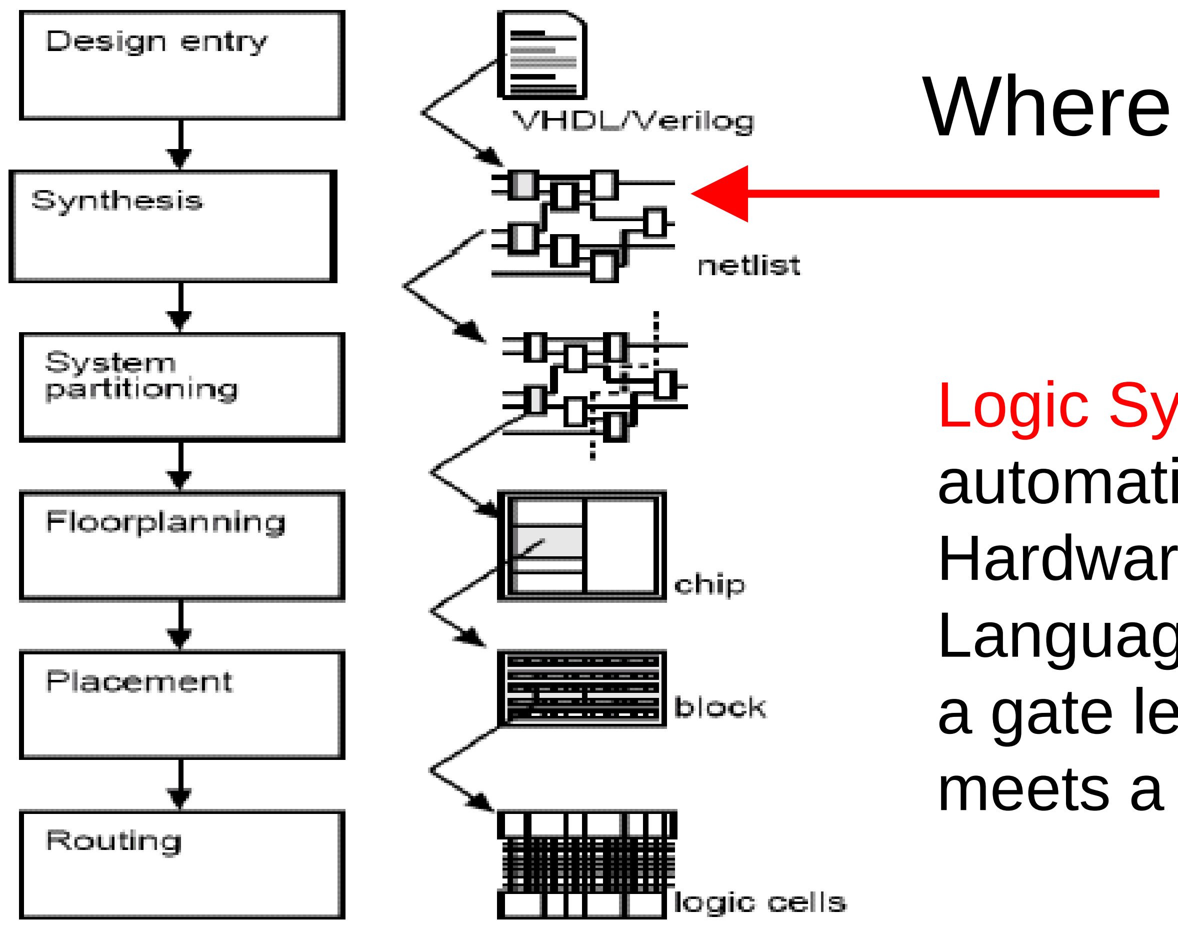
Logic Synthesis and Synopsys Design Compiler Demo

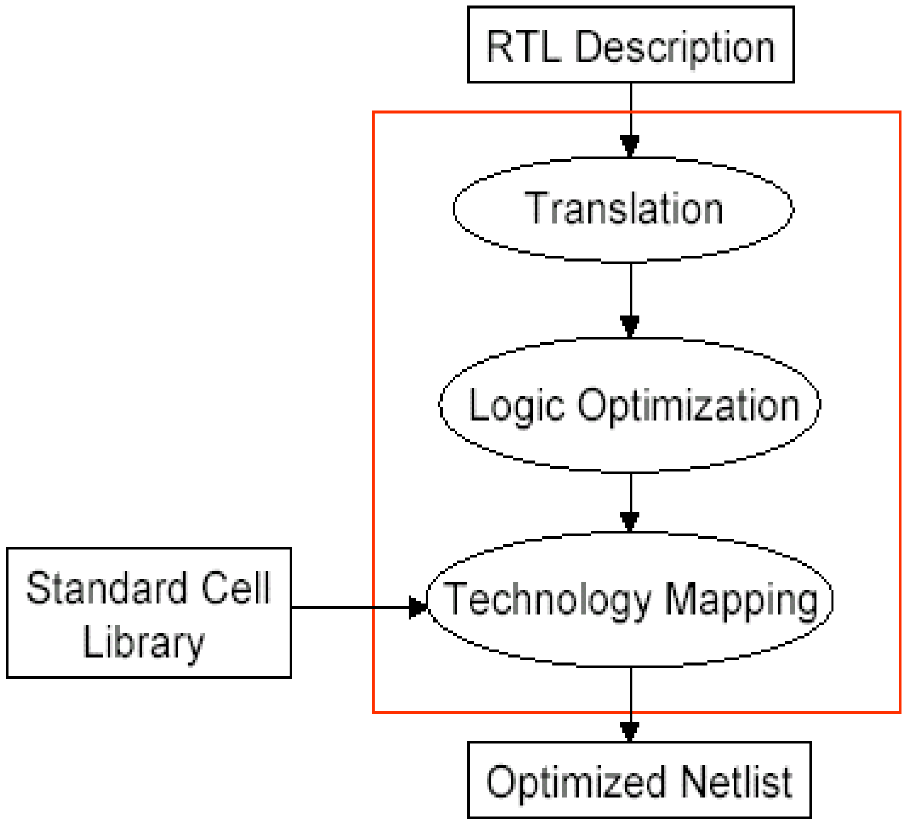
ASIC Synthesis



Where are we now?

Logic Synthesis is the automatic conversion of Hardware Description Language(HDL) model into a gate level netlist that meets a set of constraints

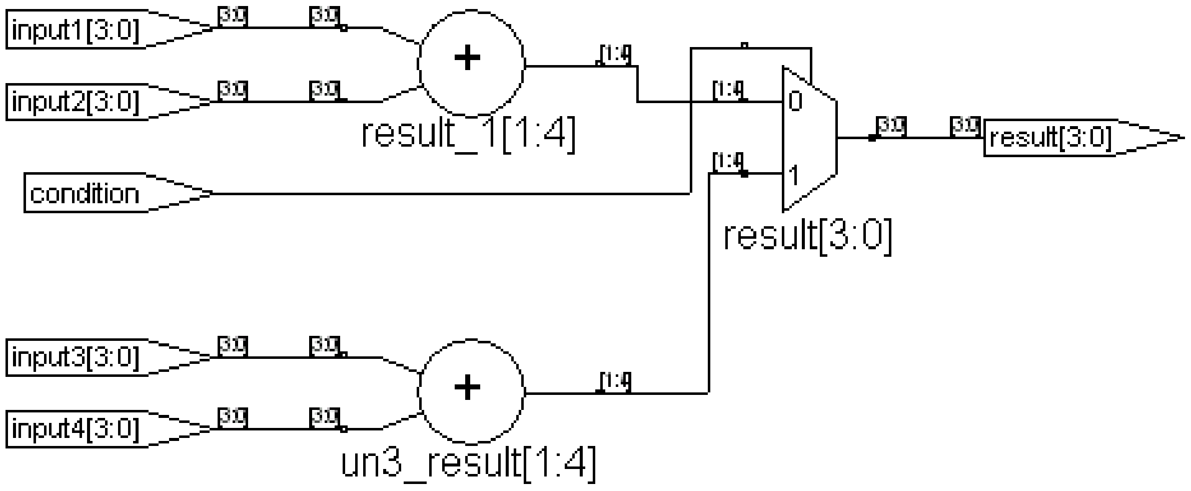
Logic Synthesis



- **Translation:** **translating HDL model into combinational Boolean equations and sequential memory elements**
- **Logic Optimization:** **reducing the delay and area by modifying (optimizing) the equations and memory elements**
- **Technology Mapping:** **mapping the equations and memory elements into a standard cell library**

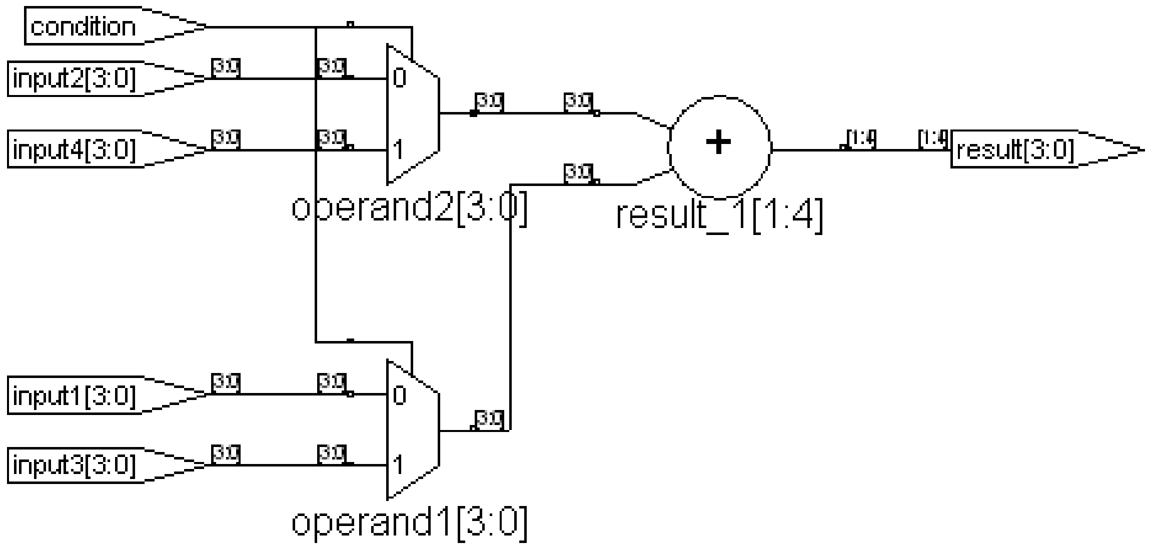
Logic Synthesis: Translation Example

```
process1: process (condition, input1,
                   input2, input3, input4) is
begin
if (condition = '0') then
   result <= input1 + input2;
else
   result <= input3 + input4;
end if;
end process;
```



Logic Synthesis: Translation Example

```
signal operand1, operand2
        : std_logic_vector(3 downto 0) ;
process2: process (condition) is
begin
if (condition = ‘0’) then
       operand1 <= input1;
       operand2 <= input2;
else
       operand1 <= input3;
       operand2 <= input4;
end if;
result <= operand1 + operand2;
end process;
```



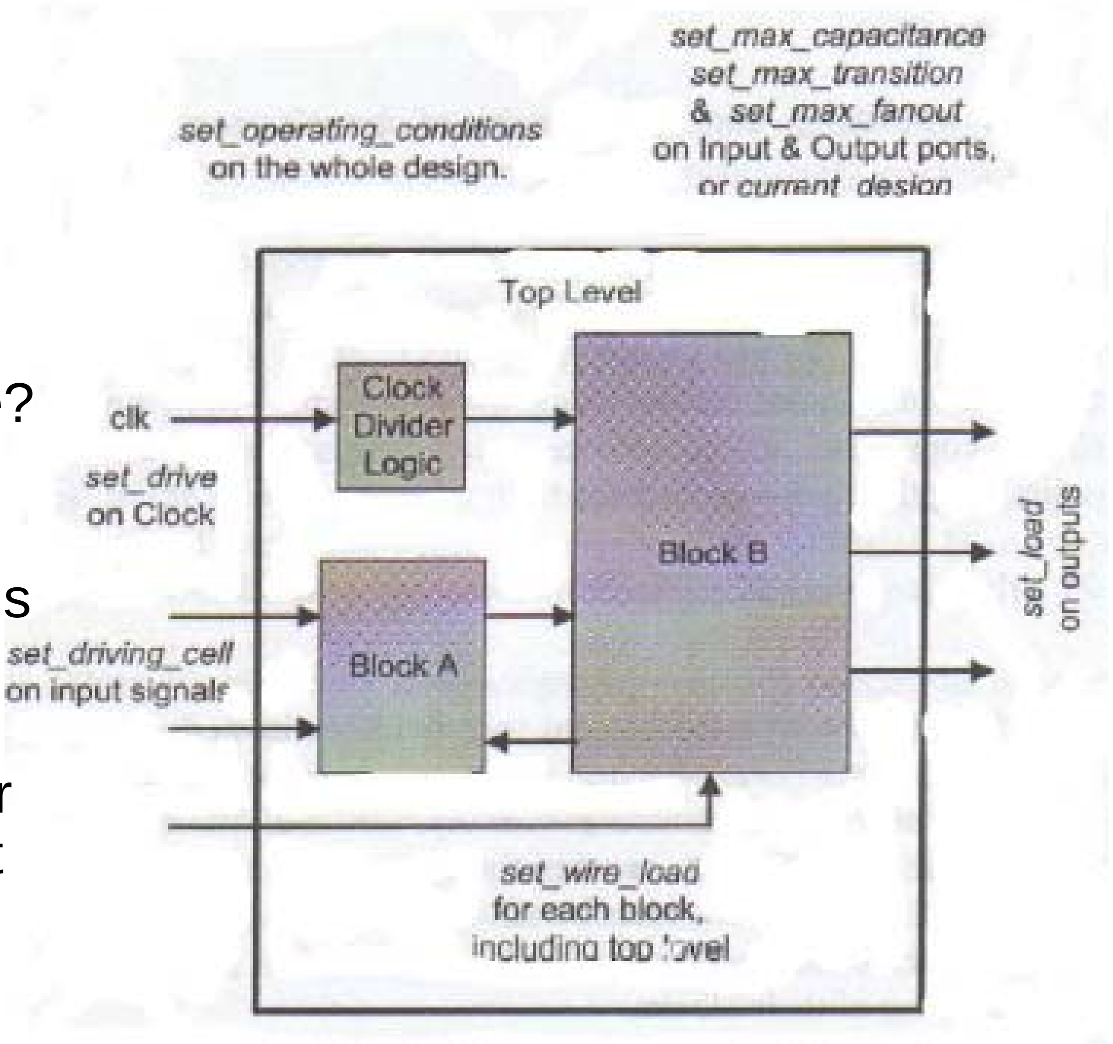
Issues in Logic Synthesis

- Specify design environment
  - Cell libraries (worst case and best case)
  - Operating conditions, wire load models, design rules
  - Input drive strengths, output loading
- Read in design (analyze and elaborate)
- Set constraints
  - Input arrival times, output expected times
  - Clock frequency, area constraint
- Compile (find a netlist that meets the constraints)
- Write netlist and synthesis reports

Logic Synthesis: Specify Design Environment

Define the parameters associated with the other blocks:

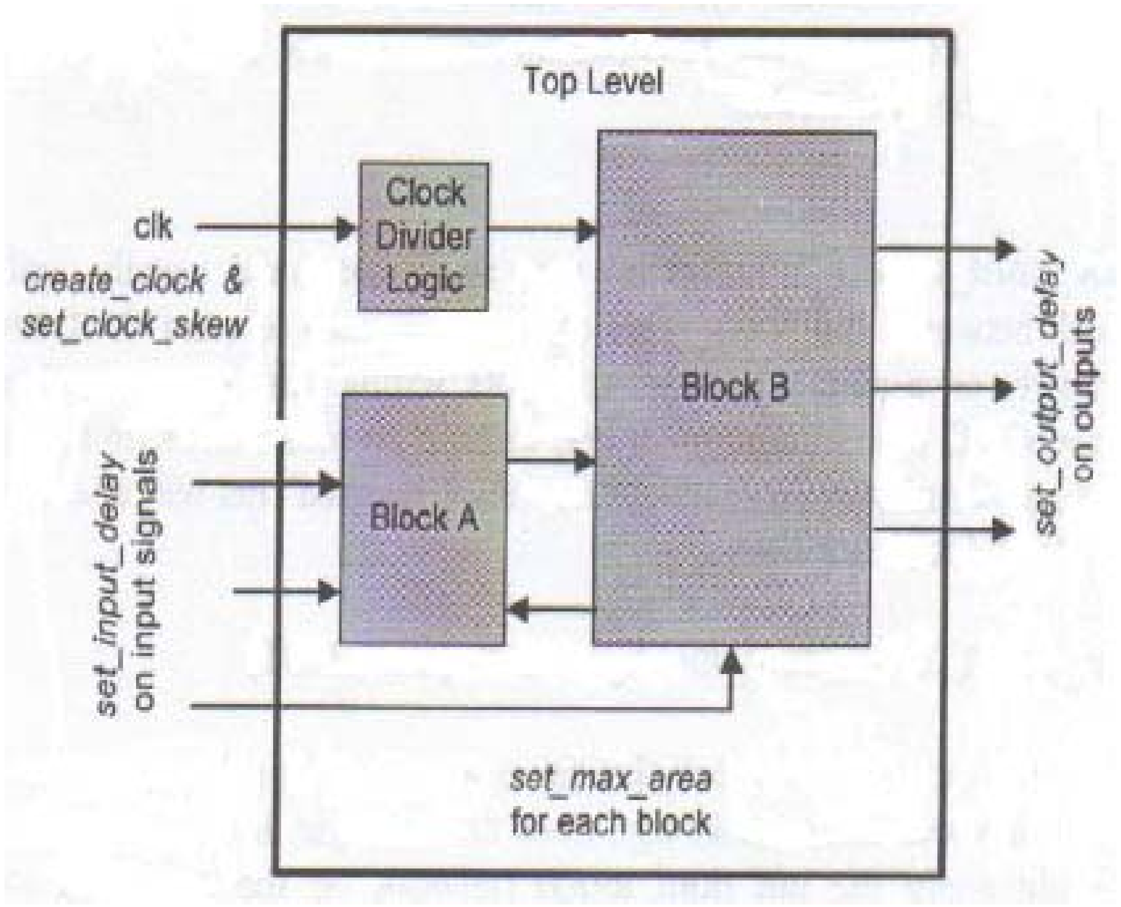
- Checking for the best case or the worst case?
- What are the drive strengths of gates driving the inputs of this block?
- What is the maximum load/maximum number of gates an output port of this block will see?



set_operating_conditions on the whole design.
set_max_capacitance
set_max_transition
& set_max_fanout
on Input & Output ports, or current design
Top Level
clk
Clock Divider Logic
set_drive on Clock
Block B
Block A
set_driving_cell on input signals
set_load on outputs
set_wire_load for each block, including top level

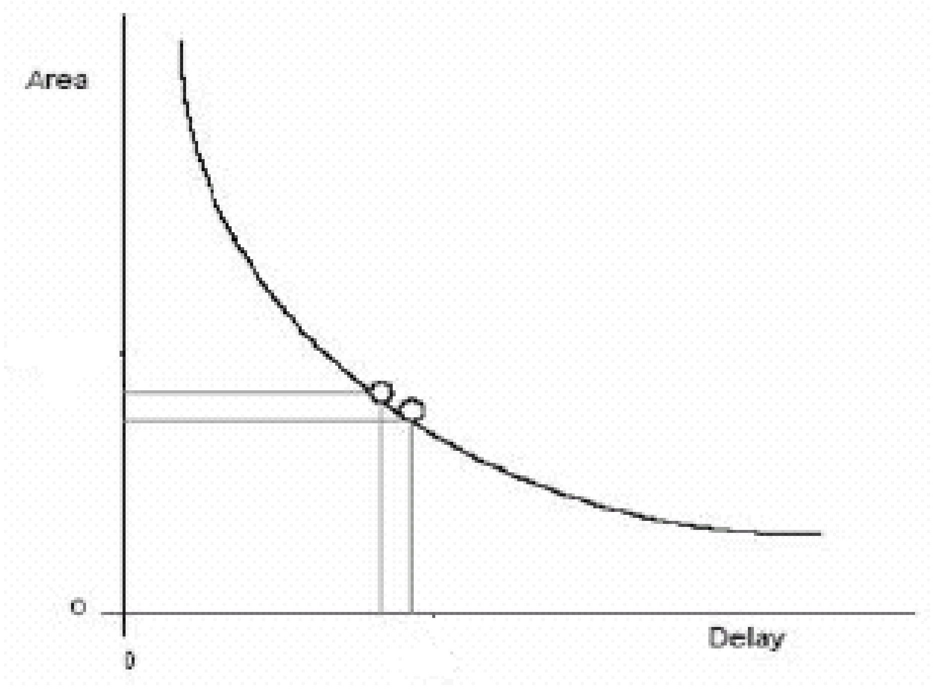
Logic Synthesis: Design Constraints

- Clock period+skew determines if your logic between registers meet your timing constraints or not.
- can also set input and output delays for your I/O ports
- Specify the max/min delay of a particular path or ignore a path for timing
- Set a maximum area for your design (not guaranteed to hold)



Logic Synthesis: Summary

- Synthesis can be an iterative process (design the RTL based on cell library performance !)
- If constraints are not met after synthesis
  - Re-write HDL code
  - Change constraints



Synopsys Design Compiler

- Technical Stuff...
  - Files you'll need to get the tool running:
    - synopsys.env
    - .synopsys_dc.setup

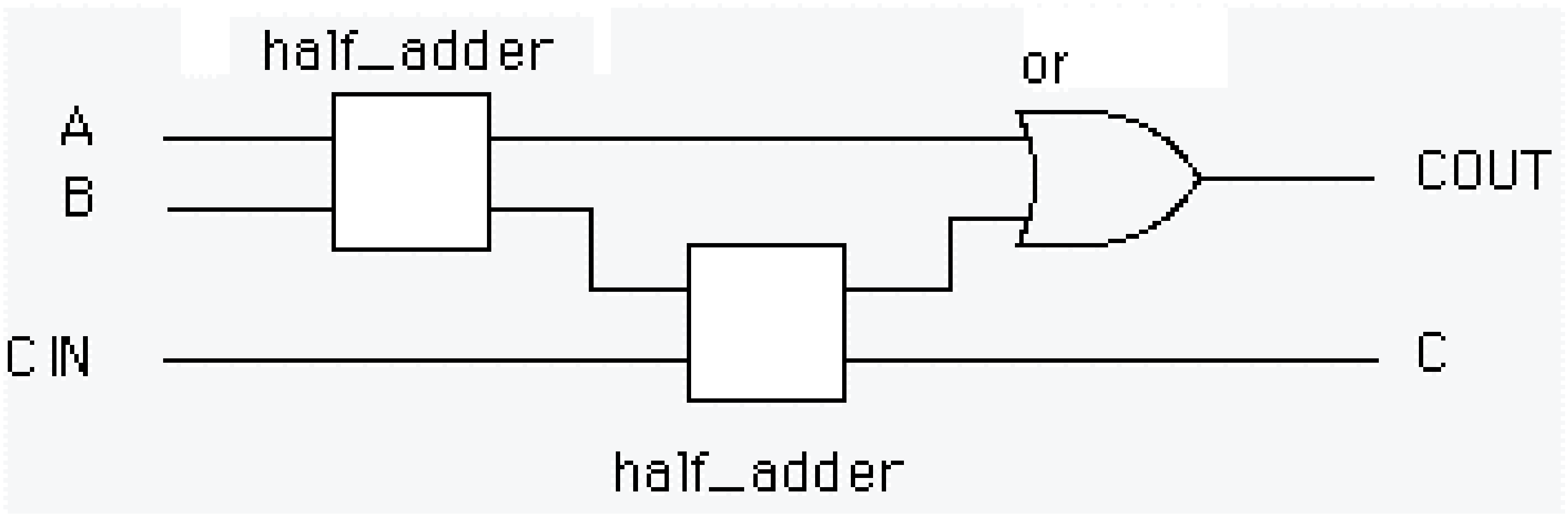
  You will find them @ /vol/ece394

  - Need to have your UNIX accounts
    - Type dc_shell to run the program

Synopsys Design Compiler

- Specify design environment
  - Cell libraries (worst case and best case)
  - Operating conditions, wire load models, design rules
  - Input drive strengths, output loading
- Read in design (analyze and elaborate)
- Set constraints
  - Input arrival times, output expected times
  - Clock frequency, area constraint
- Compile (find a netlist that meets the constraints)
- Write netlist and synthesis reports

Synopsys Design Compiler: An Example Design – Full Adder



- Full Adder
  - 2x half_adder subblocks
  - 1x OR subblock

Synopsys Design Compiler: An Example Design – Full Adder

HA.vhd

```
entity half_adder is
port (
input  : in std_logic_vector(1 downto 0);
output:out std_logic_vector(1 downto 0)
);
end entity half_adder;

architecture gate_level of half_adder is
begin
     output(0) <= input(0) xor input(1);
     output(1) <= input(0) and input(1);
end architecture gate_level;
```

OR.vhd

```
entity or_gate is
port (
input  : in std_logic_vector(1 downto 0);
output : out std_logic);
end entity or_gate;

architecture gate_level of or_gate is
begin
     output <= input(0) or input(1);
end architecture gate_level;
```

Synopsys Design Compiler: An Example Design – Full Adder

FA.vhd

```
entity full_adder is
port (in1, in2, cin : in std_logic;
       sum, cout : out std_logic);
end entity full_adder;

architecture structural of full_adder is

component half_adder is
port( input  : in std_logic_vector(1 downto 0);
     output : out std_logic_vector(1 downto 0));
end component half_adder;

component or_gate is
port( input  : in std_logic_vector(1 downto 0);
      output : out std_logic);
end component or_gate;

signal s1, s2, s3: std_logic;
       .
       .
       .
```

```
begin
A0: half_adder port map(input(0)=>in1,
                        input(1)=>in2,
                        output(0)=>s2,
                        output(1)=>s1);

HA1: half_adder port map(input(0)=>s2,
                        input(1)=>cin,
                        output(0)=>sum,
                        output(1)=>s3);

OR1:  or_gate port map  (input(0)=>s1,
                        input(1)=>s3,
                        output=>cout);
end architecture structural;
```

Synopsys Design Compiler: Commands 1

- Specify design environment
  - Cell libraries (worst case and best case)
  - Operating conditions, wire load models, design rules
    - set_operating_conditions <condition>
      - set_operating_conditions WORST
    - set_wire_load_model –name <name>
      - set_wire_load_model –name SMALL
  - Input drive strengths, output loading
    - set_drive <value> <signal list>
      - set_drive 0 [in1 in2 cin]
    - set_driving_cell –cell <cell name>
      -pin <pin name> <object list>
    - set_load <value> <object list>
      - set_load 0.5 [all_outputs]
    - set_max_fanout <value> <object list>
      - set_max_fanout 5 [sum cout]
    - ...

Synopsys Design Compiler: Commands 2

- Read in design (analyze and elaborate)
  - analyze –format vhdl your_file.vhd
    - analyze –format vhdl HA.vhd
    - analyze –format vhdl OR.vhd
    - analyze –format vhdl FA.vhd
    - Checks for syntax errors before going any further
  - elaborate your_entity –arch your_architecture
    - elaborate full_adder –arch structural
    - The top-level entity
  - current_design top_level_entity
    - current_design full_adder
  - uniquify
    - use the same netlist elements for different instances of the same component

Synopsys Design Compiler: Commands 3

- Set constraints
  - Create clock with skew
    - create_clock –period <value> -waveform <duty> <signal_name>
      - create_clock –period 10 –waveform [list 0 5] –name vCLK
    - set_clock_uncertainty –setup <value> -hold <value> <signal_name>
  - Set input and output delays
    - set_input_delay <value> –clock <signal_name> <object list>
    - set_output_delay <value> –clock <signal_name> <object list>
  - max/min delay of a path or ignore a path for timing
    - set_max_delay <value> -from <object list> -to <object list>
    - set_dont_touch_network <object list>
  - Set a maximum area for your design (not guaranteed to hold)
    - set_max_area <value>
      - set_max_area 0

Synopsys Design Compiler: Commands 4

- Compile
  - compile (actually once you set the environment and constraints, building the netlist is as simple as this ☺)
    - Some parameters for compile
      - -map_effort <low|medium|high>
      - -incremental_mapping
      - -scan etc..
- Write netlist and synthesis reports
  - write –format vhdl –hierarchy –output FA.vhdnetlist
  - report_area > filename
  - report_timing -max_paths 5 > filename

Wrapping it up

- Good News
  - We don’t have to write these commands every time we want to synthesize a design. We can write them in a script file (ie synthesis_script.scr) and tell Design Compiler to use that file for synthesis:

```
>>dc_shell –f synthesis_script.scr > log_file
```

- Even better news:
  - If don’t want to use command line, you can use your script file from a graphical interface –unfortunately we cannot access it from here, but you can run the GUI on workstations by typing:

```
>>design_analyzer
```

A Sample Script File for Full Adder

```
#Design entry
analyze –format vhdl HA.vhd
analyze –format vhdl OR.vhd
analyze –format vhdl FA.vhd
elaborate full_adder –arch structural
current_design full_adder
uniquify

#Setup operating conditions, wire loads, clocks
create_clock –period 10 –waveform [list 0 5] –name vCLK

# Input Drives and Output Loads
set_drive 0  “in1”
set_load 0.5 ”sum”
# Compile and write the netlist
compile
write –format vhdl –hierarchy –output FA.vhdnetlist

#Create Reports
report_area
report_timing -max_paths 5
```

Wrapping it up

- How to use this generated .vhdnetlist file?
  - We will use this file for Post-Synthesis Functional and Timing Verification
  - Going down the ASIC design cycle, we will use this file in placement and routing tools which will bring us one more step closer to fabricated chip (thus $$$ ☺)

Design Analyzer Simulations

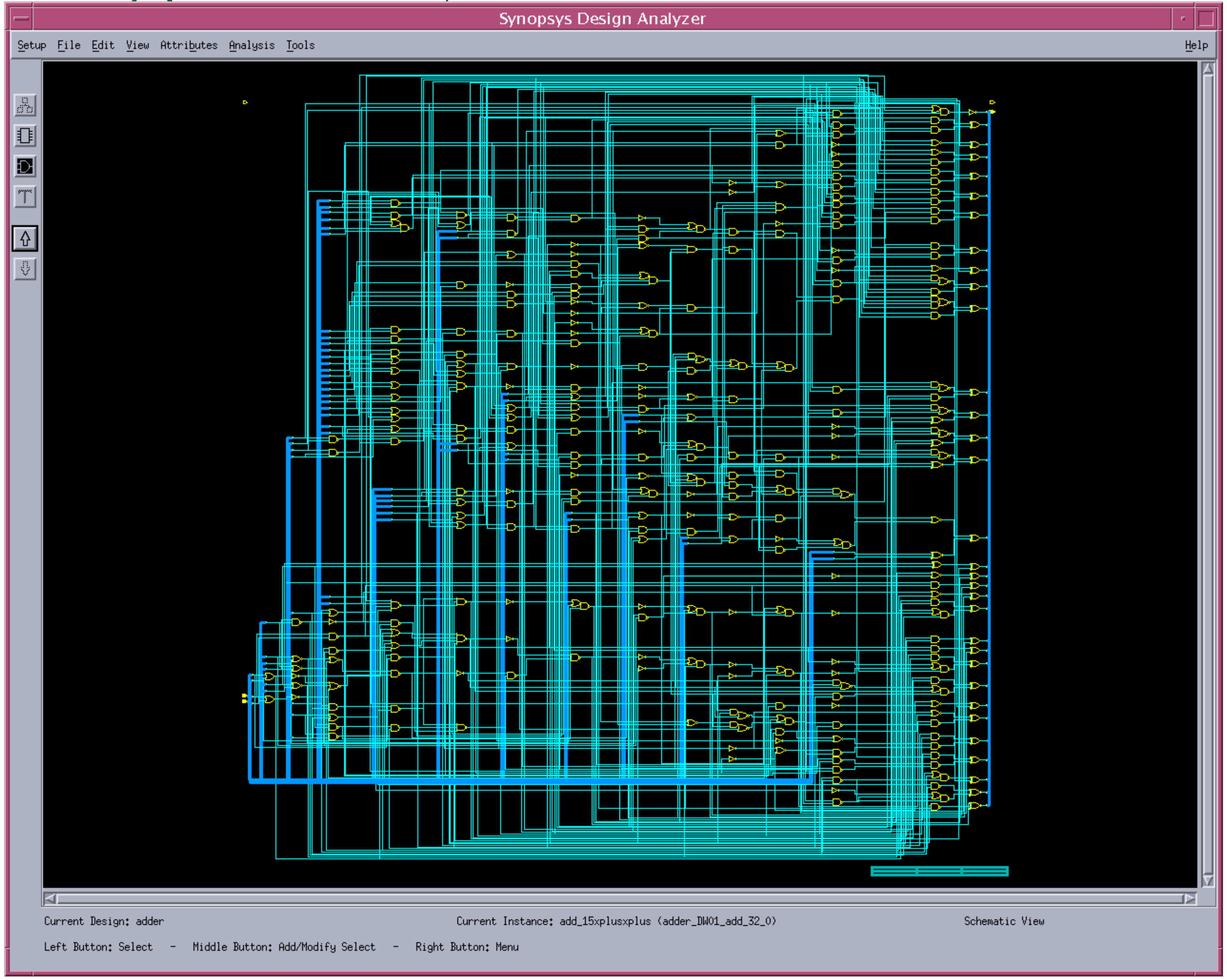
```
analyze –format vhdl adder8b.vhd
elaborate adder –arch behavioral
current_design adder
uniquify

set_drive 0 "in1"
set_drive 0 "in2"
set_load 0.5 "sum"

set_max_delay x -from "in1" -to "sum"
set_max_delay x -from "in2" -to "sum"
set_max_area 0

compile
write –format vhdl –hierarchy –output adder.svhd

report_area > area_log
report_timing > timing_log
quit
```

Three different runs for

X=10, 25 and 100

X	Area	Time
10	487	9.86
25	423	21.11
100	252	38.28

Design Analyzer Simulations X=10



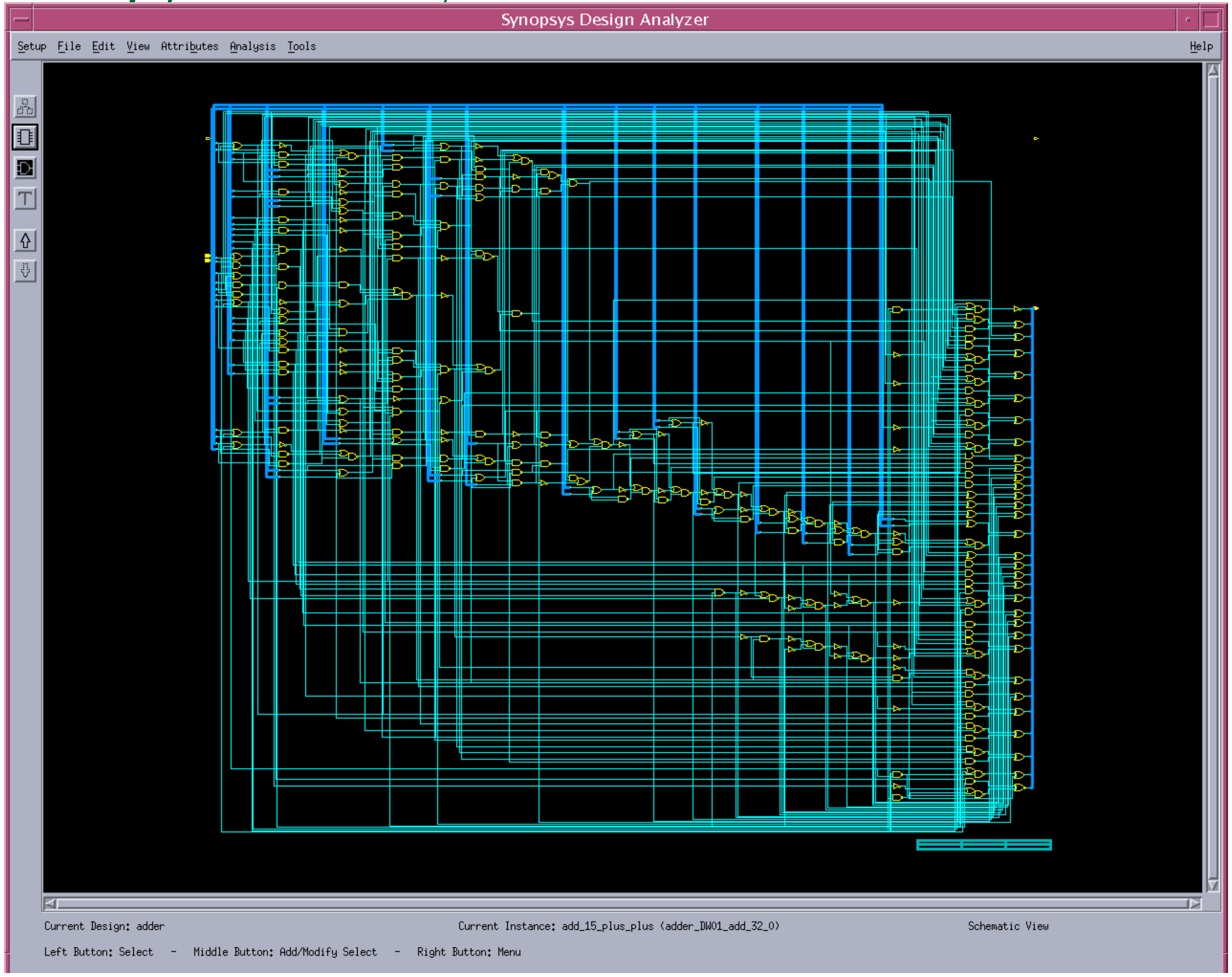
Current Design: adder

Current Instance: add_15xplusxplus (adder_DW01_add_32_0)

Schematic View

Left Button: Select - Middle Button: Add/Modify Select - Right Button: Menu

Design Analyzer Simulations X=25



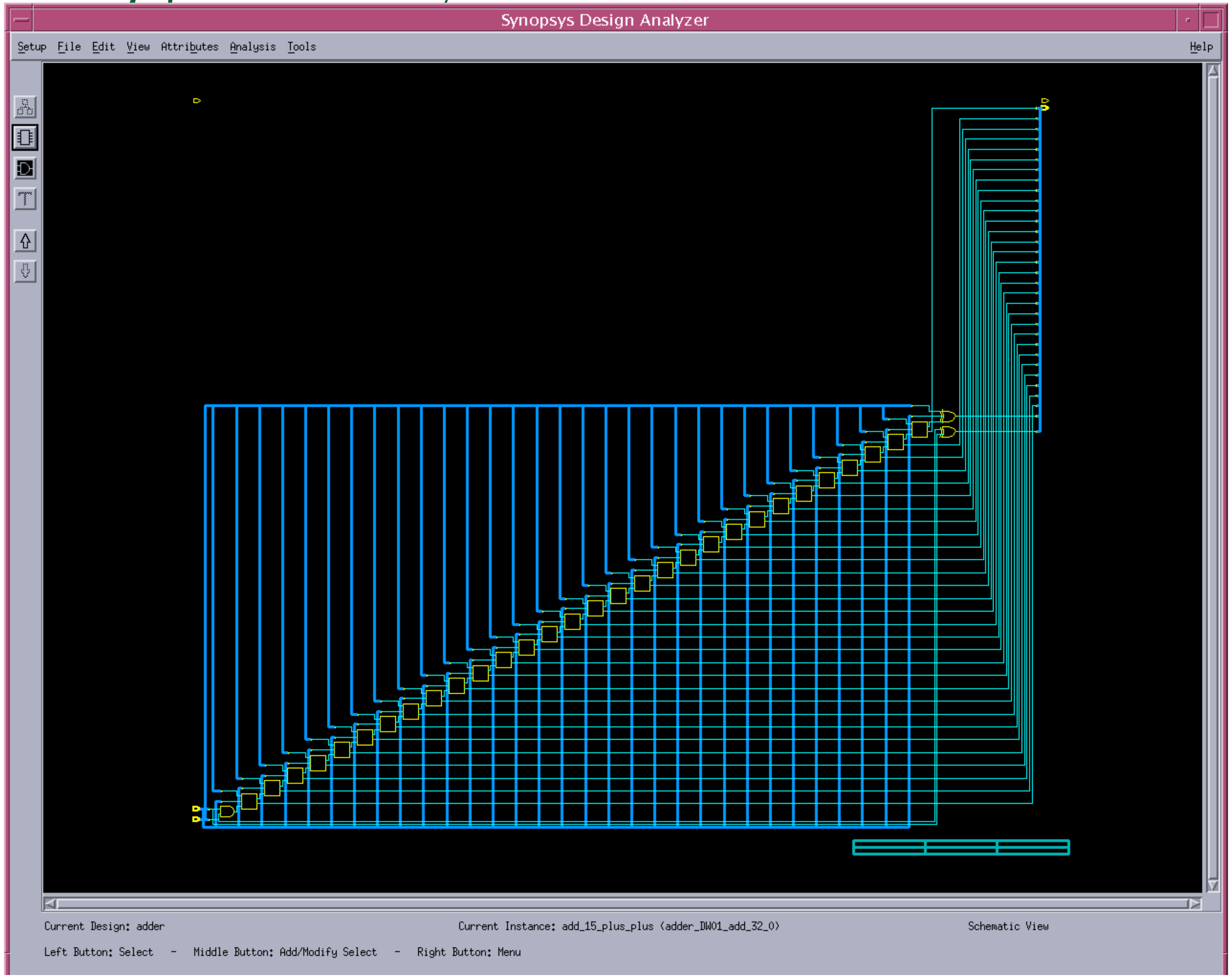
Current Design: adder

Current Instance: add_15_plus_plus (adder_DW01_add_32_0)

Schematic View

Left Button: Select - Middle Button: Add/Modify Select - Right Button: Menu

Design Analyzer Simulations X=100



Current Design: adder

Current Instance: add_15_plus_plus (adder_DW01_add_32_0)

Schematic View

Left Button: Select - Middle Button: Add/Modify Select - Right Button: Menu