

Chapter 9

Verilog Synthesis

```

/* -- c -- */
/*****
/* Defaults can be found in $SYNOPTSYS/admin/setup/.synopsys_dc.setup */
/*****
set company {University of Utah}
set designer {Erik Brunvand}
/*****
set plot_command {lpr -Pcsp}
set text_print_command {lpr -Pcsp}
set text_editor_command {emacs %s &}
set command_log_file "./synopsys-dc_shell.log"
set view_command_log_file "./synopsys-dc_shell-view.log"
set find_converts_name_lists "false"
/*****

set SynopsysInstall [getenv "SYNOPTSYS"]

# You can add to this search path if you have your libraries
# stored somewhere else...
set search_path [list . \
[format "%s%s" $SynopsysInstall /libraries/syn] \
[format "%s%s" $SynopsysInstall /dw/sim_ver] ]

/* ===== */
/* Define a work library in the current project dir */
/* to hold temporary files and keep the project area */
/* uncluttered. Note: You must create a subdirectory */
/* in your project directory called WORK. */
/* ===== */
define_design_lib WORK -path ./WORK

/* ===== */
/* General configuration settings. */
/* ===== */

set hdlin_check_no_latch true
set compile_fix_multiple_port_nets true
set hdlin_translate_off_skip_text true

set verilogout_write_components true
set verilogout_architecture_name "structural"
set verilogout_no_tri true

set hdlin_translate_off_skip_text true
set bus_naming_style {%s[%d]}

set target_library [list UofU_Digital_v1_2.db]
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]

```

Figure 9.1: Example Synopsys `synopsys_dc.setup` file

```
/* Behavioral Model of an Inverter */
module INV_Test(INV_in,INV_out);
    input INV_in;
    output INV_out;
    assign INV_out = ~INV_in;
endmodule
```

Figure 9.2: Verilog behavioral description of an inverter

```
> behv2str

CORRECT>beh2str (y|n|e|a)? yes
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
```

Figure 9.3: Usage information for the **beh2str** script

```
module INV_Test ( INV_in, INV_out );
    input INV_in;
    output INV_out;

    INVX1 U2 ( .A(INV_in), .Y(INV_out) );
endmodule
```

Figure 9.4: Synthesized inverter using a cell from the target library

```
//-----  
// Very simple moore-style FSM  
// Erik Brunvand, University of Utah  
//-----  
module moore (clk, clr, insig, outsig);  
input clk, clr, insig;  
output outsig;  
  
// define state encodings as parameters  
parameter [1:0] s0 = 2'b00, s1 = 2'b01, s2 = 2'b10, s3 = 2'b11;  
  
// define reg vars for state register and next_state logic  
reg [1:0] state, next_state;  
  
//define state register (with asynchronous active-low clear)  
always @(posedge clk or negedge clr)  
begin  
    if (clr == 0) state = s0;  
    else state = next_state;  
end  
  
// define combinational logic for next_state  
always @(insig or state)  
begin  
    case (state)  
        s0: if (insig) next_state = s1; else next_state = s0;  
        s1: if (insig) next_state = s2; else next_state = s1;  
        s2: if (insig) next_state = s3; else next_state = s2;  
        s3: if (insig) next_state = s1; else next_state = s0;  
        default: next_state = s0;  
    endcase  
end  
  
// now set the outsig. This could also be done in an always  
// block... but in that case, outsig would have to be  
// defined as a reg.  
assign outsig = ((state == s1) || (state == s3));  
endmodule
```

Figure 9.5: Simple state machine

```

module moore ( clk, clr, insig, outsig );
  input clk, clr, insig;
  output outsig;
  wire  state_1_, n12, n13, n14, n15, n16, n17, n18;
  wire  [1:0] next_state;

  dff_qb state_reg_0_ ( .D(next_state[0]), .G(clk), .CLR(clr), .Q(outsig), .QB(n12) );
  dff_qb state_reg_1_ ( .D(next_state[1]), .G(clk), .CLR(clr), .Q(state_1_) );
  mux2 U13 ( .A(n13), .B(n12), .S(state_1_), .Y(next_state[1]) );
  nor2 U14 ( .A(n14), .B(n12), .Y(n13) );
  nand2 U15 ( .A(n15), .B(n16), .Y(next_state[0]) );
  inv U16 ( .A(n17), .Y(n16) );
  nor2 U17 ( .A(n14), .B(n18), .Y(n17) );
  nand2 U18 ( .A(n18), .B(n14), .Y(n15) );
  inv U19 ( .A(insig), .Y(n14) );
  nor2 U20 ( .A(n12), .B(state_1_), .Y(n18) );
endmodule

```

Figure 9.6: Result of running `beh2str` on `moore.v`

```

# 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

compile -ungroup_all
check_design

/* always do change_names before write...             */
redirect change_names { change_names -rules verilog -hierarchy -verbose }

write -f verilog -output [getenv "OUTFILE"]
quit

```

Figure 9.7: `beh2str.tcl` basic synthesis command script

```
#####
# General Synthesis Script Template for Synopsys Design Compiler
# Erik Brunvand, 2007
#####
# "Your library path" may be empty if your library will be in
# your synthesis directory because `.` is already on the path.
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]

# 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-files!!]
set fileFormat verilog ;# verilog or VHDL
set basename !!basename!! ;# choose a basename for the output files
set myClk !!clk!! ;# The name of your clock
set virtual 0 ;# 1 if virtual clock, 0 if real clock

# Timing (in ns) and loading information
set myPeriod_ns !!10!! ;# desired clock period (sets speed goal)
set myInDelay_ns !!0.5!! ;# delay from clock to inputs valid
set myOutDelay_ns !!0.5!! ;# delay from clock to output valid
set myInputBuf !!INVX4!! ;# name of cell driving the inputs
set myLoadLibrary !!UofU_Digital!! ;# name of library the cell comes from
set myLoadPin !!A!! ;# name of pin that outputs drive

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

# 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 (XG-mode)
set write_sdf 0 ;# sdf file for back-annotated timing sim
set write_sdc 0 ;# sdc constraint file for place and route
set write_rep 1 ;# report file from compilation
set write_pow 0 ;# report file for power estimate
```

Figure 9.8: Part 1 of the `syn-script.tcl` synthesis script

```

module moore ( clk, clr, insig, outsig );
  input clk, clr, insig;
  output outsig;
  wire  n3, n5, n6;
  wire  [1:0] next_state;

  DCX1 state_reg_0_ ( .D(next_state[0]), .CLK(clk), .CLR(clr), .Q(outsig) );
  DCBX1 state_reg_1_ ( .D(next_state[1]), .CLK(clk), .CLR(clr), .QB(n3) );
  XNOR2X1 U6 ( .A(insig), .B(n5), .Y(next_state[0]) );
  NAND2X1 U3 ( .A(n3), .B(outsig), .Y(n5) );
  NAND2X1 U7 ( .A(insig), .B(n3), .Y(n6) );
  MUX2NX1 U8 ( .A(n3), .B(n6), .S(outsig), .Y(next_state[1]) );
endmodule

```

Figure 9.9: Result of running `syn-dc` with the `syn-script.tcl` on `moore.v`

```

Inferred memory devices in process
  in routine moore line 14 in file
    './moore.v'.
=====
| Register Name | Type | Width | Bus | MB | AR | AS | SR | SS | ST |
=====
| state_reg | Flip-flop | 2 | Y | N | Y | N | N | N | N |
=====

```

Figure 9.10: Inferred memory from the `moore.v` example

```

link
Linking design 'moore'
Using the following designs and libraries:
-----
moore /home/elb/IC_CAD/syn-f06/moore.db
example (library) /home/elb/IC_CAD/syn-f06/UofU_Digital.db
dw_foundation.sldb (library) /cad_common/Synopsys/SYN-F07/libraries/syn/dw_foundation.sldb

```

Figure 9.11: Link information from the `dc_shell` synthesis process (with `dw_foundation` path shortened)

```

*****
Report : timing
       -path full
       -delay max
       -max_paths 1
Design : moore
Version: Z-2007.03-SP4
Date   : Wed Nov  5 22:10:50 2008
*****

Operating Conditions: typical   Library: UofU_Digital_v1_2
Wire Load Model Mode: top

Startpoint: state_reg_0_
            (rising edge-triggered flip-flop clocked by clk)
Endpoint:   state_reg_0_
            (rising edge-triggered flip-flop 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
state_reg_0_/CLK (DCX1)        0.00      0.00 r
state_reg_0_/Q (DCX1)         1.00      1.00 f
U3/Y (NAND2X1)                 0.48      1.48 r
U6/Y (XNOR2X1)                 0.23      1.71 f
state_reg_0_/D (DCX1)          0.00      1.71 f
data arrival time              1.71

clock clk (rise edge)          10.00     10.00
clock network delay (ideal)    0.00     10.00
state_reg_0_/CLK (DCX1)        0.00     10.00 r
library setup time             -0.44      9.56
data required time              9.56

-----
data required time              9.56
data arrival time              -1.71
-----
slack (MET)                     7.85

```

Figure 9.12: Timing report for the `moore.v` synthesis using `syn-script.tcl`

```
*****
Report : area
Design : moore
Version: Z-2007.03-SP4
Date   : Wed Nov  5 22:10:50 2008
*****

Library(s) Used:

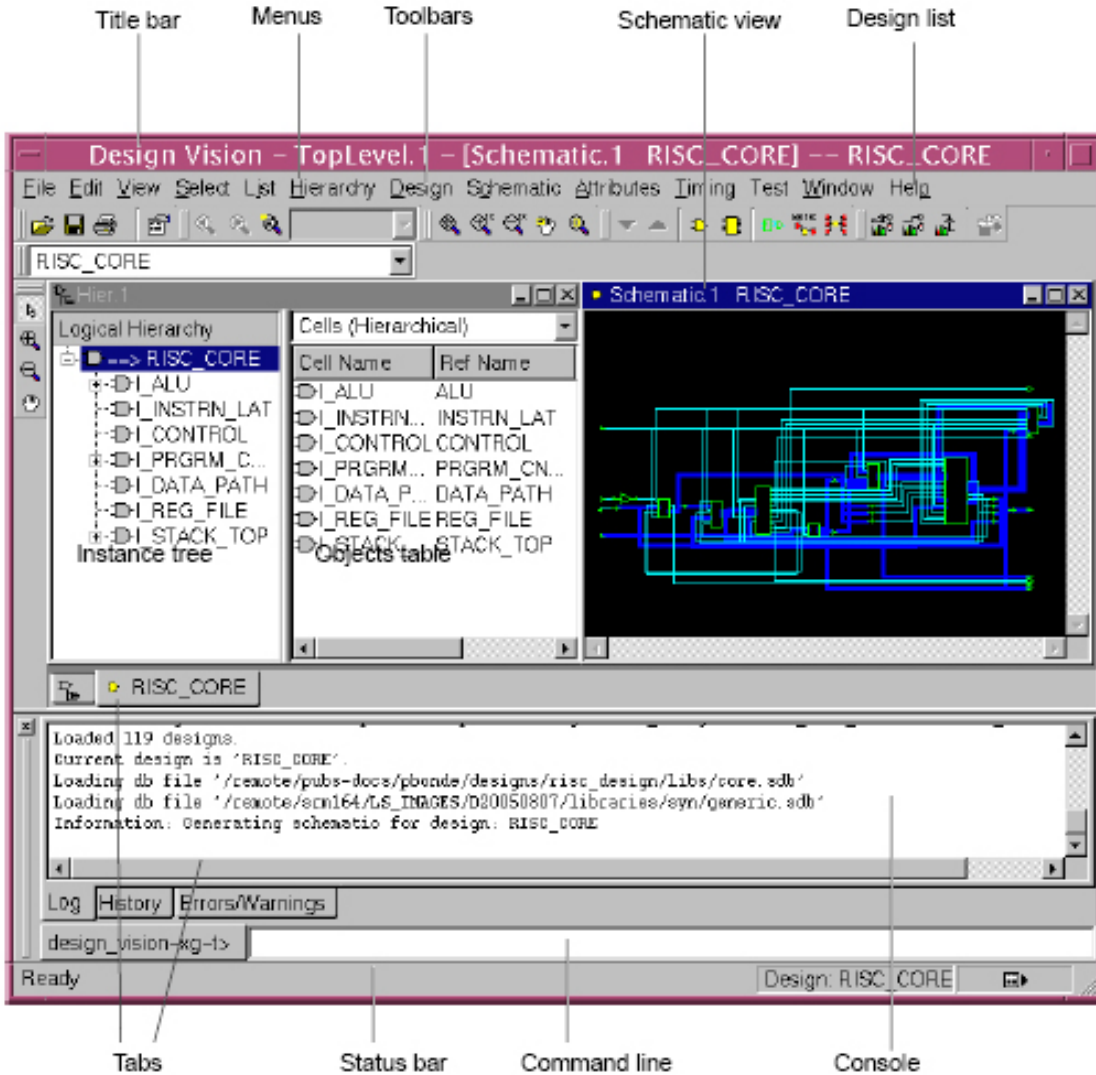
  UofU_Digital_v1_2 (File: /uusoc/facility/res/async/elb/IC_CAD/syn-f06/UofU_Digital_v1_2.db)

Number of ports:          4
Number of nets:           9
Number of cells:          6
Number of references:     5

Combinational area:      23.000000
Noncombinational area:  36.000000
Net Interconnect area:  undefined (No wire load specified)

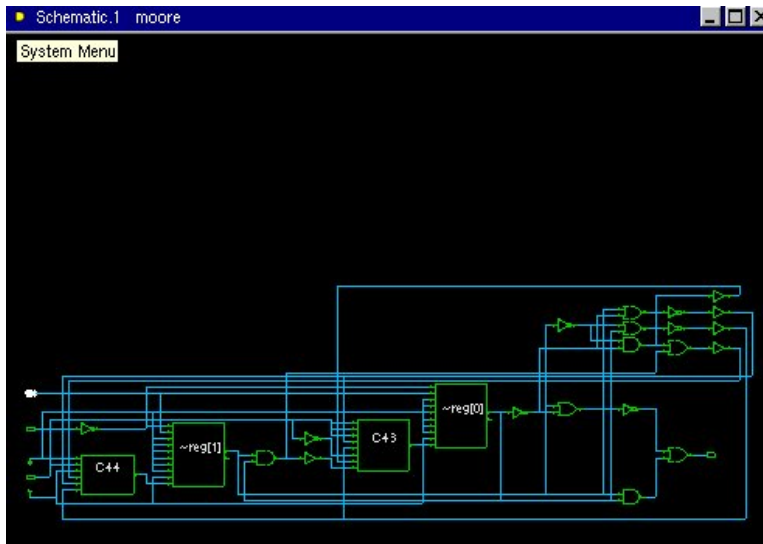
Total cell area:         59.000000
Total area:              undefined
```

Figure 9.13: Area report for the `moore.v` synthesis using `syn-script.tcl`



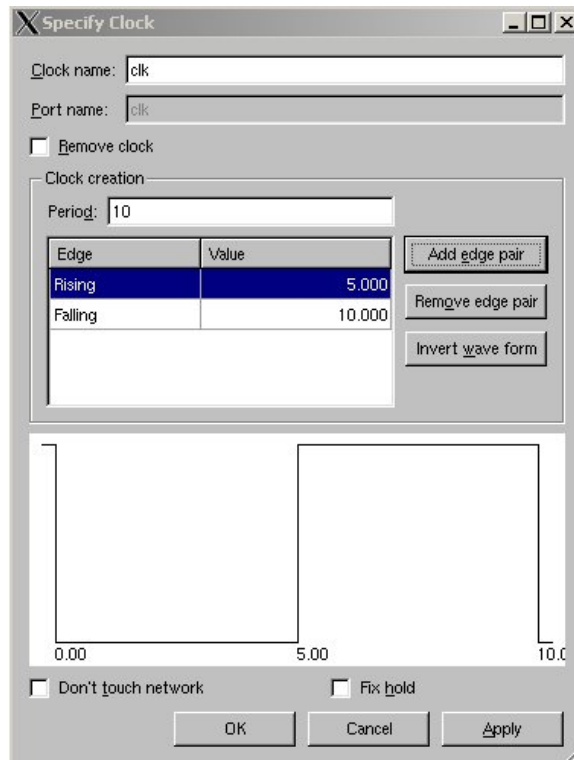
(Reprinted by permission of Synopsys, Inc. Copyright © 2007, 2010 Synopsys, Inc. All Rights Reserved)

Figure 9.14: General view of the *Design Vision* GUI



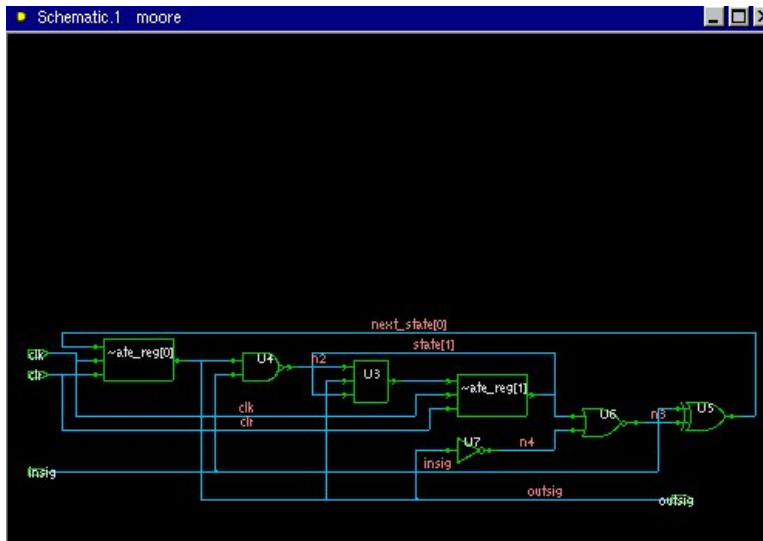
(Reprinted by permission of Synopsys, Inc. Copyright ©2007, 2010 Synopsys, Inc. All Rights Reserved)

Figure 9.15: Initial mapping of the `moore.v` example



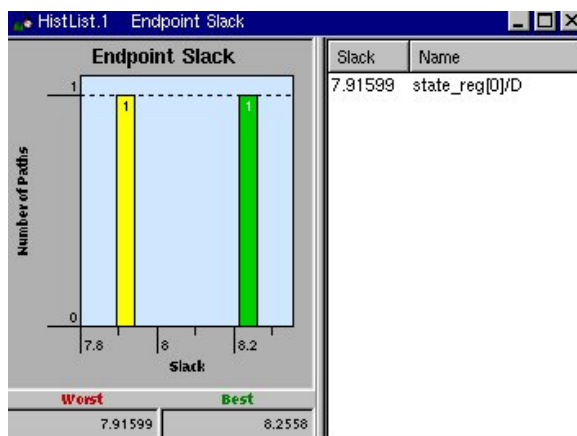
(Reprinted by permission of Synopsys, Inc. Copyright ©2007, 2010 Synopsys, Inc. All Rights Reserved)

Figure 9.16: Clock definition in *Design Vision*



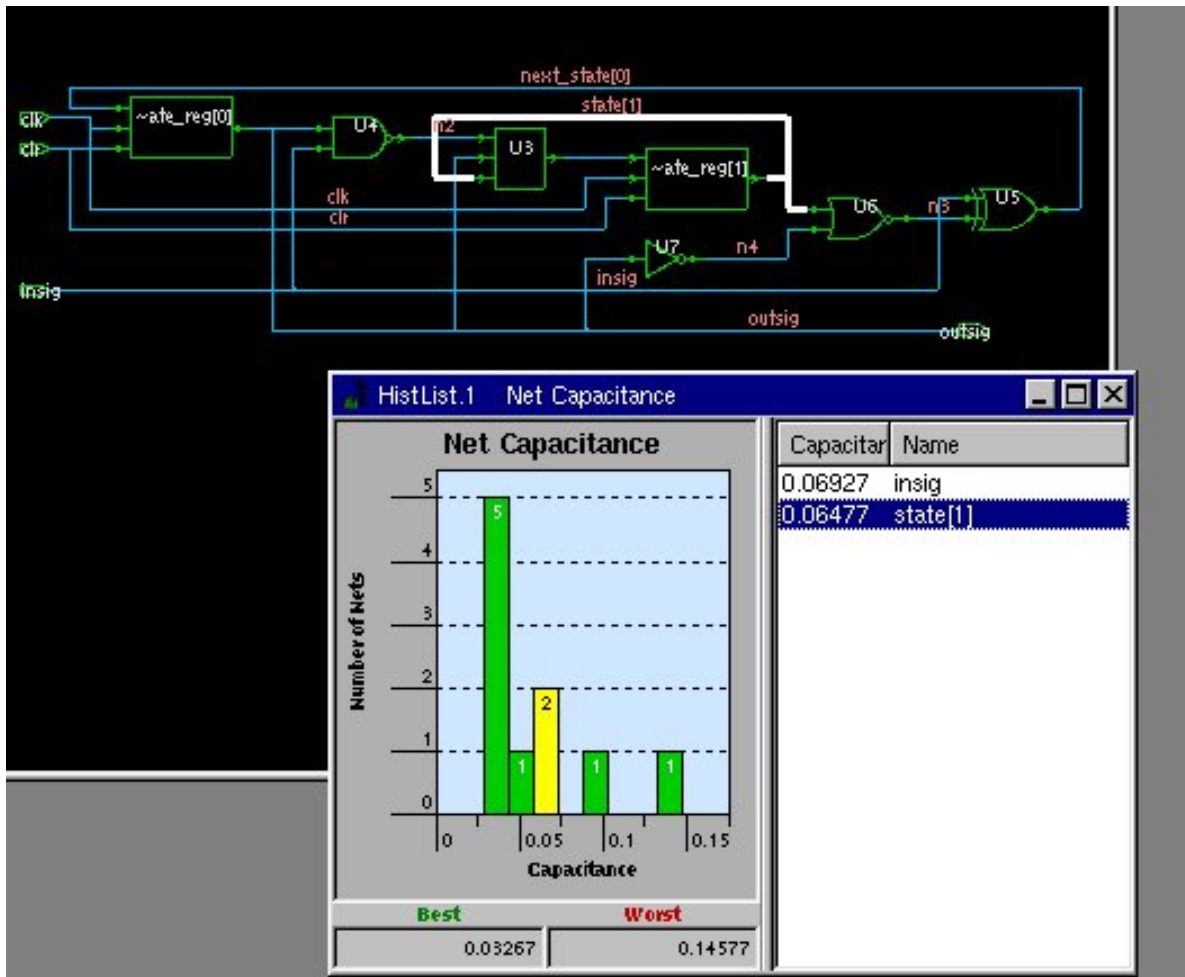
(Reprinted by permission of Synopsys, Inc. Copyright ©2007, 2010 Synopsys, Inc. All Rights Reserved)

Figure 9.17: Final mapping of the `moore.v` example



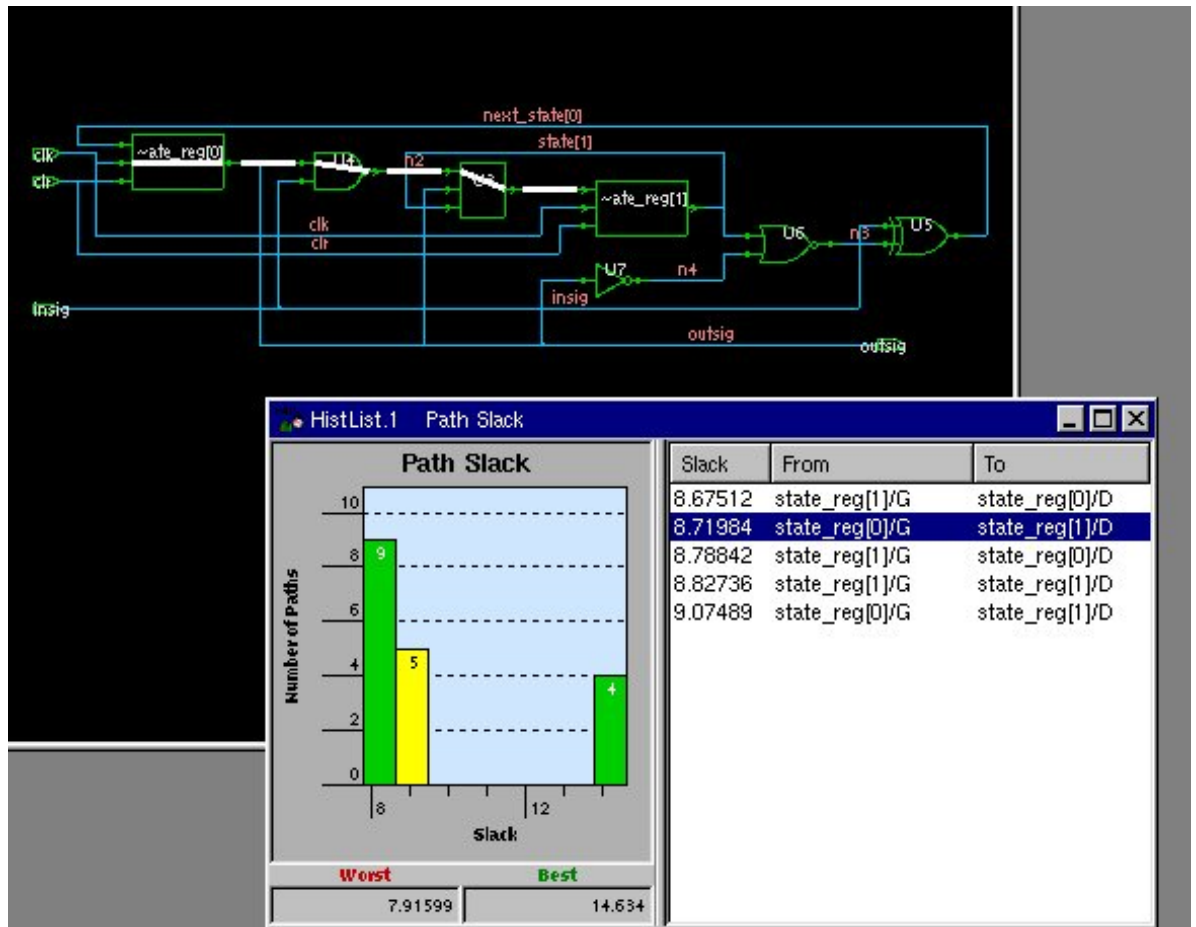
(Reprinted by permission of Synopsys, Inc. Copyright ©2007, 2010 Synopsys, Inc. All Rights Reserved)

Figure 9.18: Endpoint slack for the two endpoints in `moore`



(Reprinted by permission of Synopsys, Inc. Copyright © 2007, 2010 Synopsys, Inc. All Rights Reserved)

Figure 9.19: Wiring capacitance histogram with highlighted path



(Reprinted by permission of Synopsys, Inc. Copyright ©2007, 2010 Synopsys, Inc. All Rights Reserved)

Figure 9.20: Timing path histogram with highlighted path

```

module decrementer( inst_A, SUM_inst );

    parameter width = 8;

    input [width-1 : 0] inst_A;
    output [width-1 : 0] SUM_inst;

    // Instance of DW01_dec
    DW01_dec #(width) U1( .A(inst_A), .SUM(SUM_inst) );

endmodule

```

Figure 9.21: *DesignWare* 8-bit decrementer instantiated in Verilog code

```

module decrementer ( inst_A, SUM_inst );
    input [7:0] inst_A;
    output [7:0] SUM_inst;
    wire  n1, n2, n3, n4, n5, n6, n7, n8, n9;

    NAND2X1 U1 ( .A(SUM_inst[0]), .B(n1), .Y(n2) );
    NOR2X1 U2 ( .A(inst_A[2]), .B(n2), .Y(n4) );
    NAND2X1 U3 ( .A(n4), .B(n3), .Y(n5) );
    NOR2X1 U4 ( .A(inst_A[4]), .B(n5), .Y(n7) );
    NAND2X1 U5 ( .A(n7), .B(n6), .Y(n8) );
    INVX2 U6 ( .A(inst_A[0]), .Y(SUM_inst[0]) );
    INVX2 U7 ( .A(inst_A[1]), .Y(n1) );
    INVX2 U8 ( .A(inst_A[5]), .Y(n6) );
    INVX2 U9 ( .A(inst_A[3]), .Y(n3) );
    OAI21X1 U10 ( .A(n1), .B(SUM_inst[0]), .C(n2), .Y(SUM_inst[1]) );
    XNOR2X1 U11 ( .A(inst_A[2]), .B(n2), .Y(SUM_inst[2]) );
    OAI21X1 U12 ( .A(n4), .B(n3), .C(n5), .Y(SUM_inst[3]) );
    XNOR2X1 U13 ( .A(inst_A[4]), .B(n5), .Y(SUM_inst[4]) );
    OAI21X1 U14 ( .A(n7), .B(n6), .C(n8), .Y(SUM_inst[5]) );
    XNOR2X1 U15 ( .A(inst_A[6]), .B(n8), .Y(SUM_inst[6]) );
    NOR2X1 U16 ( .A(inst_A[6]), .B(n8), .Y(n9) );
    XOR2X1 U17 ( .A(inst_A[7]), .B(n9), .Y(SUM_inst[7]) );
endmodule

```

Figure 9.22: Structural code for the 8-bit *DesignWare* decrementer


```
#####
# Script for Cadence RTL Compiler synthesis
# Erik Brunvand, 2008
# Use with syn-rtl -f <rtl-script>
#####

# Set the search paths to the libraries and the HDL files
# Remember that "." means your current directory
set_attribute hdl_search_path {./} ;# Search path for Verilog files
set_attribute lib_search_path {./} ;# Search path for library files
set_attribute library [list UofU_Digital_v1_2.lib] ;# Target Library
set_attribute information_level 6 ;# See a lot of warnings.

set myFiles [list moore.v] ;# All your HDL files
set basename moore ;# name of top level module
set myClk clk ;# clock name
set myPeriod_ps 10000 ;# Clock period in ps
set myInDelay_ps 250 ;# delay from clock to inputs valid
set myOutDelay_ps 250 ;# delay from clock to output valid
set runname RTL ;# name appended to output files

#####
#* below here shouldn't need to be changed... *
#####

# Analyze and Elaborate the HDL files
read_hdl ${myFiles}
elaborate ${basename}

# Apply Constraints and generate clocks
set clock [define_clock -period ${myPeriod_ps} -name ${myClk} [clock_ports]]
external_delay -input $myInDelay_ps -clock ${myClk} [find / -port ports_in/*]
external_delay -output $myOutDelay_ps -clock ${myClk} [find / -port ports_out/*]

# Sets transition to default values for Synopsys SDC format,
# fall/rise 400ps
dc::set_clock_transition .4 $myClk

# check that the design is OK so far
check_design -unresolved
report timing -lint

# Synthesize the design to the target library
synthesize -to_mapped

# Write out the reports
report timing > ${basename}_${runname}_timing.rep
report gates > ${basename}_${runname}_cell.rep
report power > ${basename}_${runname}_power.rep

# Write out the structural Verilog and sdc files
write_hdl -mapped > ${basename}_${runname}.v
write_sdc > ${basename}_${runname}.sdc
quit
```

Figure 9.23: Synthesis script for Cadence *RTL Compiler* using *moore.v* as the behavioral Verilog

```
// Generated by Cadence Encounter(r) RTL Compiler v06.20-s027_1

module moore(clk, clr, insig, outsig);
  input clk, clr, insig;
  output outsig;
  wire clk, clr, insig;
  wire outsig;
  wire n_0, n_3, n_4, n_5, n_6, n_7, n_8;
  DCX1 \state_reg[0] (.CLR (clr), .CLK (clk), .D (n_8), .Q (outsig));
  OAI21X1 g192(.A (n_6), .B (n_4), .C (n_7), .Y (n_8));
  DCBX1 \state_reg[1] (.CLR (clr), .CLK (clk), .D (n_5), .Q (n_0), .QB());
  OAI21X1 g194(.A (n_6), .B (n_0), .C (insig), .Y (n_7));
  NOR2X1 g195(.A (n_3), .B (n_6), .Y (n_5));
  INVX1 g197(.A (n_3), .Y (n_4));
  XNOR2X1 g196(.A (outsig), .B (n_0), .Y (n_6));
  NOR2X1 g198(.A (insig), .B (n_0), .Y (n_3));
endmodule
```

Figure 9.24: Result of running `syn-rtl` with the `moore.tcl` synthesis script

```
=====
Generated by:      Encounter (r) RTL Compiler v06.20-s027_1
Generated on:     Nov 03 2008 03:34:13 PM
Module:          moore
Technology library: UofU_Digital_v1_2
Operating conditions: typical (balanced_tree)
Wireload mode:   enclosed
=====
```

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)
(clock clk)	launch					0 R
state_reg[1]/CLK				400		0 R
state_reg[1]/Q	DCBX1	3	102.8	508	+1492	1492 F
g196/B					+0	1492
g196/Y	XNOR2X1	3	93.2	306	+694	2186 F
g194/A					+0	2186
g194/Y	OAI21X1	1	21.4	259	+370	2556 R
g192/C					+0	2556
g192/Y	OAI21X1	1	26.4	186	+309	2865 F
state_reg[0]/D	DCX1				+0	2865
state_reg[0]/CLK	setup			400	+207	3072 R
(clock clk)	capture					10000 R

```
-----
Timing slack : 6928ps
Start-point : state_reg[1]/CLK
End-point : state_reg[0]/D
```

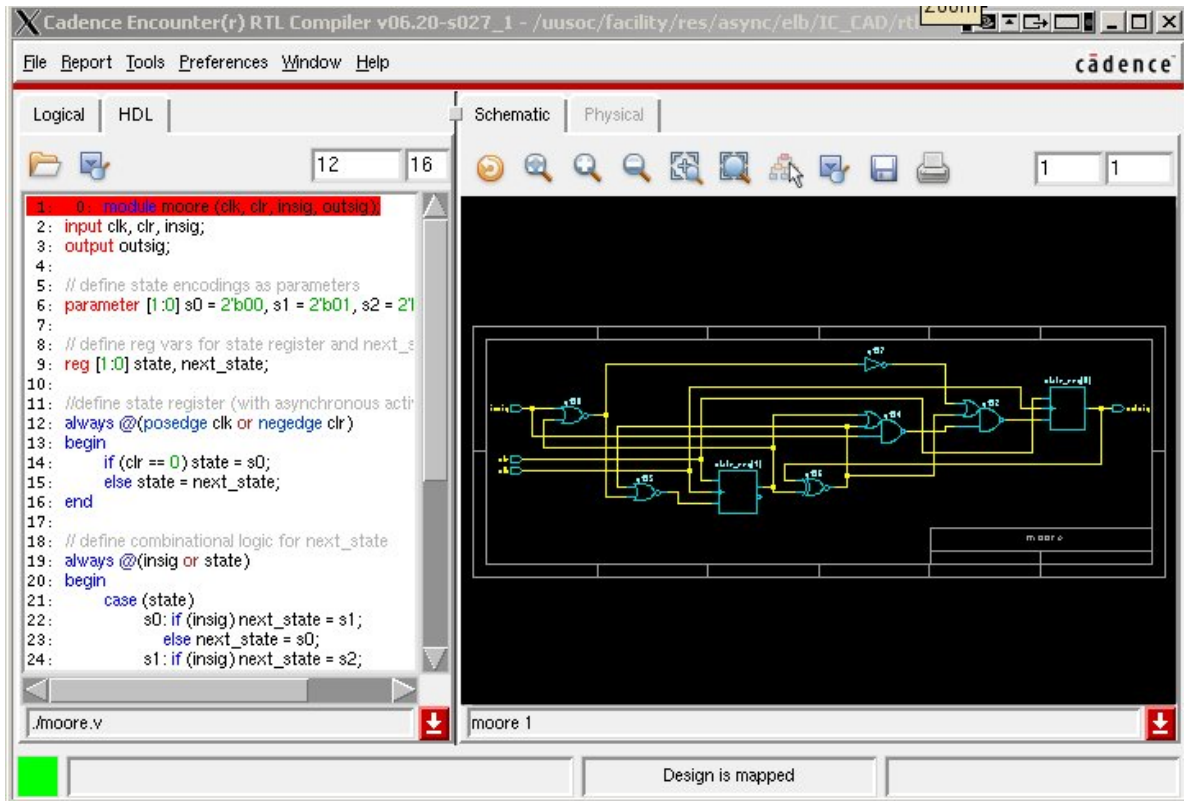
Figure 9.25: Timing report for the `moore.v` synthesis using `moore.tcl`

```
=====
Generated by:      Encounter(r) RTL Compiler v06.20-s027_1
Generated on:     Nov 03 2008  03:34:13 PM
Module:          moore
Technology library: UofU_Digital_v1_2
Operating conditions: typical (balanced_tree)
Wireload mode:   enclosed
=====
```

Gate	Instances	Area	Library
DCBX1	1	18.000	UofU_Digital_v1_2
DCX1	1	18.000	UofU_Digital_v1_2
INVX1	1	3.000	UofU_Digital_v1_2
NOR2X1	2	8.000	UofU_Digital_v1_2
OAI21X1	2	10.000	UofU_Digital_v1_2
XNOR2X1	1	8.000	UofU_Digital_v1_2
total	8	65.000	

Type	Instances	Area	Area %
sequential	2	36.000	55.4
inverter	1	3.000	4.6
logic	5	26.000	40.0
total	8	65.000	100.0

Figure 9.26: Area report for the `moore.v` synthesis using `moore.tcl`



(Copyright ©2007, 2010, Cadence Design Systems, Inc. All rights reserved worldwide. Reprinted with permission.)

Figure 9.27: General view of the *RTL Compiler* GUI with the *moore.v* example

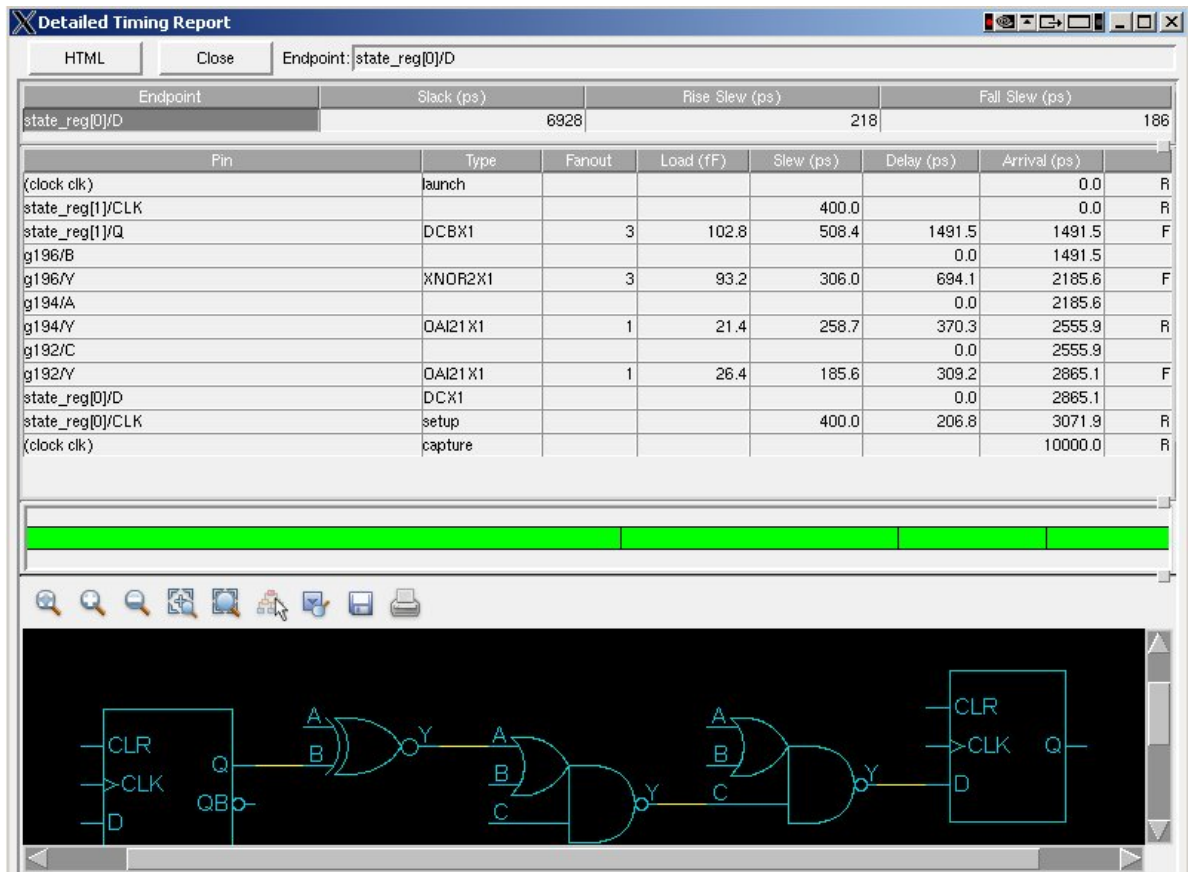
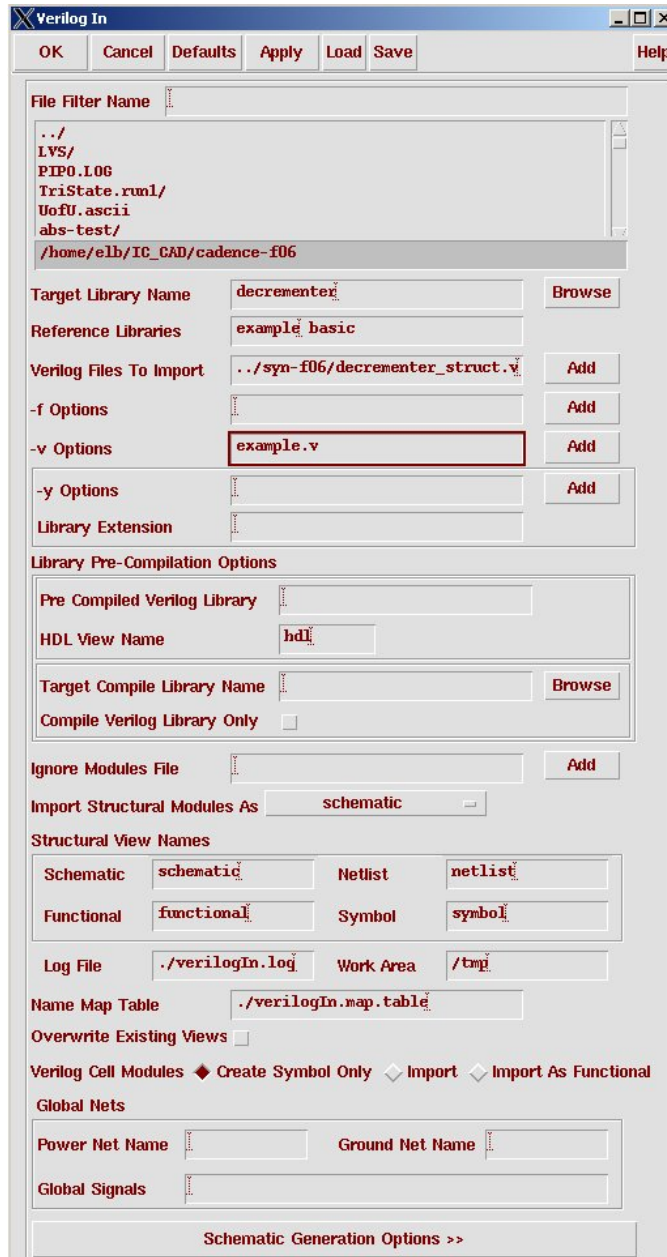
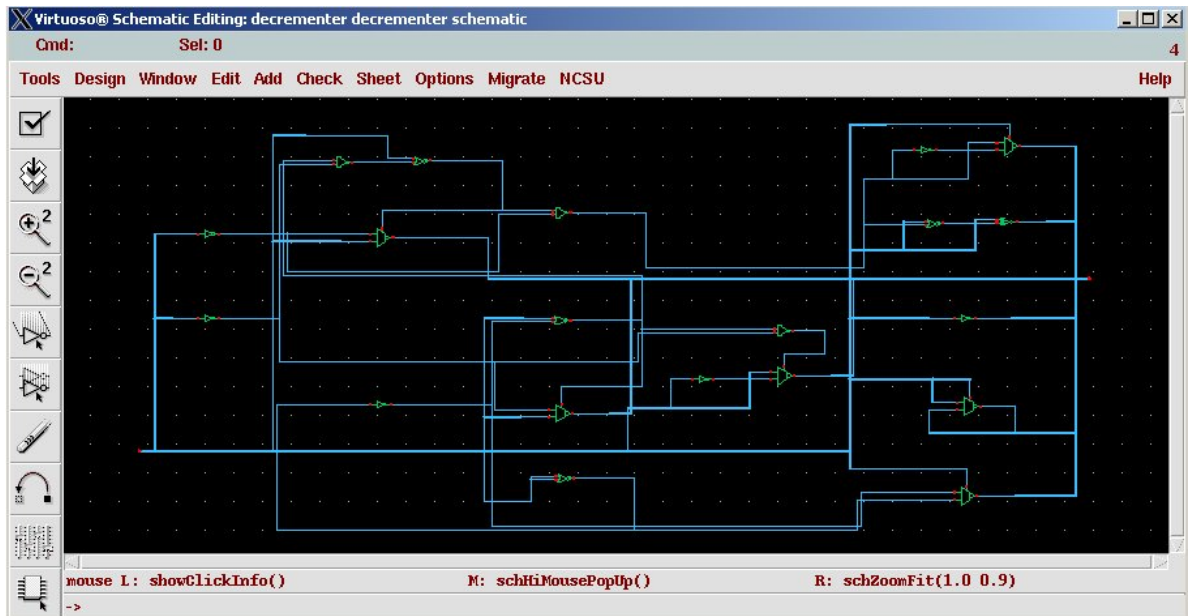


Figure 9.28: Timing report showing the worst-case path in the moore example



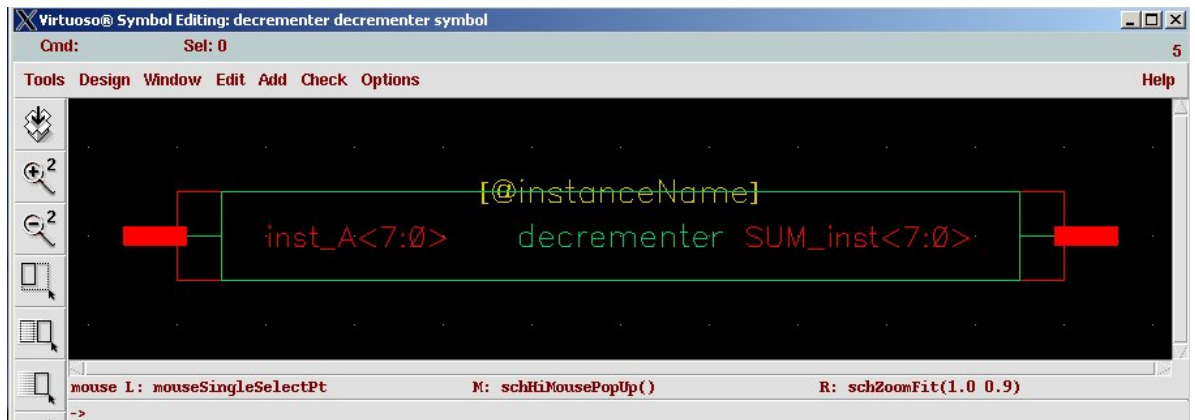
(Copyright ©2005, 2010, Cadence Design Systems, Inc. All rights reserved worldwide. Reprinted with permission.)

Figure 9.29: Dialog box for importing structural Verilog into a new schematic view



(Copyright © 2005, 2010, Cadence Design Systems, Inc. All rights reserved worldwide. Reprinted with permission.)

Figure 9.30: Schematic that results from importing the decrementer into Composer



(Copyright © 2005, 2010, Cadence Design Systems, Inc. All rights reserved worldwide. Reprinted with permission.)

Figure 9.31: Symbol that is created for the decrementer

```

// Verilog stimulus file.
// Please do not create a module in this file.

// Default verilog stimulus.

initial
begin

    clk = 1'b0;
    clr = 1'b0;
    insig = 1'b0;

#10 clr = 1;
    $display("clr=%b, insig=%b, outsig=%b, ns=%b", clr, insig, outsig, top.next_state);
#10 clk = 1;
#10 clk = 0;
    $display("clr=%b, insig=%b, outsig=%b, ns=%b", clr, insig, outsig, top.next_state);
#10 insig = 1;
#10 clk = 1;
#10 clk=0;
    $display("clr=%b, insig=%b, outsig=%b, ns=%b", clr, insig, outsig, top.next_state);
#10 clk = 1;
#10 clk=0;
    $display("clr=%b, insig=%b, outsig=%b, ns=%b", clr, insig, outsig, top.next_state);
#10 clk = 1;
#10 clk=0;
    $display("clr=%b, insig=%b, outsig=%b, ns=%b", clr, insig, outsig, top.next_state);
#10 clk = 1;
#10 clk=0;
    $display("clr=%b, insig=%b, outsig=%b, ns=%b", clr, insig, outsig, top.next_state);
end

```

Figure 9.32: Simple testbench for the simple finite state machine

```

// Verilog HDL for "UofU_Digital_v1_2", "NAND2X1" "behavioral"

module NAND2X1 (Y, A, B);
    output Y;
    input A;
    input B;

    nand _i0(Y, A, B);

    specify
        (A => Y) = (1.0, 1.0);
        (B => Y) = (1.0, 1.0);
    endspecify
endmodule

```

Figure 9.33: Behavioral Verilog code for NAND2X1 with `specify` blocks


```
(CELL
  (CELLTYPE "NAND2X1")
  (INSTANCE U7)
  (DELAY
    (ABSOLUTE
      (IOPATH A Y (0.175:0.175:0.175) (0.130:0.130:0.130))
      (IOPATH B Y (0.301:0.301:0.301) (0.282:0.287:0.287))
    )
  )
)
```

Figure 9.34: SDF delay information for an instance of the NAND2X1 cell

```
module test;

wire outsig;
reg clk, clr, insig;

`include "testbench.v"

moore top(clk, clr, insig, outsig);

endmodule
```

Figure 9.35: Top-level Verilog file with one instance of the module `moore`

```
-v UofU_Digital_behv.v
time.v
moore_struct.v
top.v
```

Figure 9.36: File list (named `files.f`) for stand-alone Verilog simulation of the structural Verilog of the `moore` finite state machine

```

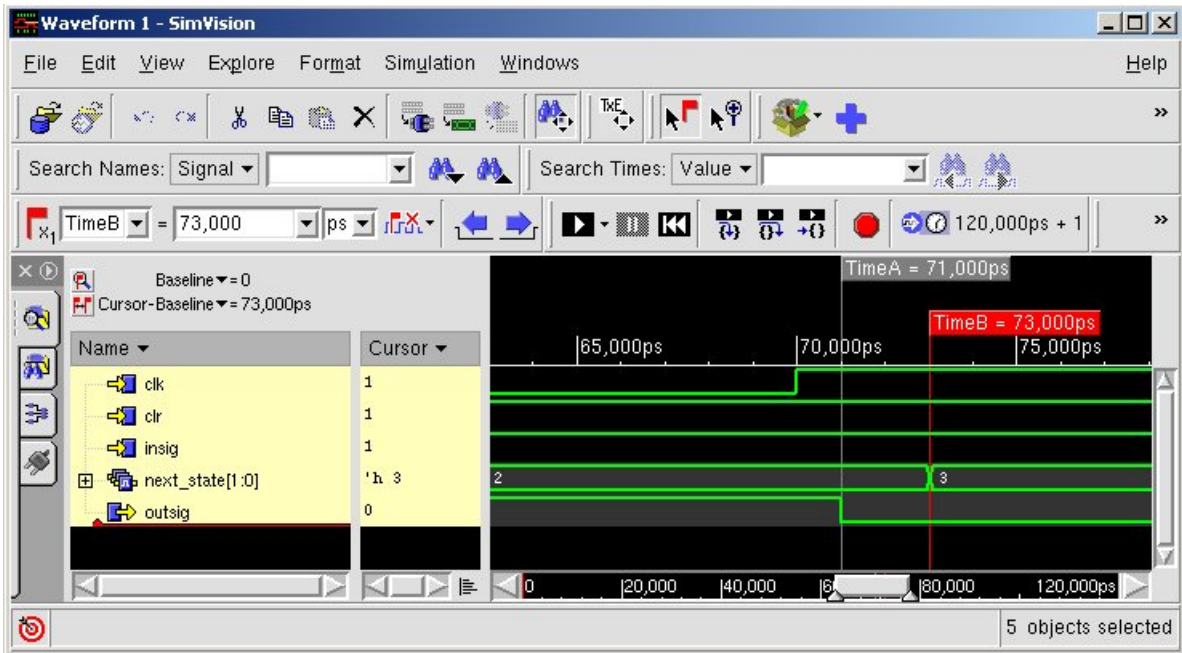
module moore ( clk, clr, insig, outsig );
  input clk, clr, insig;
  output outsig;
  wire  n3, n5, n6;
  wire  [1:0] next_state;

  // Annotate the cell instances with timing from the sdf file
  // produced by the synthesis procedure
  initial $sdf_annotate("moore_struct.sdf");

  DCX1 state_reg_0_ ( .D(next_state[0]), .CLK(clk), .CLR(clr), .Q(outsig) );
  DCBX1 state_reg_1_ ( .D(next_state[1]), .CLK(clk), .CLR(clr), .QB(n3) );
  XNOR2X1 U6 ( .A(insig), .B(n5), .Y(next_state[0]) );
  NAND2X1 U3 ( .A(n3), .B(outsig), .Y(n5) );
  NAND2X1 U7 ( .A(insig), .B(n3), .Y(n6) );
  MUX2NX1 U8 ( .A(n3), .B(n6), .S(outsig), .Y(next_state[1]) );
endmodule

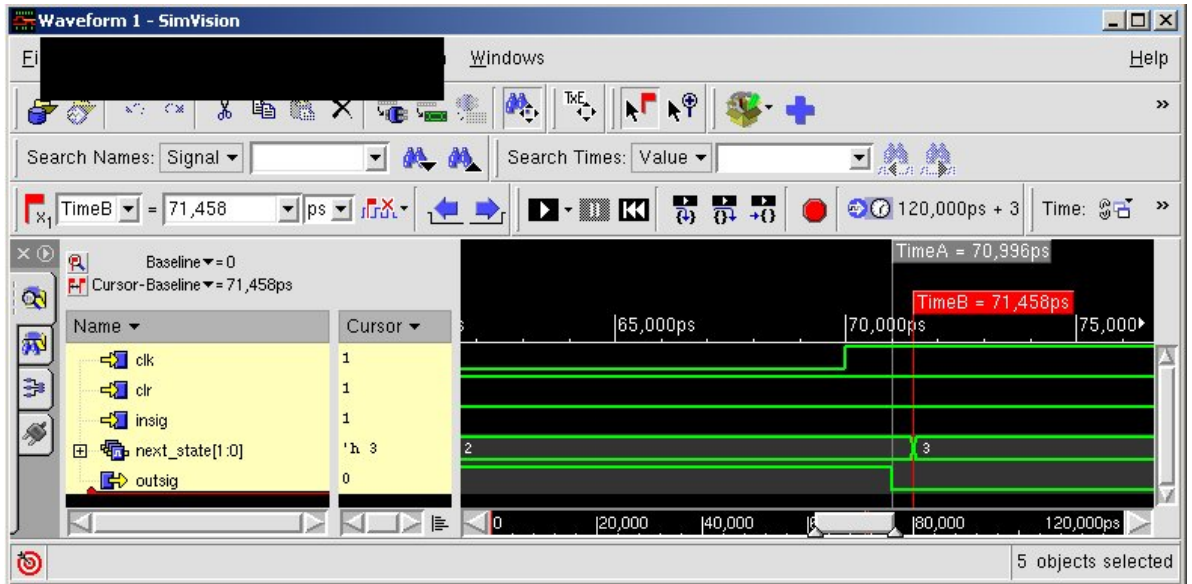
```

Figure 9.37: Example of including *sdf* annotation into a structural Verilog description of the *moore* example



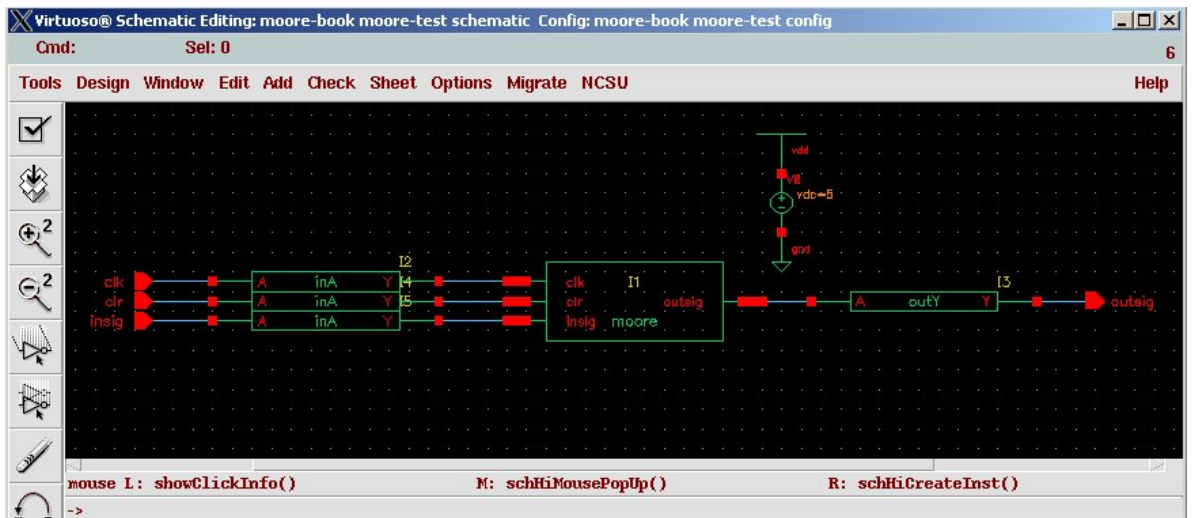
(Copyright ©2005, 2010, Cadence Design Systems, Inc. All rights reserved worldwide. Reprinted with permission.)

Figure 9.38: *NC_Verilog* waveform window showing unit-delay timing in a simulation of the *moore* FSM



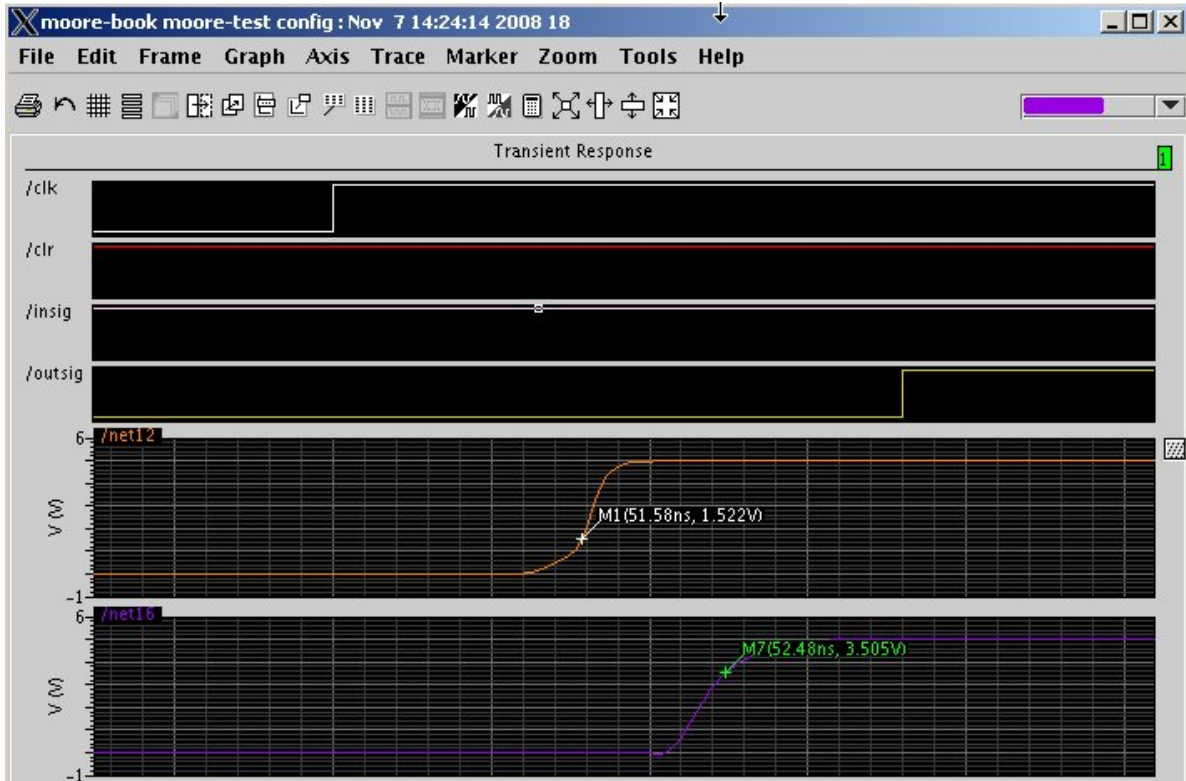
(Copyright © 2005, 2010, Cadence Design Systems, Inc. All rights reserved worldwide. Reprinted with permission.)

Figure 9.39: *NC_Verilog* waveform window showing annotated SDF timing in a simulation of the moore FSM



(Copyright © 2005, 2010, Cadence Design Systems, Inc. All rights reserved worldwide. Reprinted with permission.)

Figure 9.40: Schematic view for mixed analog/digital simulation of the moore FSM



(Copyright ©2005, 2010, Cadence Design Systems, Inc. All rights reserved worldwide. Reprinted with permission.)

Figure 9.41: Waveforms from mixed analog/digital simulation of the moore FSM