

### SAN JOSE STATE UNIVERSITY Electrical Engineering Department

Using IIT's Standard Cell Library to Synthesize Top Down designs authored with Verilog

#### IC DESIGN GROUP SAN JOSE STATE UNIVERSITY

# A supplemental guide for using CDS tools for IC design

David W. Parent Assistant Professor Electrical Engineering, SJSU One Washington Square San Jose, CA 95192-0084 Phone 408.924.3963 • Fax 408.924.2925

# Table of Contents

#### 

Background:	4
Background:	5
Getting ready for Synthesis:	7
Setting up your DFII Data Base:	
Place and Route:	
Converting from SEULTRA fomrat to DFII (icfb)	15
Checking the timing of the circuit using Spice:	
Running a Spice Simulation:	

#### LIST OF FIGURES:

Figure 2: Files to copy if you are using the TSMC025 process       6         Figure 3: Files to copy if you are using the TSMC018 process       6         Figure 4: Copying the verilog file into the proper directory.       6
Figure 3: Files to copy if you are using the TSMC018 process
Figure 4: Conving the verilog file into the proper directory 6
Figure 5: Code for lfsr
Figure 6: Getting the correct .cshrc file (Remember to log out and log back in.)
Figure 7: opening compile.scr
Figure 8: The compile.scr file before editing
Figure 9: The compile.scr file after editing
Figure 10: Running Synthesizer
Figure 11: Output from synthesis
Figure 12: Creating a DFII library
Figure 13: Library Manager after DFII data base is created13
Figure 14: Editing the seultra.scr file
Figure 15: Sourcing the evn.opts file
Figure 16: Running the place and rout tool
Figure 17: Place and route finished
Figure 18: Conversion Script
Figure 19: Output of Conversion Process
Figure 20: Ifsr converted
Figure 21: Final Layout of Ifsr
Figure 22: Creating another library to run extraction on the layout of the lfsr
Figure 23: Copying the lfsr layout view
Figure 24: Layout view of lfsr in lfsr_test library
Figure 25:DRC Check
Figure 26: Acceptable errors from DRC
Figure 27: Creating Pins
Figure 28: Extracting with parasitic Capacitances
Figure 29: Starting Äffirma
Figure 30: Seting up the stimulus file
Figure 31: Setting Stimulus File
Figure 32: Setting the Clock
Figure 33: Setting Enable to logic 1 (5V)
Figure 34: Setting gnd to zero volts
Figure 35: Changing reset
Figure 36: Setting VDD to 5 Volts
Figure 37: Selecting Outputs to be Plotted
Figure 38: Final Simulation

#### Acknowledgements:

This tutorial is based on the NCSU design kit. For more information, see <u>http://www.ece.ncsu.edu/cadence/CDK.html</u>. This tutorial also follows the design flow used by WPI at <u>http://vlsi.wpi.edu/cadence/</u>.

This would not be possible without the excellent library and documentation developed by James Stine and his Graduate Student *Johannes Grad* at the Illinois institute of Technology (<u>http://www.ece.iit.edu/~cad/cadence/seultra/</u>).

# Chapter

# Standard Cell Design:

#### Background:

The standard cell design methodology is a top down approach to digital design. It is also called the ASIC approach and is used in EE287. Any time you start with a HDL description of the logic and work your way down you are following this kind of approach. The advantages of this approach are that you can describe logic in words without committing to a specific circuit. for instance A <= B+C, would define and adder, but you have not committed to a ripple carry of carry look ahead architecture. You can optimize your adding at a higher level without have to do a n adder circuit by hand. This approach along with the use of standard cell library which has an estimate of the delay and power for various circuits that are already laid out, can be used to automatically generate layouts from a Verilog or vhdl net-list.

The design flow for this approach is:

- 1. Describe the functionality of your circuit in an HDL (In this case Verilog)
  - a. Keep re-writing code and simulating until the logic meets specification (No timing information is created at this time.)
- 2. Synthesize your HDL description of your circuit to a library that has timing (delay) information.
  - a. Optimize the performance of your design until you meet your logical, timing, and power specification.
- 3. Place and route your design with area and topology information from a physical library.
- 4. Create a physical description of your circuit in "GDSII" format
- 5. Import this physical description into a DFII database (The view your are familiar with if you have done the bottom up cadence tutorial. For example: the layout view.)
- 6. Run a DRC check on the layout. (draw a do not DRClayer over the pins)
- 7. Create a Spice netlist from the layout view by extracting the layout.

- 8. Create a symbol for the design
- 9. Create a test bench for this design, use the symbol to set up the spice test bench
- 10. Verify the logic, timing, and power specification for the corners (If you have them, if not use the nominal values)
- 11. Send the design out
- 12. Verify the circuit when it comes back from manufacturing.

It is very important to remember that each step in the design flow you are estimating the performance of your circuit. The accuracy of the simulation increases as you go down the design flow, but so does the simulation time. Furthermore, it takes more time to fix an error the farther you are along the design flow. As a rule of thumb it is better to give your design a 30% margin of error at the beginning of the design, so you do not have implement costly fixes at the end of the design flow or god forbid after the circuit has been manufactured. Currently the design cycles in industry are so short that you do not have the luxury of fixing a bug in a later release.

For more on various design flows see chapter 1 of the new Kang and Leblebici book.

#### Getting Started:

This tutorial will take you through the basics of taking a design through synthesis and place and route. At this time, it will not cover how to actually optimize a circuit in the higher level tools. It assumes that you know how to optimize a Verilog description of a circuit from a Top-Down design class.

Choose which technology you want to use. The available technologies are:

- AMI06
- TSMC025
- TSMC018

Copy the start up files into your account, depending on which technology you selected (Figure 1, Figure 2 and Figure 3.). In this case, we will be doing a 10 bit lfsr. No matter what type of circuit you are doing you need these files.

🗐 eecad28 - default - SSH Secure Shell		
File Edit View Window Help		
D 🛩 🖬 🚨 🎉 🖿 🖻 🖨 🖉 💭 🦠 🙌		
eecad28.engr.sjsu.edu% eecad28.engr.sjsu.edu% eecad28.engr.sjsu.edu% eecad28.engr.sjsu.edu% cp -rf /apps/cadence/iit eecad28.engr.sjsu.edu%	*/ami06/main lfsr_ami06	
Connected to eecad28	SSH2 - 3des-cbc - hmac-md5 - none 92x5	NUM //

Figure 1: Files to copy if you are using the AMI06 process.

🕮 eecad28 - default - SSH Secure Shell			
File Edit View Window Help			
] D 🛩 🖬   <i>9</i> 💋 陷 6: 6   <i>1</i> 🍎 🛸 🛠			
eecad28.engr.sjsu.edu% eecad28.engr.sjsu.edu% eecad28.engr.sjsu.edu% cd eecad28.engr.sjsu.edu% cp -rf /apps/cadence/iit* eecad28.engr.sjsu.edu%	/tsmc025/main lfsr_tsmc025		•
Connected to eecad28	SSH2 - 3des-cbc - hmac-md5 - none	92x5	NUM //

Figure 2: Files to copy if you are using the TSMC025 process

🚈 eecad28 - default - SSH Secure Shell		
File Edit View Window Help		
] D 🛩 🖬   🧟 🎉 陷 🛍 🍘 💭 🦓 🙌		
eecad28.engr.sjsu.edu% eecad28.engr.sjsu.edu% eecad28.engr.sjsu.edu% eecad28.engr.sjsu.edu% cp -rf /apps/cadence/iit7 eecad28.engr.sjsu.edu%	*/tsmc018/main lfsr_tsmc018	×
Connected to eecad28	SSH2 - 3des-cbc - hmac-md5 - none 92x5	

Figure 3: Files to copy if you are using the TSMC018 process

We will be following the AMI06 process for the rest of the tutorial.

Now cd into the directory you just created and copy the actual verilog file into it (Figure 4).

🗐 eecad28 - default - SSH Secure Shell		
File Edit View Window Help		
D 🛩 🖬 🧾 🎉 🍋 🛍 🖨 🏚 🎒 🦓 🙌		
eecad28.engr.sjsu.edu% eecad28.engr.sjsu.edu% eecad28.engr.sjsu.edu% cd lfsr_ami06 eecad28.engr.sjsu.edu% cp /apps/cadence/iit*/lfs eecad28.engr.sjsu.edu%	sr.v .	•
Connected to eecad28	SSH2 - 3des-cbc - hmac-md5 - none 92x5	NUM //

Figure 4: Copying the verilog file into the proper directory.

Use a text editor to make sure the code matches the code in Figure 5.

Getting ready for Synthesis:

```
🗐 eecad28 - default - SSH Secure Shell
File Edit View Window Help
 🗅 📽 🔚 🔎 🎉 🖻 🖻 🖷 🧾 🍎 🦓 |
                             N?
eecad28.engr.sjsu.edu% cd lfsr ami06
eecad28.engr.sjsu.edu% cp /apps/cadence/iit*/lfsr.v .
eecad28.engr.sjsu.edu% nedit lfsr.v &
[2] 5508
eecad28.engr.sjsu.edu%
Connected to eecad28
                                  SSH2 - 3des-cbc - hmac-md5 - noi
Ifsr.v (read only)
 File Edit Search Preferences
                         Shell Macro Windows
// 10-bit Linear Feedback Shift-Register
//
// Johannes Grad, IIT
module lfsr(q, e, reset, clk);
                            // Global clock
input clk;
                          // Counter enable
// Counter reset
input
         e;
input
        reset;
                           // Counter output
output [9:0] q;
req [9:0] q;
always @(posedge clk) begin
  if (reset)
      q <= 1;
  else if(e)
      q \le \{ q[8:0], \sim (q[9] \land q[6]) \};
end
endmodule
```

Figure 5: Code for lfsr.

Now we have to make sure you are running the correct version of Synopsis tools.

If your account was created before the 19'th of September 2003, copy a new .cshrc file into your home directory, and then log out and log back in(Figure 6).

🗐 eecad28 - default - SSH Secure Shell	
File Edit View Window Help	
🗅 🛩 🖬   🤐 🎉 🖻 🖻 🖨 💋 📁 🧠 🙌	
eecad28.engr.sjsu.edu% eecad28.engr.sjsu.edu% eecad28.engr.sjsu.edu% cd eecad28.engr.sjsu.edu% cp /apps/cadence/iit*/.cshrc . eecad28.engr.sjsu.edu%	

Figure 6: Getting the correct .cshrc file (Remember to log out and log back in.)

Change back into you lfsr\_ami06 directory.

Before we synthesyze the lfsr we have to modify the script file that the Synopsis tool uses to compile your design into a net list.

Open the text editor and open the file compile.scr

```
eecad28.engr.sjsu.edu% pwd
/home/dparent/lfsr_ami06
eecad28.engr.sjsu.edu% nedit compile.scr &
[2] 5522
eecad28.engr.sjsu.edu%
```

Figure 7: opening compile.scr.

The script file is slef explanatory. You have have to search and replace TOP\_LEVEL\_NAME and FILE\_NAME with lfsr.

```
/* Compile Script for Synopsys
                                              *
                                              ×
/* Johannes Grad, IIT
/*
                                              ×
/*
                                              *
/* Change "TOP_LEVEL NAME" to your design name
                                              ×
/* Change "FILE_NAME" to your verilog sourc
                                              ×
/*
            code file. If you have more than
                                              ×
/*
            one, you need one line for each
                                              ×
/*
                                              *
            Do not include the testbench
link library=target library={iit06 stdcells pads.db
define design lib WORK -path .
read -f verilog FILE NAME.v
set flatten true
verilogout show unconnected pins = "true";
max area 0.0
current design TOP LEVEL NAME
compile -ungroup all -map effort medium
set max fanout 8.0 TOP LEVEL NAME
compile -incremental mapping -map effort medium
check design
/* If you want pads, uncomment these 2 lines and */
/* list all your pins/busses that you want have as
/* set port is pad {clk,load,Cin,Z,A,B} */
/* insert pads */
write -f verilog -output TOP LEVEL NAME.vh
write -hier -output TOP LEVEL NAME.db
```

Figure 8: The compile.scr file before editing.

/\* Compile Script for Synopsys \*/ /\* Johannes Grad, IIT \*/ /\* \*/ /\* \*/ /\* Change "lfsr" to your design name \*/ /\* Change "lfsr" to your verilog sourc \*/ /\* code file. If you have more than one, you need one line for each \*/ /\* \*/ /\* \*/ Do not include the testbench link library=target library={iit06 stdcells pads.db} define design lib WORK -path . read —f verilog lfsr.v set flatten true verilogout show unconnected pins = "true"; max area 0.0 current design lfsr compile -ungroup all -map effort medium set max fanout 8.0 lfsr compile -incremental mapping -map effort medium check design /\* If you want pads, uncomment these 2 lines and \*/ /\* list all your pins/busses that you want have as pads ' /\* set port is pad {clk,load,Cin,Z,A,B} \*/ /\* insert pads \*/ write -f verilog -output lfsr.vh write -hier -output lfsr.db

Figure 9: The compile.scr file after editing.

Now we actually run the synthesizer from the command line in Synopsys(Figure 10).



Figure 10: Running Synthesizer.

You should see output like Figure 11.

At this point you should check the timing of the circuit and eliminate any long paths that make the circuit fail specification. You should be within 30% of your spec at this time to allow for routing induced delays.

```
pedimind heral obcimization suase
-----
```

ELAPSED TIME	AREA	WORST NEG SLACK		DESIGN RULE COST
		esign Rule		max_fanout)
ELAPSED TIME	AREA	WORST NEG SLACK	SLACK	DESIGN RULE COST
0:00:01	15192.0		0.0	2.0
		very Phase	(max_are	a 0)
ELAPSED TIME	AREA	WORST NEG SLACK	TOTAL NEG SLACK	RULE COST
	ign is 'lf n mt pads, p your pins _is_pad {c	'lfsr' to sr'. uncomment : /busses tha	these 2 lin at you wan	nes and */ t have as p
write -f ver Information: 1 write -hier Writing to 1 1	Please m	ake sure tl fsr.db	nat you ha	
report_timin l report_cell l report_power l quit l dc_shell> Thank you	> cell. c > power	rep .rep		

Figure 11: Output from synthesis.

#### Setting up your DFII Data Base:

Ultimately we are going to load in the gds2 file from the place and route tool into icfb. WE must create a library with icfb before we run the seultra tool script. The reason is that the se\_shell script creates a cds.lib file that will not allow you to view the technology file information.

To create this directory: We have to start icfb and create a directory with the proper techfile and same name of the verilog file (**Error! Reference source not found.**).

OK Cancel Apply
Library
Name:
Path: I
Technology Library
If this library will not contain physical design (i.e., layout) data you do not need a tech library. Otherwise, you must either attach to an existing tech library or compile one. Choose option:
◇ No tech library needed
Attach to existing tech library     AMI 0.60u C5N (3M, 2P, high-res)     AMI 0.60u C5N (3M, 2P, high-res)     □
Misc.

Figure 12: Creating a DFII library.

If it is done correctly, you should see the library manager look like Figure 13.

File	Edit	View	<u>D</u> esign	Mana	iger
🔲 Sh	ow Cat	tegories	]	Show	Files
— Libr	ary —				
1					
ANALO					
		g_Parts			
		al_Part	:s		
NCSU	Sheet	s_8ths			
NCSU	TechL:	ib_amiC	16		
NCSU	TechL:	ib_tsmc	:03		
PADFF	<b>AME</b>				
SEAG					
SJSU	AMI06	_IP			
basio	5				
cdsDe	efTechl	Lib			
lfsr					
synth	n_iit0	6			

Figure 13: Library Manager after DFII data base is created.

#### Place and Route:

Now we need to run the place and route tool from Cadence called seultra. Before we do this, we have to edit the script file seultra.scr according to Figure 14.

Figure 14: Editing the seultra.scr file.

It is simple; just follow the instructions of the script file.

Now we have to run a set up file that tells the shell where the binaries are for the Cadence Place and rout tool Figure 16).

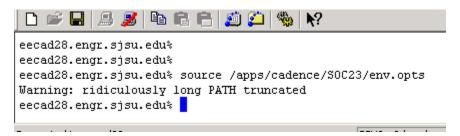


Figure 15: Sourcing the evn.opts file.

Now run the place and route tool from the command line (Figure 16). This will take a long time even for this simple design.

D 🛩 🖬 🔎 🎉 🗈 🖻 🍘 💭 🦓 🙌
dc_shell> Thank you eecad28.engr.sjsu.edu% nedit seultra.scr & [2] 5526 eecad28.engr.sjsu.edu% se_shell -f seultra.scr

Figure 16: Running the place and rout tool.

When it is finished, your terminal should look like Figure 17

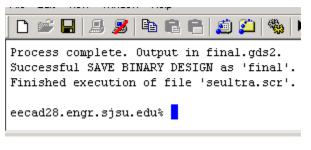


Figure 17: Place and route finished.

#### Converting from SEULTRA fomrat to DFII (icfb)

The format of the project is not in the correct DFII database format to convert it we run a special script developed by IIT. To run the script type in the command iitcells\_se2icfb, as in Figure 18.

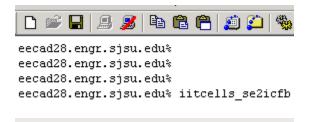


Figure 18: Conversion Script.

You should get a listing as in Figure 19.

Checking if final.gds2 exists.....OK Determining top-level name.....OK (lfsr) Creating temporary cds.lib.....OK Determining Technology.....OK (AMI 0.5um) Removing old library.....OK (lfsr) Creating new DFII library.....OK (lfsr) Creating PIPO script file.....OK Running PIPO (GDS Stream-In)..... \* \* \* CADENCE Design Systems, Inc. \* \* Virtuoso(R) Physical Data Translator 4.4.6 \* EXEC TIME : 22-Sep-2003 20:10:36 \* \* \* \* 

\*\*\* There were 0 error and 1 warning messages \*\*\*

Statistic and more information, please check /home/dparent/lfsr\_ami06/PIPO.LOG fi le.

NORMAL EXIT ...

\* Warning \* Years in non-standard 4 digit format found in the stream file. Such y ears have been converted to standard format. The tool that wrote the stream file should write the years in standard format, which is the number of years since 190 0. For more information, please see the documentation/KPNS.

Cleaning up.....0K Good by. eecad28.engr.sjsu.edu%

Figure 19: Output of Conversion Process.

You can now see the layout view of the lfsr in icfb's layout tool (Figure 20).

🛛 Show Categories 👘 🔲	Show Files		
Library	Cell	View	
lfsr	lfsr	layout	
ANALOG NCSU_Analog_Parts NCSU_Digital_Parts NCSU_Sheets_8ths NCSU_TechLib_amiO6 NCSU_TechLib_tsmcO3 PADFRAME SEAG SJSU_AMIO6_IP basic cdsDefTechLib lfsr synth_iitO6	NOR3X1 OAI21X1 OAI22X1 OR2X1 OR2X2 PADFC PADGND PADINC PADINC PADINC PADOUT PADVDD TBUFX1 TBUFX2 XNOR2X1 XOR2X1 Lfsr Lfsr_VIA1 Lfsr_VIA2	△ layout	
Messages			
Loading NCSU Library Ma	nager customizations	. done .	

Figure 20: lfsr converted.

We can see the final layout in Figure 21.

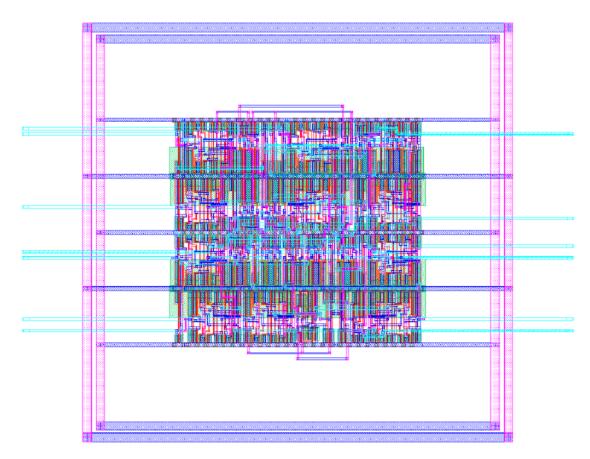


Figure 21: Final Layout of lfsr.

#### Checking the timing of the circuit using Spice:

I am not quite sure why but the extraction from the library you create your design in (in this case lfsr) does not work. You have to create another library like lfsr\_test and copy the layout into it, and then run extract. You can do this from the icfb session you started before you sourced the env.opts file to het the paths for the place and route tool to work. If you closed that icfb session, then start another terminal and then start icfb again.

Create a library according to Figure 22.

XCreat	e Library	,				×
ок	Cancel	Apply				Help
Library						
Name	ilfs:	_test				
Path:						
Technol	logy Libr	ary				
Other		u must	• •	· ·	e., layout) data you do not need a tech library. g tech library or compile one.	
	🔷 No te	ch libra	ry needed			
	<ul> <li>♦ Comp</li> </ul>			>	AMI 0.60u C5N (3M, 2P, high-res)	-
Misc.						
	I/O Pa	d Type:	🔶 Perimeter 🗸	> Area a	array	L V

Figure 22: Creating another library to run extraction on the layout of the lfsr.

Copy the lfsr, layout view into the library lfsr\_test according to Figure 23. (If you get any warnings, just click on fix errors and then proceed.)

🗙 Copy Vier	N	×
- From -		
Library	<u>l</u> ifer	
Cell	ļlfsc	
View	layout	
То		
Library	lfor_teotį	
Cell	lfsr	
View	layou₫	
- Options	·	
💷 Сору	Hierarchical	
M	Skýr Liðinavien	AMALOO NCSU_Analog_Pacts
×	Dopy AD Wests	
W	isws its Gapy	(layout
🔲 Updat	e Instances:	Of Enline Likeary 😑
ОК	Apply	Cancel Help

Figure 23: Copying the lfsr layout view.

Now open up the lfsr layout view from the lfsr\_test library (Figure 24).

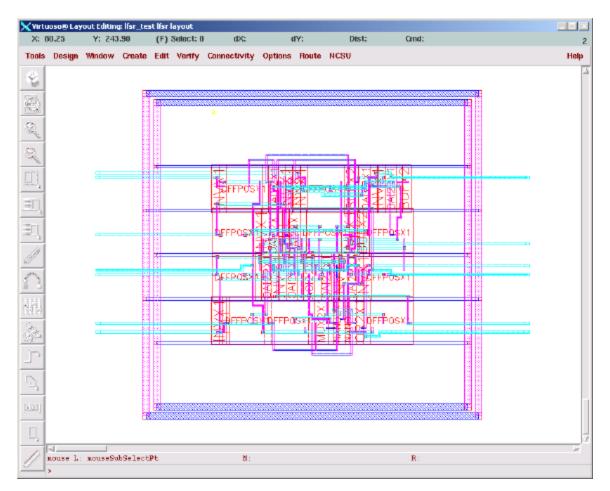


Figure 24: Layout view of lfsr in lfsr\_test library.

Go to Verify, DRC check and clcik on ok in the pop-up like Figure 25.

XDRC					
ОК	Cancel	Defaults	Apply		
Checking	Method	🔶 flat	🔷 hierard	:hical 🔷 hier v	v/o opt
Checking	Limit	🔶 🔶 full 🦂	¢ increm	ental 🔷 by an	ea
		Coortân	ate		
Switch N	lames				
Run-Spe	cific Comr	nand File			
Inclusion	Limit		1000		
Join Nets	With Sam	ne Name			
Echo Con	nmands				
Rules File	9		divaDF	RC. rulį́	
Rules Lib	rary		■  lfs	r_testį́	[
Machine			🔶 local	🔷 remote	Machi

Figure 25:DRC Check.

If you get some errors about malformed metals as in Figure 26, do not worry about it.

```
******** Summary of rule violation for cell
# errors Violated Rules
2 Improperly formed shape - metal1
13 Improperly formed shape - metal3
15 Total errors found
```

Figure 26: Acceptable errors from DRC.

The pins are have to be created from labels that have been automatically generated.

To create these pins first delete all the DRC error markers by going to Verify...Markers... Delete All.

Go to Create... Pins from labels in the Virtuoso editor. You should see a pop-up like Figure 27.

OK       Cancel       Hi         Creation Options       Labels Within:       Cellview $\checkmark$ Selected Instances         Labels:       All       Selected         Labels Found       Pin Layer       Width       Length       Ignore         Imetal1       I	🗙 Create Pins From Labe	els				X
Labels Within:        Cellview        Selected Instances         Labels:       All       Selected       Length       Ignore         Labels Found       Pin Layer       Width       Length       Ignore         Imetal1	OK Cancel					Help
Labels:     All     Selected       Labels Found     Pin Layer     Width     Length     Ignore       Imetal1     dg     I.0     I.0     I.0	Creation Options					
Labels Found     Pin Layer     Width     Length     Ignore       metal1     dg     1.0     1.0     1.0	Labels Within:	🕨 Cellview 📣 Selected In	stances			
metall dg 1.0 1.0	Labels:	All 🔷 Selected				
	Labels Found	Pin Layer	Width	Length	Ignore	
	metal1 dg	💓 metall dg 🖃	1.0	1. Q <u>ĭ</u>		
	metal3 dg			1. 0 <u>ĭ</u>		
			-			

Figure 27: Creating Pins

Just click ok.

Now go to Verify..Extract to extract the circuit for simulation (Figure 28).

X Extractor			×
OK Cancel	Defaults	Apply	Help
Extract Method	🔶 flat	t $\diamondsuit$ macro cell $\diamondsuit$ full hier $\diamondsuit$ incremental hie	er
Join Nets With San	ne Name	Echo Commands	
Switch Names		Extract_parasitic_caps Set Swi	tches
Run-Specific Com	mand File		
Inclusion Limit		1000	
View Names	Extracted	extracted Excell excell	
Rules File		divaEXT. rul	
Rules Library		Ifsr_test	
Machine		🔶 local 🔷 remote 🛛 Machine 📗	

Figure 28: Extracting with parasitic Capacitances.

Note: The NCSU kit will not do high speed clock trees very accurately. Be Carefull!

The extraction report should contain no errors.

#### Running a Spice Simulation:

Open up the extracted view and start the analog environment (Figure 29).

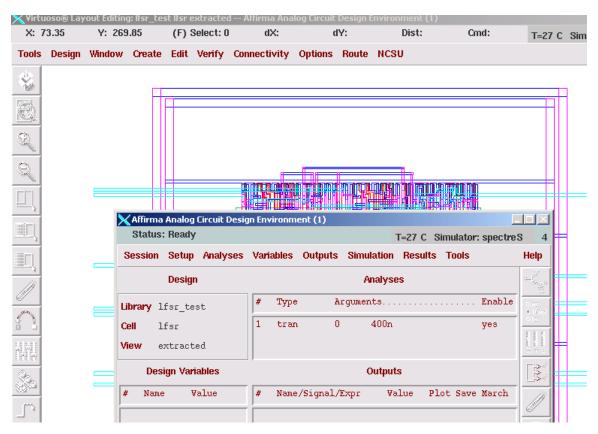


Figure 29: Starting Affirma.

Make sure extracted is in the view list under Setup... Environment.

Make sure the transient analysis is set to 400ns.

To set up the test vectors in the Affirma window go to Setup... Stimulus.. Edit Analog.

🗙 spectreS0: Edit Stimulus File							
ОК	Cancel	Defaults	Apply				
Editor	$\diamond$ tex	:t 🔶 graph	ical				
File Name	-						

Figure 30: Seting up the stimulus file.

Click graphical in the pop-up (Figure 30). and Figure 31should appear.

🔀 Setup Analog Stimuli	×
OK Cancel Apply	Help
Stimulus Type 🔹 Inputs 🔷 Global Sources	
DFFclk /gnd! Voltage dcOFFe /gnd! Voltage dcOFFgnd /gnd! Voltage dcOFFq[0] /gnd! Voltage dcOFFq[1] /gnd! Voltage dcOFFq[2] /gnd! Voltage dcOFFq[3] /gnd! Voltage dcOFFq[4] /gnd! Voltage dcOFFq[5] /gnd! Voltage dcOFFq[6] /gnd! Voltage dcOFFq[7] /gnd! Voltage dc	
Change Enabled I Function dc I Type Voltage AC magnitude	1

Figure 31: Setting Stimulus File.

Now we need to change the inputs to DC sources such as VDD and Gound and inputs to Vpulses. If it is an output leave it alone (q[0]...q[9]

OK Cancel Apply He
Stimulus Type 🛛 🔷 Inputs 🔷 Global Sources
OFF clk /gnd! Voltage pulse "Voltage 1"=0.0 "Volta OFF e /gnd! Voltage dc OFF gnd /gnd! Voltage dc OFF q[0] /gnd! Voltage dc OFF q[1] /gnd! Voltage dc OFF q[2] /gnd! Voltage dc OFF q[3] /gnd! Voltage dc OFF q[4] /gnd! Voltage dc OFF q[5] /gnd! Voltage dc OFF q[6] /gnd! Voltage dc OFF q[6] /gnd! Voltage dc
Enabled Function Type Voltage
AC magnitude
AC phase
DC voltage
Voltage 1
Voltage 2
Delay time
Rise time     1r <sup>×</sup> <sub>i</sub> Fall time     1r <sup>×</sup> <sub>i</sub>
- ×
Pulse width 5rg Period 10rg

Figure 32: Setting the Clock.

OK Cancel Apply He
Stimulus Type 🔺 Inputs 🔷 Global Sources
OFF clk /gnd! Voltage pulse "Voltage 1"=0.0 "Volta
OFF e /gnd! Voltage dc "DC voltage"=5 OFF gnd /gnd! Voltage dc
OFF g[0] /gnd! Voltage dc
OFF q[1] /gnd! Voltage dc
OFF q[2] /gnd! Voltage dc
OFF q[3] /gnd! Voltage dc
OFF q[4] /gnd! Voltage dc
OFF q[5] /gnd! Voltage dc
OFF q[6] /gnd! Voltage dc
J
Change
Enabled 🗐 Function dc 🖃 Type Voltage 🖃
AC magnitude
AC phase
DC voltage

Figure 33: Setting Enable to logic 1 (5V).

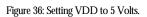
OK Cancel Apply	Help
Stimulus Type 🛛 🔶 Inputs 🔷 Global Sources	
ON clk /gnd! Voltage pulse "Voltage 1"=0.0 "Volta	5
ON e /qnd! Voltage dc "DC voltage"=5	
ON gnd /gnd! Voltage dc "DC voltage"=0	
OFF q[0] /gnd! Voltage dc	
OFF q[1] /gnd! Voltage dc	
OFF q[2] /gnd! Voltage dc	
OFF q[3] /gnd! Voltage dc	1
OFF q[4] /gnd! Voltage dc	
OFF q[5] /gnd! Voltage dc	
OFF q[6] /gnd! Voltage dc	
I	
Change	
Enabled Function dc = Type Voltage =	
AC magnitude	-
AC phase	
DC voltage	

Figure 34: Setting gnd to zero volts.

OFF q[2] /gnd! Voltage	
OFF q[3] /gnd! Voltage	
OFF q[4] /gnd! Voltage	
OFF q[5] /gnd! Voltage	
OFF q[6] /gnd! Voltage	
OFF q[7] /gnd! Voltage OFF q[8] /gnd! Voltage	
OFF q[9] /gnd! Voltage	
	pulse "Voltage 1"=0.0 "Vol
OFF vdd /gnd! Voltage d	
1	
<u>م</u>	
Г. Г.	Channes
	Change
Enabled Function	nuise 🗉 👘 Tyne Voltage 🖃 🗌
Enabled 🔟 Function	pulse 💷 🛛 Type 🛛 Voltage 💷
Enabled _ Function	pulse = Type Voltage =
AC magnitude	
AC magnitude AC phase	
AC magnitude AC phase DC voltage Voltage 1	<u> </u>
AC magnitude AC phase DC voltage Voltage 1 Voltage 2	Image: Control of the second secon
AC magnitude AC phase DC voltage Voltage 1	
AC magnitude AC phase DC voltage Voltage 1 Voltage 2	Image: Control of the second secon
AC magnitude AC phase DC voltage Voltage 1 Voltage 2 Delay time	
AC magnitude AC phase DC voltage Voltage 1 Voltage 2 Delay time Rise time	
AC magnitude AC phase DC voltage Voltage 1 Voltage 2 Delay time Rise time Fall time	V.     V.       V.

Figure 35: Changing reset.

OFF q[2] /gnd! Voltage dc
OFF q[3] /gnd! Voltage dc
OFF q[4] /gnd! Voltage dc
OFF q[5] /gnd! Voltage dc
OFF q[6] /gnd! Voltage dc
OFF q[7] /gnd! Voltage dc
OFF q[8] /gnd! Voltage dc
OFF q[9] /gnd! Voltage dc
ON reset /gnd! Voltage pulse "Voltage 1"=0.0 "Vol
ON vdd /gnd! Voltage dc "DC voltage"=5
J
Change
Enabled Function dc = Type Voltage =
AC magnitude
AC phase
DC voltage



#### Make sure vdd, reset, e, clock, and gnd are all enabled!

Click on apply, but do not close the window.

In the Affirma window Go to Outputs... To be Plotted... Select from schematic and click on all the metal3 lines until the high light (Figure 37)

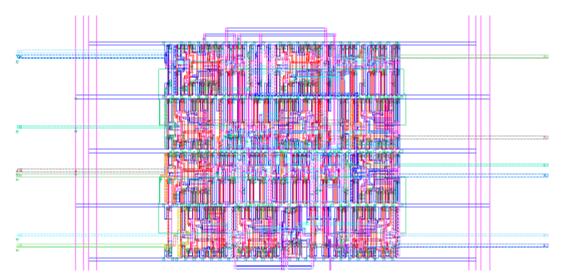
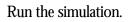


Figure 37: Selecting Outputs to be Plotted.

Look in the Afirma window to make sure all the inputs and outputs are listed.



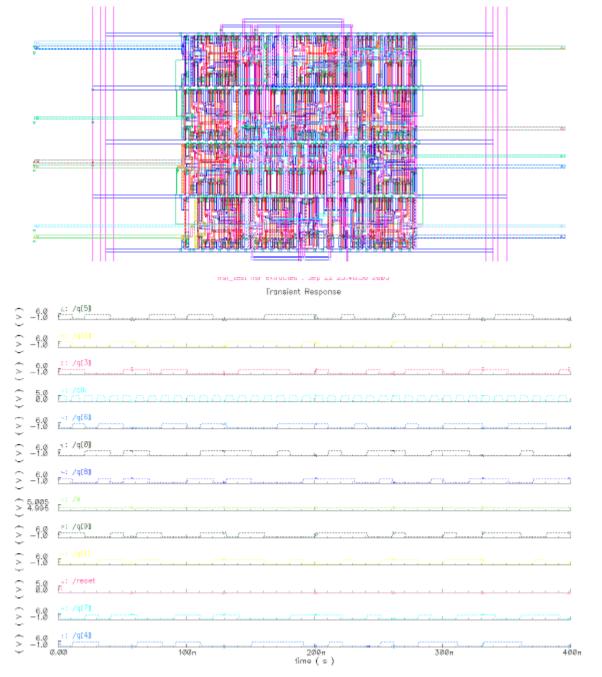


Figure 38: Final Simulation.

You should get something like Figure 38.

# This is the very basics of this flow. I have not shown you how to set any synthesis or place and route parameters. You use it at your own risk.