Introduction to CMOS VLSI Design

Lab04: Logic Synthesis and Place and Route

Peter Kogge
Joseph Nahas
University of Notre Dame
Fall 2010

Eric Brunvand, *Digital VLSI Chip Design with Cadence and Synopsys CAD Tools*,
Section 9.2 and Chapter 11

Synthesis and Place and Route Process

- Synthesis (Cadence RTL Compile):
  - Convert Behavioral Verilog to Structural Verilog
- Place and Route (Cadence SOC Encounter)
  - Import the design and design files
  - Floorplan
  - Add Power Rings
  - Place Cells
  - First Timing Optimization
  - Add Clock Tree
  - Second Timing Optimization
  - Final Routing

Getting Started

- Start Xterm (not PuTTY) on UNIX/Linux system.
  - Do not start the Cadence environment – you will not need it.
- Copy the example files to your working directory
  - Change to your working directory
  - `cp -r /opt/und/cadence/ncsu-cdk-1.5.1/lib/Lab04_Example .`
  - The directory Lab04_Example should be copied to your directory.
  - *cd* to the Lab04_Example directory
- The directory has the following files (among others):
  - counter.v – a verilog description of a counter.
  - counter_RTL.tck – a tcl script for the Cadence RTL Compiler.
  - UofU_Digital_v1_2.lib – cell library (timing, power, functionality) file
  - UofU_Digital_v1_2.lef– cell abstract file (cell layout)
  - syn-rtl – command file for RTL Compiler
  - syn-rtlg – command file for RTL Compiler starting GUI
  - cad-soc – command file for Encounter
  - counter.io – list of pins in counter.v and desired location in layout

Note space and period
i.e. copy to current directory

Structural Logic Synthesis Using the Cadence Encounter RTL Compiler

Encounter RTL Compiler Synthesis

- Inputs
  - Behavioral Verilog
  - Control Script (tcl)
- Outputs
  - Structural Verilog
  - List of cells
  - Power Estimate
  - Timing Estimate
  - SDC Timing File
  - RC Log File

8 bit Counter

```
module counter (clk, clr, load, in, count);
parameter width=8;
input clk, clr, load;
input [width-1 : 0] in;
output [width-1 : 0] count;
reg [width-1 : 0] tmp;

always @(posedge clk or negedge clr)
begin
  if (!clr)
     tmp = 0;
  else if (load)
     tmp = in;
  else
     tmp = tmp + 1;
end
assign count = tmp;
endmodule
```


syn-rtlg
This is the top-level run script

```
#!/bin/tcsh
#####################################################################
# Startup script for Cadence RTL Compiler
# Erik Brunvand, University of Utah
# Modified by J. Nahas, University of Notre Dame
#####################################################################

source /opt/und/cadence/.cadencerc

# remind the user where they are...
echo "Working directory is"  $PWD

# Start icfb (pass args through if there are any)
rc -gui $argv
```

argv will be tcl script on next page

counter_RTL.tcl (1/2)

This is the tcl script run by the top level script.

You will need to modify this part of the script for your project.

```
#####################################################
# Script for Cadence RTL Compiler synthesis
# Erik Brunvand, 2008
# Modified by J. Nahas, UND, 2009
# Use with syn-rtl -f <rtl-script>
#####################################################

# Set the search paths to the libraries and the HDL files
# Remember that "." means your current directory

# Attributes and Parameters are set:
set_attribute hdl_search_path {./} ;# Search path for Verilog files
set_attribute lib_search_path {/afs/nd.edu/i386_linux24/opt/und/cadence/ncsu-cdk-1.5.1/lib/
UofU_Digital_v1_2} ;# Search path for library files
set_attribute library [list UofU_Digital_v1_2.lib] ;# Target Library in working directory
set_attribute information_level 6 ;# See a lot of warnings.

set myFiles [list counter.v]    ;# All your HDL files
set basename counter            ;# 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...      *
#*****************************************************
```

counter_RTL.tcl (2/2)

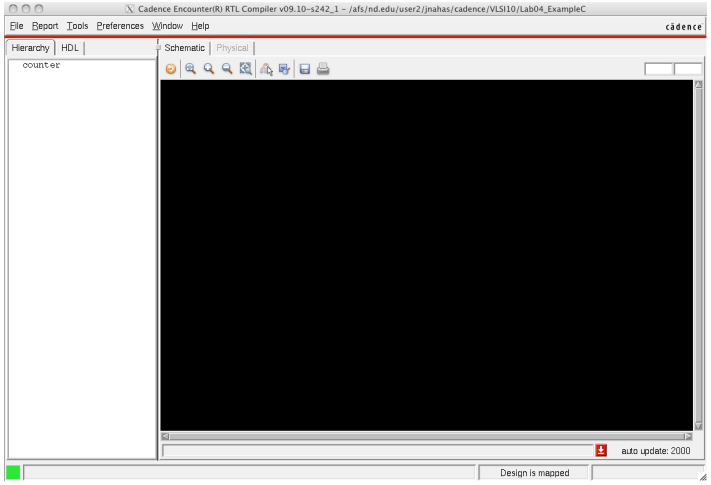
This is the working part of the script which should not need to be modified.

```
# Analyze and Elaborate the HDL files
read_hdl ${myFiles}                    # Read the files in the file list (counter.v)
elaborate ${basename}                  # start at module counter

# Apply Constraints and generate clocks
set clock [define_clock -period ${myPeriod_ps} -name ${myClk} [clock_ports]] #define clock
external_delay -input $myInDelay_ps -clock ${myClk} [find / -port ports_in/*] # set delays
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 for use by Place and Route tool
write_hdl -mapped >  ${basename}_${runname}.v          # structural file: counter_RTL.v
write_sdc >  ${basename}_${runname}.sdc               # clock information file
```


Running RTL Compiler

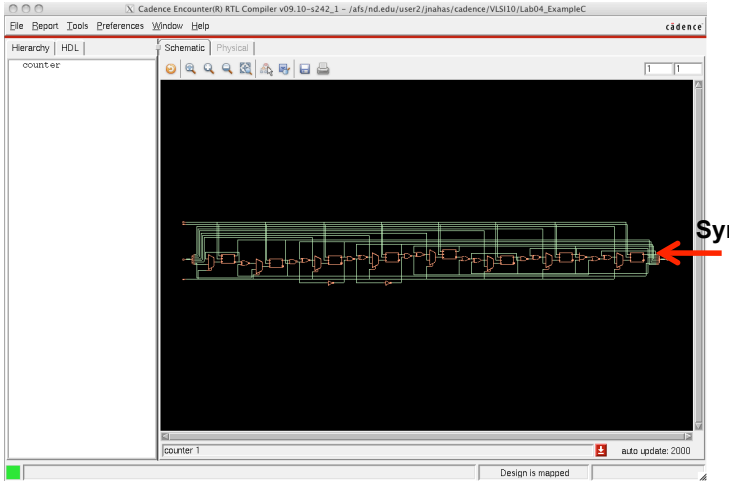
- To make sure that you have the required Cadence environment, run the Cadence environment script:
  - `source /opt/und/cadence/.cadencerc`
- start the RTL Compiler GUI with the counter_RTL.tcl script:
  - `syn-rtlg -f counter_RTL.tcl`
  - The RTL Compiler will generate a number of files and open its GUI:
    - counter_RTL_cell.rep
    - counter_RTL_power.rep
    - counter_RTL_timing.rep
    - counter_RTL.sdc
    - counter_RTL.v
    - rc.cmd
    - rc.log

Encounter RTL Window opens

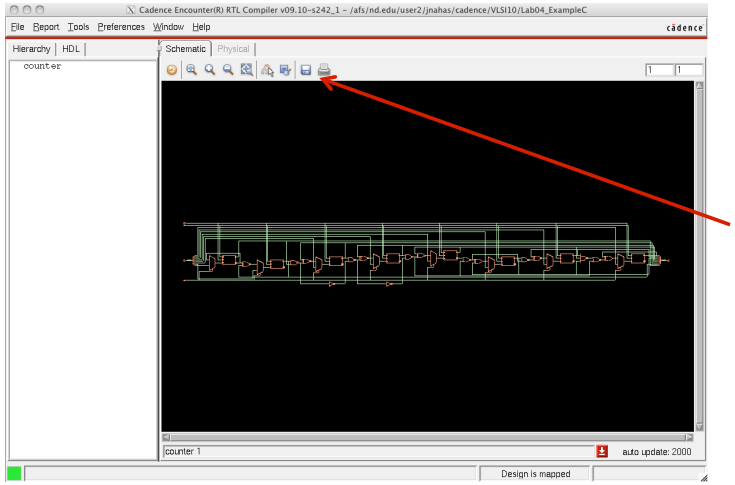


File->Update GUI to see synthesized logic

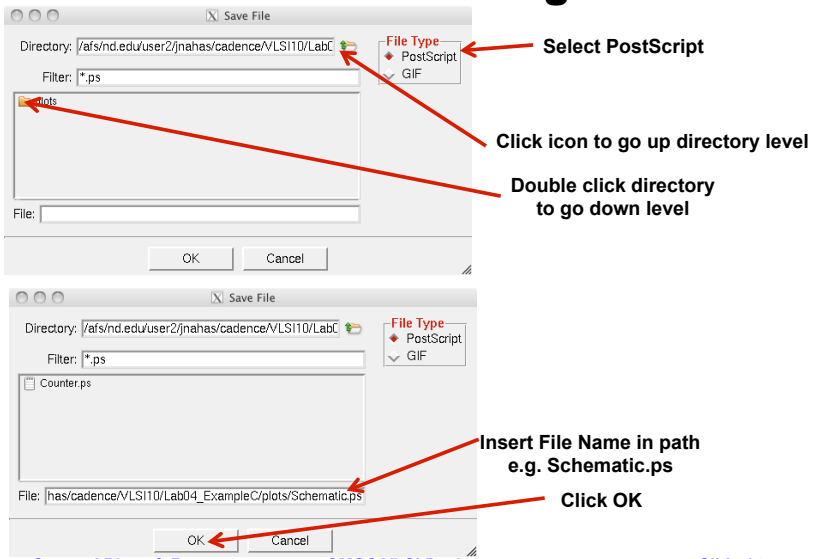
RTL Compiler GUI



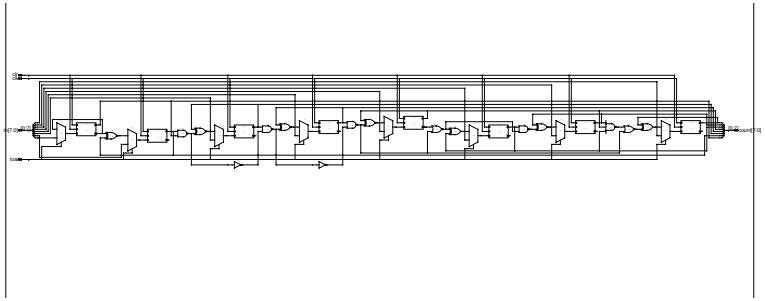
Save a Postscript of the Schematic



Save File Dialog



Schematic



Generated Files

Cell Report

Cells used in Structural Verilog

Gate	Instances	Area	Library
AND3X1	1	8.000	UofU_Digital_v1_2
DCBX1	8	144.000	UofU_Digital_v1_2
INVX2	6	18.000	UofU_Digital_v1_2
MUX2NX1	1	7.000	UofU_Digital_v1_2
MUX2X2	7	63.000	UofU_Digital_v1_2
NAND2X1	3	9.000	UofU_Digital_v1_2
NAND3X1	1	7.000	UofU_Digital_v1_2
NOR2X1	2	8.000	UofU_Digital_v1_2
NOR3X1	1	9.000	UofU_Digital_v1_2
XNOR2X1	3	24.000	UofU_Digital_v1_2
XOR2X1	4	32.000	UofU_Digital_v1_2
total	37	329.000	

Type	Instances	Area	Area %
sequential	8	144.000	43.8
inverter	6	18.000	5.5
logic	23	167.000	50.8
total	37	329.000	100.0

Timing Report

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)	
(clock clk)	launch					0	R
tmp_reg[1]/CLK				400		0	R
tmp_reg[1]/QB	DCBX1	2	88.6	496	+938	938	R
g96/A					+2	940	
g96/Y	NOR2X1	3	119.5	484	+835	1776	F
g91/B					+1	1777	
g91/Y	NAND2X1	3	147.7	472	+783	2560	R
g88/C					+2	2561	
g88/Y	NOR3X1	1	61.7	494	+624	3185	F
g81/A					+3	3188	
g81/Y	XOR2X1	1	46.4	220	+594	3783	F
g73/A					+2	3785	
g73/Y	MUX2X2	1	45.6	135	+464	4249	F
tmp_reg[7]/D	DCBX1				+2	4252	
tmp_reg[7]/CLK	setup			400	+202	4453	R
(clock clk)	capture					10000	R

Timing slack : 5547ps
Start-point : tmp_reg[1]/CLK
End-point : tmp_reg[7]/D

counter_RTL.v

Generated Structural Verilog for use in Place and Route

(1/4)

```
// Generated by Cadence Encounter(r) RTL Compiler v06.10-s032_1

module counter(clk, clr, load, in, count);
  input clk, clr, load;
  input [7:0] in;
  output [7:0] count;
  wire clk, clr, load;
  wire [7:0] in;
  wire [7:0] count;
  wire n_0, n_12, n_13, n_14, n_15, n_16, n_18, n_19;
  wire n_20, n_21, n_22, n_23, n_24, n_25, n_26, n_27;
  wire n_28, n_29, n_30, n_31, n_32, n_34, n_35, n_36;
  wire n_37, \tmp[0] , \tmp[1] , \tmp[2] , \tmp[6] ;
```


counter_RTL.v (2/4)

```
DCBX1 \tmp_reg[6] (.CLR (clr), .CLK (clk), .D (n_37), .Q (), .QB  (\tmp[6] ));
DCBX1 \tmp_reg[7] (.CLR (clr), .CLK (clk), .D (n_36), .Q (count[7]),  .QB ());
MUX2X2 g72(.A (n_34), .B (in[6]), .S (load), .Y (n_37));
DCBX1 \tmp_reg[5] (.CLR (clr), .CLK (clk), .D (n_35), .Q (count[5]), .QB ());
MUX2X2 g73(.A (n_31), .B (in[7]), .S (load), .Y (n_36))
DCBX1 \tmp_reg[4] (.CLR (clr), .CLK (clk), .D (n_32), .Q (count[4]), .QB ());
DCBX1 \tmp_reg[3] (.CLR (clr), .CLK (clk), .D (n_30), .Q (count[3]), .QB ());
MUX2X2 g74(.A (n_29), .B (in[5]), .S (load), .Y (n_35));
XNOR2X1 g80(.A (count[6]), .B (n_25), .Y (n_34));
MUX2X2 g77(.A (n_27), .B (in[4]), .S (load), .Y (n_32));
DCBX1 \tmp_reg[2] (.CLR (clr), .CLK (clk), .D (n_28), .Q (), .QB (\tmp[2] ));
DCBX1 \tmp_reg[1] (.CLR (clr), .CLK (clk), .D (n_26), .Q (), .QB (\tmp[1] ));
```

counter_RTL.v (3/4)

```
XOR2X1 g81(.A (n_22), .B (count[7]), .Y (n_31));
MUX2X2 g76(.A (n_23), .B (in[3]), .S (load), .Y (n_30));
XOR2X1 g82(.A (n_20), .B (count[5]), .Y (n_29));
MUX2X2 g79(.A (n_18), .B (in[2]), .S (load), .Y (n_28));
XOR2X1 g85(.A (n_24), .B (count[4]), .Y (n_27));
MUX2X2 g83(.A (n_15), .B (in[1]), .S (load), .Y (n_26));
DCBX1 \tmp_reg[0] (.CLR (clr), .CLK (clk), .D (n_16), .Q (), .QB(\tmp [0] ));
NAND3X1 g89(.A (count[4]), .B (count[5]), .C (n_24),(n_25));
XNOR2X1 g84(.A (count[3]), .B (n_21), .Y (n_23));
NOR3X1 g88(.A (n_12), .B (n_19), .C (n_21), .Y (n_22));
NOR2X1 g87(.A (n_19), .B (n_21), .Y (n_20));
XNOR2X1 g90(.A (n_13), .B (count[2]), .Y (n_18));
MUX2NX1 g93(.A (count[0]), .B (n_0), .S (load), .Y (n_16));
AND3X1 g92(.A (count[3]), .B (count[2]), .C (n_14), .(n_24));
```


counter_RTL.v (4/4)

```
  XOR2X1 g94(.A (count[0]), .B (count[1]), .Y (n_15));
  NAND2X1 g91(.A (count[2]), .B (n_14), .Y (n_21));
  INVX2 g95(.A (n_14), .Y (n_13));
  NAND2X1 g97(.A (count[6]), .B (count[5]), .Y (n_12));
  INVX2 g111(.A (\tmp[1] ), .Y (count[1]));
  NAND2X1 g98(.A (count[3]), .B (count[4]), .Y (n_19));
  INVX2 g102(.A (\tmp[0] ), .Y (count[0]));
  NOR2X1 g96(.A (\tmp[1] ), .B (\tmp[0] ), .Y (n_14));
  INVX2 g107(.A (in[0]), .Y (n_0));
  INVX2 g109(.A (\tmp[6] ), .Y (count[6]));
  INVX2 g106(.A (\tmp[2] ), .Y (count[2]));
endmodule
```

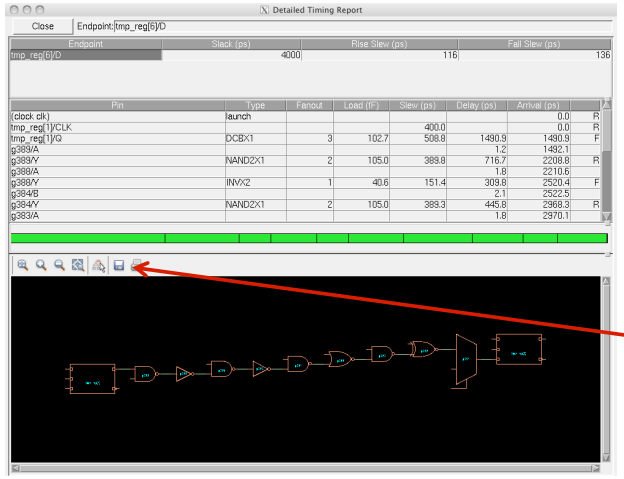

SDC Timing File counter_RTL.sdc

```
set sdc_version 1.5


# Set the current design
current_design counter

create_clock -name "clk" -add -period 10.0 -waveform {0.0 5.0} [get_ports clk]
set_clock_transition 0.4 [get_clocks clk]
set_input_delay -clock [get_clocks clk] -add_delay 0.25 [get_ports {in[0]}]
set_input_delay -clock [get_clocks clk] -add_delay 0.25 [get_ports {in[1]}]
set_input_delay -clock [get_clocks clk] -add_delay 0.25 [get_ports {in[2]}]
set_input_delay -clock [get_clocks clk] -add_delay 0.25 [get_ports {in[3]}]
set_input_delay -clock [get_clocks clk] -add_delay 0.25 [get_ports {in[4]}]
set_input_delay -clock [get_clocks clk] -add_delay 0.25 [get_ports {in[5]}]
set_input_delay -clock [get_clocks clk] -add_delay 0.25 [get_ports {in[6]}]
set_input_delay -clock [get_clocks clk] -add_delay 0.25 [get_ports {in[7]}]
set_input_delay -clock [get_clocks clk] -add_delay 0.25 [get_ports load]
set_input_delay -clock [get_clocks clk] -add_delay 0.25 [get_ports clr]
set_input_delay -clock [get_clocks clk] -add_delay 0.25 [get_ports clk]
set_output_delay -clock [get_clocks clk] -add_delay 0.25 [get_ports {count[0]}]
set_output_delay -clock [get_clocks clk] -add_delay 0.25 [get_ports {count[1]}]
```

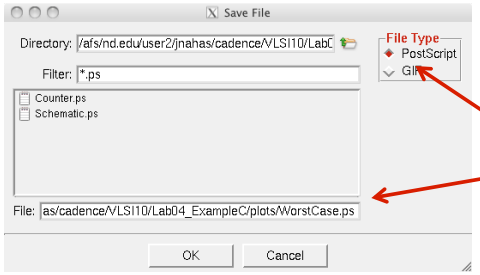
This file is used in the Place and Route

Worst Timing Path

- In the RTL Compiler GUI, select Report->Timing->Worst Path



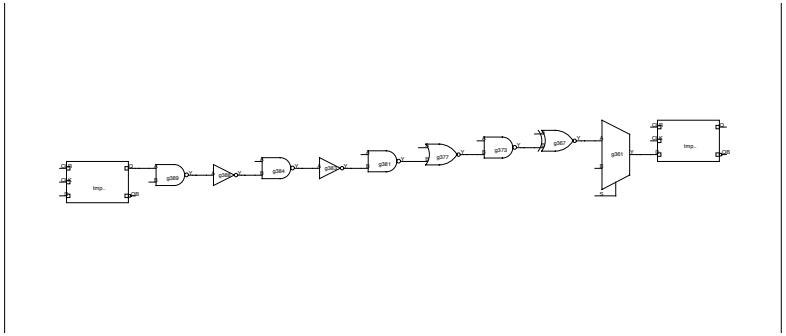
Save PostScrip Window



Make sure that you check your save location.

Save Postscript of WorstCasePath

Worst Case Path



Exit the RTL Synthesizer

- File->Exit Tool

Place and Route

Using the Cadence Encounter RTL to GDSII System

Place and Route

Place and Route is the process of:

- Selecting and placing standard logic cells in a layout based on a structural HDL (VHDL or Verilog) logic description.
- Interconnecting the cells into a circuit with metal interconnect.

Encounter Place and Route Process

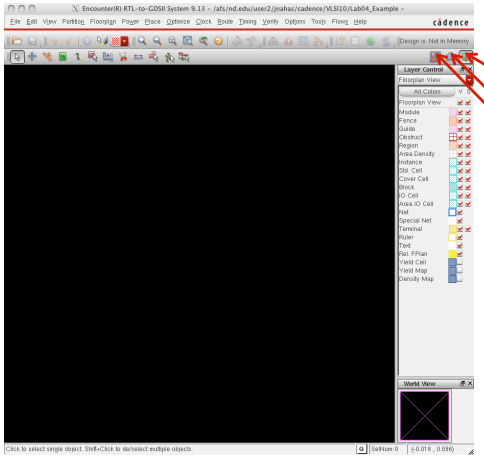
- Open the Encounter Gui
  - `cad-soc`
    - This is a command script
    - This must be in an xTerm window since X-Windows will be generated.
- Steps in Encounter
  - Import the design and design files
  - Floorplan
  - Power Planning - Add Power Rings
  - Place Cells
  - First Timing Optimization
  - Add Clock Tree
  - Second Timing Optimization
  - Final Routing

cad-soc script

```
#!/bin/tcsh
#########################################################
# Start the SOC Encounter place and route tool
# Erik Brunvand, University of Utah
#########################################################

source /opt/und/cadence/.cadencerc

# remind the user where they are...
echo "Working directory is"  $PWD

# Start encounter (pass args through)
encounter $argv
```

Note: typing encounter would do the same thing!

Encounter GUI



Importing the Design

- The Design Import window sets up the Place and Route
- The following files are imported:
  - From Synthesis:
    - counter_RTL.v
    - counter_RTL.sdc
  - Input/Output Location file:
    - counter.io
  - Library Files:
    - UofU_Digital_v1_2.lef
      - Library physical information
    - UofU_Digital_v1_2.lib
      - Library timing information

counter.io File

```
# Pin assignments for counter example
Pin: clk E
Pin: clr E
Pin: load E

Pin: in[7] S
Pin: in[6] S
Pin: in[5] S
Pin: in[4] S
Pin: in[3] S
Pin: in[2] S
Pin: in[1] S
Pin: in[0] S

Pin: count[7] N
Pin: count[6] N
Pin: count[5] N
Pin: count[4] N
Pin: count[3] N
Pin: count[2] N
Pin: count[1] N
Pin: count[0] N
```

E = east or right side

S = south or bottom

N = north or top

UofU_Digital_v1_2.lib

Over 8000 lines of code
This is the beginning of the library

```
/*
  delay model :       typ
  check model :       typ
  power model :       typ
  capacitance model : typ
  other model :       typ
*/
library(UofU_Digital_v1_2) {

  delay_model : table_lookup;
  in_place_swap_mode : match_footprint;

  /* unit attributes */
  time_unit : "1ns";
  voltage_unit : "1V";
  current_unit : "1mA";
  pulling_resistance_unit : "1kohm";
  leakage_power_unit : "1nW";
  capacitive_load_unit (1,pf);

  slew_upper_threshold_pct_rise : 80;
  slew_lower_threshold_pct_rise : 20;
  slew_upper_threshold_pct_fall : 80;
  slew_lower_threshold_pct_fall : 20;
  input_threshold_pct_rise : 30;
  input_threshold_pct_fall : 70;
  output_threshold_pct_rise : 70;
  output_threshold_pct_fall : 30;
  nom_process : 1;
  nom_voltage : 5;
  nom_temperature : 25;
  operating_conditions ( typical ) {
     process : 1;
     voltage : 5;
     temperature : 25;
```


Start of NAND2X1 in UofU_Digital_v1_2.lib

```
/* ---------------- *
 * Design : NAND2X1 *
 * ---------------- */
cell (NAND2X1) {
  cell_footprint : nand2;
  area : 3;
  cell_leakage_power : 0.0567011;
  pin(A)  {
    direction : input;
    capacitance : 0.0201485;
    rise_capacitance : 0.0201485;
    fall_capacitance : 0.019937;
    rise_capacitance_range ( 0.0190239, 0.0212732) ;
    fall_capacitance_range ( 0.0186932, 0.0211807) ;
    internal_power() {
      rise_power(passive_energy_template_5x1) {
        index_1 ("0.06, 0.18, 0.42, 0.6, 1.2");
        values ("0.23538, 0.234957, 0.234854, 0.234994, 0.235103");
      }
      fall_power(passive_energy_template_5x1) {
        index_1 ("0.06, 0.18, 0.42, 0.6, 1.2");
        values ("0.242091, 0.236781, 0.235478, 0.235391, 0.235861");
```

UofU_Digital_v1_2.lef

5700 lines of code

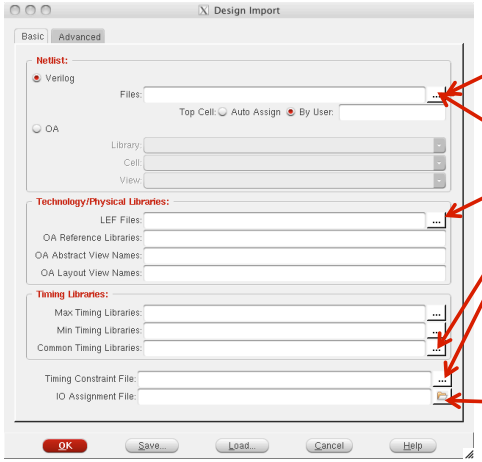
```
##########################
#
# This is the TechHeader.lef file that contains the
# technology information for the AMI C5N 0.5 micron
# CMOS technology (SCN3M_SUBM when using MOSIS)
#
# Erik Brunvand
##########################

VERSION 5.5 ;
NAMESCASESENSITIVE ON ;
BUSBITCHARS "[]" ;
DIVIDERCHAR "/" ;
UNITS
  DATABASE MICRONS 100 ;
END UNITS

MANUFACTURINGGRID 0.15 ;

LAYER poly
  TYPE    MASTERSLICE ;
END poly

LAYER cc
  TYPE    CUT ;
  SPACING 0.9 ;
END cc
```


Start of NAND2X2 in UofU_Digital_v1_2.lef

```
MACRO NAND2X2
    CLASS CORE ;
    FOREIGN NAND2X2 0 0 ;
    ORIGIN 0.00 0.00 ;
    SIZE 12.00 BY 27.00 ;
    SYMMETRY X Y ;
    SITE core ;
    PIN A
        DIRECTION INPUT ;
        PORT
        LAYER via ;
        RECT  0.90 10.20 1.50 10.80 ;
        LAYER metal2 ;
        RECT  0.60 9.90 1.80 14.10 ;
        LAYER metal1 ;
        RECT  0.60 9.90 1.80 11.10 ;
        END
    END A
```


Importing the Design

- File->Import Design…



Select Structural Verilog File on next 2 slides

Browser Button Click to select file (see next 4 slides)

Browser Button Click to select file (In later slide)

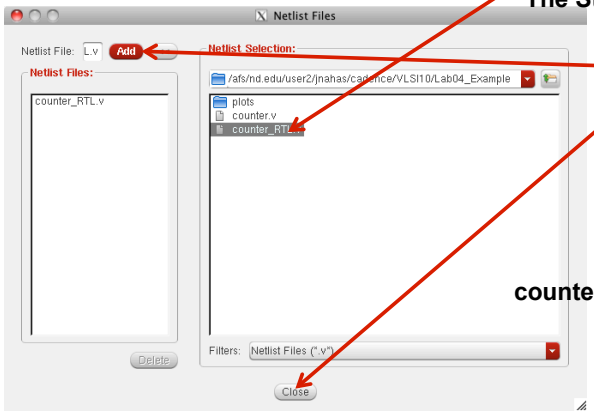
Adding Structural Verilog File

- Selecting Files

Click to Expand selection box

Adding Structural Verilog File

- Selecting Files



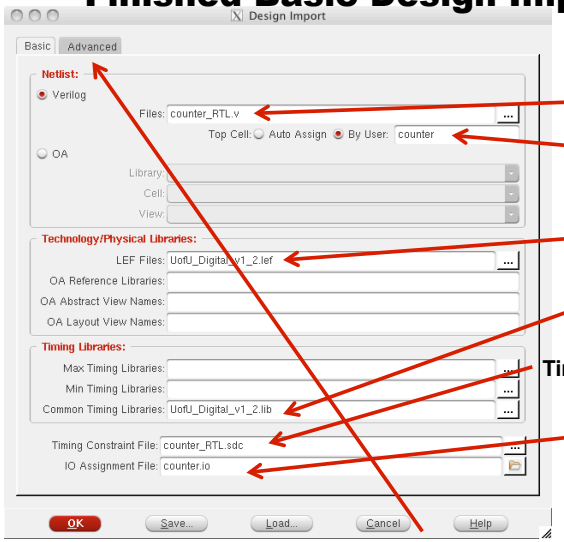
Select counter_RTL.v
The Structural Verilog File

Click Add

Click Close

Do not select
counter.v the behavioral file

Finished Basic Design Import Window



Structural Verilog File

Fill in top cell name

Select abstract file (xx.lef)

Leave Max and Min blank
Select library file (xx.lib)

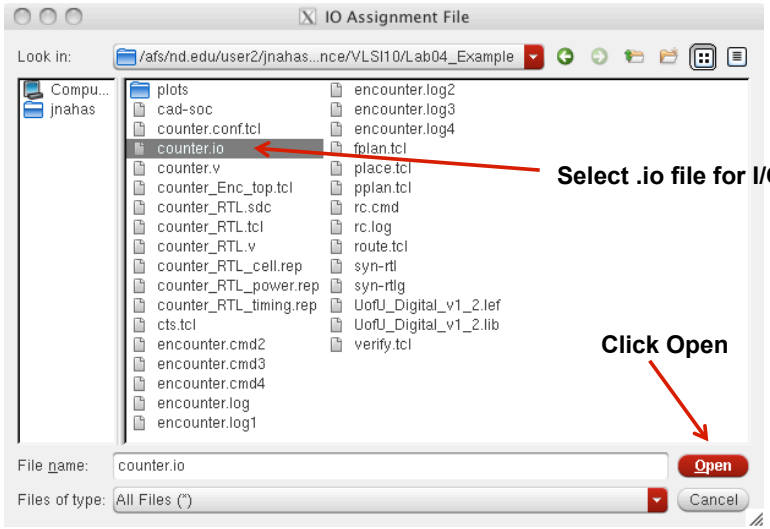
Timing Constraint File (xx.sdc)

I/O File (xx.io)
see next page

We will use the
UofU_Digital_v1_2
.lib and .lef files

When done, Click Advanced Tab

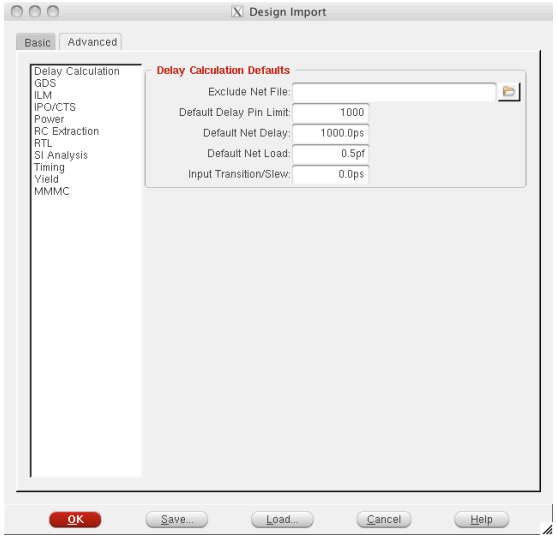
I/O File Addition Window



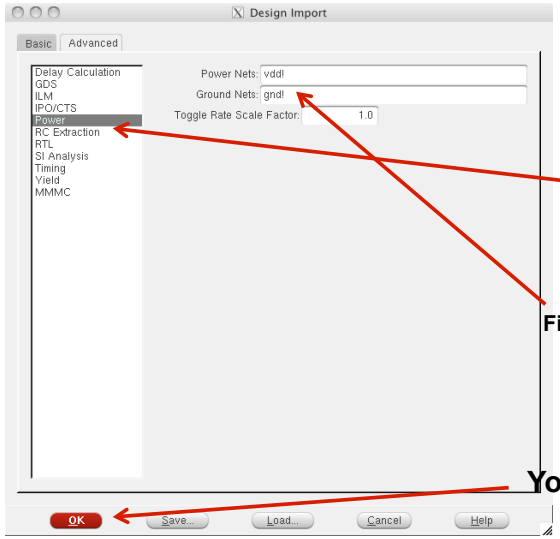
Select .io file for I/O

Click Open

Design Import Advanced Tab



Design Import Advanced Power page

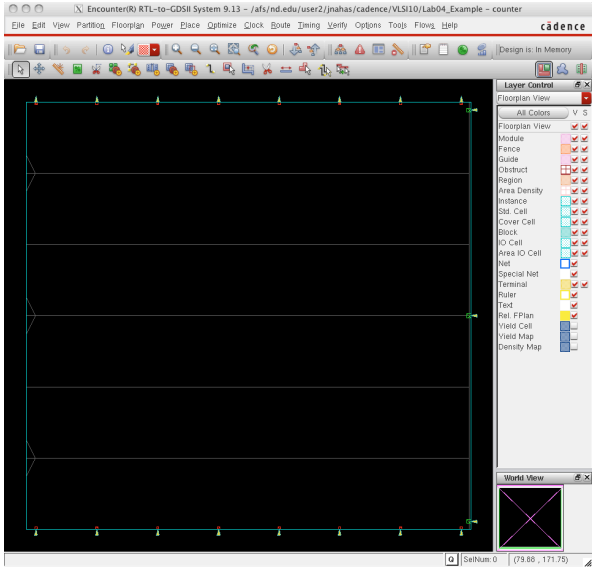


Select Power

Fill in Power and Ground Nets
vdd! and gnd!

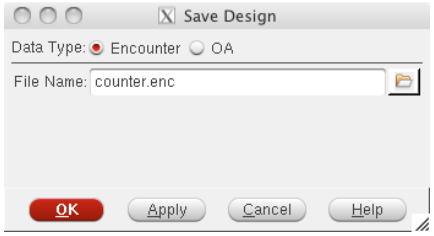
You can now click OK

Encounter window with floorplan



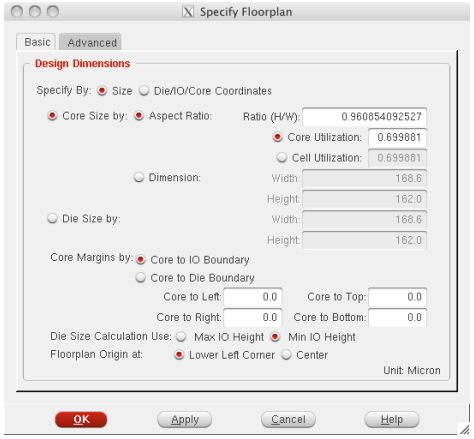
Save the Design

- Design->Save Design …
- Do this after each step!!!!

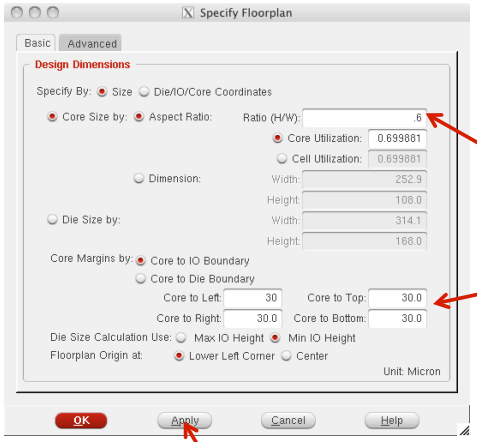


Floorplanning

- Floorplan –> Specify Floorplan …



Floorplan with margins added

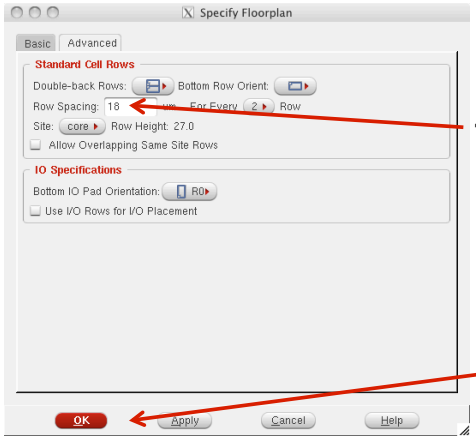


Aspect Ratio Can be adjusted if desired

30 micron margins added
Left
Right
Top
Bottom

Click Apply to see how floorplan changes affect layout on main screen

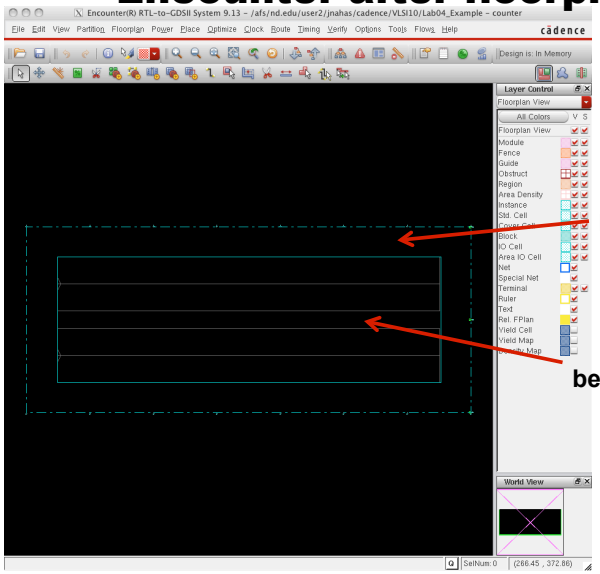
Floorplan Advanced Tab with spacing added



18 micron spacing added

Click OK when done

Encounter after floorplanning



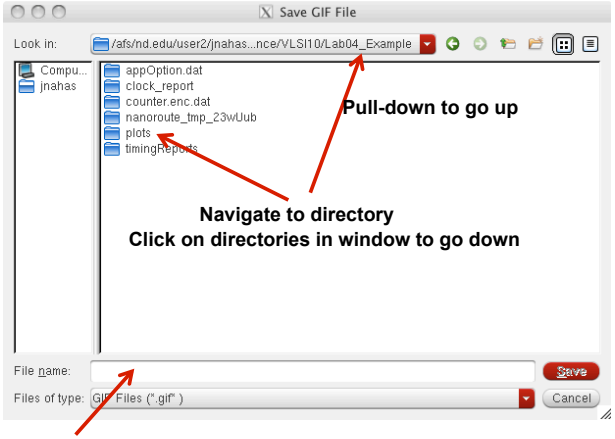
Note Margin

Note space between pairs of rows

Reminder
Save the design

Save plot of Floorplan

- Tools->Screen Capture->Write to GIF File…

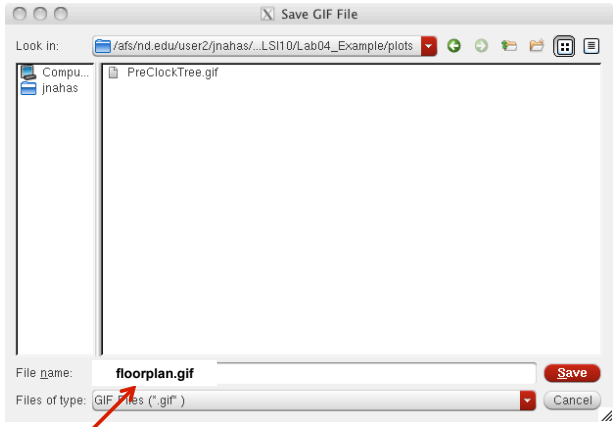


Pull-down to go up

Navigate to directory
Click on directories in window to go down

File Name

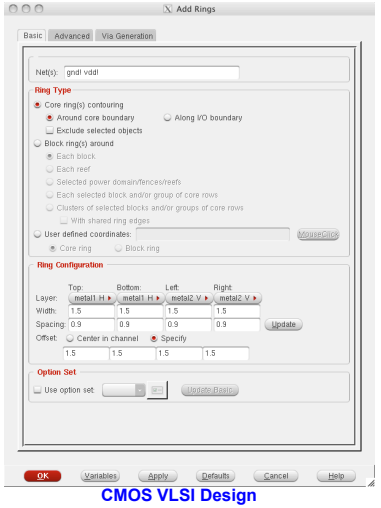
Save plot of Floorplan



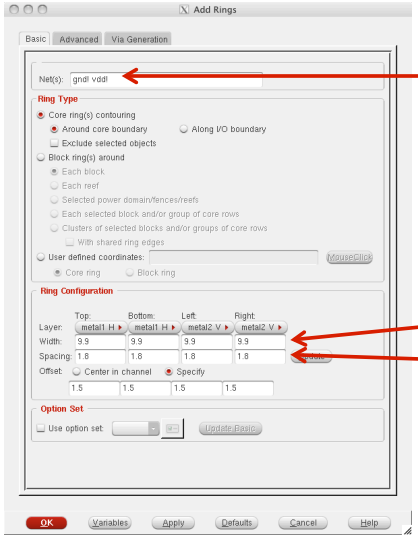
File Name

Power Planning

- Power->Power Planning->Add Rings…



Add Power Rings

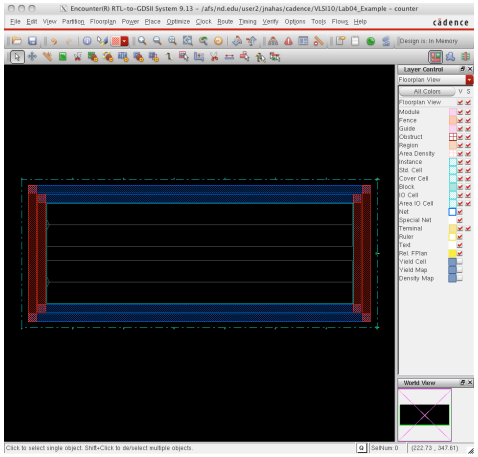


gnd! vdd!

Ring widths to 9.9
Ring spacing to 1.8

Click OK

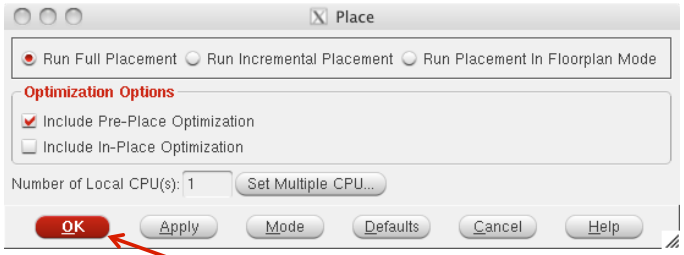
Floorplan with Power rings



Save a plot called PowerRings.gif
Tools->Screen Capture->Write to GIF File…

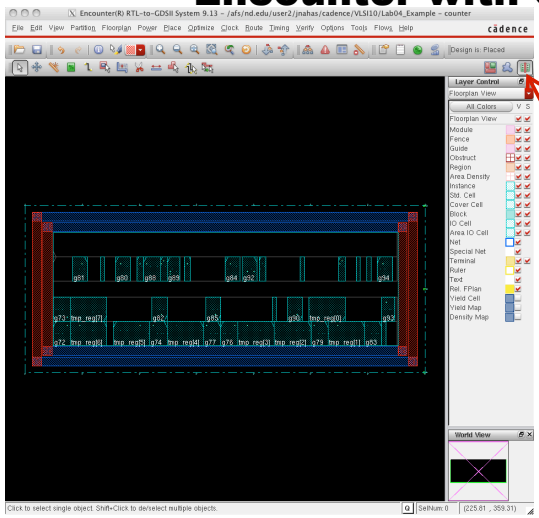
Place Standard Cells

- Place->Place Standard Cells…



Leave Defaults – just click OK

Encounter with Cells



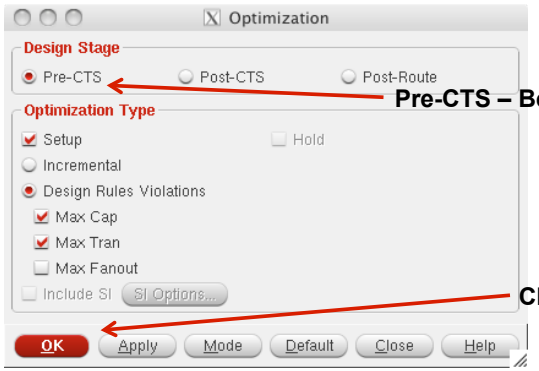
Click Physical View to see cells

Save the Design!

Save a plot called Cells.gif
Tools->Screen Capture->Write to GIF File…

First Timing Optimization

- Optimize->Optimize Design….



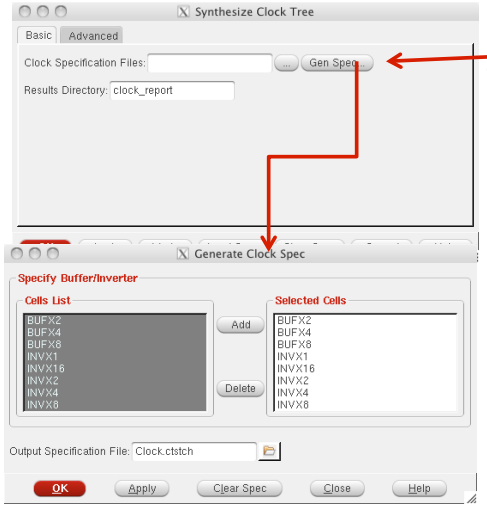
Pre-CTS – Before Clock Tree Synthesis

Click OK

Note: You might get an error message
Ignore for now.

Clock Tree Synthesis

- Clock->Synthesize…

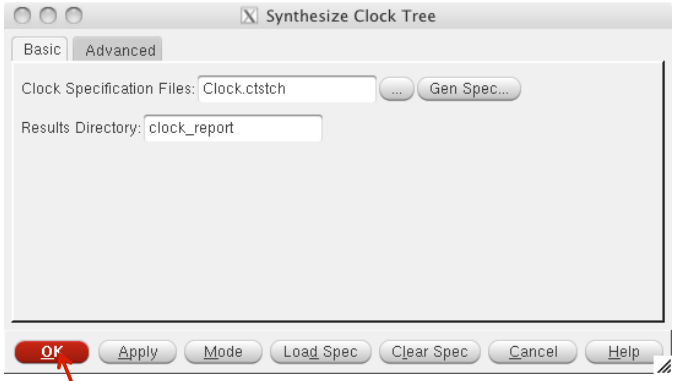


Select Gen Spec to open Generate Clock Spec window

Select BUF and INV cells
Click ADD
Click OK

Cells are added to Selected Cells List
Clock.ctstch file created

Synthesize Clock Tree



Click OK

You might get diamonds across your layout!!

Ignore for now.

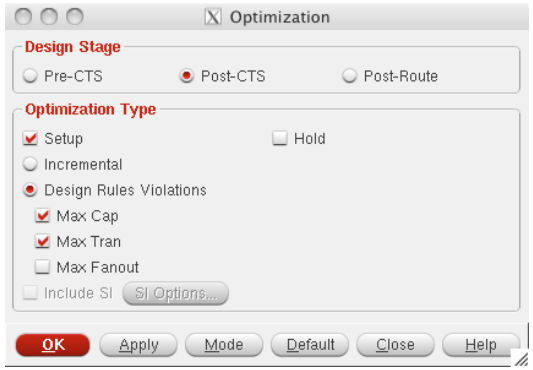
See the Clock Tree

- Clock->Display->Display Clock Tree
- Turn off with
  - Clock->Display->Clear Clock Tree Display

Save a plot called ClockTree.gif
Tools->Screen Capture->Write to GIF File…

Post CTS Optimization

- Optimize->Optimize Design…
  - Select Post CTS
  - Click OK

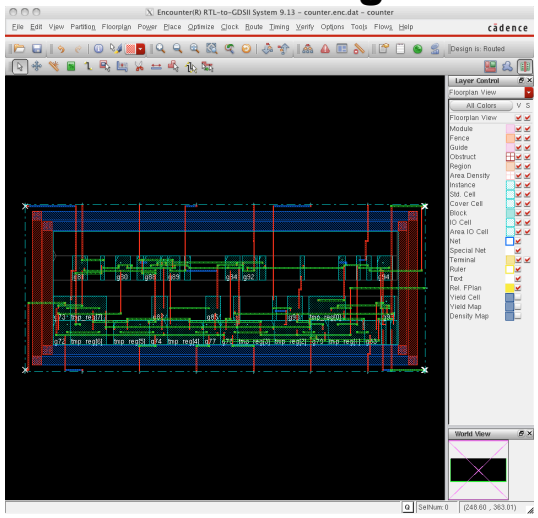


Final Routing

- Route->NanoRoute->Route

Click OK

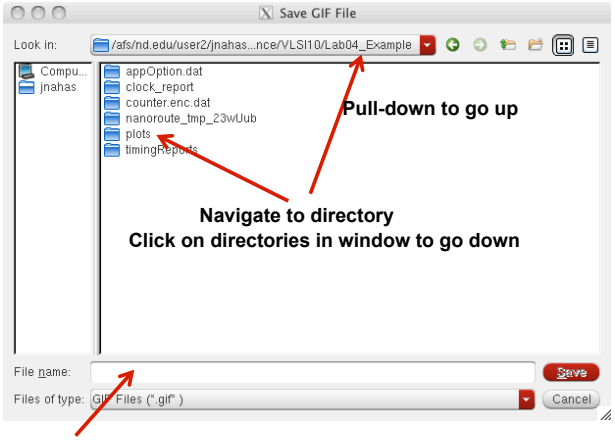
Routed Design



Save your Design

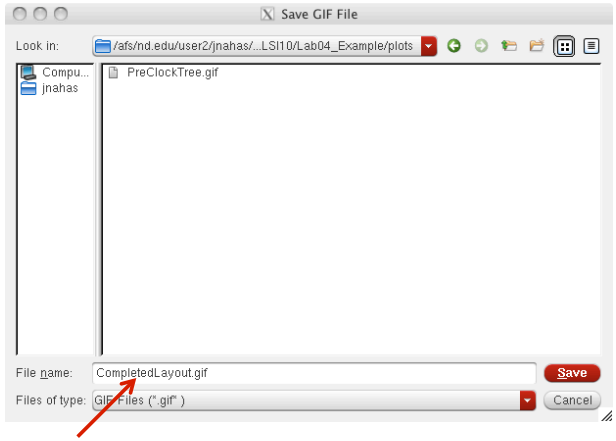
Save plot of layout

- Tools->Screen Capture->Write to GIF File…



File Name

Save plot of layout



File Name

Documentation

- Create a folder in the drop box named Lab04
- In the folder place the following:
  - counter.v
  - counter_RTL.v
  - Schematic.ps
  - WorstCasePath.ps
  - floorplan.gif
  - PowerRings.gif
  - Cells.gif
  - ClockTree.gif
  - CompletedLayout.gif