北京大学 PEKING UNIVERSITY

AI ASIC: Design and Practice

(ADaP)

Fall 2024

HDL Simulators & Logic Synthesis

燕博南

Outline

北京大学 PEKING UNIVERSITY

- Verilog HDL Simulators
- Logic Synthesis - A instrumentalism Perspective
  - How to synthesis RTL code to logic gates?

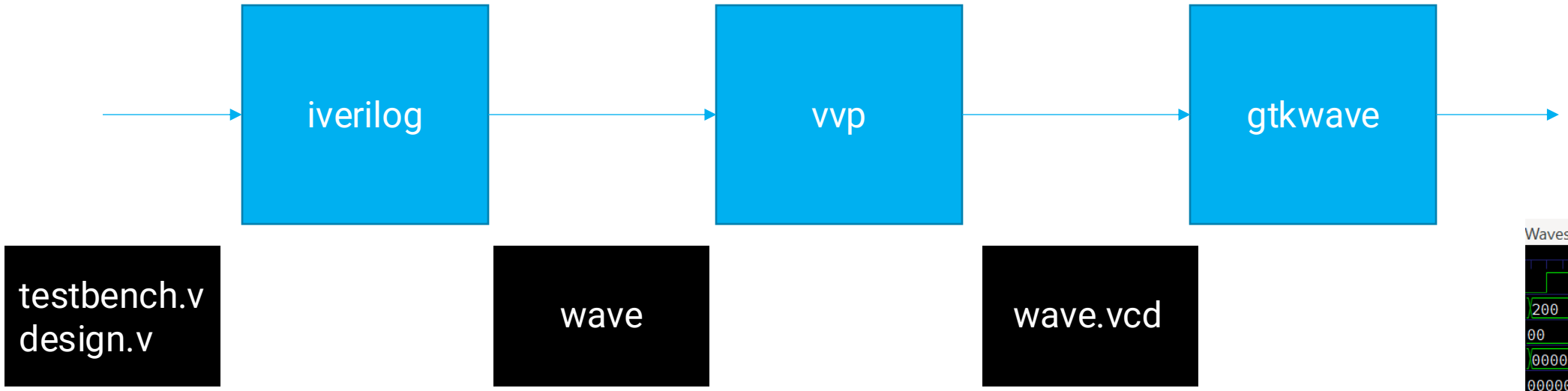
RTL: Register Transfer Level

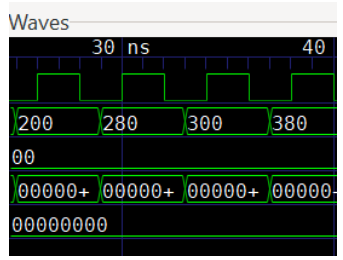
北京大学
PEKING UNIVERSITY

Part 1

Open-source simulators for HDL

Verilog Simulator Workflow





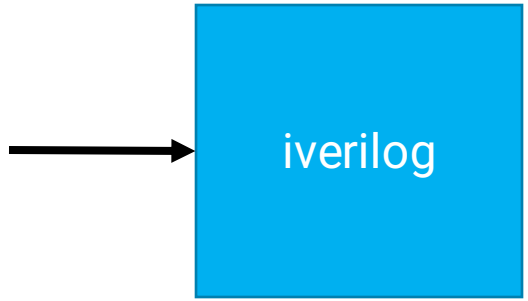
Runnable commend lines:

```
1   iverilog -o wave testbench.v
2   vvp -n wave -lxt2
3   gtkwave wave.vcd -S plot.tcl
```

Verilog Simulator Workflow

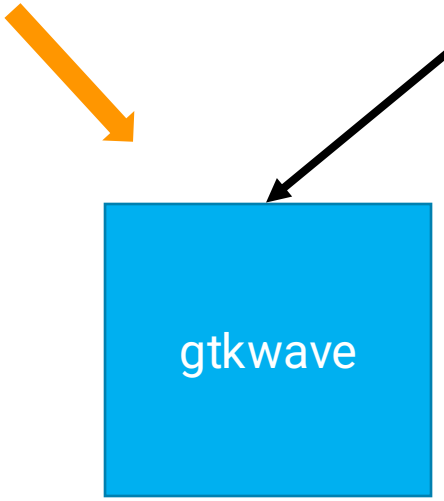
```
`include “design.v”
Module testbench;

…

design u1 (…);

endmodule
```

testbench.v



plot.tcl

```
1   set sigs [list]
2   lappend sigs "__bug_marker__"
3   lappend sigs "testbench.CLK\[0\]"
4   lappend sigs "testbench.A\[11:0\]"
5
6   set added [ gtkwave::addSignalsFromList $sigs ]
```



No need to drag outputs every time after running

Runnable command lines:

```
1   iverilog -o wave testbench.v
2   vvp -n wave -lxt2
3   gtkwave wave.vcd -S plot.tcl
```

A good tutorial:
全平台轻量开源verilog仿真工具iverilog+GTKWave使用教程

Other simulators

北京大学 PEKING UNIVERSITY

- Verilator
- PyRTL
- VCS
- Incisive
- Modelsim
- …

What are their differences?

北京大学
PEKING UNIVERSITY

Part 2

Logic Synthesis

Why synthesis

- Definition: Synthesize logic circuits into gates (gate-level netlist)
- Quick (re)design time
- Separate functionality, technology-dependent parameters, and design constraints
- Fast timing/area/power estimates

Why "synthesis" works? Why can we synthesize logic?

A Symbolic Analysis of Relay and Switching Circuits

By CLAUDE E. SHANNON
ENROLLED STUDENT AIEE

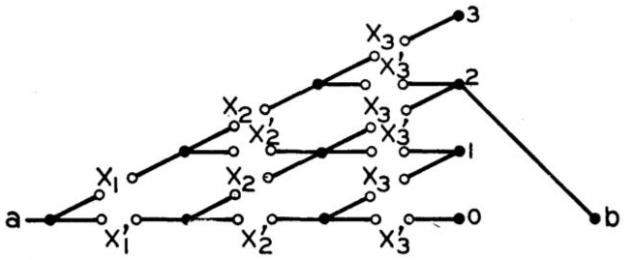


Figure 23. Circuit for realizing $S_2(X_1, X_2, X_3)$

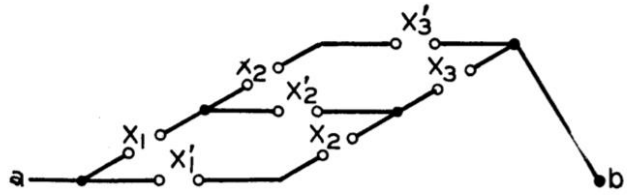
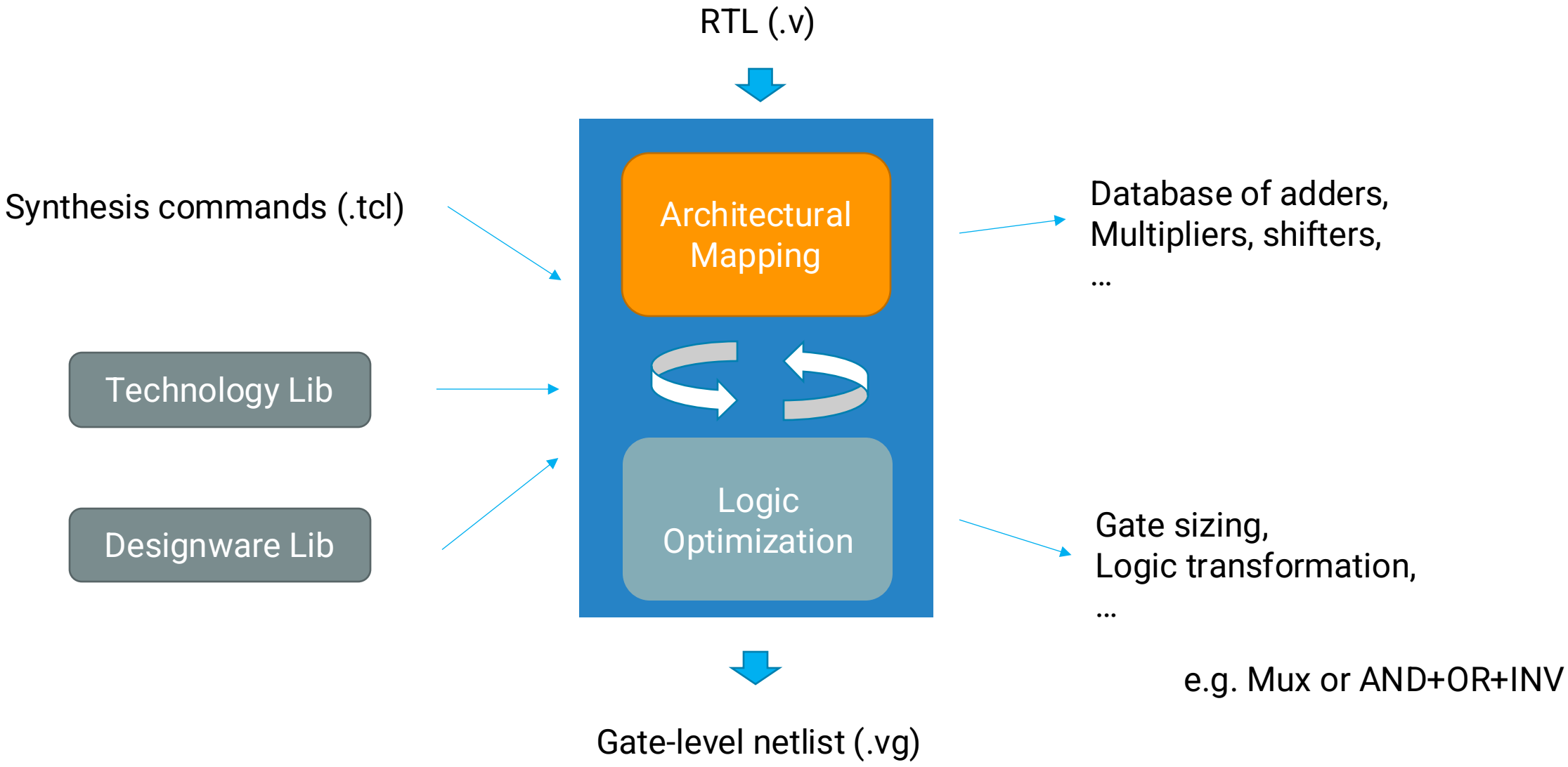


Figure 24. Simplification of figure 23

How to do logic synthesis?



Outputs of logic synthesis?

- Timing/Area/Power reports
- Gate-level netlist (.vg) & timing info. In standard delay format (.sdf) for timing-aware simulation

- Pre-syn (RTL) sim needs: testbench.v, design.v
- Post-syn sim needs: testbench.v, design.vg, design.sdf, stdlib_cells.v

e.g. Assign C=A+B

Synthesis Tool: Synopsys Design Compiler

- dc_shell-t or dc_shell-xg-t
- Command line approach (no gui)
- Other synthesis tools also work fine:
  - Cadence Genus
  - Open Road: yosys

Script Example (.tcl file)

```
1   remove_design -all
2   source ./SET_CON/T40GP_LibSetup.tcl
3   lappend search_path ./HDL ./SYN
4   analyze -format verilog Adder.v
5   set PROCESS _T40GP
6   set DESIGN_NAME ADD
7   elaborate $DESIGN_NAME
8   link
9   set_operating_conditions -min ff1p16vn40c -max ss0p95v125c
10  set_wire_load_selection_group "predcaps"
11  set TCLK 1.9
12  set TCU 0.1
13  source ./SET_CON/T40GP_VarSetup.tcl
14  set_false_path -from [get_ports ACLR_]
15  uniquify
16  compile -only_design_rule
17  compile_ultra -no_autoungroup
18  compile -inc -only_hold_time
19  set_fix_multiple_port_nets -all -buffer_constants [get_designs *]
20  remove_unconnected_ports -blast_buses [get_cells -hier]
21  report_timing -path full -delay min -max_paths 10 > $LOGPATH$TOPLEVEL$PROCESS.holdtiming
22  report_timing -path full -delay max -max_paths 10 > $LOGPATH$TOPLEVEL$PROCESS.setuptiming
23  report_area -hierarchy > $LOGPATH$TOPLEVEL$PROCESS.area
24  report_power -hier -hier_level 2 > $LOGPATH$TOPLEVEL$PROCESS.power
```

1. Setup technology library

2. Read RTL deign & do architecture mapping

3. Add constraints

4. Compile design

5. Get reports

Logic Synthesis

北京大学 PEKING UNIVERSITY

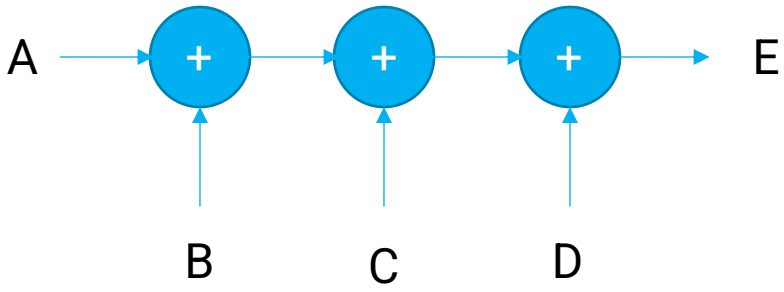
1. RTL coding
2. Technology lib setup
3. Constraints
4. Major synthesis commands
5. Gate-level simulation

1. **RTL coding**
2. Technology lib setup
3. Constraints
4. Major synthesis commands
5. Gate-level simulation

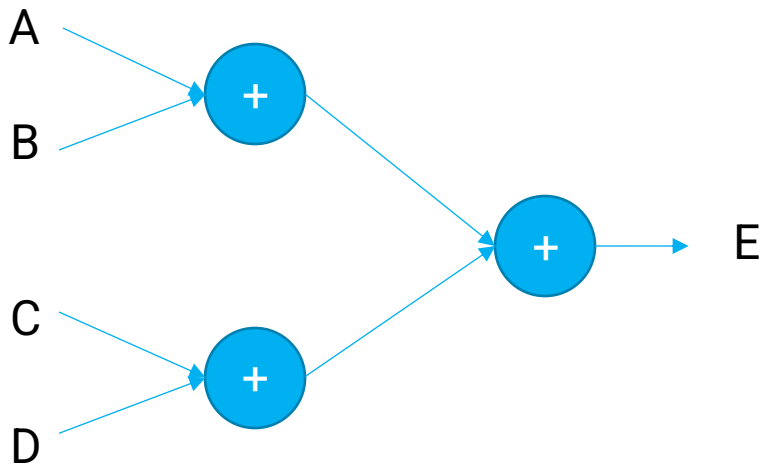
Write Good HDL

- Different coding styles that are functionally eqvl. may be mapped into HWs having different timing/area

E=A+B+C+D



E=(A+B)+(C+D)



Which is better?

Logic Synthesis

北京大学 PEKING UNIVERSITY

1. RTL coding
2. **Technology lib setup**
3. Constraints
4. Major synthesis commands
5. Gate-level simulation

Technology Lib is about:

- Process
- Operating Condition (ss/tt/ff design corner, temperature, VDD voltage)
- Cell Name
- Area
- Functionality
- Pin Capacitance
- Leakage Power
- Look-up Tables (LUT) for Dynamic Power & Propagation Delay of all i/o Paths
- (idx1: input transition time; idx2: o/p capacitance)
- …

Setup Tech Lib

PEKING UNIVERSITY

```
1   set search_path “$search_path \
2   /w/apps2/public.2/tech/synopsys/32-28nm/SAED32_EDK/lib/stdcell_rvt/db_nldm”
3   set target_library “saed32rvt_ff1p16vn40c.db saed32rvt_ss0p95v125c.db”
4   set link_library “* saed32rvt_ff1p16vn40c.db saed32rvt_ss0p95v125c.db dw_foundation.sldb”
5   set synthetic_library “dw_foundation.sldb”
6
7   # Define work path (note: The work path must exist, so you need to create a folder WORK first)
8   define_design_lib WORK -path ./WORK
9   set alib_library_analysis_path “./alib-52/”
```

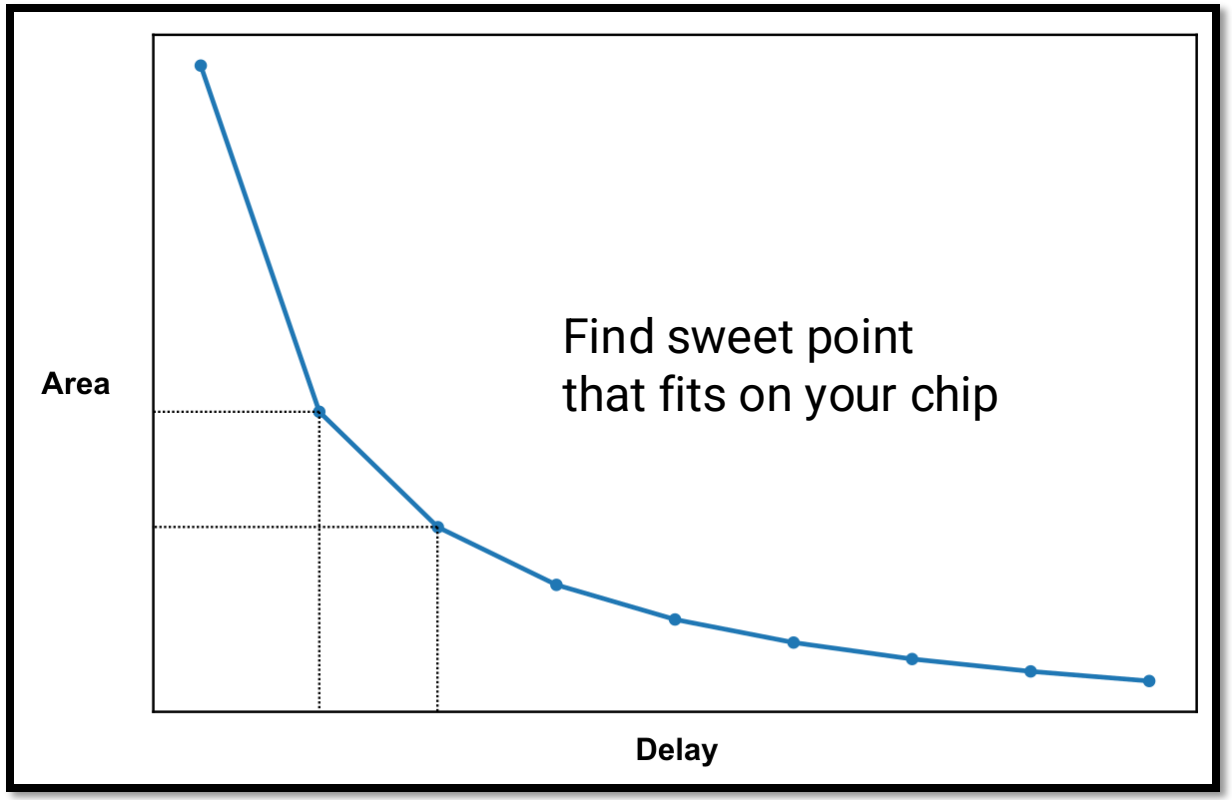
- DC follows **search_path** to find libraries you specify
- DC uses cells in target library for logic optimization (so we need to do **set_target_library** first)
- **Remember:** Use backslash **** before changing lines to avoid compilation errors.

Logic Synthesis

北京大学 PEKING UNIVERSITY

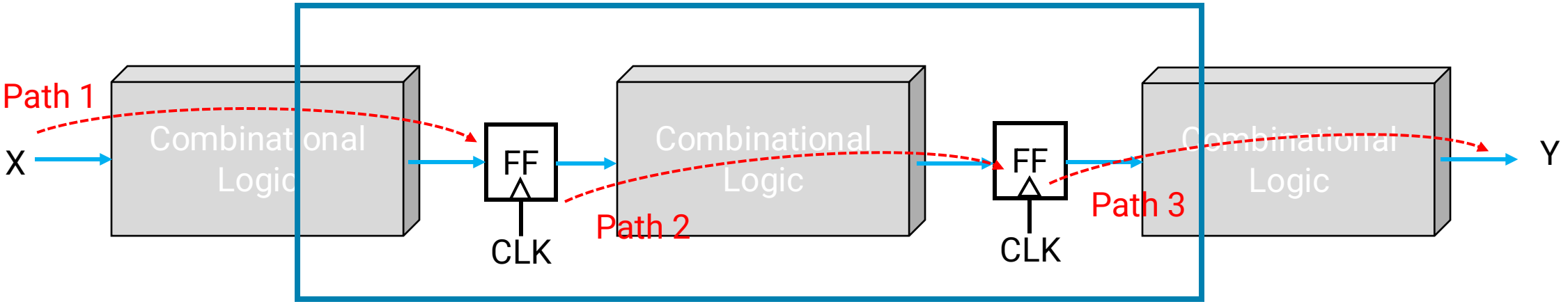
1. RTL coding
2. Technology lib setup
3. **Constraints**
4. Major synthesis commands
5. Gate-level simulation

Synthesis is Constraint-driven



- You set the goals (through **design constraints**)
- DC optimizes the design to meet your goals

Timing Analysis for VLSI



"set_input_delay" : affects input logic

"create_clock": affects internal logic

"set_output_delay": affects output logic

Input delay: Arrival of an external path w.r.t. the clock edge
Output delay: timing from an output port of the current design to a register input port of other modules

Describe Constraints

- **Clocks**
  - Period, latency, uncertainty
- **Design rules**
  - Maximum transition
  - Maximum capacitance
  - Maximum fanout
- **Input-related**
  - Driving cells, Input delay
- **Output-related**
  - Load, Output delay
- **Exception paths**
  - False paths, multi-cycle paths
- **Optimization goal**
  - Maximum area/power

 # Accurate constraints
(not too tight/loose):
Good integ. results

Clock Description

PEKING UNIVERSITY

- create_clock: define clock's waveform (e.g. period)

```
create_clock -name "sysclk" -period 2.0 [get_ports "CLK"]
```

- set_fix_hold: respect the hold time requirement of all clocked flip-flops

```
set_fix_hold sysclk
```

- set_dont_touch_network: do not buffer clock network

```
set_dont_touch_network [get_clocks "sysclk"]
```

- Specify uncertainty (skew + jitter) of clock network

```
set_clock_uncertainty 0.1 [get_clocks "sysclk"]
```

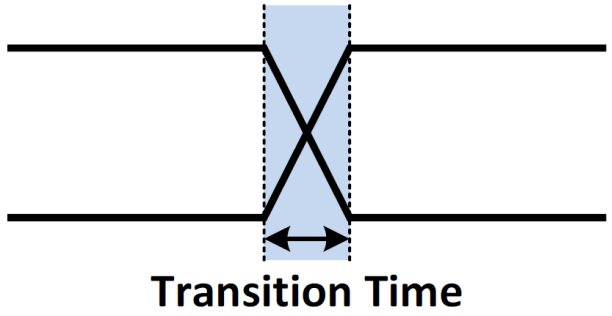
- Sets the max_transition attribute to a specified value on specified clocks group, ports or designs

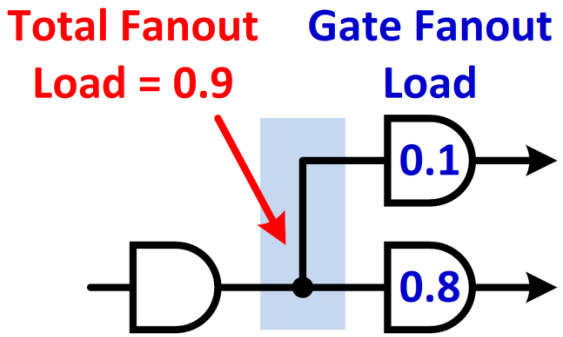
```
set_max_transition 0.1 current_design
```

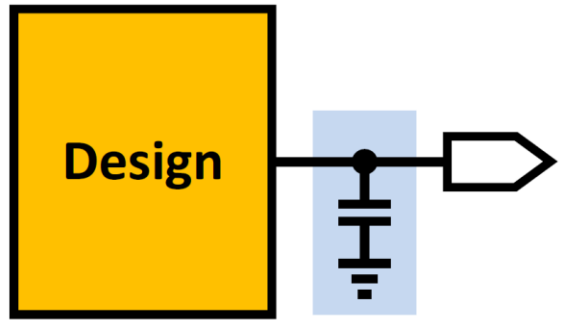
Constraints: Max transitions, fanout & output loads

- Sets the max_transition, fanout load & output load attribute :

```
set_max_transition 0.1 current_design

set_max_fanout 1.0 current_design

set_load 1.0 [all_outputs]
```







Constraints: Driving Cells

- Sets attributes on input or inout ports of the current design, specifying that a library cell or output pin of a library cell drives specified ports:

```
set_driving_cell [-library lib] [-lib_cell lib_cell_name] [-pin pin_name]
```

```
set ALL_IN_BUT_CLK [remove_from_collection [all_inputs] “CLK"]

set_driving_cell -no_design_rule –library \
        saed32rvt_ss0p95v125c.db:saed32rvt_ss0p95v125c -lib_cell \
        DFFASRX2_RVT -pin Q $ALL_IN_BUT_CLK
```

Constraints: False Paths

- Removes timing constraints from particular paths, but still needs to satisfy design rule (transition, capacitance, fanout):

```
set_false_path –from [from_list]
```

```
set_false_path -from [get_ports “TEST_OUT1”]
```

Constraints: Maximum Area/Power

- Optimization goals for your design (DC will do it best to satisfy them, w/o violating the three design rules):

desired area

```
set_max_area 0.0
```

um^2

```
set_max_total_power 0.0
```

uW

desired power

```
set_max_area 10000
set_max_total_power 100
```

Constraints: Operating Condition

- Defines the operating conditions for the current design:

```
set_operating_conditions [-min min_condition] [-max max_condition]
```

hold time check condition

Setup time check condition

```
set_operating_conditions -min ff1p16vn40c -max ss0p95v125c
```

Constraints: Wire-load Models and Modes

- Specify a selection group to use for determining a wire load model to be assigned to designs and cells or to a specified cluster:

```
set_wire_load_selection_group group_name [-max] [-min]
```

```
set_wire_load_selection_group group_name “predcaps”
```

DC will select proper model based on synthesis area (again, lookup this name in .lib file)

1. RTL coding
2. Technology lib setup
3. Constraints
4. **Major synthesis commands**
5. Gate-level simulation

Commands Preview

PEKING UNIVERSITY

- Not all commands introduced are necessarily needed
- Things MUST do

Analyze, elaborate, and link design
Compile design

- Things can do based on your needs

Ungroup; Uniquify
Clock gating creation
Remove unconnected ports

Suggestion: Learn by playing w/ different commands and observing the differences

Analyze Designs:

北京大学 PEKING UNIVERSITY

- Analyzes the HDL files and stores the intermediate format in the specified library.

```
analyze [-format vhdl | verilog | sverilog] file_list
```

```
analyze –format Verilog {adder.v}
elaborate
```

Or

```
read_verilog adder.v
```

Elabrate & Link Designs

北京大学 PEKING UNIVERSITY

- Builds a design from the intermediate format of a Verilog module, a VHDL entity and architecture, or a VHDL configuration:

```
set design_name adder
elaborate $design_name
```

or

```
link
```

Ungroup & Uniquify

北京大学 PEKING UNIVERSITY

- Removes a level of hierarchy:

```
ungroup cell_list | -all [-flatten]
```

```
ungroup –flatten -all
```

- Removes multiple-instantiated hierarchy in the current design by creating a unique design for each of the cell instances:

```
uniquify
```

Compile

- Performs logic-level and gate-level synthesis and optimization on the current design:

```
compile [-no_design_rule | -only_design_rule | -only_hold_time] [-
        map_effort medium | high] [-boundary_optimization]
```

```
compile –map_effort high
```

```
compile_ultra
```

Remove Unconnected Ports

北京大学 PEKING UNIVERSITY

- Removes unconnected ports or pins from cells, references, and sub-designs

```
remove_unconnected_ports -blast_buses [get_cells -hier]
```

Report Designs

北京大学 PEKING UNIVERSITY

```
report_timing -path full -delay min -max_paths 10 -nworst 2 > Design.holdtiming
report_timing -path full -delay max -max_paths 10 -nworst 2 > Design.setuptiming
report_area -hierarchy > Design.area
report_power -hier -hier_level 2 > Design.power
report_resources > Design.resources
report_constraint -verbose > Design.constraint
check_design > Design.check_design
check_timing > Design.check_timing
```

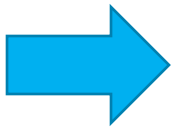
Logic Synthesis

1. RTL coding
2. Technology lib setup
3. Constraints
4. Major synthesis commands
5. **Gate-level simulation**

Slight Modification of Original Testbench.v

- RTL simulation: **design.v** + **testbench.v**
- Gate-level simulation:
  - **design.vg**
  - **design.sdf** (from DC)
  - **testbench.v** (w/ slight modification)

```
`include “design.v”
Module Testbench;

….

Design u1 (…);

…



endmodule
```



```
`include “design.vg”
`include “stdlib_cells.v”
Module Testbench;
….

initial $sdf_annotate(“design.sdf”, u2);

Design u2 (…);

…

endmodule
```

DC Help

- If you meet any questions about some commands
- Use

```
man set_input_delay
```

- Or

- Search on Baidu/Bing/Google/…

Summary

PEKING UNIVERSITY

- Synthesis tool makes VLSI design a lot easier
  - Easy to use: **.v** files + **.tcl** files
  - Easy to switch designs to any technology by changing associated libraries
  - Easy to have accurate area/timing/power estimate
  - Easy to match the best archit. with design constraints

- **But… not a substitute for thinking**
  - Mind your coding styles
  - Handmade optimization (e.g. using shift-and-add for variables' multiplication) still needed