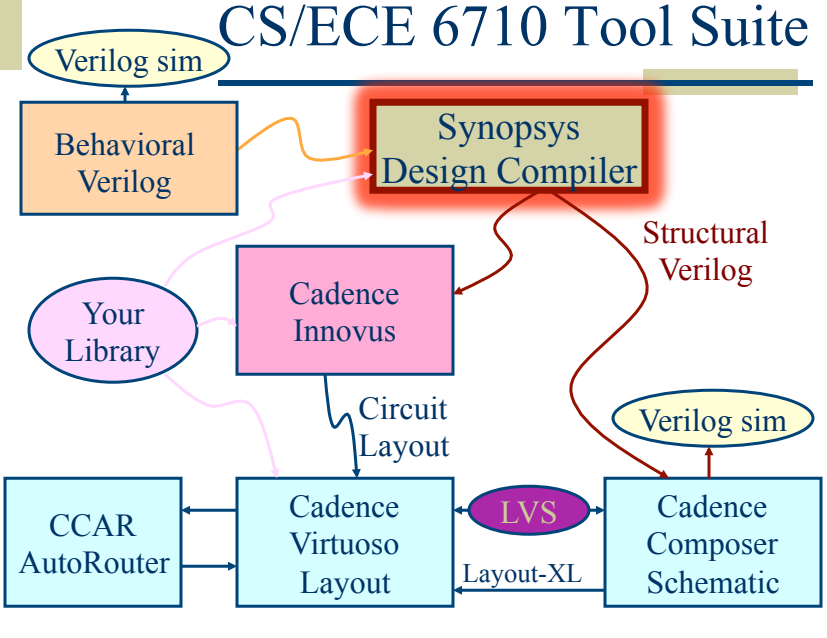
Synthesis

Synopsys Design Compiler

CS/ECE 6710 Tool Suite



In the CAD Book

- Chapter 9
  - Specifically Section 9.1 on Design Compiler
    - Section 9.2 is the Cadence version of Design Compiler, called RTL Compiler (or maybe something else now…)
    - It works well, but Synopsys DC is the industry standard…
  - Also 9.3 – importing structural Verilog back to Cadence
  - And 9.4 Post-Synthesis Verilog Simulation

Requirements

- Synthesis is the process of turning behavioral Verilog into structural Verilog
- Structural Verilog requires a library of cells
  - That's where CAD5 comes in!
  - Initial library has only 5 cells: INV, NAND2, NOR2, TIEHI, TIELO
  - Characterize cells with Liberate
  - Convert to .lib (timing) and .db (synthesis target) formats

Minor Fix for .lib File

- Turns out that Liberate produces a .lib file with references to vdd and vss
  - The old ELC didn't do this…
- It may reduce warnings later if those are named vdd! and gnd!
- Script in /uusoc/facility/cad_common/local/bin/F17 to do this
  - **`fix-libfile <filename>`**

Synopsys Design Compiler

- Synthesis of behavioral to structural
- Three ways to go:
  1. Type commands to the design compiler shell
     - Start with syn-dc and start typing
  2. Write a script
     - Use syn-script.tcl as a starting point
  3. Use the Design Vision GUI
     - Friendly menus and graphics...

Synthesis Process: Design Compiler

1. Define synthesis environment
2. Read in your behavioral Verilog
3. Set synthesis constraints (speed, area, etc.)
4. Compile (synthesize) the design
5. Write out the results

Design Compiler – Basic Flow

1. Define environment
   - target libraries – your cell library
   - synthetic libraries – DesignWare libraries
   - link-libraries – libraries to link against
2. Read in your behavioral Verilog
   - Usually split into analyze and elaborate
3. Set constraints
   - timing – define clock, loads, etc.

Design Compiler – Basic Flow

4. Compile the design
   - compile or compile_ultra
   - Does the actual synthesis
5. Write out the results
   - Make sure to change_names
   - Write out structural verilog, report, ddc, sdc files

beh2str – the simplest script!

```
[elb@lab2-12 cadence]$ beh2str
beh2str - Synthesizes a verilog RTL code to a structural code
        based on the synopsys technology library specified.
Usage   : beh2str f1 f2 f3
        f1 is the input verilog behavioral file
        f2 is the output verilog structural file
        f3 is the compiled synopsys technology library file
```

beh2str – the simplest script!

```
[elb@lab2-12]$ beh2str
beh2str - Synthesizes a verilog RTL code to a structural code
        based on the synopsys technology library specified.
Usage   : beh2str f1 f2 f3
          f1 is the input verilog RTL file
          f2 is the output verilog structural file
          f3 is the compiled synopsys technology library file

[elb@lab2-12]$ beh2str addsub.v addsub_dc.v Lib6710_00.db
….
Thank you…
[elb@lab2-12]$ …results in addsub_dc.v and addsub_dc.v.rep
```

addsub.v

```
module addsub (a, b, addnsub, result);
  parameter SIZE = 8;        // default word size is 8
  input  [SIZE-1:0] a, b;        // two SIZE-bit input
  input  addnsub;                // Control bit: 1 = add, 0 = sub
  output reg [SIZE:0] result;  // SIZE+1 bit result

always @(a, b, addnsub)
   begin
      if (addnsub)
          result = a + b;
      else result = a – b;
   end
endmodule
```

addsub_dc.v

```
module addsub ( a, b, addnsub, result );
  input [7:0] a;
  input [7:0] b;
  output [8:0] result;
  input addnsub;
  wire   n8, n9, n10, n11, n12, n13, n14, n15, n16, n17, n18, n19, n20, n21,
        n22, n23, n24, n25, n26, n27, n28, n29, n30, n31, n32, n33, n34, n35,
        n36, n37, n38, n39, n40, n41, n42, n43, n44, n45, n46, n47, n48, n49,
        n50, n51, n52, n53, n54, n55, n56;
  XNOR2X2 U4 ( .A(addnsub), .B(n8), .Y(result[8]) );
  AOI21X2 U5 ( .A(n9), .B(n10), .C(n11), .Y(n8) );
  AOI21X2 U6 ( .A(n12), .B(n13), .C(a[7]), .Y(n11) );
  …
  INVX1 U60 ( .A(a[0]), .Y(n52) );
  XNOR2X2 U61 ( .A(b[0]), .B(addnsub), .Y(n54) );
endmodule
```

58 cells used

addsub_dc.v.rep

```
Operating Conditions: typical   Library: foo_typ
Wire Load Model Mode: top
  Startpoint: b[0] (input port)
  Endpoint: result[8] (output port)
  Path Group: (none)
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  input external delay                     0.00       0.00 r
  b[0] (in)                                0.00       0.00 r
  U61/Y (XNOR2X2)                          0.67       0.67 r
  U56/Y (INVX1)                            0.57       1.24 f
…
  U5/Y (AOI21X2)                           0.42       8.62 r
  U4/Y (XNOR2X2)                           0.47       9.09 r
  result[8] (out)                          0.00       9.09 r
  data arrival time                                   9.09
  -----------------------------------------------------------
  (Path is unconstrained)
```

addsub_dc.v.rep

```
Library(s) Used:
    foo_typ (File: /home/elb/VLSI/cadence-f13/syn-f13/trythis/Lib6710_00.db)

Number of ports:                          26
Number of nets:                           75
Number of cells:                          58
Number of combinational cells:            58
Number of sequential cells:                0
Number of macros:                          0
Number of buf/inv:                        17
Number of references:                      3

Combinational area:        331.000000
Buf/Inv area:               51.000000
Noncombinational area:       0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:           331.000000
Total area:                 undefined
```

beh2str – the simplest script!

```
#!/bin/tcsh
setenv SYNLOCAL /uusoc/facility/cad_common/local/class/6710/F17/synopsys
#set the path of dc shell script file
setenv SCRIPTFILE ${SYNLOCAL}/beh2str.tcl
#store the arguments
setenv INFILE $1
setenv OUTFILE $2
setenv LIBFILE $3
#setup to run synopsys
source /uusoc/facility/cad_common/local/setups/F17/setup-synopsys
# run (very simple) Synopsys Design Compiler synthesis
dc_shell-xg-t -f $SCRIPTFILE
```

Beh2str.tcl – the actual script

```
# beh2str script
set target_library [list [getenv "LIBFILE"]]
set link_library [concat [concat "*" $target_library] $synthetic_library]
read_file -f verilog [getenv "INFILE"]
#/* This command will fix the problem of having */
#/* assign statements left in your structural file. */
set_fix_multiple_port_nets -all -buffer_constants
#do the actual compilation (synthesis)
compile -ungroup_all
check_design
#/* always do change_names before write... */
redirect change_names { change_names -rules verilog -hierarchy -verbose }
# write out the structural Verilog
write -f verilog -output [getenv "OUTFILE"]
quit
```

What beh2str leaves out...

- Timing!
  - No clock defined so no target speed
  - No wire load model, so fewer placement constraints
  - No input drive defined so assume infinite drive
  - No output load define so assume something

Copy this from /uusoc/facility/cad_common/local/class/6710/F17/synopsys

.synopsys_dc.setup

```
…
set SynopsysInstall [getenv "SYNOPSYS"]

set search_path [list . \
[format "%s%s"  $SynopsysInstall /libraries/syn] \
[format "%s%s"  $SynopsysInstall /dw/sim_ver] \
]
define_design_lib WORK -path ./WORK
set synthetic_library [list dw_foundation.sldb]
set synlib_wait_for_design_license [list "DesignWare-Foundation"]
set link_library [concat  [concat  "*" $target_library] $synthetic_library]
set symbol_library [list generic.sdb]
…
```

syn-script.tcl

- */uusoc/facility/cad_common/local/class/6710/F17/synopsys*

```
#/* search path should include directories with memory .db files */
#/* as well as the standard cells */
set search_path [list . \
[format "%s%s" SynopsysInstall /libraries/syn] \
[format "%s%s" SynopsysInstall /dw/sim_ver] \
!!your-library-path-goes-here!!]
#/* target library list should include all target .db files */
set target_library [list !!your-library-name!!.db]
#/* synthetic_library is set in .synopsys_dc.setup to be */
#/* the dw_foundation library. */
set link_library [concat [concat "*" $target_library] $synthetic_library]
```

syn-script.tcl

```
#/* below are parameters that you will want to set for each design */
#/* list of all HDL files in the design */
set myFiles [list !!all-your-structural-Verilog-files!! ]
set fileFormat verilog              ;# verilog or VHDL
set basename !!basename!!           ;# Name of top-level module
set myClk !!clk!!                   ;# The name of your clock
set virtual 0                       ;# 1 if virtual clock, 0 if real clock
#/* compiler switches... */
set useUltra 1                       ;# 1 for compile_ultra, 0 for compile
                                    # mapEffort, useUngroup are for
                                    # non-ultra compile...
set mapEffort1 medium               ;# First pass - low, medium, or high
set mapEffort2 medium               ;# second pass - low, medium, or high
set useUngroup 1                    ;# 0 if no flatten, 1 if flatten
```

syn-script.tcl

```
#/* Timing and loading information */
set myPeriod_ns !!10!!              ;# desired clock period (sets speed goal)
set myInDelay_ns !!0.25!!           ;# delay from clock to inputs valid
set myOutDelay_ns !!0.25!!          ;# delay from clock to output valid
set myInputBuf !!INVX4!!            ;# name of cell driving the inputs
set myLoadLibrary !!Lib!!           ;# name of library the cell comes from
set myLoadPin !!A!!                 ;# pin that outputs drive

#/* Control the writing of result files */
set runname struct ;# Name appended to output files
```

syn-script.tcl

```
#/* the following control which output files you want. */
#/* They should be set to 1 if you want the file, 0 if not */
set write_v 1      ;# compiled structural Verilog file
set write_ddc 0  ;# compiled file in ddc format
set write_sdf 0   ;# sdf file for back-annotated timing sim
set write_sdc 1  ;# sdc constraint file for place and route
set write_rep 1   ;# report file from compilation
set write_pow 0 ;# report file for power estimate
```

syn-script.tcl

```
# analyze and elaborate the files
analyze -format $fileFormat -lib WORK $myfiles
elaborate $basename -lib WORK -update
current_design $basename
# The link command makes sure that all the required design
# parts are linked together.
# The uniquify command makes unique copies of replicated modules.
link
uniquify
# now you can create clocks for the design
if { $virtual == 0 } {
   create_clock -period $myPeriod_ns $myClk
   } else {
   create_clock -period $myPeriod_ns -name $myClk
   }
```

syn-script.tcl

```
# Set the driving cell for all inputs except the clock
# The clock has infinite drive by default. This is usually
# what you want for synthesis because you will use other
# tools (like EDI) to build the clock tree (or define it by hand).
set_driving_cell -library $myLoadLibrary -lib_cell $myInputBuf \
    [remove_from_collection [all_inputs] $myClk]
# set the input and output delay relative to myclk
set_input_delay $myInDelay_ns -clock $myClk \
    [remove_from_collection [all_inputs] $myClk]
set_output_delay $myOutDelay_ns -clock $myClk [all_outputs]
# set the load of the circuit outputs in terms of the load
# of the next cell that they will drive, also try to fix hold time issues
set_load [load_of  [format "%s%s%s%s%s" $myLoadLibrary \
                                "/" $myInputBuf "/" $myLoadPin]] [all_outputs]
set_fix_hold $myClk
```

syn-script.tcl

```
# now compile the design with given mapping effort
# and do a second compile with incremental mapping
# or use the compile_ultra meta-command
if { $useUltra == 1 } {
    compile_ultra
    } else {
    if { $useUngroup == 1 } {
        compile -ungoup_all -map_effort $mapEffort1
        compile -incremental_mapping -map_effort $mapEffort2
        } else {
        compile -map_effort $mapEffort1
        compile -incremental_mapping -map_effort $mapEffort2
        }
    }
```

syn-script.tcl

```
# Check things for errors
check_design
report_constraint -all_violators
set filebase [format "%s%s%s" $basename "_"  $runname]
# structural (synthesized) file as verilog
if { $write_v == 1 } {
    set filename [format "%s%s" $filebase ".v"]
    redirect change_names { change_names -rules verilog \
                                -hierarchy -verbose }
    write -format verilog -hierarchy -output $filename
}
# write the rest of the desired files... then quit
```

Using Scripts

- Modify syn-script.tcl or write your own
- syn-dc –f scriptname.tcl
- Make sure to check output!!!!

addsub_struct.v – 10ns spec

```
Startpoint: addnsub (input port clocked by clk)
Endpoint: result[8] (output port clocked by clk)
Path Group: clk
Path Type: max

Point                                    Incr       Path
-----------------------------------------------------------
clock clk (rise edge)                    0.00       0.00
clock network delay (ideal)              0.00       0.00
input external delay                     0.25       0.25 f
addnsub (in)                             0.00       0.25 f
U79/Y (INVX4)                            0.37       0.62 r
U20/Y (NOR2X1)                           0.64       1.26 f
...
U16/Y (NOR2X1)                           0.33       8.30 r
U17/Y (NOR2X1)                           0.53       8.83 f
result[8] (out)                          0.00       8.83 f
data arrival time                                   8.83

clock clk (rise edge)                   10.00      10.00
clock network delay (ideal)              0.00      10.00
output external delay                   -0.25       9.75
data required time                                  9.75
-----------------------------------------------------------------------------

data required time                                  9.75
data arrival time                                  -8.83
-----------------------------------------------------------------------------
slack (MET)                                         0.92
```

```
Number of ports:                 26
Number of nets:                 111
Number of cells:                 94
Number of combinational cells:   94
Number of sequential cells:       0
Number of macros:                 0
Number of buf/inv:               21
Number of references:             7
```

addsub_struct.v – 4ns spec

```
Startpoint: b[3] (input port clocked by clk)
Endpoint: result[6] (output port clocked by clk)
Path Group: clk
Path Type: max

Point                                    Incr       Path
-----------------------------------------------------------
clock clk (rise edge)                    0.00       0.00
clock network delay (ideal)              0.00       0.00
input external delay                     0.25       0.25 r
b[3] (in)                                0.00       0.25
U16/Y (INVX4)                            0.07       0.32 f
U117/Y (NAND2X1)                         0.41       0.73 r
...
U152/Y (NAND2X1)                         0.24       3.74 r
result[6] (out)                          0.00       3.74 r
data arrival time                                   3.74

clock clk (rise edge)                    4.00       4.00
clock network delay (ideal)              0.00       4.00
output external delay                   -0.25       3.75
data required time                                  3.75
-----------------------------------------------------------

data required time                                  3.75
data arrival time                                  -3.74
-----------------------------------------------------------
slack (MET)                                         0.01
```

```
Number of ports:                 26
Number of nets:                 248
Number of cells:                231
Number of combinational cells:  231
Number of sequential cells:       0
Number of macros:                 0
Number of buf/inv:               79
Number of references:             9
```

addsub_struct.v – 3ns spec

```
Startpoint: a[1] (input port clocked by clk)
 Endpoint: result[7] (output port clocked by clk)
 Path Group: clk
 Path Type: max
 Point                                    Incr       Path
 -----------------------------------------------------------
 input external delay                     0.25       0.25 r
 a[1] (in)                                0.00       0.25 r
 U147/Y (INVX4)                           0.07       0.32 f
 U80/Y (NAND2X1)                          0.24       0.55 r
…
 U132/Y (NAND2X1)                         0.30       3.59 r
 result[7] (out)                          0.00       3.59 r
 data arrival time                                   3.59
 clock clk (rise edge)                    3.00       3.00
 clock network delay (ideal)              0.00       3.00
 output external delay                   -0.25       2.75
 data required time                                  2.75
-----------------------------------------------------------
 data required time                                  2.75
 data arrival time                                  -3.59
 -----------------------------------------------------------
 slack (VIOLATED)                                   -0.84
```

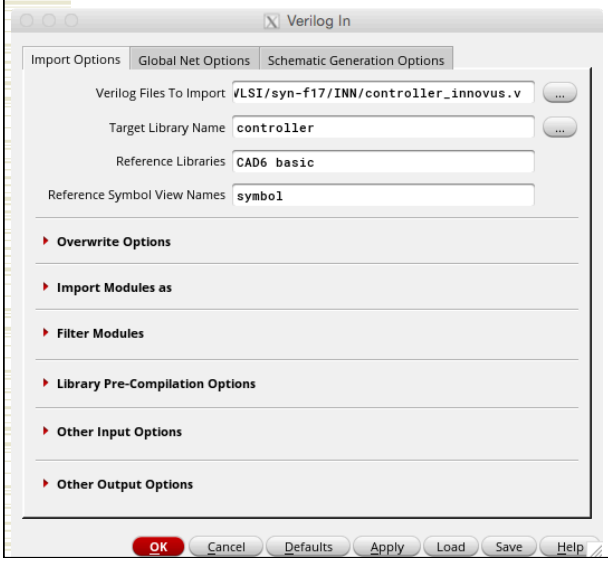
```
Number of ports:                 26
Number of nets:                 247
Number of cells:                230
Number of combinational cells:  230
Number of sequential cells:       0
Number of macros:                 0
Number of buf/inv:               77
Number of references:             9
```

addsub_struct.v – 3ns spec

In the running log information…

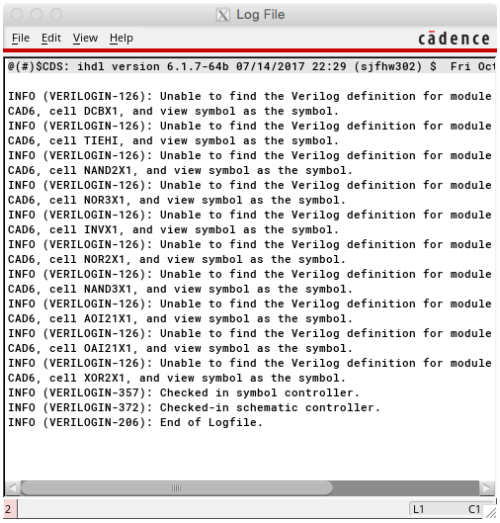
```
max_delay/setup ('clk' group)

                          Required        Actual
  Endpoint                Path Delay      Path Delay        Slack
  ---------------------------------------------------------------
  result[7]                  2.75            3.59 r        -0.84  (VIOLATED)
  result[6]                  2.75            3.58 r        -0.83  (VIOLATED)
  result[5]                  2.75            3.56 r        -0.81  (VIOLATED)
  result[3]                  2.75            3.46 r        -0.71  (VIOLATED)
  result[8]                  2.75            3.40 r        -0.65  (VIOLATED)
  result[4]                  2.75            3.39 r        -0.64  (VIOLATED)
  result[2]                  2.75            3.29 r        -0.54  (VIOLATED)
  result[1]                  2.75            3.26 f        -0.51  (VIOLATED)
```

Import Structural Verilog - Sch



Slightly different Example – controller.v Synthesized with Synopsys, placed and routed with Innovus, but an example of importing structural Verilog…

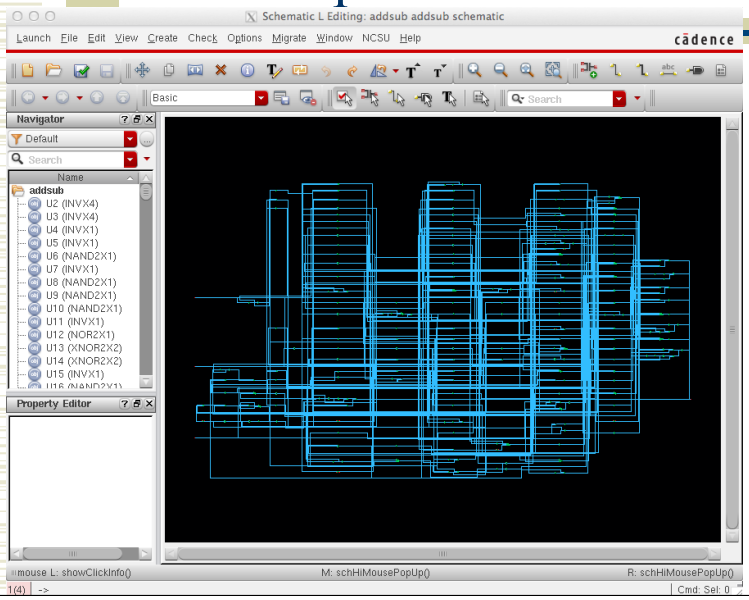
Verilog Import Log



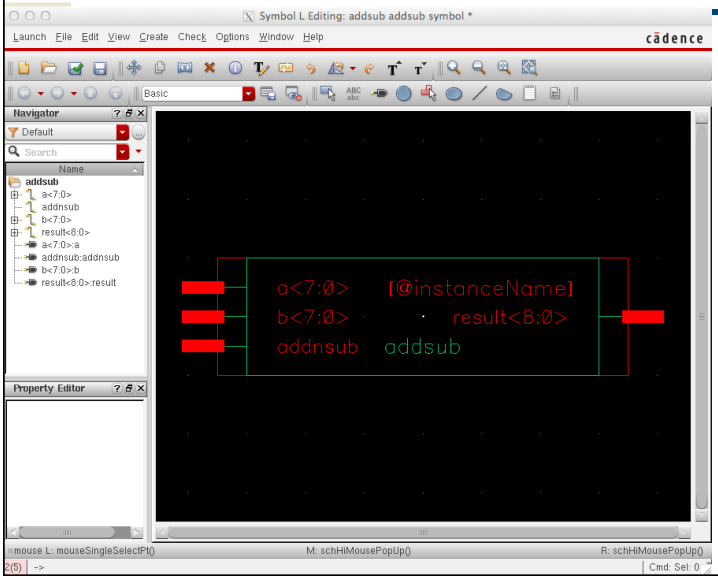
If you want to avoid these warnings, you need a text file with Verilog module names and interfaces for all your cells.

But, if you don't have this, Cadence will use your symbols which is fine…

Imported Schematic



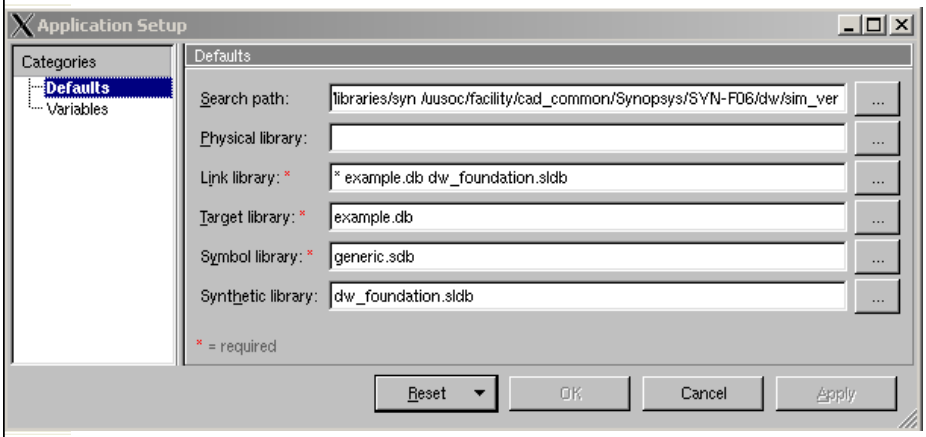
Imported Symbol



Using Design Vision

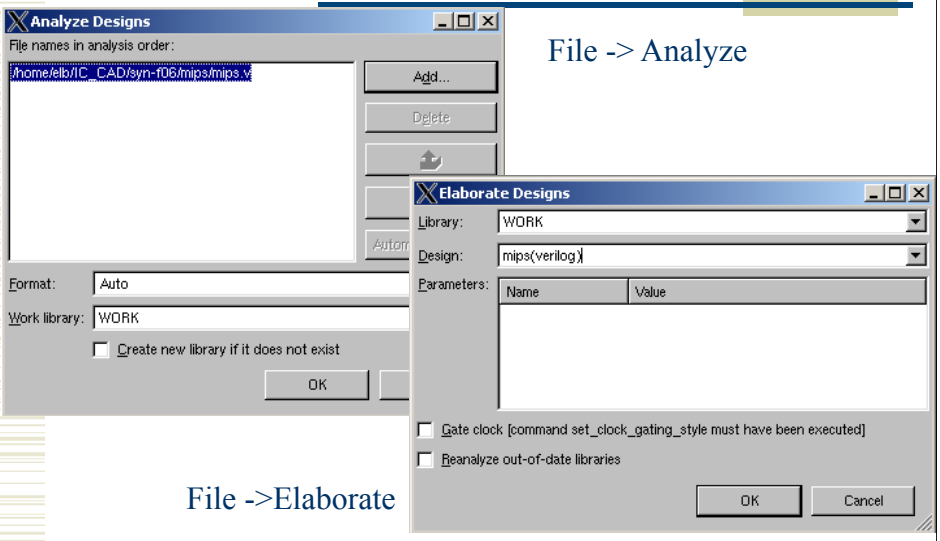
- You can do all of these commands from the design vision gui if you like
- syn-dv
- Follow the same steps as the script
  - Set libraries in your own .synopsys_dc.setup
  - analyze/elaborate
  - define clock and set constraints
  - compile
  - write out results

Setup



File ->Setup

analyze/elaborate

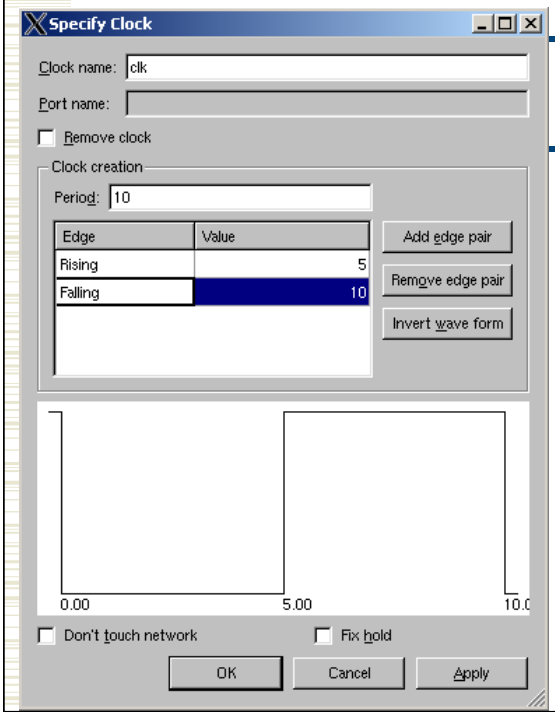


File -> Analyze

File ->Elaborate

Look at results...

```
|       RAM_reg       | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  | N  |
|       RAM_reg       | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  | N  |
|       RAM_reg       | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  | N  |
|       RAM_reg       | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  | N  |
|       RAM_reg       | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  | N  |
==========================================================================
Statistics for MUX_OPs
==========================================================================
|      block name/line         | Inputs | Outputs | # sel inputs | MB |
==========================================================================
| regfile_WIDTH8_REGBITS3/304 |   8    |    8    |      3       | N  |
| regfile_WIDTH8_REGBITS3/305 |   8    |    8    |      3       | N  |
==========================================================================
Presto compilation completed successfully.
Information: Building the design 'alu' instantiated from design 'datapath_WIDTH8_REGBITS3' with
        the parameters "8". (HDL-193)
Statistics for case statements in always block at line 279 in file
        '/home/elb/IC_CAD/syn-f06/mips/mips.v'
===============================================
|           Line           |  full/ parallel  |
===============================================
|           280            |     auto/auto     |
===============================================
Presto compilation completed successfully.
Information: Building the design 'zerodetect' instantiated from design 'datapath_WIDTH8_REGBITS3' with
 the parameters "8". (HDL-193)
Presto compilation completed successfully.
design_vision-xg-t>
```

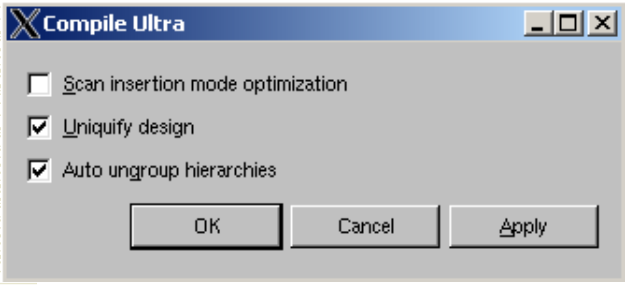


Define clock

attributes -> specify clock

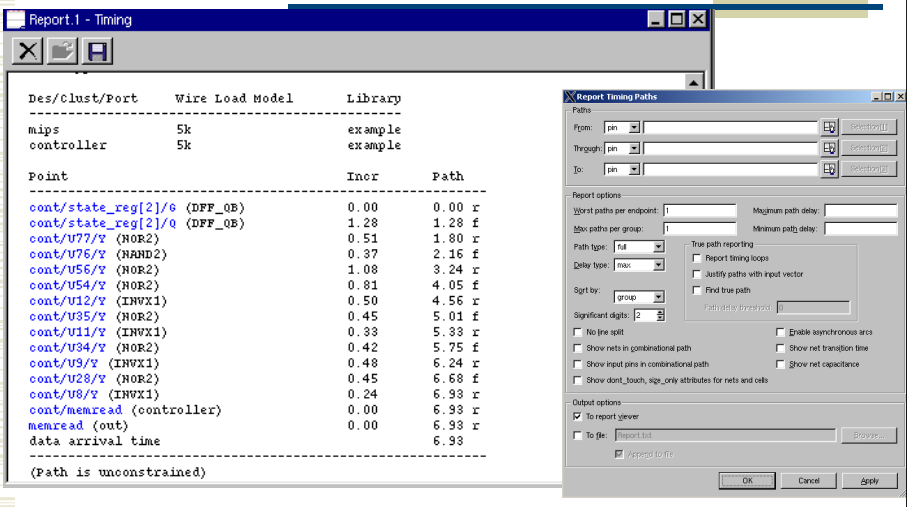
Also look at other attributes...

Compile



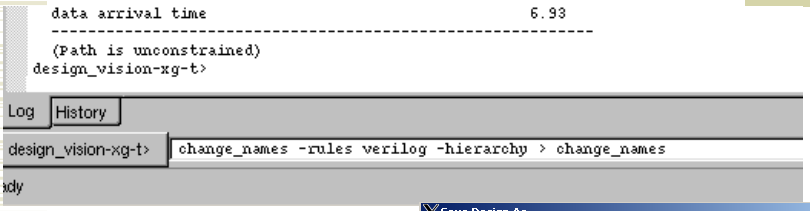
Design -> Compile Ultra

Timing Reports



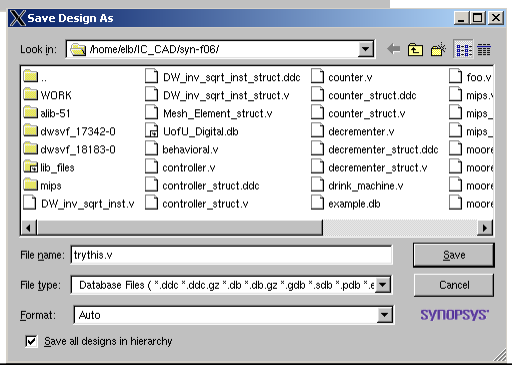
Timing -> Report Timing Path

Write Results



change_names

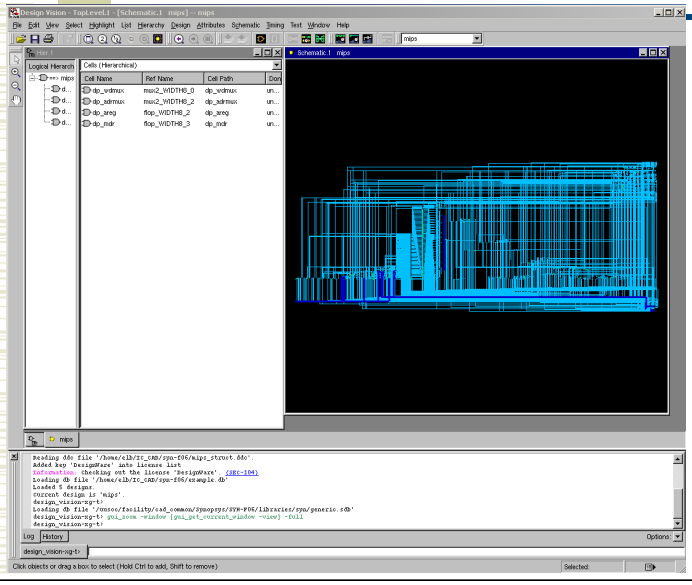
File -> Save As...



Or, use syn-dv after script...

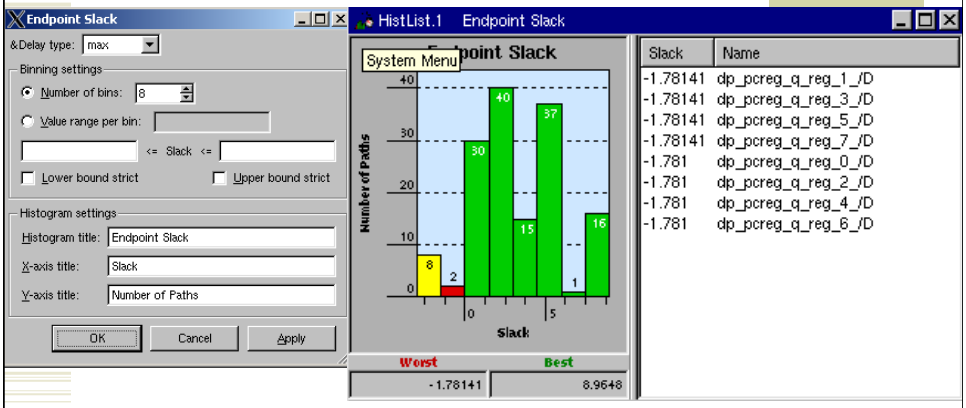
- syn-dc –f mips.tcl
- results in .v, .ddc, .sdc, .rep files
- Read the .ddc file into syn-dv and use it to explore timing...

syn-dv with mips_struct.v



File -> Read

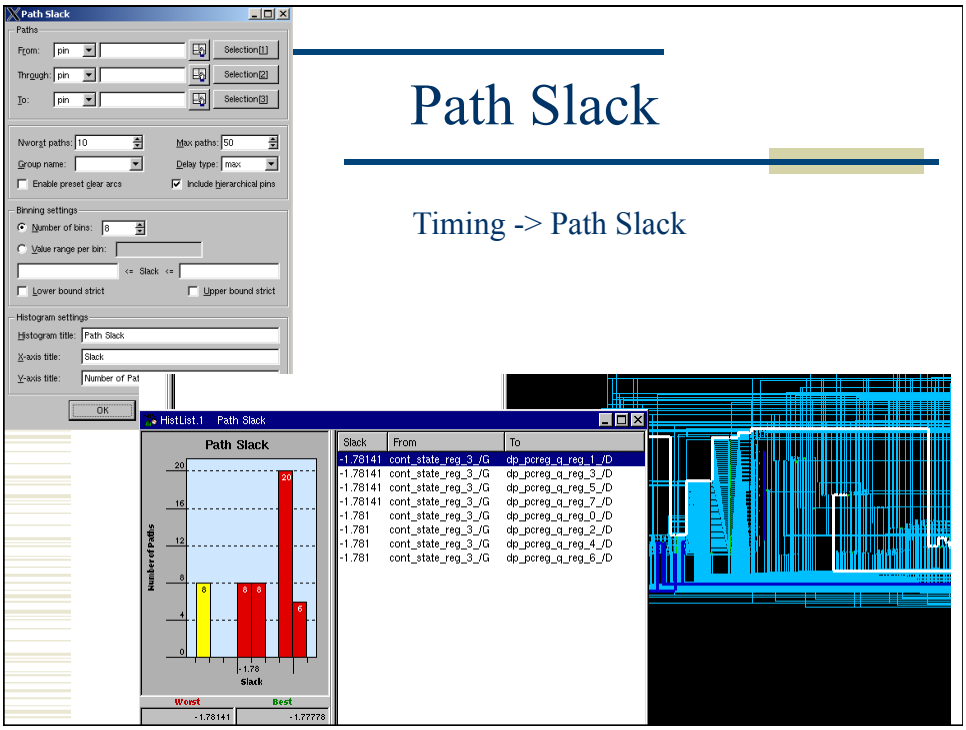
Endpoint slack...



Timing -> Endpoint Slack

Path Slack

Timing -> Path Slack



Summary

- Behavioral -> Structural -> Layout
- Can be automated by scripting, but make sure you know what you're doing
  - on-line tutorials for TCL
    - Google "tcl tutorial"
  - Synopsys documentation for design_compiler
- End up with structural Verilog based on your cells
  - Perfect input for place and route! ☺