ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN-ĐIỆN TỬ
BỘ MÔN KỸ THUẬT ĐIỆN TỬ

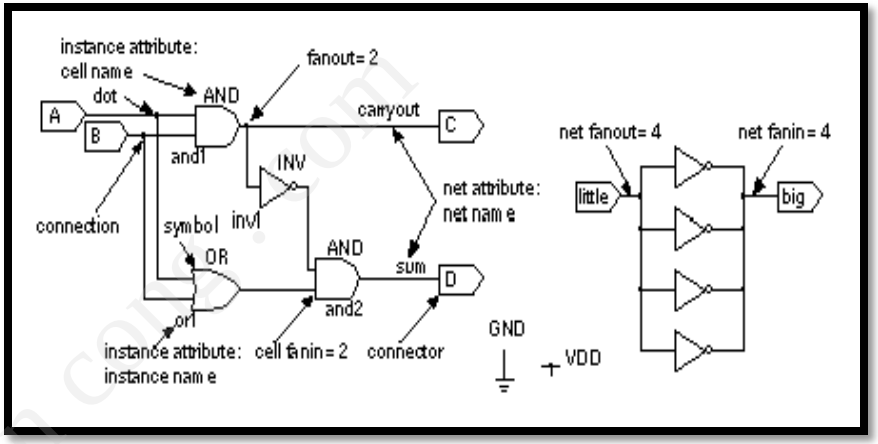
ASIC CHIP AND IP CORE DESIGN

Chapter 3: Architecture Design

3.1 Design Entry
3.2 Design Constraints
3.3 Optimizing the design
3.4 Hardware Description Languages

3.1 Design Entry

- Design entry:
  - Graphical tools:
    - Schematic entry
  - HDLs tools:
    - Verilog
    - VHDL

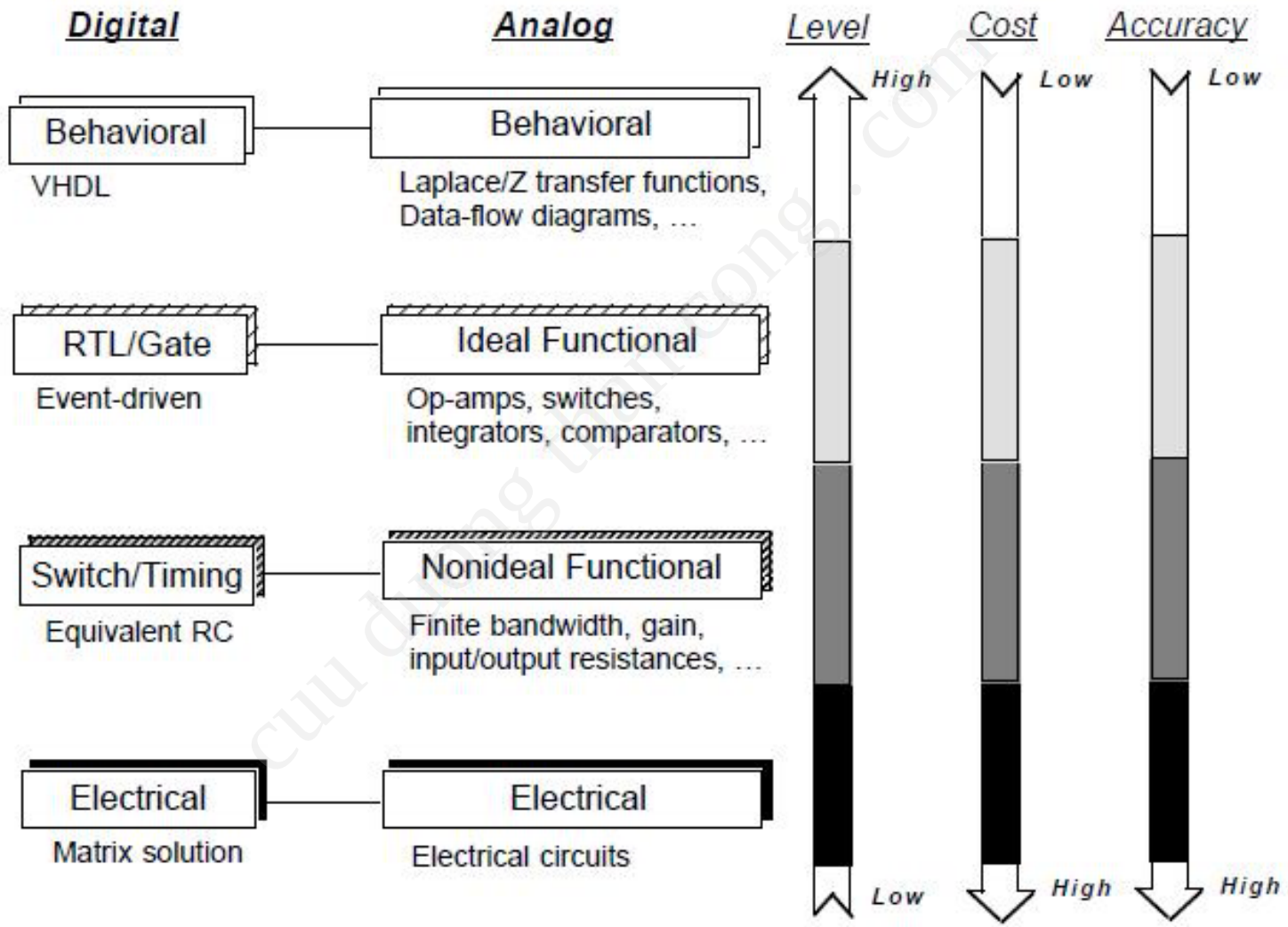


Schematic entry

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY logic IS
PORT( a,b,c : IN std_logic;
              x : OUT std_logic);
 END logic;
 ARCHITECTURE a OF logic IS
BEGIN
PROCESS (a,b,c)
   BEGIN
        x<=(a and b) or c;
   END PROCESS;
END;
```

HDL code entry

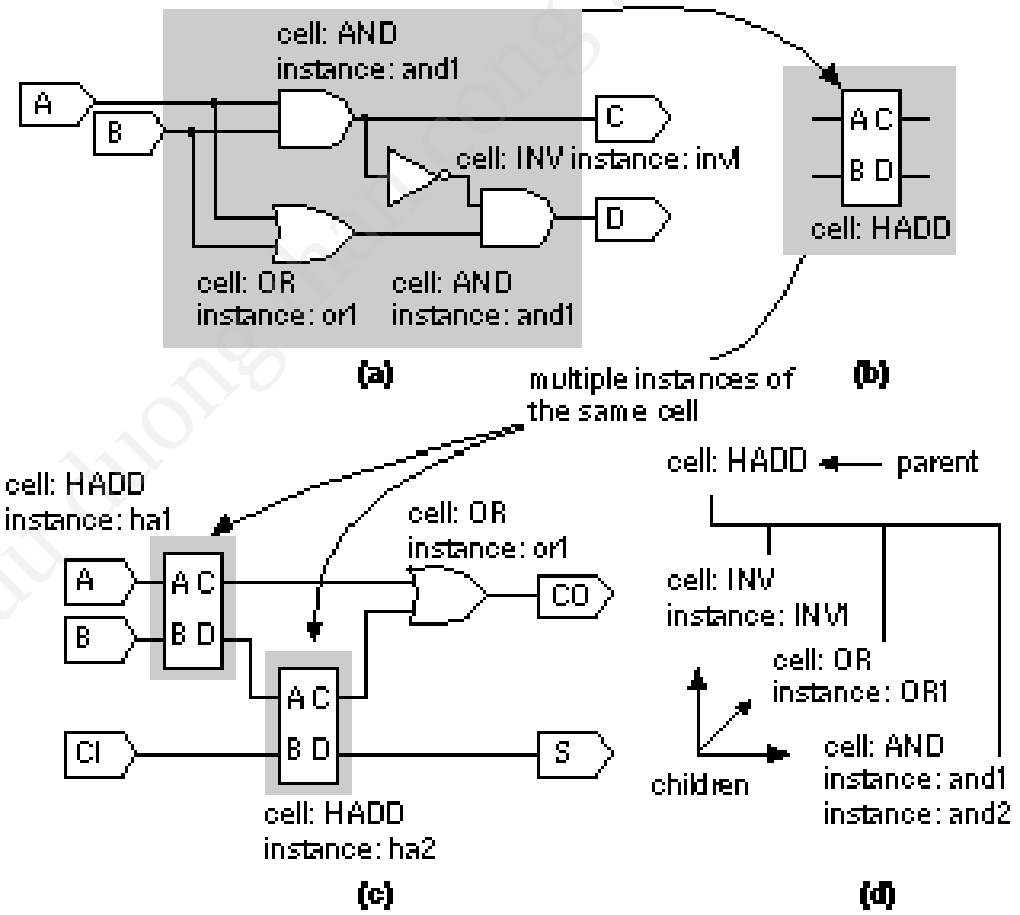
Level of Abstraction in IC Design



Hierarchical Design

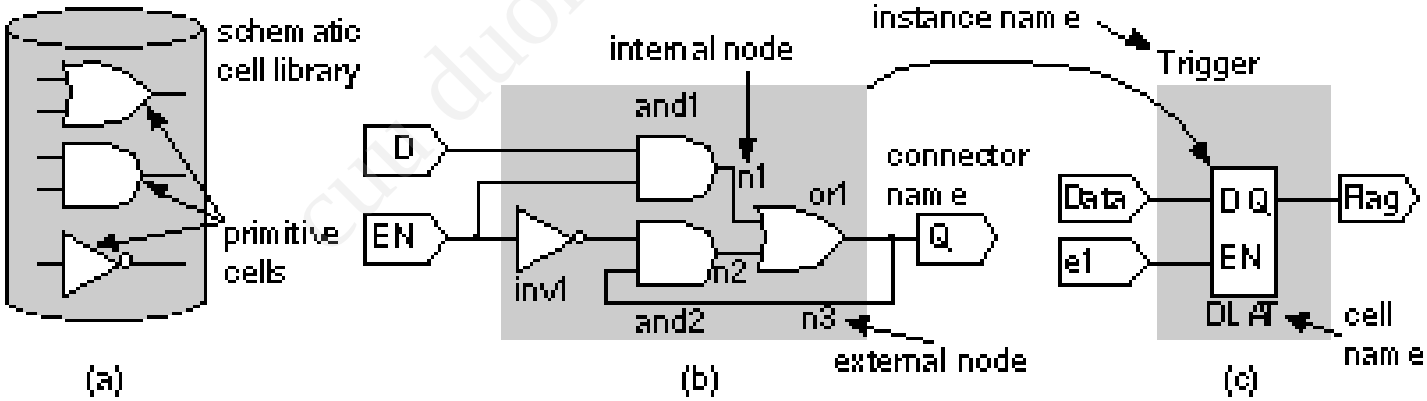
- **Hierarchy** reduces the size and complexity of a schematic
- **Subschematic** is a child of the parent schematic

Hierarchical design of half adder



Schematic Icon & Symbol

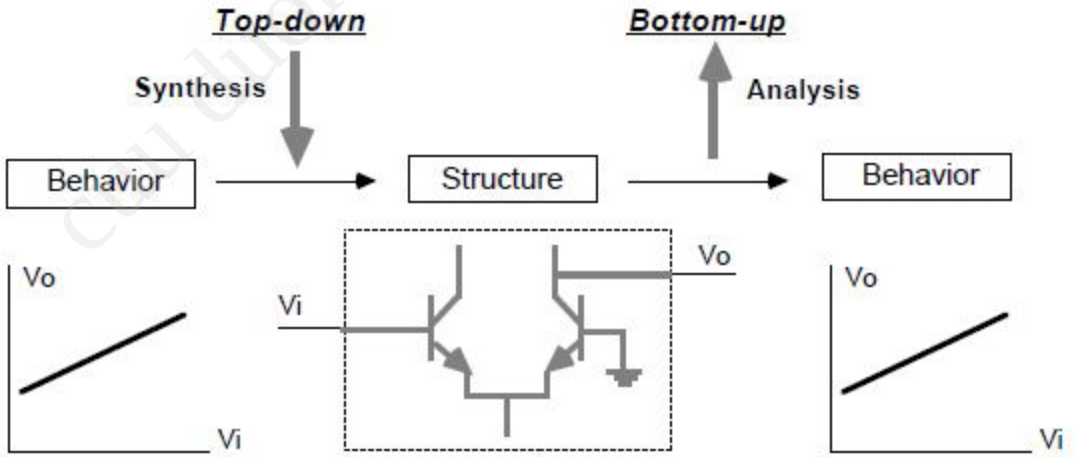
- Most schematic-entry programs allow the designer to draw special or custom icons.
- In addition, the schematic-entry tool will also usually create an icon automatically for a subschematic that is used in a higher-level schematic.
- This is a derived icon , or derived symbol .
- The external connections of the subschematic are automatically attached to the icon, usually a rectangle.



Hierarchical Design

Two fundamental approaches of a design

- Bottom-up
  - Each module is individually synthesized
  - Uses manual time budgets or a default budget
  - May not produce optimal design
  - Prone to error in manually specified time budget
- Top-Down
  - Entire design hierarchy is optimized together
  - Only top level time budgets are required
  - Better results due to optimization across entire design



Class Assignments

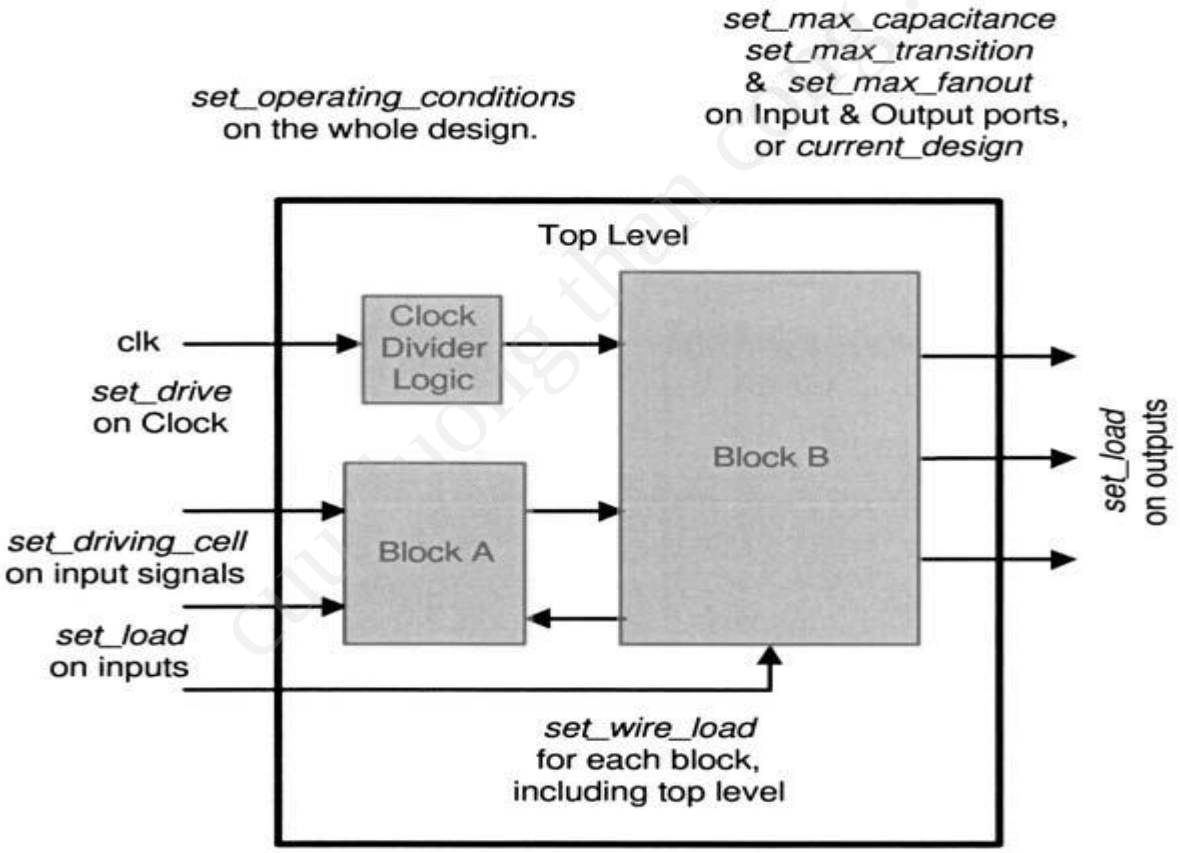
1. Design the following digital circuit by behavioral model and RTL/gate model
   a) Multiplexer 8 to 1
   b) Decoder 3 to 8
   c) 3-bit comparator
2. Design the following digital circuit by using hierarchical methodology
   a) 1D filter with 5 taps
   b) 2D filter using 1D filter unit for 8x8 block
   c) Image filter for 3x3 block

3.2 Design Constraints

- In order to obtain optimum design, designers have to methodically constrain their designs by
  - Design environment:
    - Set library
    - Set operating condition
    - Set wire load model
  - Target objectives
    - Set clock latency/transition/uncertainty
    - Set input/output delay
  - Design rules
    - Set don’t touch network
    - Set max transition

Design Environment

- **set_min_library:** specify the worst-case and the best case library
  - set_min_library <max library filename> –min_version <min library filename>
  - dc_shell -t> set_min_library “ex25_worst.db” –min_version “ex25_best.db”



Design Environment

- **Set_operating_conditions:** describe the process, voltage, and temperature conditions of the design
  - set_operating_conditions <name of operating conditions>
- Examples:
  - dc_shell-t> set_operating_conditions –min BEST –max WORST
  - dc_shell-t> set_operating_conditions –min BEST –max WORST
- **set_wire_load_model:** provide estimated statistical wire-load information to DC
  - set_wire_load_model -name<wire-load model>
- Examples:
  - dc_shell -t> set_wire_load_model -name MEDIUM
- **set_wire_load_mode** defines the three modes associated for modeling wire loads
  - dc_shell -t> set_wire_load_mode top

Design Environment

- **set_drive:** specify the drive strength at the input port
- **set_driving_cell** is used to model the drive resistance of the driving cell to the input ports.
  - set_drive <value> <object list>
  - set_driving_cell –cell <cell name> –pin <pin name> <object list>
- Examples:
  - set_drive 0 {CLK RST}
  - set_drive 1.5 {I1 I2}
  - set_driving_cell –cell BUFF1 –pin Z [all_inputs]
- **set_load:** sets the capacitive load in the units defined in the technology library (usually pico Farads - pF)
  - set_load <value> <object list>
  - set_load 1.5 [all_outputs]
  - set_load 0.3 [get_nets blockA/n1234]

Design constraints

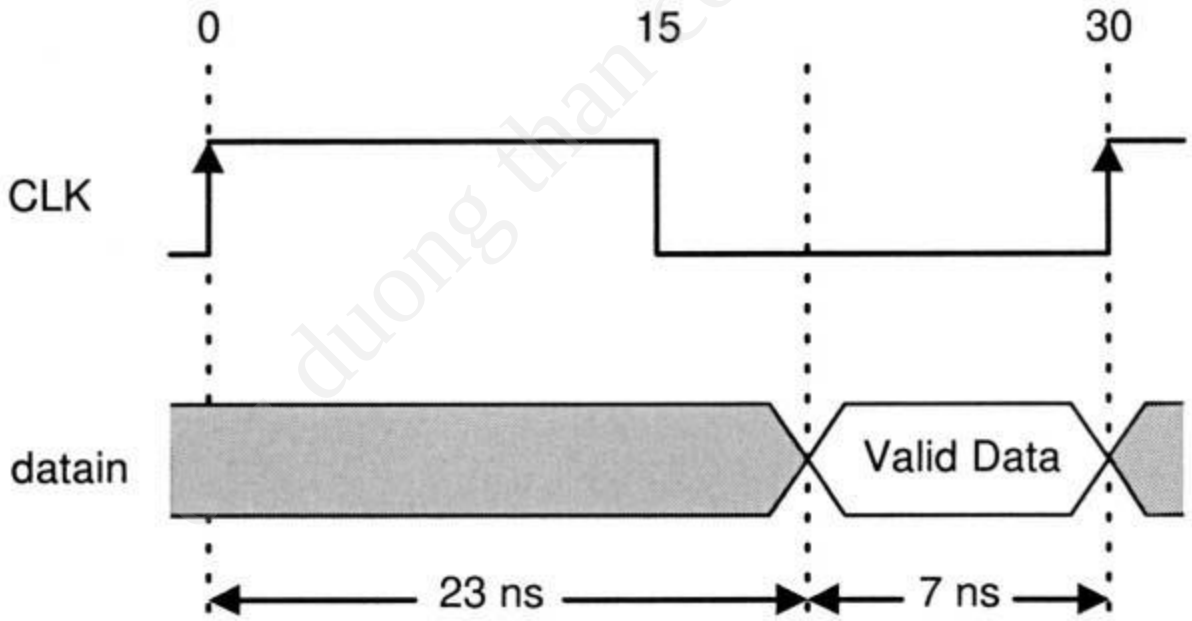
- **create_clock**: is used to define a clock object with a particular period and waveform.
  - –period option: defines the clock period
  - –waveform option: controls the duty cycle and the starting edge of the clock.
  - dc_shell-t> create_clock–period 40 –waveform [list 0 20] CLK
- **create_generated_clock**: is used for clocks that are generated internal to the design
  - create_generated_clock –name <clock name> –source <clock source> -divide_by <factor> | –multiply_by <factor>
- **set_dont_touch_network** is used for clock networks and resets to prevent DC from buffering the net, in order to meet DRCs
  - dc_shell-t> set_dont_touch_network {CLK, RST}

Design constraints

- **set_dont_touch** is used to set a dont_touch property on the current_design, cells, references or nets.
  - dc_shell-t> set_dont_touch current_design
  - dc_shell-t> set_dont_touch [get_cells sub1]
  - dc_shell -t> set_dont_touch [get_nets gated_rst]
- **set_dont_use** command is generally set in the .synopsys_dc.setup environment file.
  - dc_shell -t> set_dont_use [list mylib/SDFF* mylib/RSFF*]

Design constraints

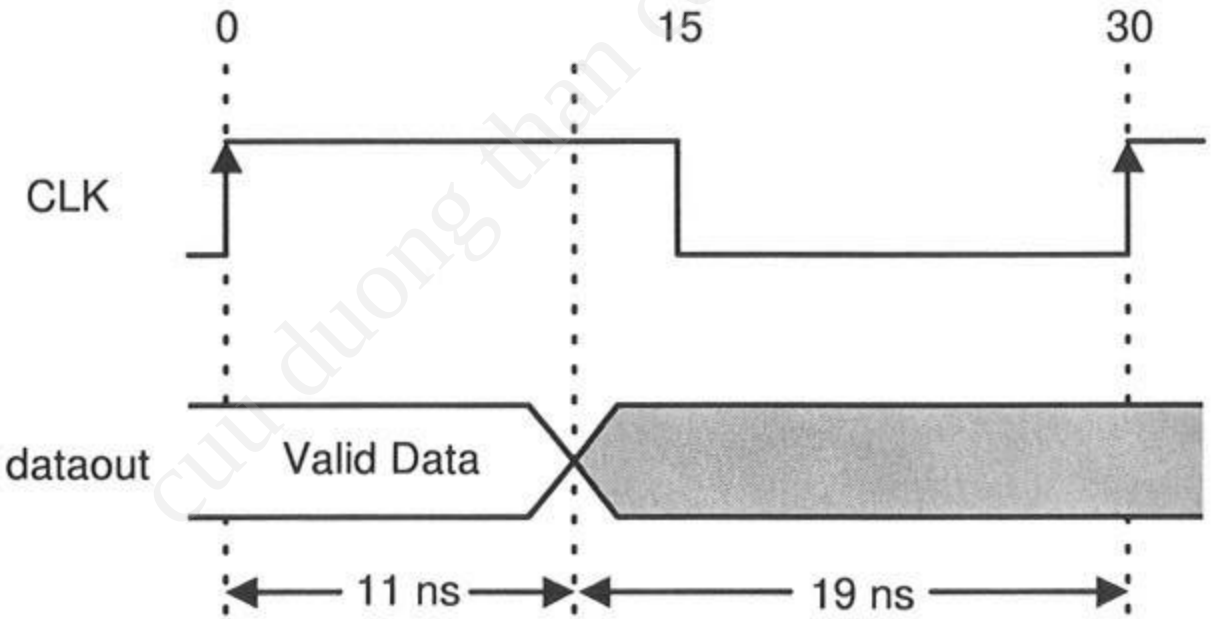
- set_input_delay specifies the input arrival time of a signal in relation to the clock.
  - dc_shell-t> set_input_delay –max 23.0 –clock CLK {datain}
  - dc_shell-t> set_input_delay –min 0.0 –clock CLK {datain}



Specification of Input Delay

Design constraints

- set_output_delay command is used at the output port, to define the time it takes for the data to be available before the clock edge.
  - dc_shell-t> set_output_delay –max 19.0 –clock CLK {dataout}



Specification of Output Delay

Design constraints

- set_clock_latency command is used to define the estimated clock insertion delay during synthesis. This is primarily used during the prelayout synthesis and timing analysis.
  - dc_shell -t> set_clock_latency 3.0 [get_clocks CLK]
- set_clock_uncertainty command lets the user define the clock skew information. Basically this is used to add a certain amount of margin to the clock, both for setup and hold times.
  - dc_shell -t> set_clock_uncertainty –setup 0.5 –hold 0.25 [get_clocks CLK]
- set_clock_transition for some reason does not get as much attention as it deserves. However, this is a very useful command, used during the pre-layout synthesis, and for timing analysis.
  - dc_shell -t> set_clock_transition 0.3 [get_clocks CLK]

Design constraints

- **set_propagated_clock** is used during the post layout phase when the design has undergone the insertion of the clock tree network. In this case, the latency is derived using the traditional method of delay calculation.
  - dc_shell -t> set_propagated_clock [get_clocks CLK]
- **set_false_path** is used to instruct DC to ignore a particular path for timing or optimization.
  - dc_shell -t> set_false_path –from in1 –through U1/Z –to out1
- **set_multicycle_path** is used to inform DC regarding the number of clock cycles a particular path requires in order to reach its endpoint.
  - dc_shell-t> set_multicycle_path 2 –from U1/Z –through U2/A –to out1

Design Rules

- **set_max_delay** defines the maximum delay required in terms of time units for a particular path.
  - dc_shell -t> set_max_delay 0 –from CK2 –to [all_registers –clock_pins]
- **set_min_delay** is the opposite of the set_max_delay command, and is used to define the minimum delay required in terms of time units for a particular path
  - dc_shell-t> set_min_delay 3 –from [all_inputs] –to [all_outputs]
- **group_path** command is used to bundle together timing critical paths in a design
  - dc_shell-t> group_path –to [list out1 out2] –name grp1

Design Rules

- **Set_max_transition <value> <object list>**
  - dc_shell-t> Set_max_transition 0.3 current_design
- **Set_max_fanout <value> <object list>**
  - dc_shell-t> Set_max_fanout 3.0 [all_outputs]
- **Set_max_capacitance <value> <object list>**
  - dc_shell-t> Set_max_capacitance 1.5 [get_port out1]

Clocking Issues

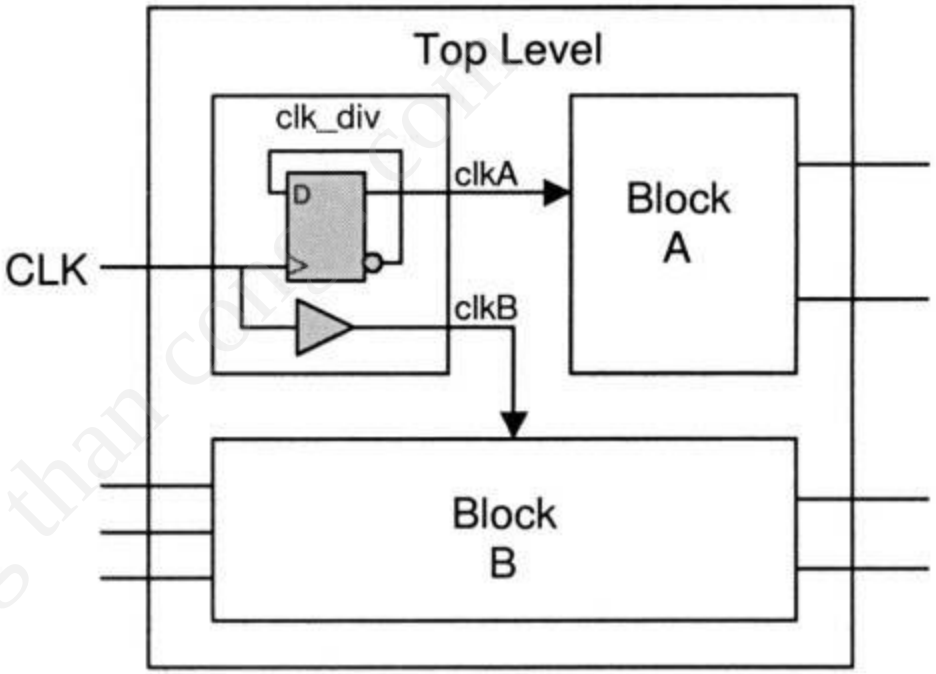
- **Clock issues for Pre-layout:** estimate the clock tree latency and skew
  - dc_shell-t> create_clock –period 40 –waveform {0 20}C LK
  - dc_shell-t> set_clock_latency 2.5 CLK
  - dc_shell-t> set_clock_uncertainty –setup 0.5 –hold 0.25 CLK
  - dc_shell -t> set_clock_transition 0.1 CLK
  - dc_shell -t> set_dont_touch_network CLK
  - dc_shell -t> set_drive 0 CLK

Clocking Issues

- **Clock issues for Post-layout:** the user does not need to worry about the clock latency and skews. They are determined by the quality of the routed clock tree.
  - dc_shell> create_clock –period 40 –waveform [list 0 20] CLK
  - dc_shell> set_propagated_clock CLK
  - dc_shell> set_clock_uncertainty –setup 0.25 –hold 0.05 CLK
  - dc_shell> set_dont_touch_network CLK
  - dc_shell> set_drive 0 CLK

Generated Clocks

A clock divider circuit in ***clk_div*** block, is used to divide the frequency of the primary clock *CLK by half, and* then generate the divided clock that drives ***Block A.*** The primary clock is also used to clock, ***Block B*** *and is buffered internally (in the clk_div block), before* feeding *Block B.*



- dc_shell> create_clock –period 40 –waveform {0 20} CLK
- dc_shell> create_clock –period 80 –waveform {0 40} find(port, “clk_div/clkA”)

- dc_shell-t> create_generated_clock –name clkA –source CLK –divide_by 2

Examples for Design Constraints

```
#-------------------------------------------------
# Design entry
analyze –format verilog sub1 .v
analyze –format verilog sub2.v
analyze –format verilog top_block.v
elaborate top_block
current_design top_block
uniquify
check_design
```

Examples for Design Constraints

```
# Setup operating conditions, wire load, clocks, resets
set_wire_load_model large_wl
set_wire_load_mode enclosed
set_operating_conditions WORST
create_clock –period 40 –waveform [list 0 20] CLK
set_clock_latency 2.0 [get_clocks CLK]
set_clock_uncertainty –setup 1.0 –hold 0.05 [get_clocks CLK]
set_dont_touch_network [list CLK RESET]
```

Examples for Design Constraints

```
#Input drives
set_driving_cell –cell [get_lib_cell buff3] –pin Z [all_inputs]
set_drive 0 [list CLK RST]
#-------------------------------------
#Output loads
set_load 0.5 [all_outputs]
#-------------------------------------
#Set input & output delays
set_input_delay 10.0 –clock CLK [all_inputs]
set_input_delay –max 19.0 –clock CLK { IN1 IN2}
set_input_delay –min –2.0 –clock CLK IN3
set_output_delay 10.0 –clock CLK [all_outputs]
```

Examples for Design Constraints

```
#Advanced constraints
group_path –from IN4 –to OUT2 –name grp1
set_false_path –from IN5 –to sub1/dat_reg*/*
set_multicycle_path 2 –from sub1/addr_reg/CP \
–to sub2/mem_reg/D
#Compile and write the database
compile
current_design top_block
write –hierarchy –output top_block.db
write –format verilog –hierarchy –output top_block.sv
# Create reports
report_timing –nworst 50
```

Class Assignments

1. Design constraints for the 1D filter with the following requirements
   a) Design name **filter_1D.v**
   b) Using the library **my_lib1.db**
   c) Using wire load model **“10x10”**
   d) Maximum operating clock **100MHz**
   e) Clock latency **2ns**
   f) Input delay **5n**s
   g) Output delay **5ns**
   h) Fanout **10** for all outputs
   i) Capacitive Load **0.3 pF**
   j) Maximum transition **3.0**

   Write the script file into a file named “run.scr”