

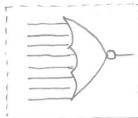
Logic Synthesis

- ▶ Logic synthesis transforms RTL code into a gate-level netlist
 - ▶ RTL Verilog converted into Structural Verilog

```
Assign zero_det = (data_bus == '0') ? 1'b0 : 1'b1
```

Verilog source code

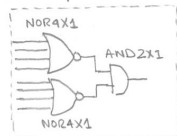
TRANSLATION
(RTL-to-Verilog)



Generic Boolean Gates
(GTECH)

- no timing or loading information
- A virtual library, not "real gates"

OPTIMIZATION +
MAPPING
(compile)



Target Library
(SAED Cell Library)

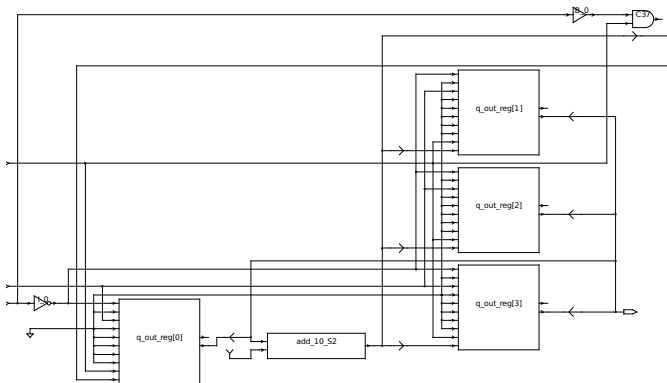
Logic Synthesis - The process and steps

► Translation

- Check RTL for valid syntax
- Transform RTL to unoptimized generic (GTECH) gates
- Parameters applied
- Replace arithmetic operators with DesignWare components
- Link all the parts of the design

Logic Synthesis - The process and steps

- 4-bit Counter, Translated, no Mapping or Optimization



Logic Synthesis - Process and Steps

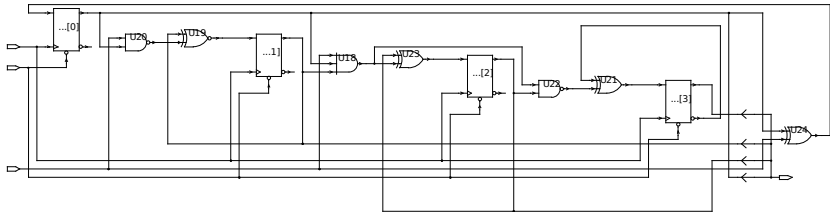
► Optimization and Mapping

- Optimization and Mapping driven by constraints and cell library
- Choose best DesignWare implementation
- Factor out common logical sub-expressions and share terms
- Flatten logic into 2-level realization - hurts area
- Map logic into best fit implementation for given cell library
- Iterate to find "best" realization

Logic Synthesis - Process and Steps

- 4-bit Counter, optimized, mapped, with constraints:

```
create_clock -period 4 [get_ports clk]
set_input_delay 3.0 -max -clock clk [remove_from_collection [all_inputs]\
[get_ports clk]]
```



Logic Synthesis - Constraints

- ▶ Constraints guide optimization and mapping process
- ▶ The designer sets goals (timing, area) through *constraints*

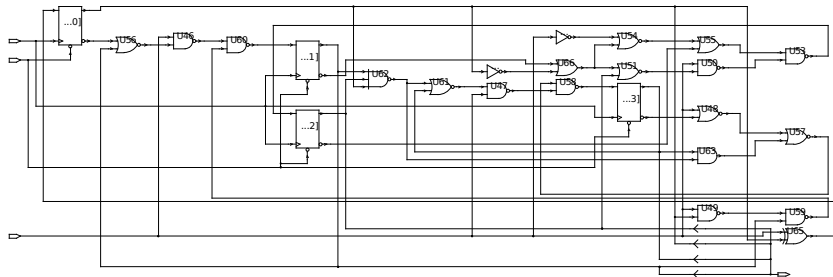
```
# design constraints below #
set_operating_conditions -max TYPICAL
set_wire_load_model -name 8000
set_wire_load_mode top
create_clock -period 20 -name my_clock [get_ports clk_50]
set_output_delay 2.0 -max -clock my_clock [all_outputs]
set_load [expr 5 * [load_of saed90nm_typ/AND2X1/IN1]] [all_outputs]
```

- ▶ DC works to build a circuit that meets your constraints.
- ▶ Typically this is the smallest design that just meets timing
- ▶ Different constraints give *different circuits* but *same function*
- ▶ We provide DC our constraints via a TCL synthesis script

Logic Synthesis - Constraints

- ▶ Back to our 4-bit counter
- ▶ Tightening the constraints even tighter...

```
set_input_delay 3.8 -max -clock clk [remove_from_collection [all_inputs] \  
    [get_ports clk ]]
```



- ▶ Same function, but bigger and faster

Logic Synthesis - Cell Library

- ▶ ASIC vendor supplies the technology file (.lib) for your library
- ▶ The .lib file is compiled into a .db file with *library_compiler*
- ▶ DC uses the .db file to guide optimization and mapping
- ▶ Cell library contains cell delay, wire table, loading table, etc.
- ▶ DC maps to the cell library given in the .synopsys_dc_setup file

Logic Synthesis - DC setup file

► .synopsys_dc_setup

```
# Tell DC where to look for files
lappend search_path ../libs
set synop_lib /nfs/guille/a1/cadlibs/synop_lib/SAED_EDK90nm
# Set up libraries
set target_library $synop_lib/Digital_Standard_Cell_Library/synopsys/models/saed90nm_typ_lt_pg.db
set link_library "*" $target_library"
```

► The .lib file we are using:

http://web.engr.oregonstate.edu/~traylor/ece474/saed90nm_typ_lt_pg.lib

► Cell delay matrix produced by SPICE,... very, very time intensive!

Logic Synthesis - Available saed90nm Libraries

Worst Case Libraries

Library Name	Temperature	Vdd	Process
saed90nm_min_hth.pg.lib(.db)	125deg C	Vdd = 1.08	P = 1.2
saed90nm_min_lth.pg.lib(.db)	-40deg C	Vdd = 1.08	P = 1.2
saed90nm_min_lt.pg.lib(.db)	-40deg C	Vdd = 0.70	P = 1.2
saed90nm_min_nth.pg.lib(.db)	25deg C	Vdd = 1.20	P = 1.2
saed90nm_min_nt.pg.lib(.db)	25deg C	Vdd = 0.70	P = 1.2

Typical Case Libraries

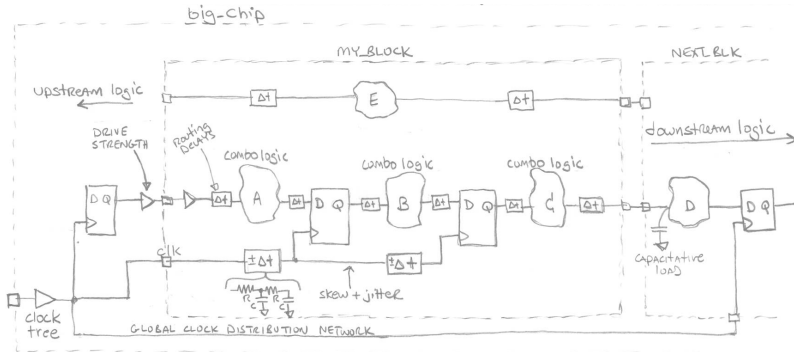
Library Name	Temperature	Vdd	Process
saed90nm_typ_htl.pg.lib(.db)	125deg C	Vdd = 0.80	P = 1.0
saed90nm_typ_ltl.pg.lib(.db)	-40deg C	Vdd = 0.80	P = 1.0
saed90nm_typ_ht.pg.lib(.db)	125deg C	Vdd = 1.20	P = 1.0
saed90nm_typ_ntl.pg.lib(.db)	25deg C	Vdd = 0.80	P = 1.0
saed90nm_typ_lt.pg.lib(.db)	-40deg C	Vdd = 1.20	P = 1.0
saed90nm_max.pg.lib(.db)	-40deg C	Vdd = 1.30	P = 1.0
saed90nm_min.pg.lib(.db)	125deg C	Vdd = 0.70	P = 1.0
saed90nm_typ.pg.lib(.db)	25deg C	Vdd = 1.20	P = 1.0

Best Case Libraries

Library Name	Temperature	Vdd	Process
saed90nm_max_htl.pg.lib(.db)	125deg C	Vdd = 0.90	P = 0.8
saed90nm_max_ltl.pg.lib(.db)	-40deg C	Vdd = 0.90	P = 0.8
saed90nm_max_ht.pg.lib(.db)	125deg C	Vdd = 1.32	P = 0.8
saed90nm_max_nt.pg.lib(.db)	25deg C	Vdd = 1.32	P = 0.8
saed90nm_max_ntl.pg.lib(.db)	25deg C	Vdd = 0.90	P = 0.8

Logic Synthesis

- ▶ Synthesis optimization works primarily on combo logic
- ▶ Design is split into paths at FF or block i/o
- ▶ These paths are broken into:
 - ▶ Input to register
 - ▶ Register to register
 - ▶ Register to output



Logic Synthesis - Register to Register Constraints

- ▶ Comprised of:
 - ▶ Clock period, clock uncertainty, FF setup time
- ▶ We define the clock period and clock uncertainty
- ▶ FF setup time will come from the library

Logic Synthesis - Register to Register Constraints

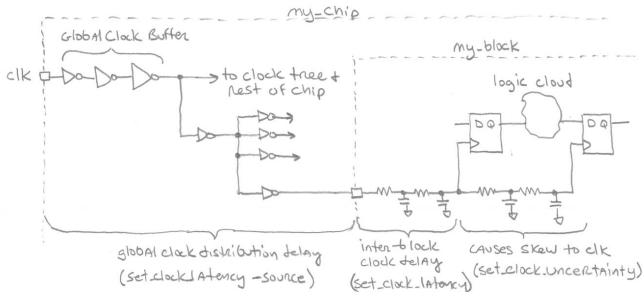
- In a synchronous design, everything is relative to the clock

```
#define the clock period and clock port (20ns clock period)
create_clock -period 20 -name my_clock [get_ports clk]
```

```
#set the clock uncertainty to +/- 10pS
set_clock_uncertainty -setup 0.01 [get_clocks my_clock]
set_clock_uncertainty -hold 0.01 [get_clocks my_clock]
```

```
#set block internal clock network delay to 100pS
set_clock_latency 0.1 [get_clocks my_clock]
```

```
#global and distributed (external) clock buffer delays (0.5ns)
set_clock_latency -source 0.5 [get_clocks my_clock]
```



Logic Synthesis - Register to Register Constraints

- ▶ The statements:

```
set_clock_uncertainty -setup 0.01 [get_clocks my_clock]
set_clock_uncertainty -hold 0.01 [get_clocks my_clock]
set_clock_latency      0.1 [get_clocks my_clock]
set_clock_latency -source 0.5 [get_clocks my_clock]
```

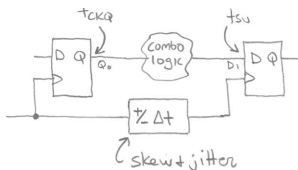
After clock tree routing will be replaced with

```
set_propagated_clock [get_clocks my_clock]
```

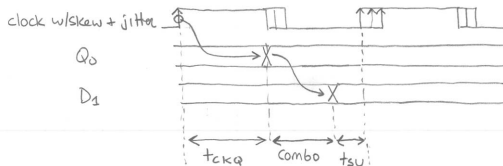
- ▶ After clock tree routing, the actual clock skew, routing and buffer delays will be known.

Logic Synthesis - Register to register constraints

- ▶ Constraining clock constrains register to register delays (Q to D)



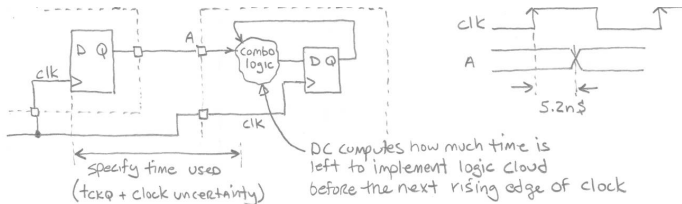
- ▶ Important relationships:
- ▶ $\text{combo_delay} \leq (\text{clock_cycle} - t_{su} - t_{CKQ} - \text{clock_skew})$
- ▶ $\text{min_cycle_time} = (t_{CKQ} + t_{su} + \text{clock_skew} + \text{combo_delay})$



Logic Synthesis - Input to Register Constraints

- Specify delay external to our block
- DC optimizes our input logic cloud in the remaining time

```
set_input_delay 5.2 -max -clock serial_clock \  
[remove_from_collection [all_inputs] [get_ports serial_clock]]  
set_driving_cell -lib_cell SDFFARX1 \  
[remove_from_collection [all_inputs] [get_ports clk_50]]
```

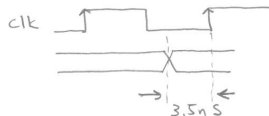
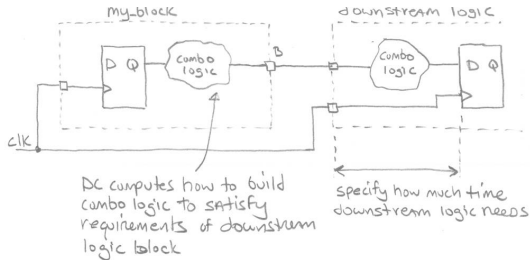


- An SDFFAR1 output asserts our input 5.2ns after clock edge

Logic Synthesis - Register to output constraints

- ▶ We specify what how much time the downstream logic requires
- ▶ DC optimizes our output decode logic in the remaining time

```
set_output_delay -max 3.5 -clock serial_clock [all_outputs]
set_load [expr 5 * [load_of saed90nm_typ/AND2X1/IN1]] [all_outputs]
```



Output At "B" must be valid
At least 3.5 ns prior to
clock edge.

- ▶ Downstream stage requires data valid 3.5ns before next clock edge
- ▶ Downstream load is equivalent to 5 AND2X1 loads

Logic Synthesis - Global Constraints

- ▶ What constraints are missing?
- ▶ These are chip-wide, global constraints
- ▶ Temperature
 - ▶ 40 deg C, 25 deg C, 125 deg C
- ▶ Voltage
 - ▶ 1.32V, 1.2V, 0.7V
- ▶ Process
 - ▶ min, max, typ
- ▶ Wire load model
 - ▶ enclosed, top
 - ▶ die area estimate

Logic Synthesis - Temp, Voltage, Process

- ▶ Cell libraries are characterized at multiple TVP corners
- ▶ CMOS fastest at: low temperature, high voltage, best process
- ▶ CMOS slowest at: high temperature, low voltage, worst process
- ▶ Actual delays/speeds vary greatly over PVT

```
library (saed90nm_max) {  
  technology ( cmos ) ;  
  delay_model      : table_lookup;  
  date : "2007 (INF CREATED ON 12-MAY-2008)" ;  
  time_unit : "1ns" ;  
  leakage_power_unit : "1pW" ;  
  voltage_unit : "1V" ;
```

```
  power_supply() {  
    power_rail(vdd, 1.32000000);  
    power_rail(vddg, 1.32000000);  
    default_power_rail : vdd ;  
  }  
  
  operating_conditions("BEST") {  
    process : 1.0;  
    temperature : -40;  
    voltage : 1.32;  
    power_rail(vdd, 1.32000000);  
    power_rail(vddg, 1.32000000);  
    tree_type : best_case_tree;  
  }
```

```
  /*****/  
  /** user supplied k_factors **/  
  /*****/  
  /*****/  
  /** PVT values used in k-factor calculations **/  
  /** Process   min,max : 0.800 1.200 **/  
  /** Voltage   min,max : 1.188 1.452 **/  
  /** Temperature min,max : -40.000 125.000 **/  
  /*****/  
  /*****/  
  /** user supplied nominals **/  
  /*****/  
  nom_voltage      : 1.320;  
  nom_temperature  : -40.000;  
  nom_process      : 1.000;
```

Logic Synthesis - Temp, Voltage, Process

- ▶ Find setup time failures with worst corner - Why?
- ▶ Find hold time failures with best corner - Why?
- ▶ Setting TVP constraints:

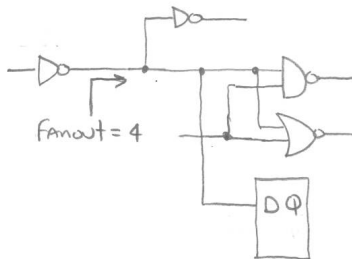
```
set_operating_conditions -max "WORST"
```

```
set_operating_conditions -max "TYPICAL"
```

```
set_operating_conditions -max "BEST"
```

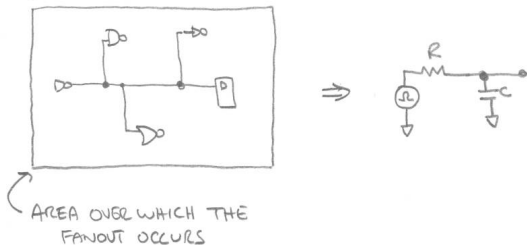
Logic Synthesis - Wire delays

- ▶ Wire delay dominates in deep sub-micron ($<0.35\mu\text{m}$) circuits
- ▶ Getting a good estimate of routing delay is vital for timing closure
- ▶ A *wireload model* relates fanout to RC parasitic prior to layout



Logic Synthesis - Wire delays

- ▶ The model also specifies a per length resistance, capacitance and area and a statistical mapping from fanout to wire length
- ▶ Wire lengths are averages of previous designs of same size and fanout
- ▶ Using the wire length and R/C values, a delay can be calculated



Logic Synthesis - Wire delays

► Wire load model from our saed90nm_typ library

```
library (saed90nm_typ) {  
  
  time_unit : "1ns" ;  
  leakage_power_unit : "1pW" ;  
  pulling_resistance_unit : "1kohm" ;  
  capacitive_load_unit(1000.000,ff) ;  
  
  wire_load("8000") {  
    capacitance : 0.000312;  
    resistance : 0.00157271;  
    area : 0.01;  
    slope : 90.646360 ;  
    fanout_length( 1 , 13.940360);  
    fanout_length( 2 , 31.804080);  
    fanout_length( 3 , 51.612120);  
    fanout_length( 4 , 73.611440);  
    ...  
    ...  
    fanout_length( 17 , 671.375880);  
    fanout_length( 18 , 749.983920);  
    fanout_length( 19 , 834.487640);  
    fanout_length( 20 , 925.134000);  
  }  
}
```

Logic Synthesis - Computing wire delays

- ▶ Determine fanout on the net
- ▶ Look up length in the wire load model
- ▶ Calculate capacitance and resistance by multiplying length by per unit R and C in the wire load model
- ▶ Calculate delay from RC
- ▶ For example, if fanout = 4, in 8000 sq micron area:

Our scenario is: `fanout_length(4 , 73.611440);`

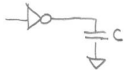
`lumped C = (0.000312 * 73.611440)`

`lumped R = (0.00157271 * 73.611440)`

`net area = (0.01 * 73.611440)`

Logic Synthesis - Computing wire delays

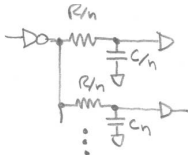
- ▶ RC delay is calculated depending on the interconnect model
- ▶ Best case tree, balanced tree, worst case tree
 - ▶ Best case tree: load is adjacent to the driver, $R=0$
 - ▶ Balanced tree: each load shares R/n and C/n where n =fanout
 - ▶ Worst case tree: lumped load at end of line



Best Case Tree

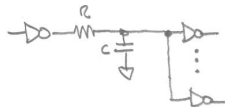
$$R=0$$

only capacitive load
no interconnect delay



Balanced Tree

each path gets C/n , R/n



Worst Case Tree

All loads at end

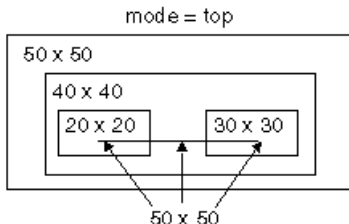
Logic Synthesis - Wire load model

- ▶ Setting the wire model
- ▶ Wire load model names (e.g. "8000") are in .lib file

```
#Setting wire load model constraint  
set_wire_load_model -name 8000
```

Logic Synthesis - Wire load mode

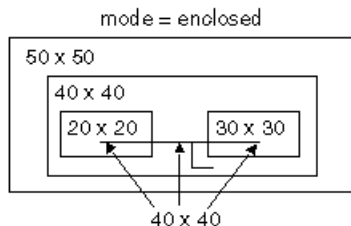
- ▶ The Wire load mode specifies the wire load model for nets that cross hierarchical boundaries
 - ▶ Top Model:



- ▶ Most pessimistic
- ▶ Uses WLM of the top level ignoring the WLMs of lower level blocks

Logic Synthesis - Wire load mode

- ▶ Enclosed Model:

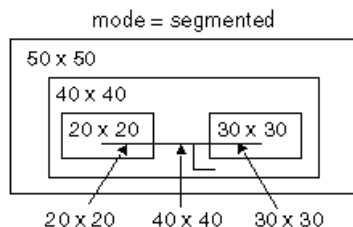


- ▶ Less pessimistic
- ▶ Uses WLM of the level that completely encloses the net

Logic Synthesis - Wire load mode

► Segmented Model:

- Sum of the cell areas of the designs containing the net is used
- Nets crossing hierarchical boundaries are divided into segments
- Each net segment is estimated from the wire load model of the design containing the segment



Logic Synthesis - Wire load mode

- ▶ Setting the wire mode

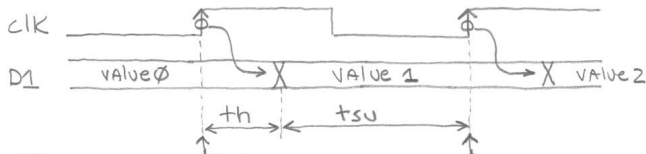
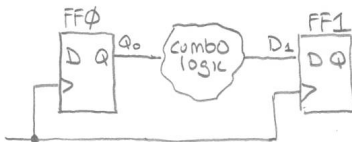
```
set_wire_load_mode top
```

Logic Synthesis

► Hold time Violations

- Running at speed is the greatest challenge
- Hold time is usually dealt with later on, Why?
 - Layout introduces unexpected delay
 - Test structures add delay in some places...
 - and introduce hold time problems elsewhere
 - Fixing hold time problems too early often fixes non-problems
 - Fixing after layout you only fix real problems
 - Best strategy: Only fix big hold time problems up front
 - These will show up under best case TVP

Logic Synthesis - Setup and Hold Review



At this edge, FF1 Acquires value 0. Value 0 must hold stable for t_h After the clock edge

Also, FF0 launches value 1

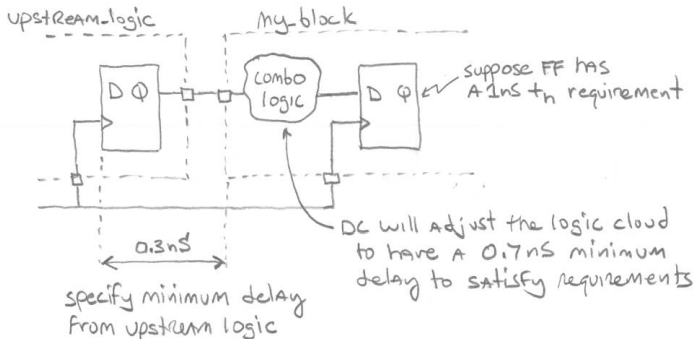
At this edge, FF1 Acquires value 1. Value 1 must have been stable for at least t_{su} before the clock edge.

Also, FF1 launches value 1

Logic Synthesis - Constraining for Hold Time

- Constrain input hold time
- Supply the minimum delay external to our block

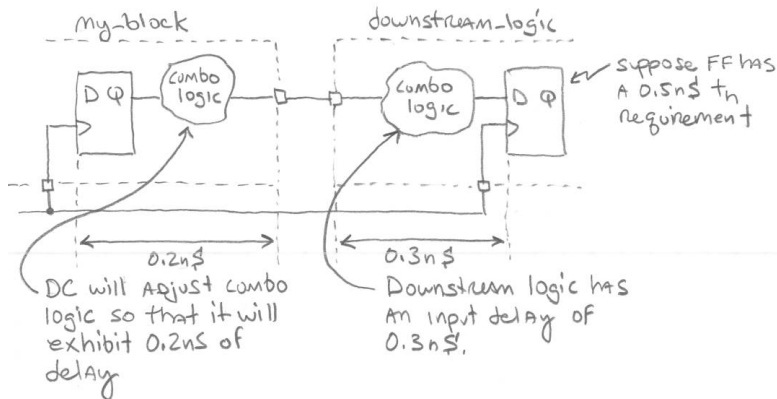
```
set_input_delay -min 0.3 -clock clk_50 \  
[remove_from_collection [all_inputs][get_ports clk_50]]
```



Logic Synthesis - Constraining for Hold Time

- ▶ Constrain output hold time
- ▶ Supply the minimum delay external to our block
- ▶ Hint: Find hold time our block must supply and negate

```
set_output_delay -min [expr 0.3 - 0.5] -clock clk_50 [all_outputs]
# [expr external_logic_delay - FF_hold_time_reqmt] OR ....
set_output_delay -min -0.2 -clock clk_50 [all_outputs]
```



Logic Synthesis - Checking Constraints

- ▶ Check to see if constraints were met

```
report_constraint:  
  [-all_violators]  (show all constraint violators)  
  [-max_delay]      (show only setup and max_delay information)  
  [-min_delay]      (show only hold and min_delay information)
```

- ▶ DC can fix hold time problems after routing by:

```
compile -incr -only_hold_time
```