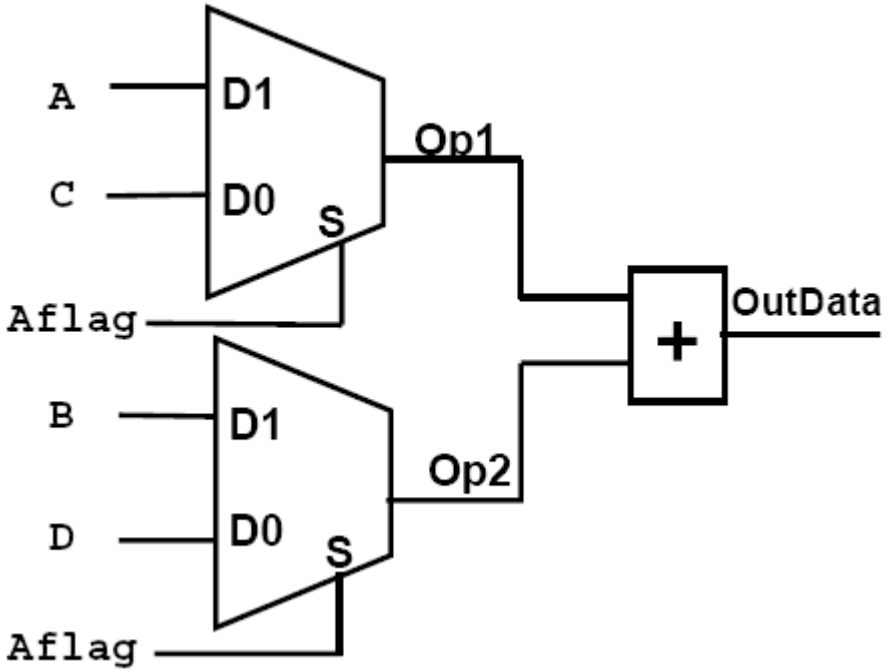
Coding Cookbook cont.

```
if (Aflag == 1'b1)
    OutData <= A + B;
else
    OutData <= C + D;
```



Coding Cookbook cont.

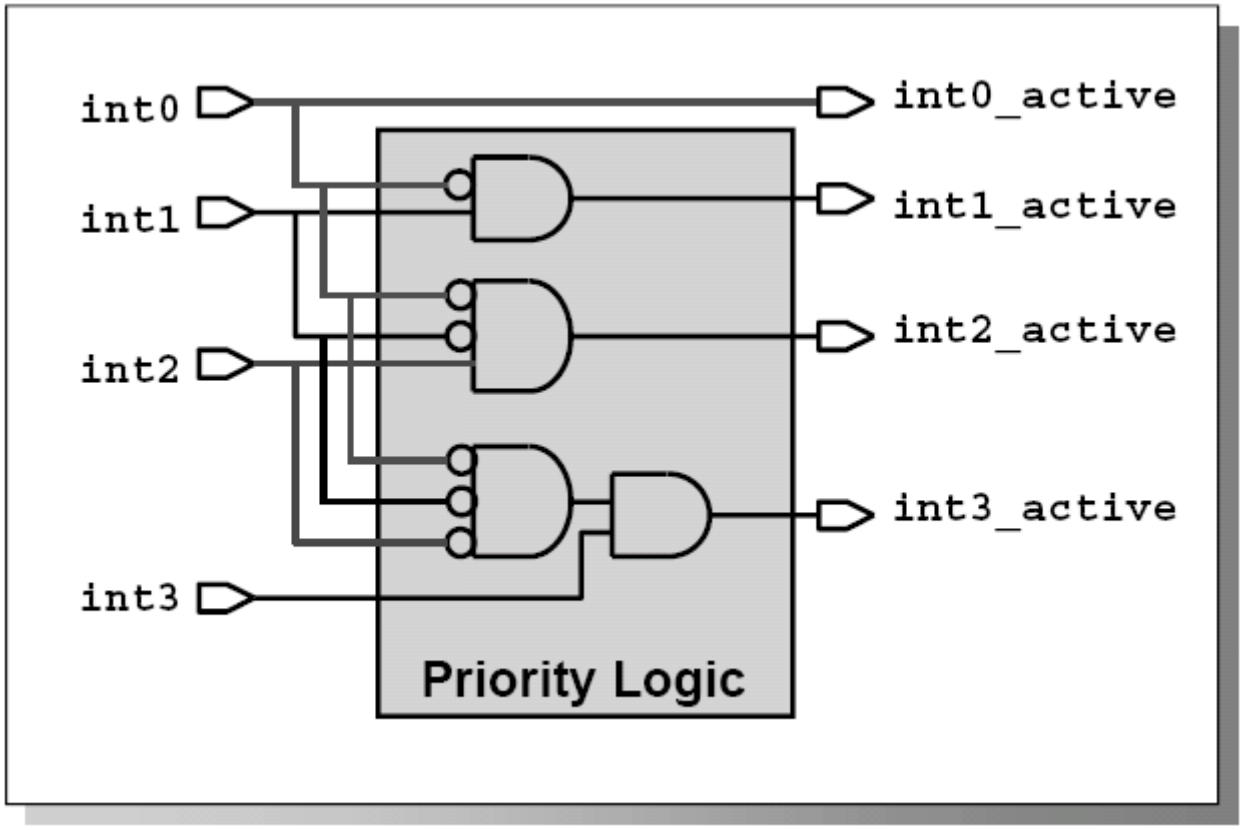
```
if (Aflag == 1‘b1)  begin
    Op1 < = A;
    Op2 < = B;
end
else begin
    Op1 <= C;
    Op2 < = D;
end
OutData <= Op1 + Op2;
```



Coding Cookbook cont.

```
int0_act <= 1'b0;
int1_act <= 1'b0;
int2_act <= 1'b0;
int3_act <= 1'b0;
if (int0) int0_act <= 1'b1;
else if (int1) int1_act <= 1'b1;
else if (int2) int2_act <= 1'b1;
else if (int3) int3_act <= 1'b1;
```

Coding Cookbook cont.

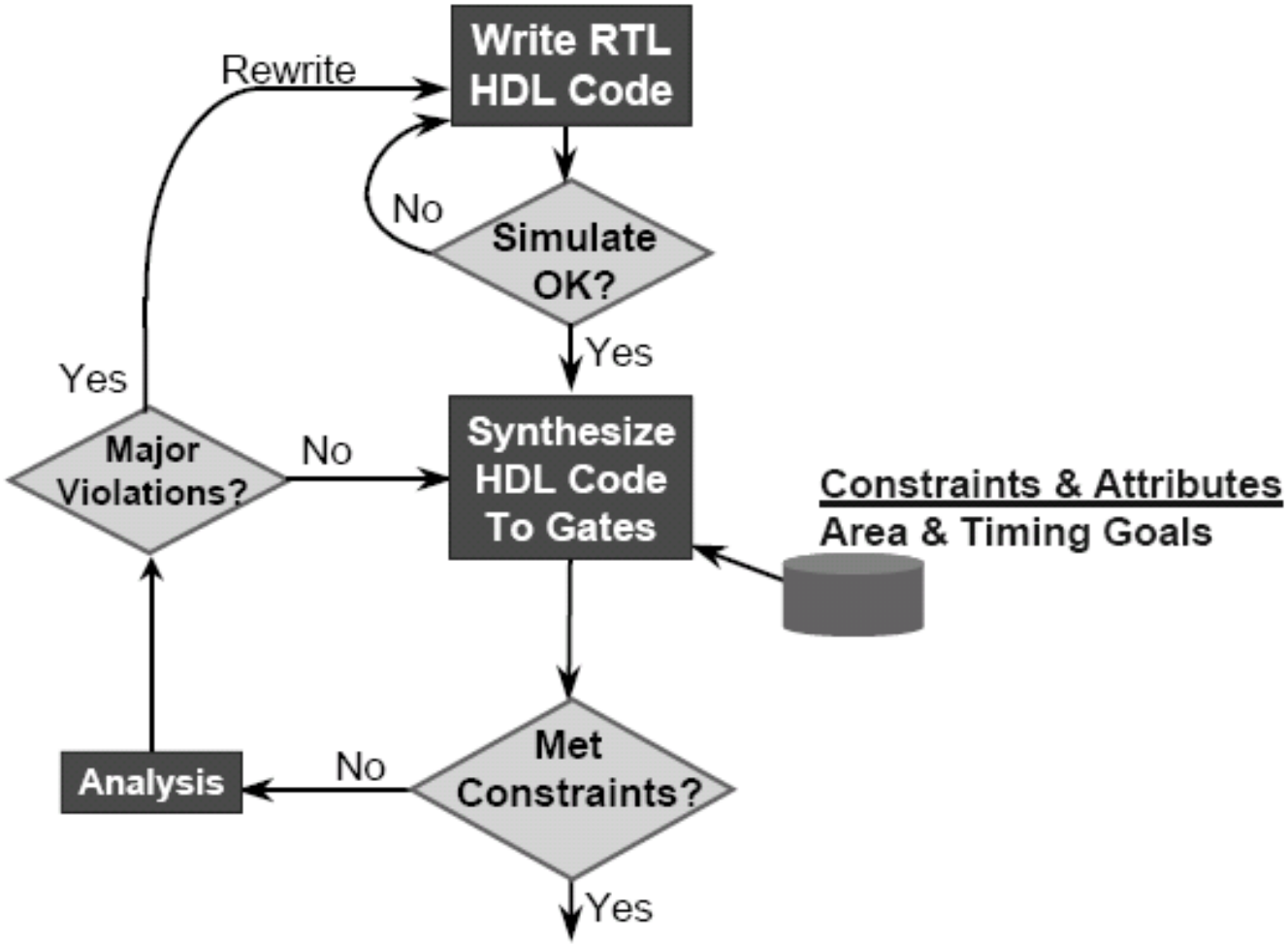


Why are the outputs initialized to 0?

Agenda

1. Introduction to Synthesis
2. Partitioning for Synthesis
3. Coding for Synthesis
4. Timing and Area
5. Environmental Attributes
6. Time and Load Budgeting
7. Timing Analysis
8. Compiling a Hierarchical Design
9. Compiling a Large Design
10. Conclusion

RTL Block Synthesis



Specifying an Area Goal

```
dc_shell-t > current_design h264_top
dc_shell-t > set_max_area 100
```

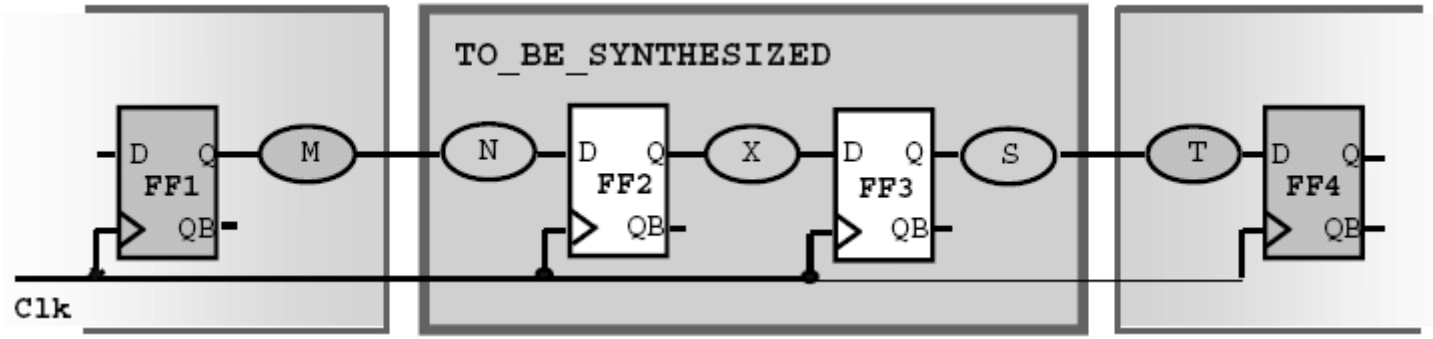
Units are those of target library, defined by the vendor

- 2-input-NAND-gate
- transistors
- square mils ….

Specifying Timing Goals for 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 (R to R) paths, and
    - all output paths

Specifying Timing Goals for Synchronous designs



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

Does the duty cycle of your clock matter?

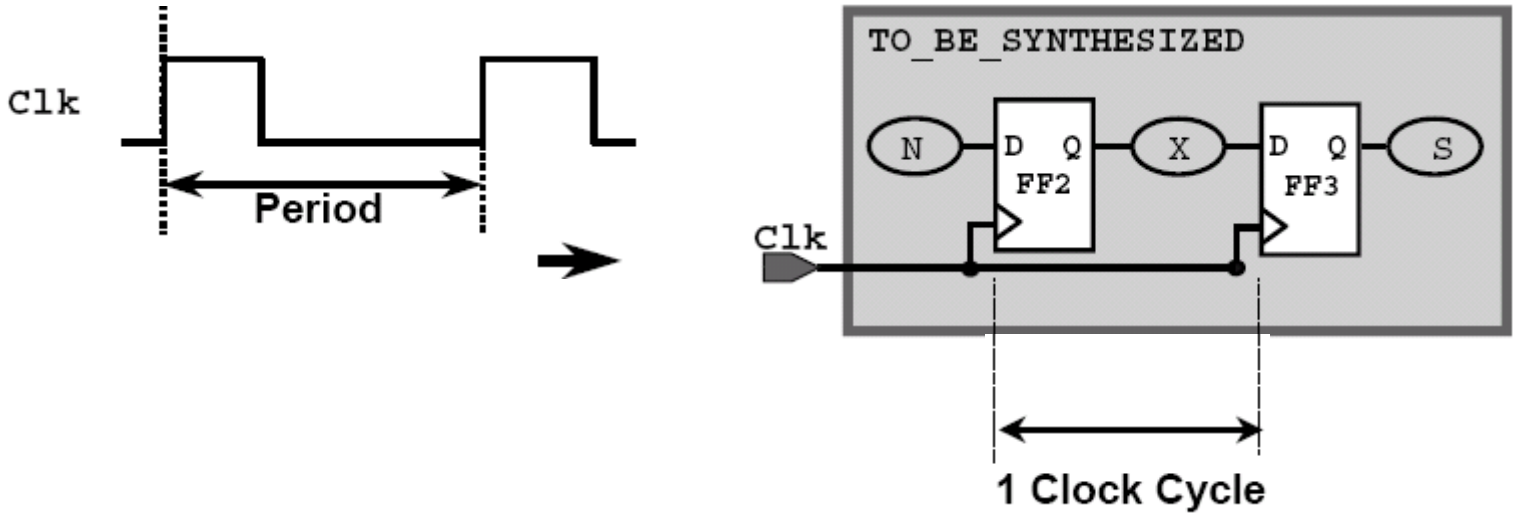
Example:

clock period = 10 ns setup = 1 ns

What is the max delay requirements for the R-T-R paths in the block TO_BE_SYNTHESIZED?

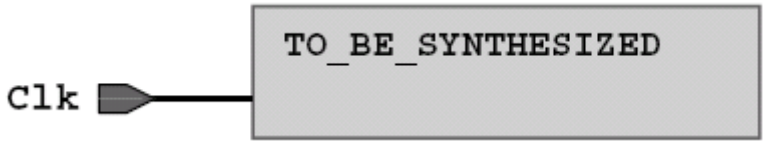
Defining a clock

User MUST Define:	User may also define:
◆Clock Source(port or pin) ◆Clock Period	◆Duty Cycle ◆Offset/Skew ◆Clock Name



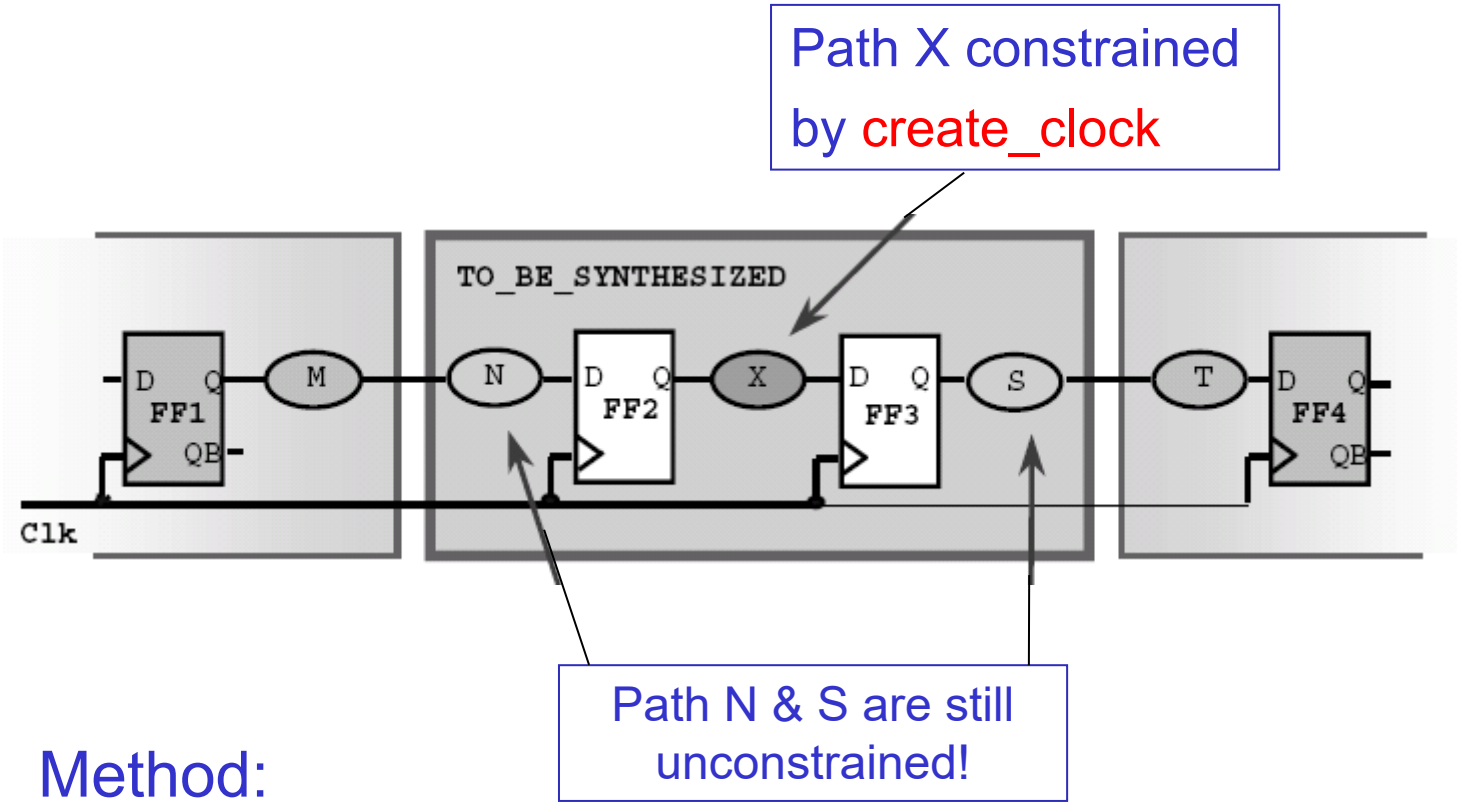
Defining a clock in Design Compiler

```
dc_shell-t > create_clock –period 10 [get_ports Clk]
dc_shell-t > set_dont_touch_network [get_clocks Clk]
```



- Create a clock constrains timing paths between registers
- use report_clock to see defined clocks and their attributes
- set_dont_touch_network tells DC not to “buffer up” the clock net, even if there are several flip-flops loading it

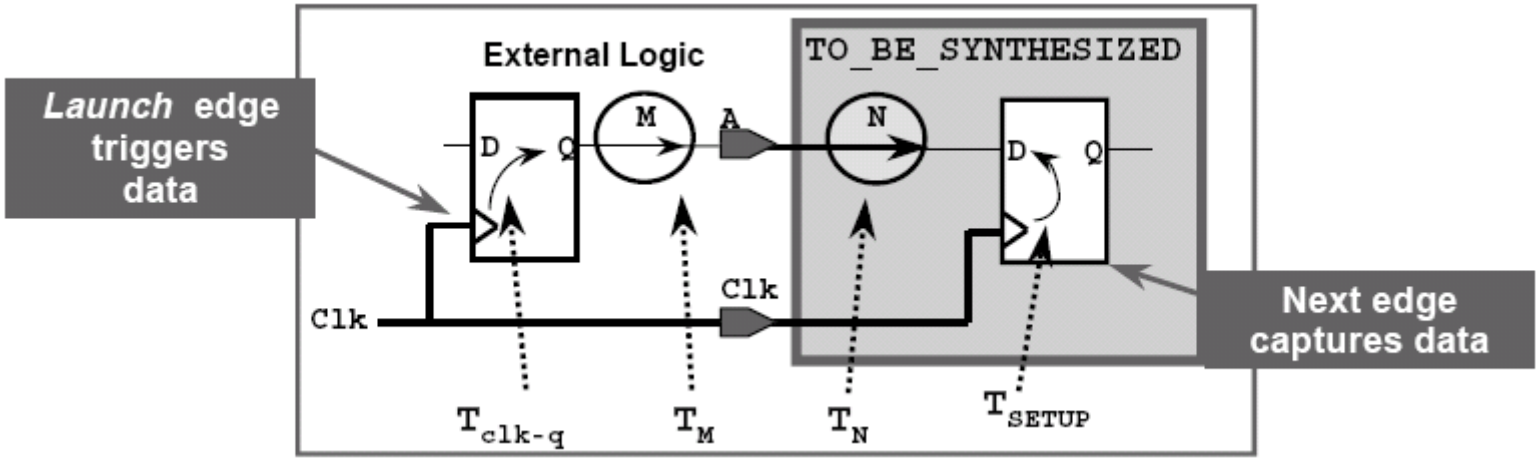
Timing Goals for Synchronous Designs



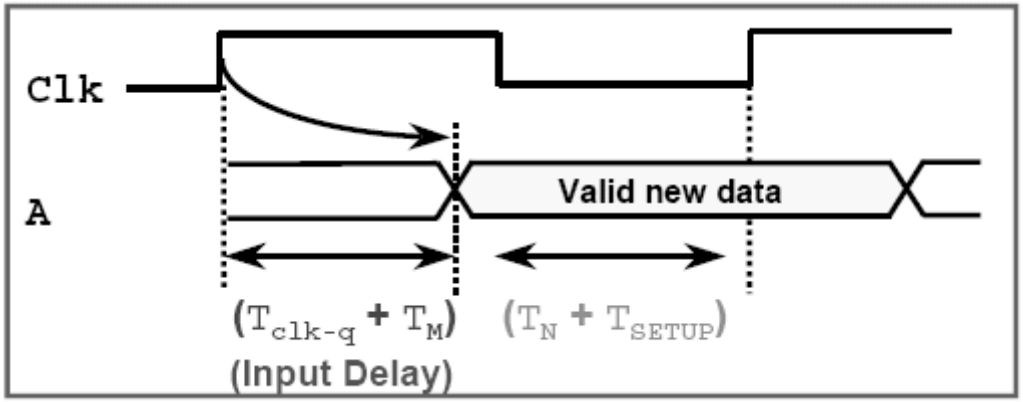
Method:

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

Constraining the Input Paths



What information must you provide to constrain the input paths?

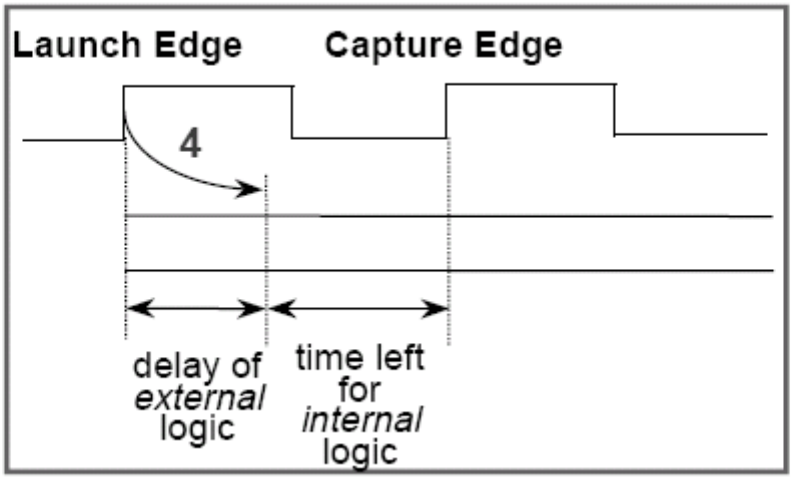


Constraining the Input Paths in DC

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

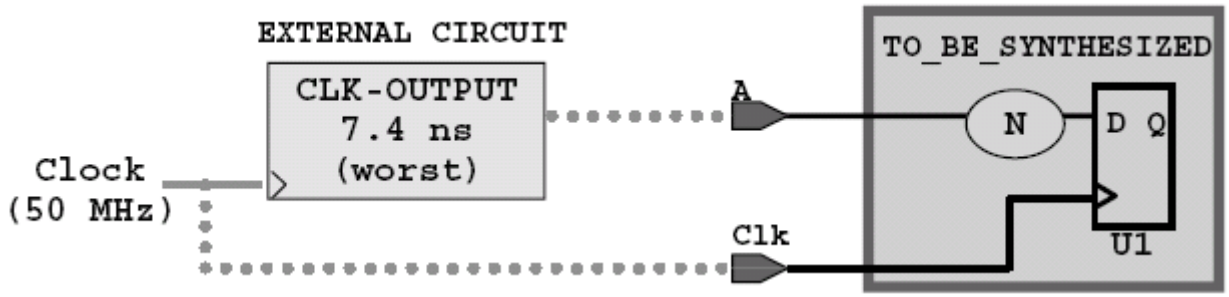
The set_input_delay command constrains input paths

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



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

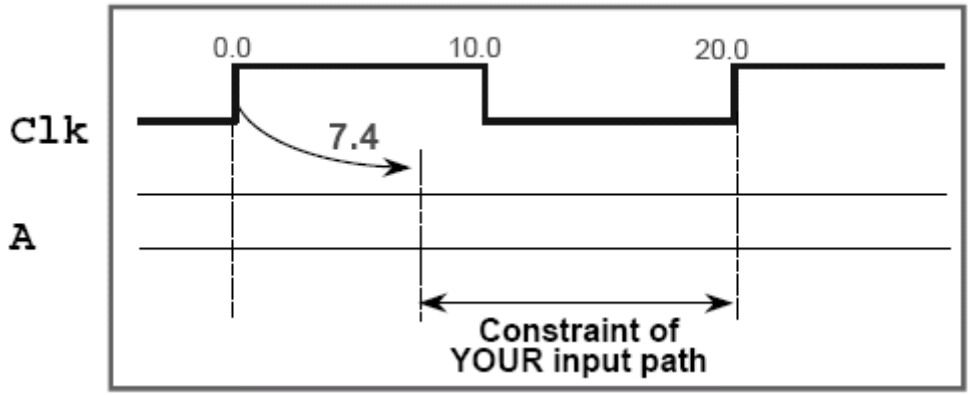
set_input_delay Exercise



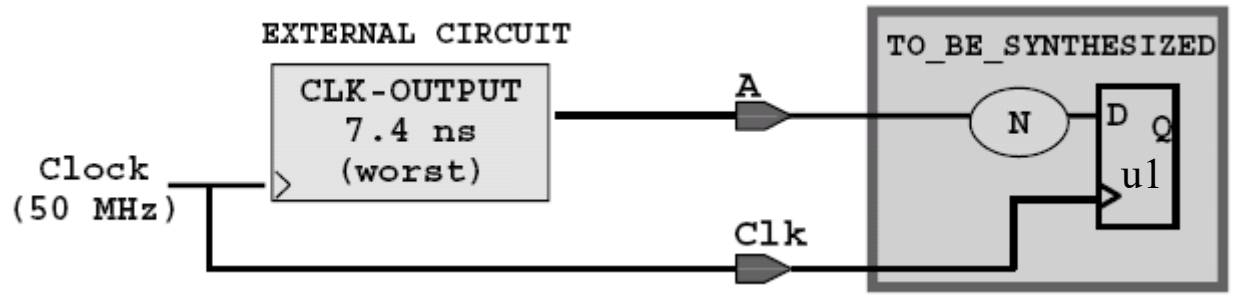
create_clock_____

set_dont_touch_network_____

set_input_delay_____



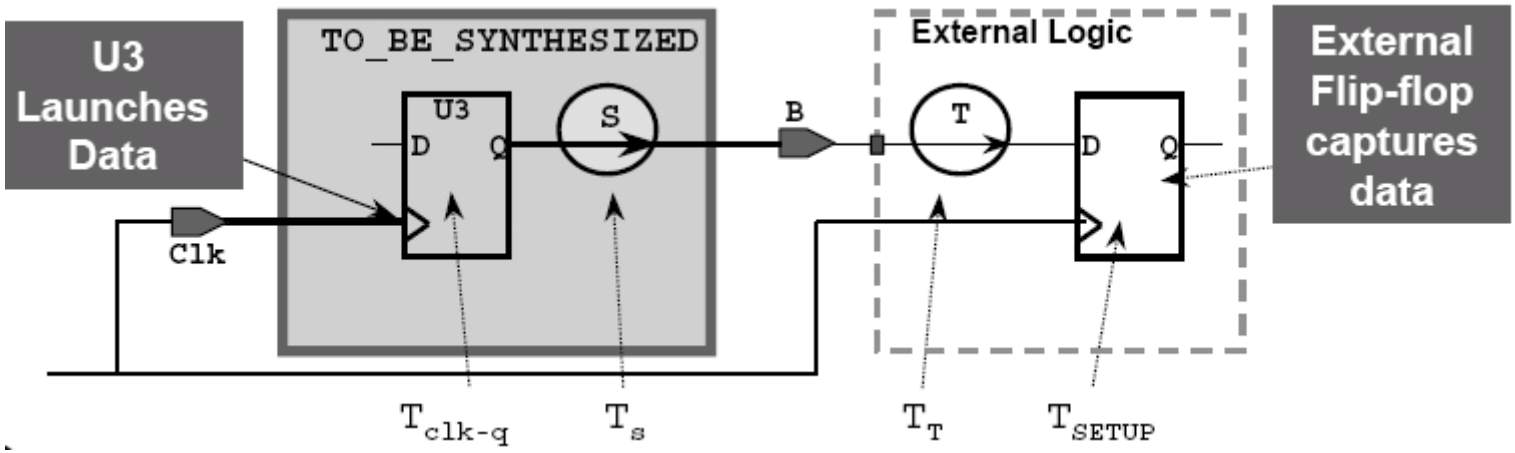
Effect of set_input_delay on input paths



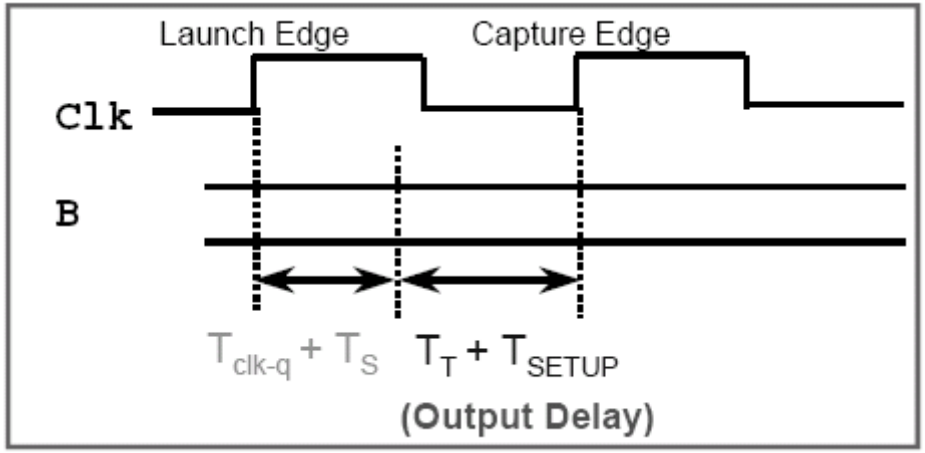

```
create_clock  -peroid 20 [get_ports Clk]
set_dont_touch_network [get_clocks Clk]
set_input_delay –max 7.4 –clock Clk [get_ports  A]
```

If U1 has a 1 ns setup requirement:
What is the maximum delay for T_N?

Constraining the Output Paths of a Design



What information must you provide to constrain the output path?

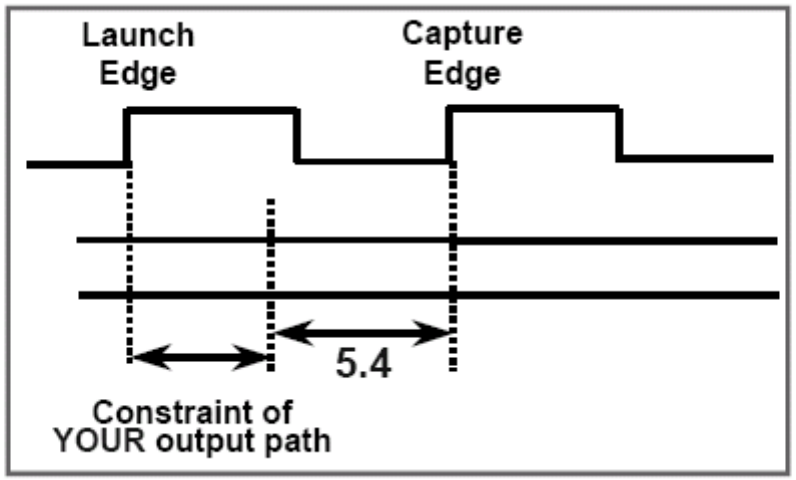


Constraining the Output Paths in DC

```
dc_shell-t > set_output_delay –max 5.4 –clock Clk [get_ports B]
```

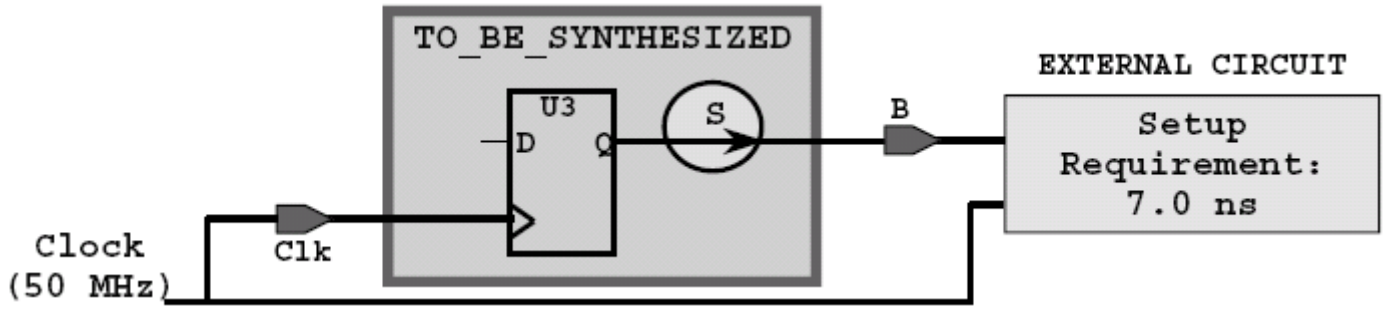
The set_output_delay command constrains output paths

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



DC calculates how much time is left for *internal* logic

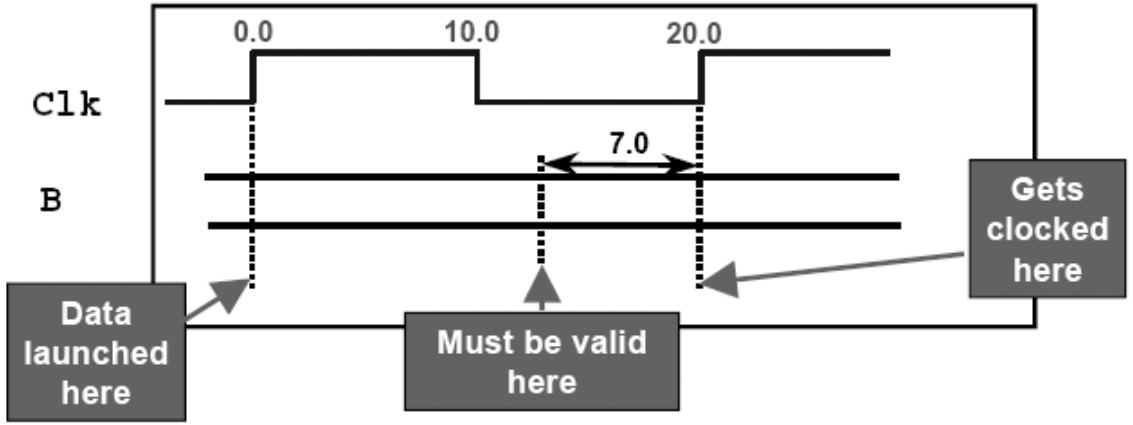
set_output_delay Exercise



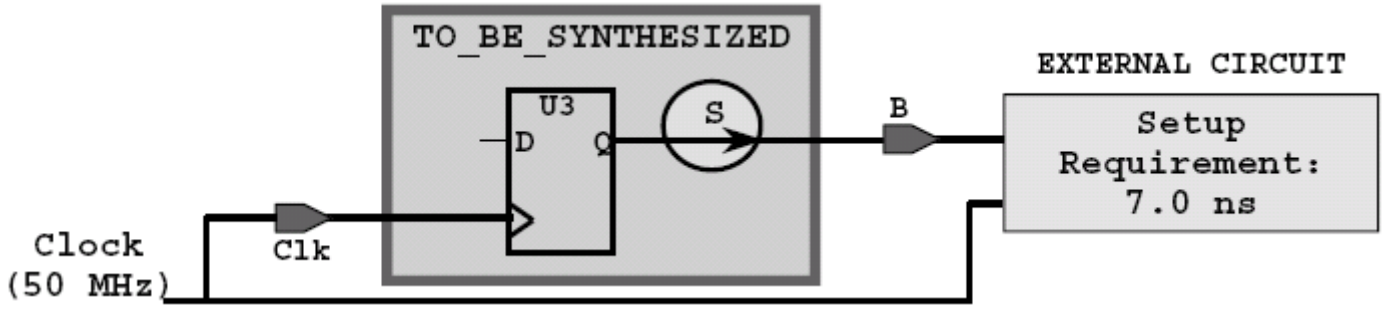
create_clock_____

set_dont_touch_network_____

set_output_delay_____



Effect of set_output_delay on output paths

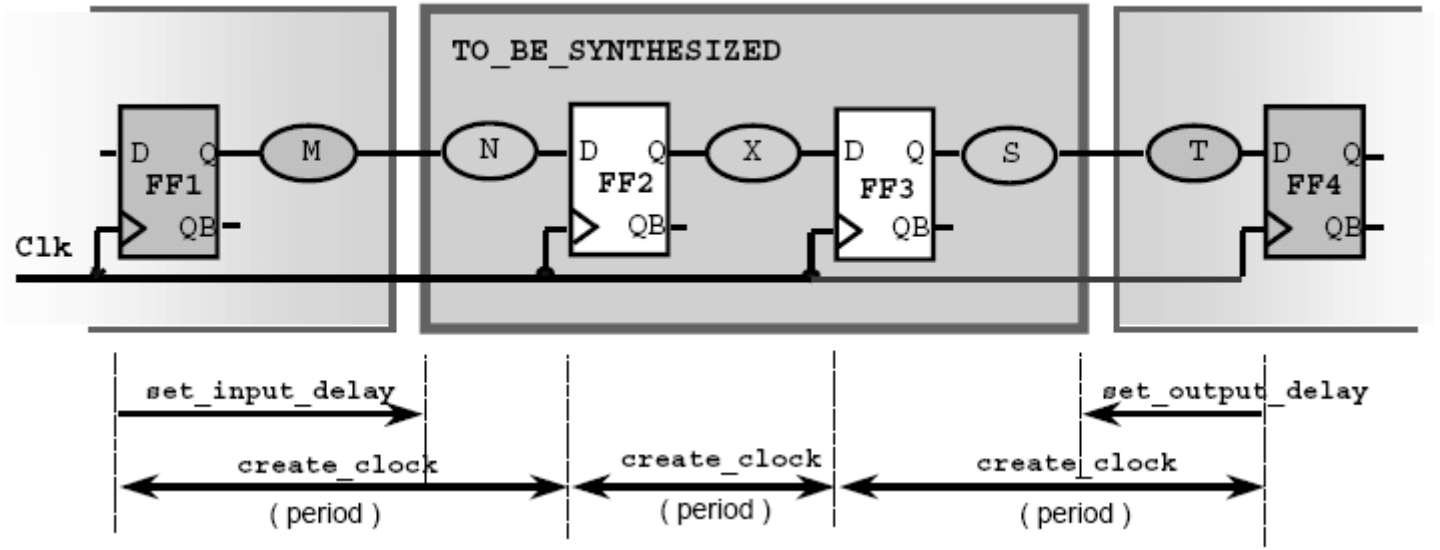



```
create_clock  -peroid 20 [get_ports Clk]
set_dont_touch_network [get_clocks Clk]
set_output_delay –max 7.0 –clock Clk [get_ports  B]
```

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

Summary of Area & Timing Goals

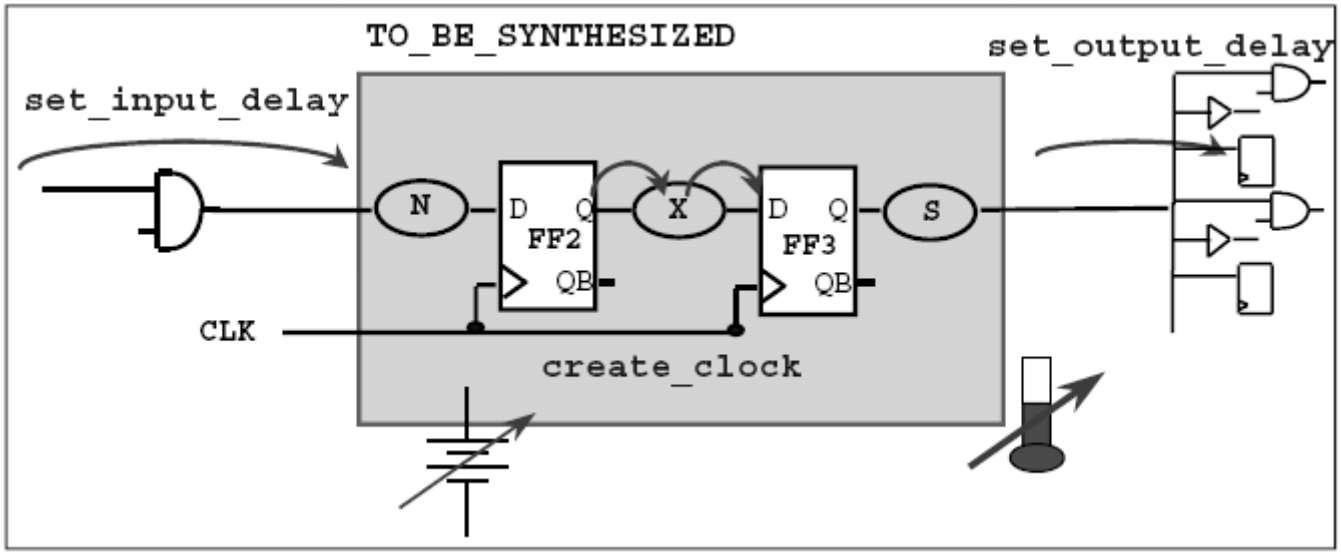
- To specify a maximum area goal (set_max_area)
- To constrain all of the timing path of a design (create_clock, set_input_delay, set_output_delay)



Agenda

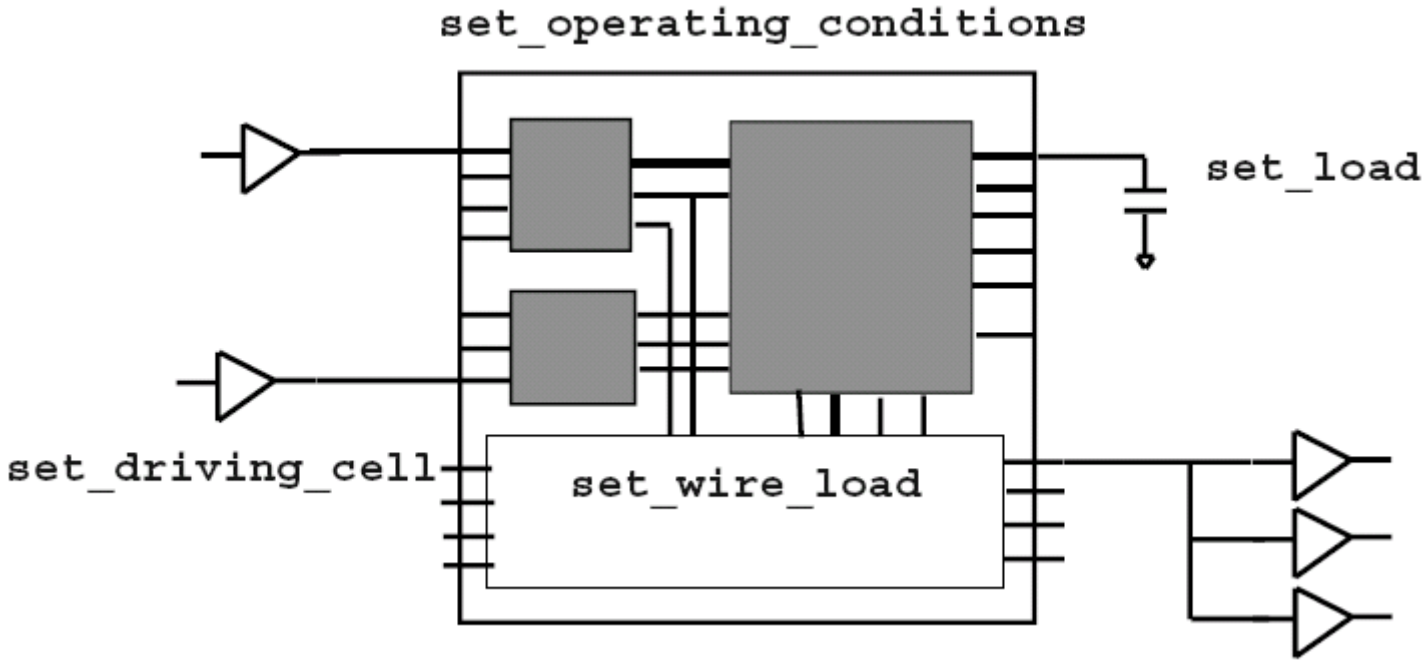
1. Introduction to Synthesis
2. Partitioning for Synthesis
3. Coding for Synthesis
4. Timing and Area
5. Environmental Attributes
6. Time and Load Budgeting
7. Timing Analysis
8. Compiling a Hierarchical Design
9. Compiling a Large Design
10. Conclusion

Constraining for Timing – What's Missing?

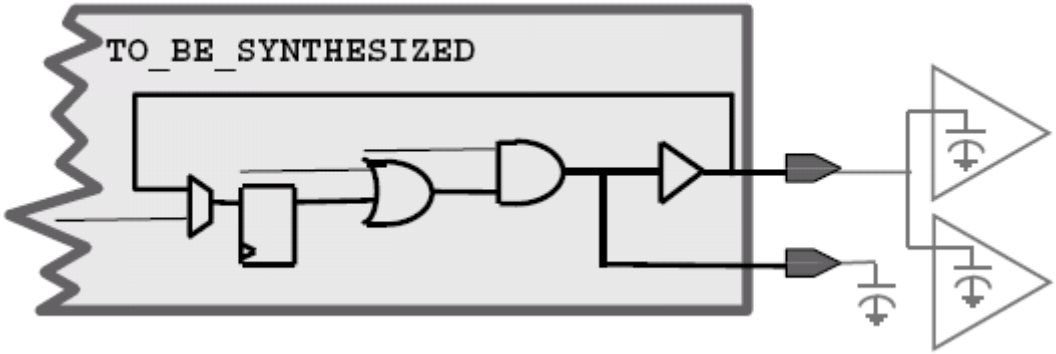


- What physical information about the chip is required in order to accurately calculate the speed of each path?
- What information (besides input delay) does DC require in order to ensure logic N meets timing?
- What information (besides external setup) does DC require in order to ensure logic S meets timing?

Describing Environmental Attribute



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 the user to specify the external capacitive load on ports
  - By default, DC assumes that the external load on ports is 0
  - You can specify some other constant value, or …
  - 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]

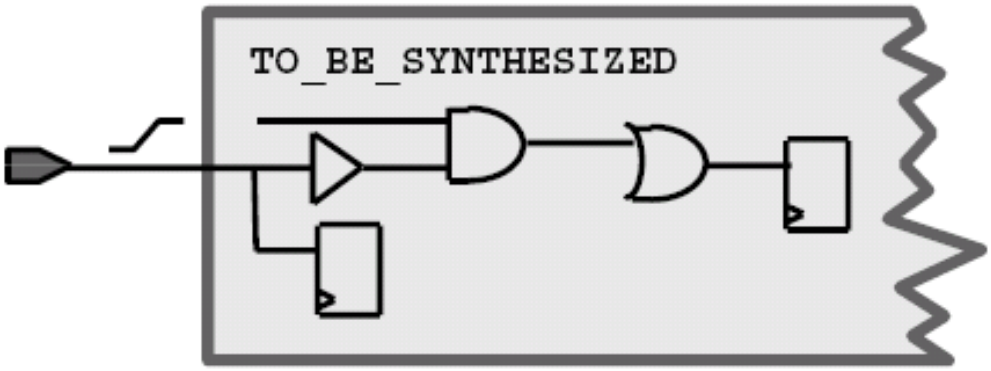
- Use set_load load_of(lib/cell/pin) to place the load of a gate from the technology library on the port

  set_load load_of (smic018/AN2/A) [get_ports OUT1]

- load_of (lib/cell/pin) can be multiplied by a scaling factor

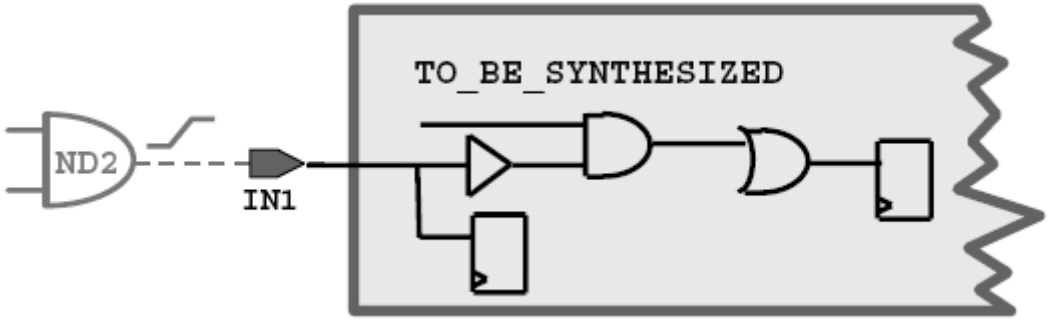
  set_load load_of (smic018/IVA/A) * 3 [get_ports OUT1]

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 the user to specify the 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 transition time on the input signal as though the specified library cell was driving it

set_driving_cell Examples



dc_shell-t > set_driving_cell –lib_cell “ND2” [get_ports IN1]

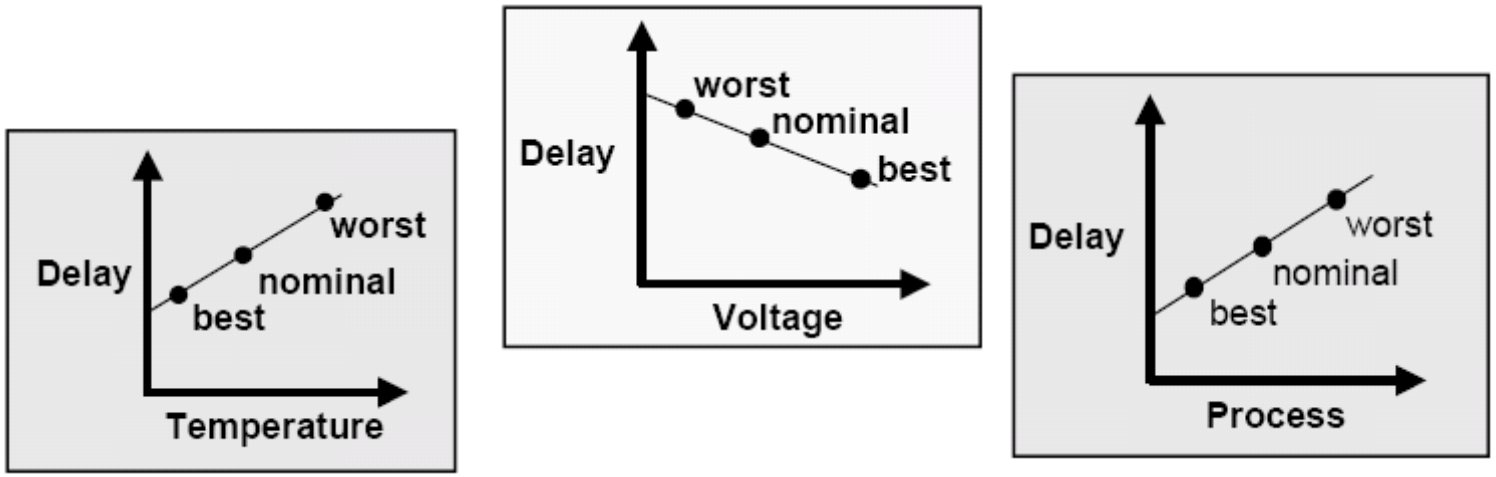
Variations in cell delays

Library cells are usually characterized using "nominal" voltage and temperature

What if the circuit is to operate at a voltage and/or temperature OTHER than nominal?

Operating Conditions

- 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



Specifying Operating Conditions

- By default, NO operating conditions are specified for a design
- Use report_lib *libname* to list the vendor-supplied operating conditions

```
Operating Conditions:

    Name            Library       Process      Temp     Volt
    ----------------------------------------------------------
    typ_25_5.00     cba_core        1.00     25.00     5.00
    typ_-40_4.50    cba_core        1.00    -40.00     4.50
    typ_-40_4.75    cba_core        1.00    -40.00     4.75
```

dc_shell-t > current_design addtwo

dc_shell-t > set_operating_conditions –max “typ_-40_4.50”

Wire Load Model

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

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

- Model is created by the Vendor
- Estimates are based on statistics from other designs the vendor has fabricated using this process

Wire Load Model Example

```
wire_load (tc6a120m2) {
    resistance : 5.0;    /* R per unit length */
    capacitance : 1.0;   /* C per unit length */
    area : 0.05;   /* Area per unit length */
    slop : 0.15;  /* extrapolation slope */

    fanout_length (1, 2.6);  /* fanout – length pairs */
    fanout_length (2, 2.9);
        ……..
    fanout_length (6, 4.7);
}
```

Specifying Wire Load in DC

```
dc_shell-t > current_design  addtwo
dc_shell-t > set_wire_load  tc6a120m2
```

Check Constraints

- Check your work before you compile
  - It can save you a lot of time!
- Use report commands:
  - report_port –verbose
    To check values specified for : set_load, set-drive, set_driving_cell, set_input_delay, set_output_delay
  - report_design
    To check wire loads and operating conditions
  - report_clock
    To verify that the clocks have been defined
  - Write_script
    To list constraints and attributes on a design in the form of an executable script

Exercise: Describe the Environment

Create a default script using following Specifications:

- The chip operate in 125MHz (8.0 ns)
  - Each input port will be constrained to use 40% of the clock period (3.2 ns)
  - Each output port will be constrained to use 40% of the clock period (3.2 ns)
- The chip will operate at 85^0C, and 4.5 volts
  - This is described by the operating condition typ_85_45
- Use a tc6a120m2 wireload model
- The inputs will be assumed to be driven by the buffer “BUF_A” from the cba_core library
  - except for the clock input
- The outputs are restricted to driving no more than 5 pf

Summary of Describing Constraints

- **Timing and Area Goals:**

  set_max_area set_input_delay

  set_output_delay

  create_clock set_dont_touch_network

- **Environmental Attributes:**

  set_driving_cell set_load load_of()

  set_operating_conditions set_wire_load

- **Reports:**

  report_clock report_port –verbose

  report_design write_script

- **Problems?**

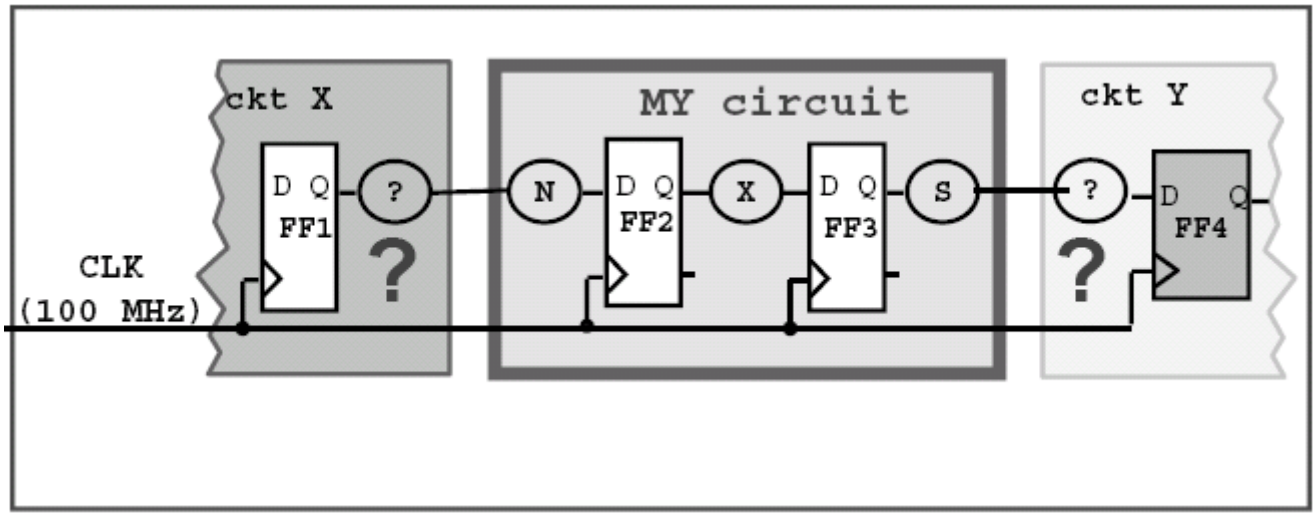
  reset_design remove_design

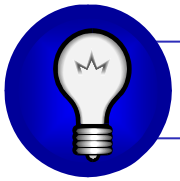
Agenda

1. Introduction to Synthesis
2. Partitioning for Synthesis
3. Coding for Synthesis
4. Timing and Area
5. Environmental Attributes
6. Time and Load Budgeting
7. Timing Analysis
8. Compiling a Hierarchical Design
9. Compiling a Large Design
10. Conclusion

Time Budgeting

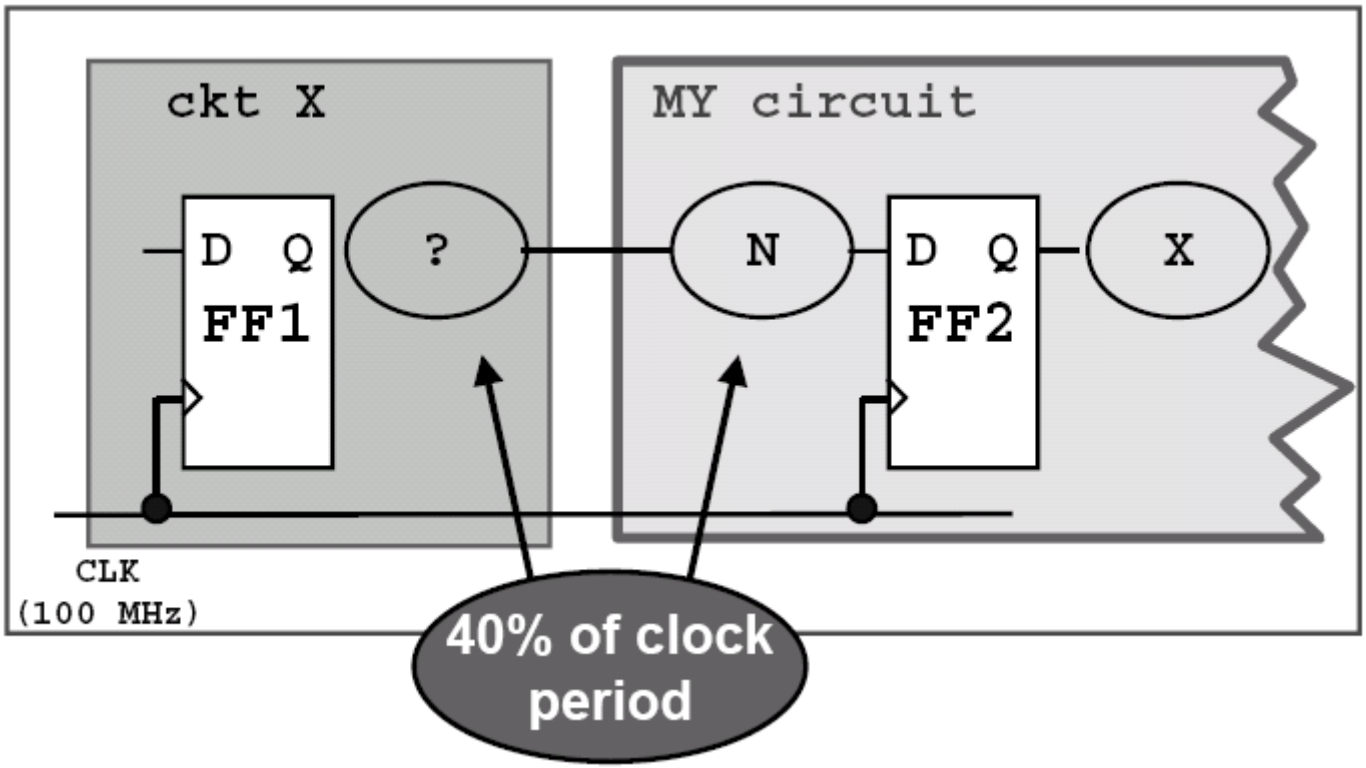
What if you don't know the delays on your inputs or the setup requirements or your outputs?

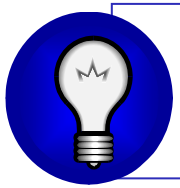




A: Create a Time Budget!

Time Budgeting cont.

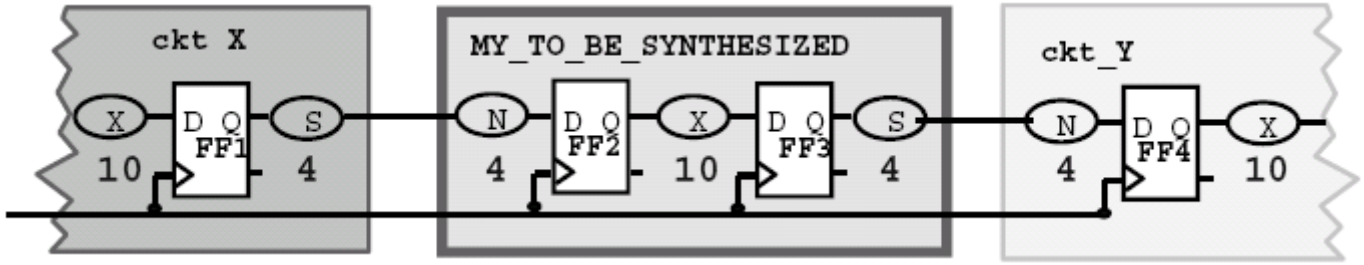




Better to budget conservatively than to compile with paths unconstrained!

Time Budgeting Example

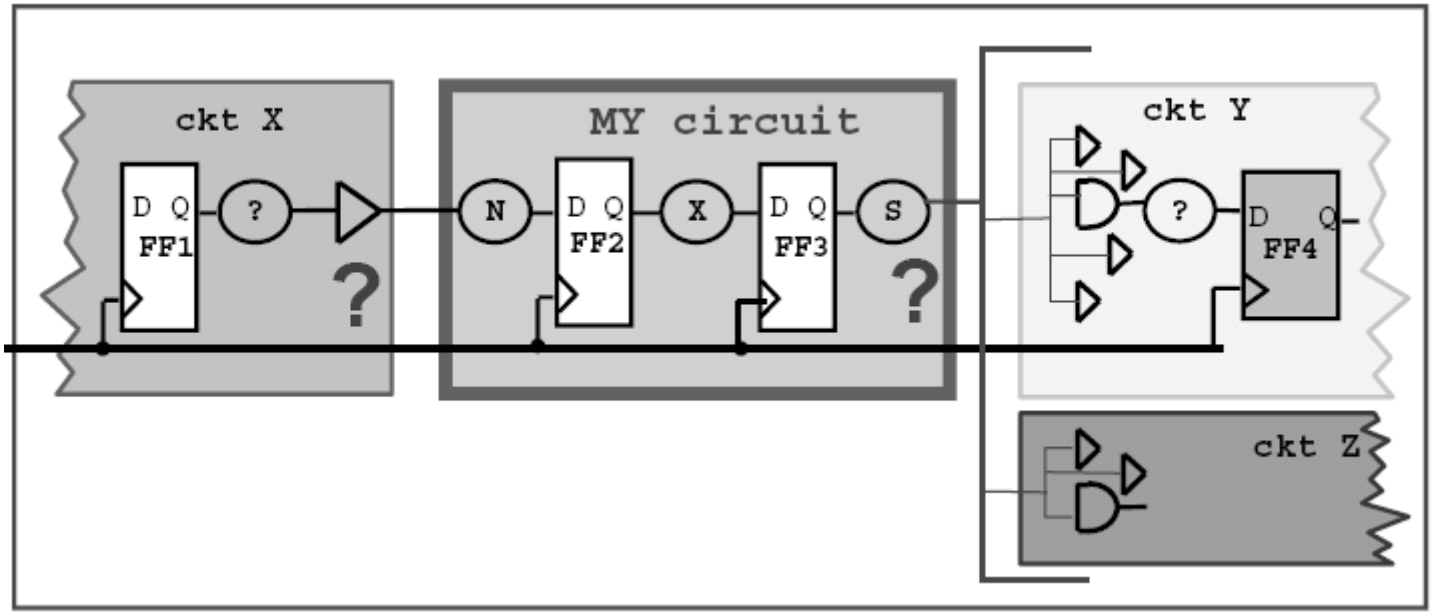
```
/* A generic Time Budgeting script file   */
create_clock –period 10 [get_ports Clk]
set_dont_touch_network [get_clocks Clk]
set_input_delay –max 6 –clock Clk [all_inputs]
remove_input_delay [get_ports Clk]
set_output_delay –max 6 –clock Clk [get_ports Clk]
```



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

Load Budgeting

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



A: Create a Load Budget!

Load Budgeting cont.

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

A : Create a load budget

- Assume a weaker cell driving the inputs, to conservative
- Limit the input capacitance of each input port
- Limit the number of other major blocks your outputs can drive

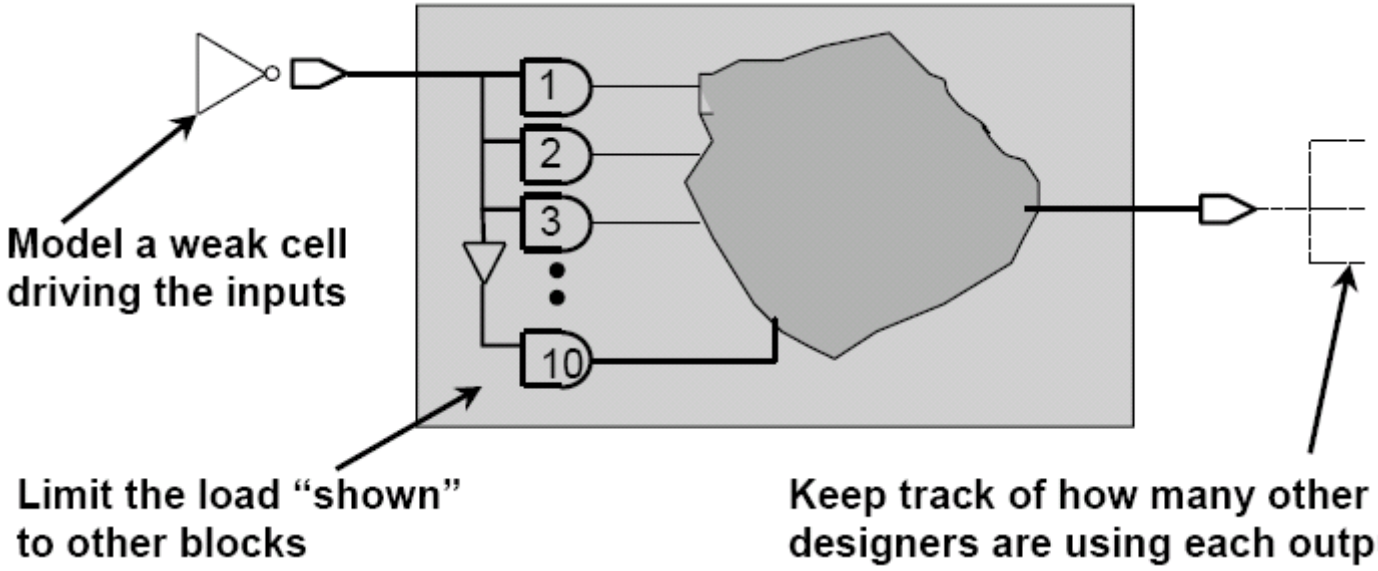
How do we limit the input capacitance of an input port?

A : Place restrictive design rules on our input ports

Load Budget Example

System Considerations:

- Inputs of all blocks can present no more than the load of 10 “AND2” gates to other blocks
- Outputs of all blocks will only be allowed to connect to a maximum of 3 other blocks
  - Otherwise, output port will need to be replicated in code



Load Budget Example cont.

```
current_design  myblock
source timing_budget.scr
/* define all inputs except Clk */
set  ALL_EXCEPT_CLK_INPUT [remove_from_collection  [all_inputs]
[get_ports Clk]]
/* Assume a weak driving buffer on the inputs */
set_driving_cell –cell IVA ${ALL_EXCEPT_CLK_INPUT}
/* Limit the input load */
set  max_input_load  load_of(cba/AND2/A) * 5
set_max_capacitance ${max_input_load}
${ALL_EXCEPT_CLK_INPUT}
/* Model the max possible load on the outputs */
set_load ${max_input_load} * 3 [all_outputs]
```

Exercise: Describe the Environment

Create a default script using following Specifications:

The chip operate in 125MHz (8.0 ns)

Environment:

- The chip will operate at 85^0C, and 4.5 volts
  - This is described by the operating condition typ_85_45
- Use a tc6a120m2 wireload model

Time Budget:

- Each input port will be constrained to use < 40% of the clock period
- Each output port will be constrained to use < 40% of the clock period

Exercise: Describe the Environment cont.

Create a default script using following Specifications:

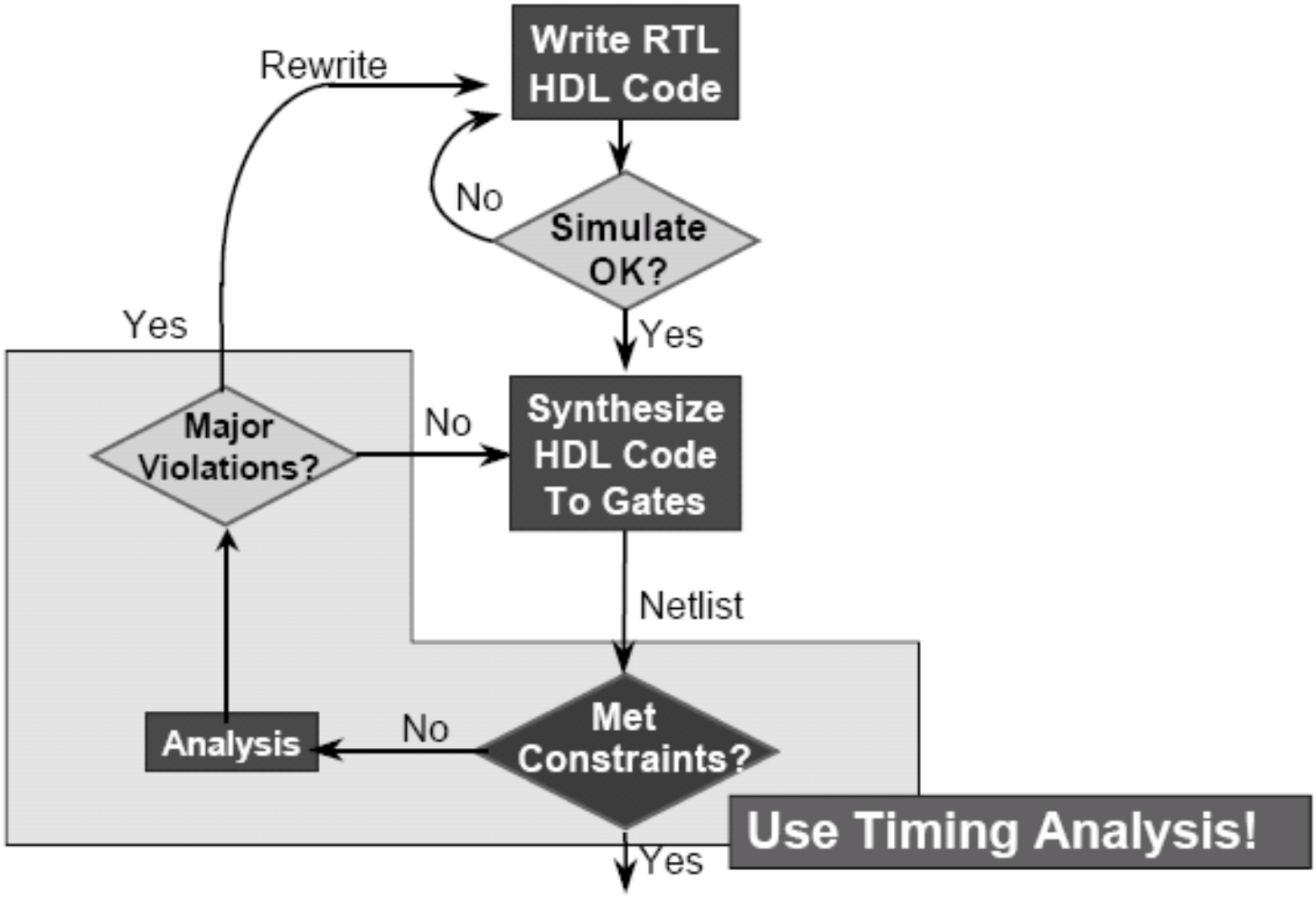
Load Budget:

- The inputs will be assumed to be driven bye the weak buffer “BUF_A” from the cba_core library(except clock input)
- The input capacitance of the non-clock input ports must be less than 10 input pin loads of the buffer “BUF_E”
  - The input pin of BUF_E is named “IA”
- The outputs are restricted to driving no more than 4 other blocks on the ASIC

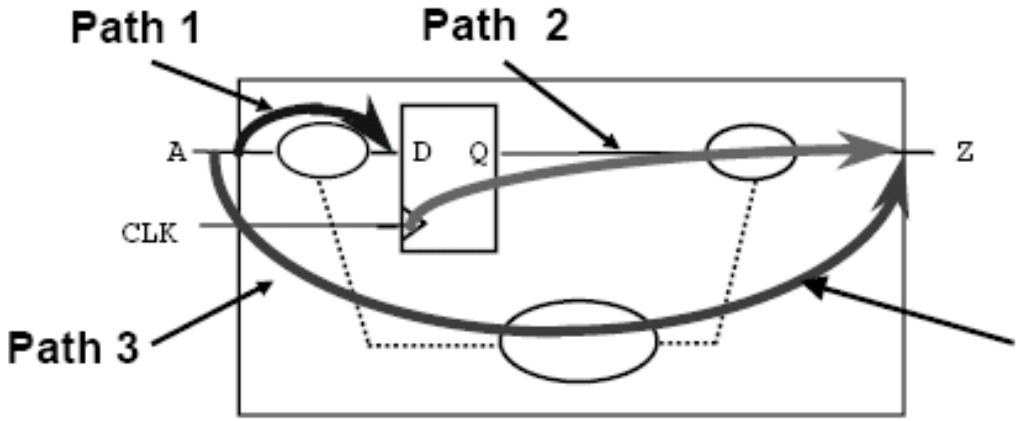
Agenda

1. Introduction to Synthesis
2. Partitioning for Synthesis
3. Coding for Synthesis
4. Timing and Area
5. Environmental Attributes
6. Time and Load Budgeting
7. Timing Analysis
8. Compiling a Hierarchical Design
9. Compiling a Large Design
10. Conclusion

Does your Design Meet its Goals?

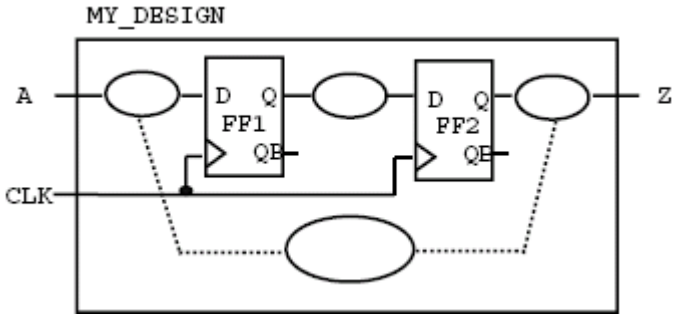


Static Timing Analysis



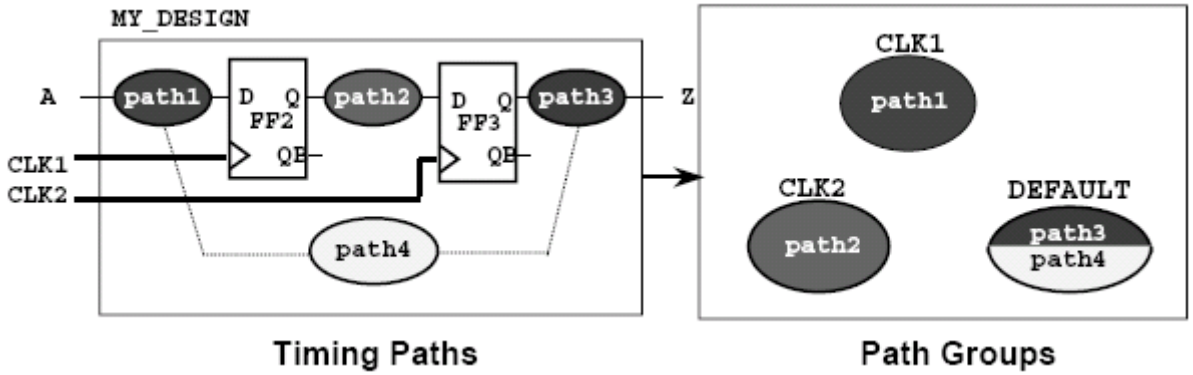
- Static Timing Analysis can determine if a circuit meets timing constraints without having to simulate
- This involves three main steps:
  - Design is broken down into sets of timing paths
  - The delay of each path is calculated
  - all path delay are checked to see if timing constraints have been met

Timing Paths in DC



- DesignTime breaks Design into sets of signal **paths**
- Each path has a startpoint and an endpoint
  - Startpoints:
    - Input ports
    - clock pins of flip flop or registers
  - Endpoints:
    - Output ports
    - Data input pins of sequential devices

Organizing Timing Paths Into Groups

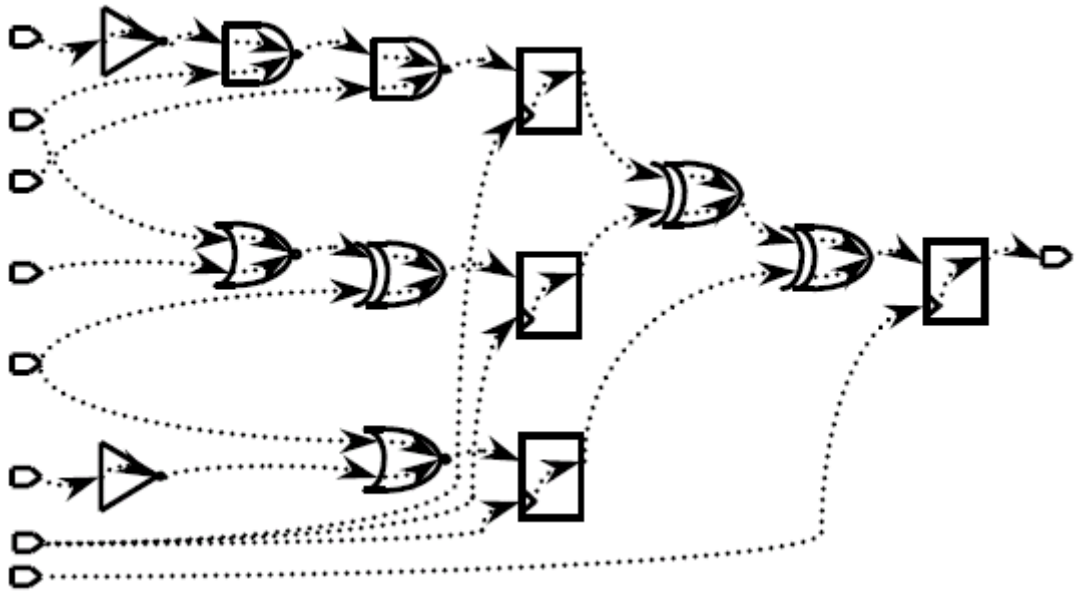


- Paths are grouped by the clocks controlling their endpoints
  - The default path group contains all paths not captured by a clock

Schematic Converted to a Timing Graph

To calculate total delay, DesignTime breaks each path into timing arcs

- Each timing arc contributes either a net delay or cell delay

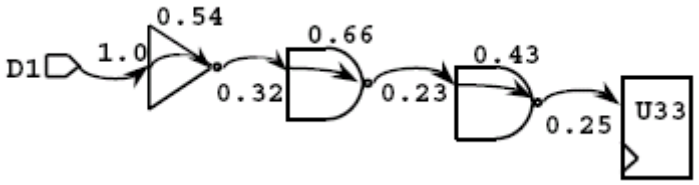


How Does DesignTime Calculate Delays?

- Cell delays are calculated using one of several cell delay models:
  - Non-linear Delay Model/Linear Delay Model/Others …
  - Cell delay model used by the technology library is chosen by the Vendor
- Net delays are calculated using one of three net delay models:
  - Best Case Tree/Balanced Tree/Worst Case Tree
  - The net delay model is contained in the operating condition, which is selected by the user

Calculating a Path Delay

Is very Simple: Just add all the net and cell timing arcs alone the path



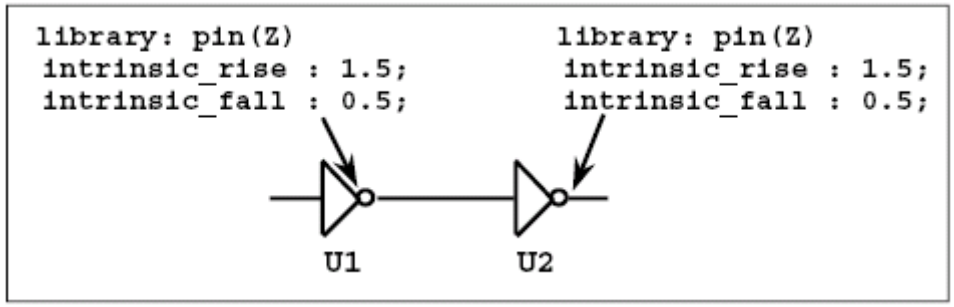
path_delay = (1.0 + 0.54 + 0.32 + 0.66 + 0.23 + 0.43 + 0.25) = 3.43 ns

Edge Sensitivity in Path Delays

What is the longest path for the circuit below?

What is the shortest?

Can we just add the longest delays and add the shortest delays?



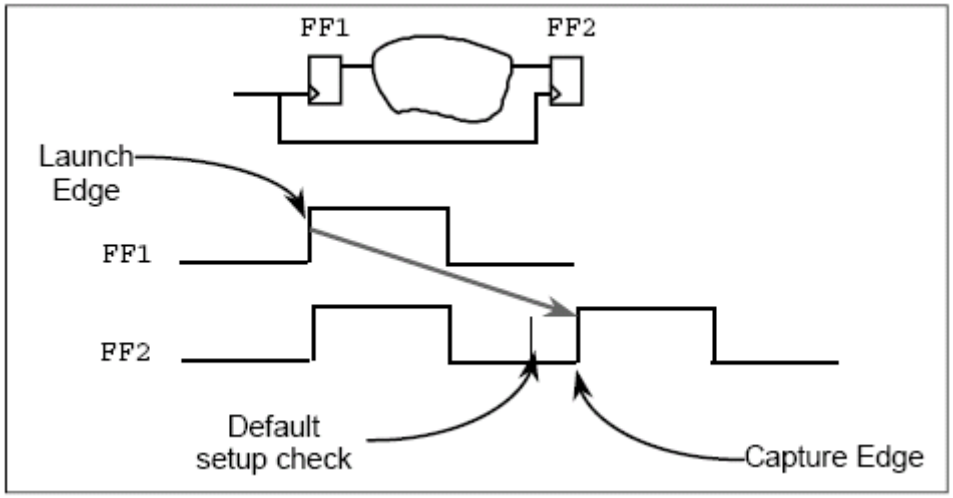
There is an “edge sensitivity” (called *unateness*) in a cell’s timing arc

- Design Time Keeps track of unateness in each timing path

Setup Relationship Between Flip-Flops

The default behavior of DC is to assume that all data must get from launch to capture edge in one clock cycle

- The path between FF1 and FF2 has a max delay constraint of $T_{CLK} - FF2_{libSetup}$



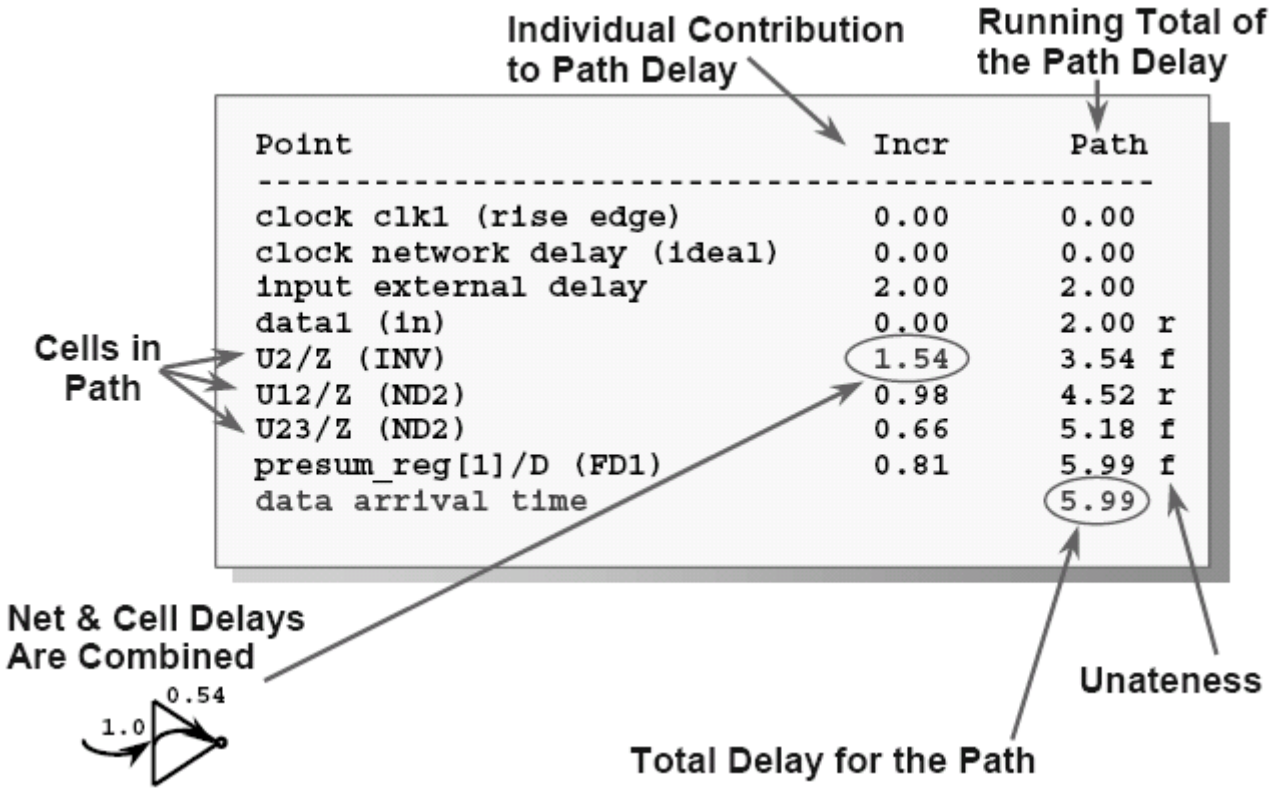
DesignTime Timing Reports

- Users will typically access DesignTime via the *report_timing* command
- The report_timing command
  - The design is broken down into individual timing paths
  - Each timing path is timed out twice:
    - once for a rising edge input, and
    - once for a falling edge input
  - The critical path (worst violator) for each clock group is found
  - A timing report for each clock group is echoed to the screen
- A DesignTime timing report has 4 major sections

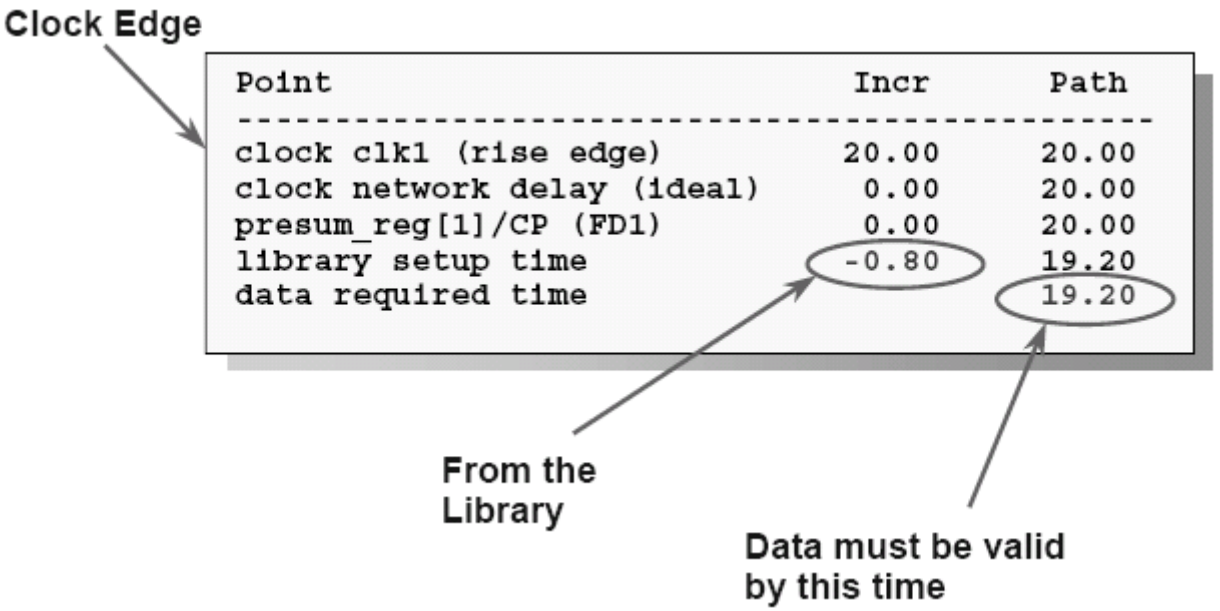
Timing Report: Path Information Section

```
****************************************
Report : timing
         -path full
         -delay max
         -max_paths 1
Design : RISC_CORE
Version: 1998.08
Date   : Wed Aug 20 20:43:45 1998
****************************************

Operating Conditions: slow_70_4.50   Library: cba_opcon
Wire Loading Model Mode: top

Design            Wire Loading Model      Library
------------------------------------------------
RISC_CORE               tc6a120m2         cba_core


  Startpoint: data1
              (input port clocked by clk1)
  Endpoint: presum_reg[1]/D
            (rising edge-triggered flip-flop clocked by clk1)
  Path Group: clk1
  Path Type: max
```

Timing Report: Path Delay Section



Timing Report: Path Required Section

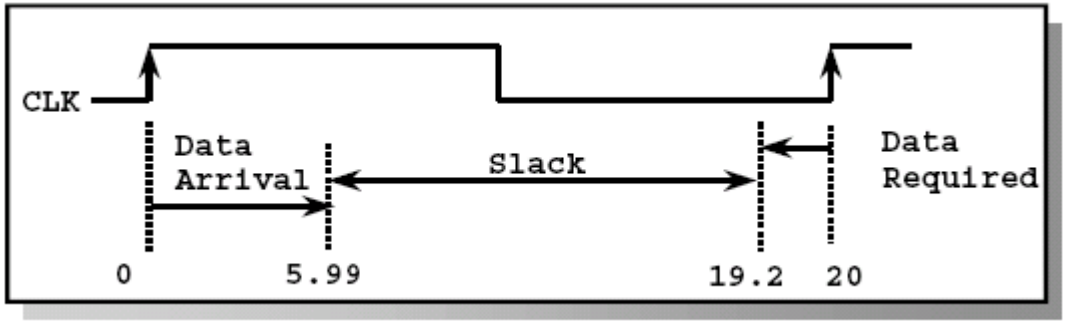


Timing Report: Summary Section

```
----------------------------------------------------
data required time                          19.20
data arrival time                           -5.99
----------------------------------------------------
slack (MET)                                 13.21
```

Either (MET) or (VIOLATED)

Timing margin (slack): negative indicates constraint violation



Example: setup check

report_timing

```
Startpoint: presum_reg[1] (rising edge-triggered flip-flop clocked by clk1)
Endpoint : preout_reg (rising edge-triggered flip-flop clocked by clk1)
Path Group : clk1
Path Type : max

Point                                    Incr         Path
-----------------------------------------------------------
clock clk1 (rise edge)                   0.00         0.00
clock network delay (ideal)              0.00         0.00
presum_reg[1]/CP                         0.00         0.00 r
presum_reg[1]/Q                          2.79         2.79 f
U23/Z (EO)                               3.66         6.45 f
U35/Z (EO)                               3.54         9.99 f
preout_reg/D (FD1)                       0.81        10.80 f
data arrival time                                    10.80

clock clk1 (rise edge)                  20.00        20.00
clock network delay (ideal)              0.00        20.00
preout_reg/CP (FD1)                      0.00        20.00 r
library setup time                      -0.80        19.20
data required time                                   19.20
-----------------------------------------------------------
data required time                                   19.20
data arrival time                                   -10.80
-----------------------------------------------------------
slack (MET)                                          8.40
```

The End

电子设计自动化作业40分

1. 如果你是老师，你会如何讲这门课？5分
2. 分别用unique和priority修饰符描述课本外的一个硬件功能 10分
3. 描述一个60人的班级、每人6门课、并包含各种名称、成绩、名次等的数据结构，给出排序成绩 25分
4. 给出一种总线的接口描述 20分
5. 给出一个层次化硬件描述的例子 10分
6. 给出逻辑综合的概念 5分
7. 给出IC的设计流程，给出流程中要进行何种验证 10分
8. 给出一个包括时序约束、面积约束、环境约束的例子 15分